UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2019-06
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defence held on 18/01/2019 in Esch-sur-Alzette
to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Emmanuel KIEFFER
Born on 30 Mai 1986 in Saint-Avold, France

CO-EVOLUTIONARY HYBRID BI-LEVEL OPTIMIZATION

Dissertation defence committee

Prof. Dr. Pascal Bouvry, dissertation supervisor
Professor, SnT, Université du Luxembourg

Prof. Dr. Franciszek Seredynski
Professor, Cardinal Stefan Wyszynski University

Prof. Dr. Ulrich Sorger, Chairman
Professor, CSC, Université du Luxembourg

Dr. Grégoire Danoy
Research scientist, SnT, University of Luxembourg

Prof. Dr. Anass Nagih, Vice Chairman
Professor, Université de Lorraine

Abstract

Multi-level optimization stems from the need to tackle complex problems involving multiple decision makers. Two-level optimization, referred as "Bi-level optimization", occurs when two decision makers only control part of the decision variables but impact each other (e.g., objective value, feasibility). Bi-level problems are sequential by nature and can be represented as nested optimization problems in which one problem (the "upper-level") is constrained by another one (the "lower-level"). The nested structure is a real obstacle that can be highly time consuming when the lower-level is $\mathcal{NP} - hard$. Consequently, classical nested optimization should be avoided. Some surrogate-based approaches have been proposed to approximate the lower-level objective value function (or variables) to reduce the number of times the lower-level is globally optimized. Unfortunately, such a methodology is not applicable for large-scale and combinatorial bi-level problems.

After a deep study of theoretical properties and a survey of the existing applications being bi-level by nature, problems which can benefit from a bi-level reformulation are investigated. A first contribution of this work has been to propose a novel bi-level clustering approach. Extending the well-know "uncapacitated k-median problem", it has been shown that clustering can be easily modeled as a two-level optimization problem using decomposition techniques. The resulting two-level problem is then turned into a bi-level problem offering the possibility to combine distance metrics in a hierarchical manner. The novel bi-level clustering problem has a very interesting property that enable us to tackle it with classical nested approaches. Indeed, its lower-level problem can be solved in polynomial time. In cooperation with the Luxembourg Centre for Systems Biomedicine (LCSB), this new clustering model has been applied on real datasets such as disease maps (e.g. Parkinson, Alzheimer). Using a novel hybrid and parallel genetic algorithm as optimization approach, the results obtained after a campaign of experiments have the ability to produce new knowledge compared to classical clustering techniques combining distance metrics in a classical manner.

The previous bi-level clustering model has the advantage that the lower-level can be solved in polynomial time although the global problem is by definition $\mathcal{NP}$-hard. Therefore, next investigations have been undertaken to tackle more general bi-level problems in which the lower-level problem does not present any specific advantageous properties. Since the lower-level problem can be very expensive to solve, the focus has been turned to surrogate-based approaches and hyper-parameter optimization techniques with the aim of approximating the lower-level problem and reduce the number of global lower-level optimizations. Adapting the well-know bayesian optimization algorithm to solve general bi-level problems, the expensive lower-level optimizations have been dramatically reduced while obtaining very accurate solutions. The resulting solutions and the number of spared lower-level optimizations have been compared to the bi-level evolutionary algorithm based on quadratic approximations (BLEAQ) results after a campaign of experiments on official bi-level benchmarks. Although both approaches are very accurate, the bi-level bayesian version required less lower-level objective function calls.

Surrogate-based approaches are restricted to small-scale and continuous bi-level problems although many real applications are combinatorial by nature. As for continuous problems, a study has been performed to apply some machine learning strategies. Instead of approximating the lower-level solution value, new approximation algorithms for the discrete/combinatorial case have been designed. Using the principle employed in GP hyper-heuristics, heuristics are trained in order to tackle efficiently the $\mathcal{NP}-hard$ lower-level of bi-level problems. This automatic generation of heuristics permits to break the nested structure into two separated phases: *training lower-level heuristics* and *solving the upper-level problem with the new heuristics*. At this occasion, a second modeling contribution has been introduced through a novel large-scale and mixed-integer bi-level problem dealing with pricing in the cloud, i.e., the Bi-level Cloud Pricing Optimization Problem (BCPOP). After a series of experiments that consisted in training heuristics on various lower-level instances of the BCPOP and using

them to tackle the bi-level problem itself, the obtained results are compared to the "cooperative coevolutionary algorithm for bi-level optimization" (COBRA).

Although training heuristics enables to *break the nested structure*, a two phase optimization is still required. Therefore, the emphasis has been put on training heuristics while optimizing the upper-level problem using competitive co-evolution. Instead of adopting the classical decomposition scheme as done by COBRA which suffers from the strong epistatic links between lower-level and upper-level variables, co-evolving the solution and the mean to get to it can cope with these epistatic link issues. The "CARBON" algorithm developed in this thesis is a competitive and hybrid co-evolutionary algorithm designed for this purpose. In order to validate the potential of CARBON, numerical experiments have been designed and results have been compared to state-of-the-art algorithms. These results demonstrate that "CARBON" makes possible to address nested optimization efficiently.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor Prof. Pascal Bouvry who gave me the chance to integrate his dynamic research team. My sincere thanks also goes to Prof. Nagih who trusted me and gave me the possibility to meet Prof. Bouvry. I also would like to express my deepest gratitude to Dr. Danoy for the continuous support during my PhD thesis, for his patience and the number of advices he gave me. Besides, I would like to thank all members of the Parallel Computing & Optimisation research team for all their precious support. In particular, I think about M. R. Brust, M. Guzek and C. Fiandrino. Thanks to the High Performance Computing team, I gained so many precious knowledge with their guidance. I would like to thank S. Varrette, V. Plugaru, H. Cartiaux, C. Parisot, X. Besseron and S. Peters. I must say that the atmosphere in the Parallel Computing & Optimisation and High Performance Computing teams could not be better and I am really enthusiastic to continue to work with all of them. Of course, my deepest gratitude go to my lovely wife, my beloved parents and my little sister for their support and patience.

Contents

List of Programs

List of Figures

List of Tables

Symbols

UL	Upper-Level
LL	Lower-Level
(x, y)	Bi-level decision vector
x	Upper-level decision vector
y	Lower-level decision vector
$F(x, y)$	Upper-level objective function
$f(x, y)$	Lower-level objective function
$G(x, y) \leq 0$	Upper-level set of constraints
$g(x, y) \leq 0$	Lower-level set of constraints
$\theta(x) := \min\{f(x, y) \quad s.t. \quad g(x, y) \leq 0\}$	Lower-level optimal value for a given x
$\Psi(x) = \{y \in Y : g(x, y) \leq 0, f(x, y) \leq \theta(x)\}$	Lower-level rational decision set solution set mapping
$\mathcal{IR} = \{(x, y), y \in \Psi(x)\}$	Inducible Region
$\phi^o(x) = \underset{y \in \mathbf{Y}}{\arg\min}\{F(x, y) : y \in \Psi(x)\}$	Optimistic rational reaction
$\phi^p(x) = \underset{y \in \mathbf{Y}}{\arg\max}\{F(x, y) : y \in \Psi(x)\}$	Pessimistic rational reaction
$\nabla_y f$	denotes the partial derivatives of f with respect to y
$Q(x)$	is the risk function, also called "The value function of the recourse problem"
KKT	Karush-Kuhn-Tucker
$relint\, S(x)$	the relative interior of $S(x)$
$bd\, S(x)$	the boundary of $S(x)$
PAM	Partition Around Medoids
HCW	Hierarchical Clustering with Ward criteria
HPC	High Performance Computing
PD map	Parkinson Disease map
GO	Gene Ontology
DO	Disease Ontology
$GO\ BP$	Gene Ontology for Biological processes
$GO\ CC$	Gene Ontology for Cellular compartments
$GO\ MF$	Gene Ontology for Molecular function
AlzPathway map	Alzheimer's Disease map
BLEAQ	Bilevel Evolutionary Algorithm based on Quadratic Approximations
$L - BFGS - B$	Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm
BO	Bayesian Optimization
$\mathcal{GP}$	Gaussian Processes
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with parameters μ and σ
MPI	Maximum Probability Improvement

EI	Expected Improvement
LCB	Lower-Confidence Bound
SLSQP	Sequential Least Squares Programming
GP hyper-heuristic	Genetic Programming hyper-heuristic
GA	Genetic Algorithm
MKP	Multi-dimensional Knapsack
ILP	Integer Linear Programming
BCPOP	Bi-level Cloud Pricing Optimization Problem
AGH	Automatic Generation of Heuristic
COBRA	A cooperative coevolutionary algorithm for bi-level optimization
BIGA	Bi-level optimisation using genetic algorithm
CARBON	CoevolutionAry HybRid Bi-level OptimizatioN algorithm
CARTESIAN	maChine leArning foR auTomatEd optimiSatIon Algorithm desigN

This dissertation is dedicated to my family

Part I

Introduction and background

Chapter 1

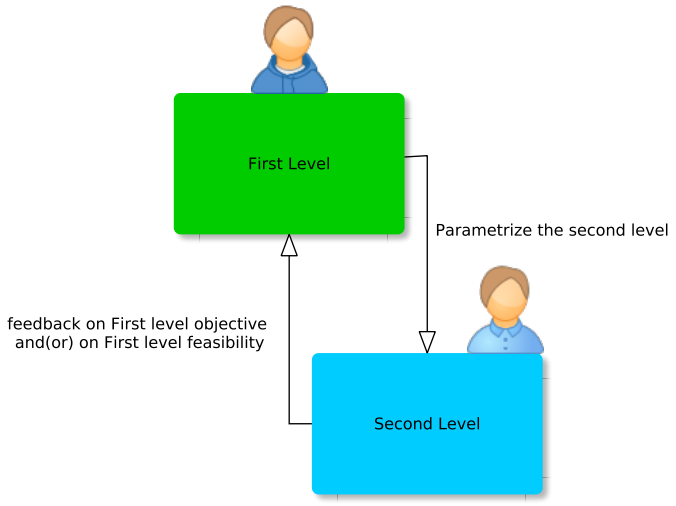
Introduction

Contents

1.1 Introduction to Bi-level optimization

In this PhD work, we focus our attention to bi-level optimization problems that stem from both Optimization and Game Theory. Real life problems such as transportation, planning and management often involve several decision makers who take optimal decisions depending on the interactions between each other. A bi-level scenario usually appears when two decisions makers have to take decisions hierarchically, i.e., one after another. Bi-level optimization problems generalize the non-cooperative Stackelberg Games (see Figure 1.1). They are two-stage models where the first player is called the "leader" and the second player is called the "follower". Taking the point of view of the leader, his problem depends on the reaction of the follower. In order to take the right decision, it has to compute the optimal strategy of the follower called " the follower rational reaction". In bi-level optimization, the leader problem is referred to as the "upper-level problem" while the follower problem is called the "lower-level problem". The order between levels is very important since they do not provide the same optimal solution. This nested structure implies that a bi-level feasible solution is necessary lower-level optimal.

Bi-level problems are intrinsically hard even for convex problems. The simplest bi-level linear programs (BLP) is strongly NP-hard [35, 26]. Exact approaches are thus not efficient but have been extensively investigated. Classical approaches are mainly based on reformulations of the bi-level program into a single-level program, which is then solved using classical optimization algorithms. Unfortunately, integer and mixed integer bi-level problems (MIBLP) have been seldom investigated and most of the studies found in the literature focus on



$$\begin{aligned} \min_{x \in \mathbf{X}} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & \min_{y \in \mathbf{Y}} \quad f(x,y) \\ & \text{s.t.} \quad g(x,y) \leq 0 \end{aligned}$$

FIGURE 1.1: A Bi-level optimization problem and its general mathematical formulation

specific problems where interesting properties can simplify levels. The last decade has seen a renew of interest concerning bi-level optimization and notably in Evolutionary Computing. Inspired by nature, these approaches have been widely used in single-level optimization cases to tackle NP-hard problems and more generally large and complex optimization systems. Indeed for $\mathcal{NP}$-hard problems and under the assumption that $\mathcal{P} \neq \mathcal{NP}$, it does not exist algorithms with polynomial complexity solving every instance to optimality. The goal is therefore to reach solutions as close as possible to the optimal ones while guaranteeing a polynomial time complexity. Since convex bi-level problems are $\mathcal{NP}$-hard contrary to their single-level equivalents, the scope of evolution computing algorithms has been extended. Nonetheless, bi-level problems are still a real challenge for these algorithms due to a high computational. Despite their high complexity, bi-level optimization problems model accurately complex situations.

Bi-level optimization has its root in two fundamental domains: Optimization and Game Theory. The next section provides a short historical background that describes these two theories that are essential tools to deal with bi-level applications.

1.2 Historical background

1.2.1 Optimization theory

Optimization is a cross-cutting domain that has applications in many areas such as applied mathematics, finance, engineering and other domains involving decision making. In fact, human beings optimize unconsciously when executing tasks. We naturally schedule our life and tend to always want to find the "right" or "best" decision. In fact, we constantly attempt to discover the "best" decisions according to some utility functions. It is not excessive to admit that optimizing is natural. In fact, the "least action principle" is quite common in Nature. Many systems attempt to reach a state of minimum energy, i.e.," a stable state. Another example is light travels that are considered as the shortest distance between two points. Optimizations problems and investigations to resolve them goes back a very long time. Ancient Greek mathematicians had already to deal with optimization queries. For example, the Heron's shortest path problem is probably one of the first mathematically stated optimization problem. Much later, J. Bernouilli challenged the scientific community to find the shape that connect two points in shortest time of a falling body which is subject to gravity. The answer seems quite simple

for us today: the line of steepest descent. Later in 1746, Maupertuis proposed the well-know "principle of least action" which unifies laws of physical motion. The French engineer, Gaspard Monge, was certainly the first one who investigated optimal transportation and resource allocation in 1781. The 19th century has seen the beginning of the optimization theory that we know today. Frederich Gauss explained that the "method of least-squares" can be used to predict orbital locations. However, Adrien Legendre in 1805 was the first one who mathematically defined the "method of least-squares". In 1847, Cauchy introduced in "Méthode générale pour la résolution des systèmes d'équations simultanées" a general method for solving systems of equations iteratively. The 20th centuries have laid the official foundations of the Optimization Theory. The book " Theory of Minima and maxima" is the first comprehensive book on Optimization and has been published by H. Hancok in 1917. The diet problem studied during the second World War was motivated by the U.S. Army to minimize the feeding cost of soldier while providing healthy diets. George Stigler investigated this problem and proposed heuristics to solve it. In 1947, George Dantzig made an important breakthrough by inventing the "Simplex method" for solving linear programming. Even nowadays, most of the existing solvers implement his simplex method that has very efficient results in practice despite its worst-case complexity. In 1984, Karmakar proposed the first algorithm with polynomial time complexity to solve continuous linear programming. In the middle of the 20th Century, Kuhn and Tucker studied non-linear optimization problems and re-developed some optimality conditions that had already been discovered by Karush in 1939. These conditions are named after the three contributors, Karush-Kuhn-Tucker (KKT) conditions. Another important step in optimization has been done by Bellman who proposed "Dynamic Programming". The second part of the 20th century has been very fruitful in terms of scientific contributions. Figure 1.2 describes a taxonomy of optimization problems that can be encountered today in the literature but also in practice. Many topics have been addressed depending on their mathematical properties. From Stochastic to Multi-objective programming, the scope of optimization has been dramatically extended and permits today to cope with very complex and specific problems.

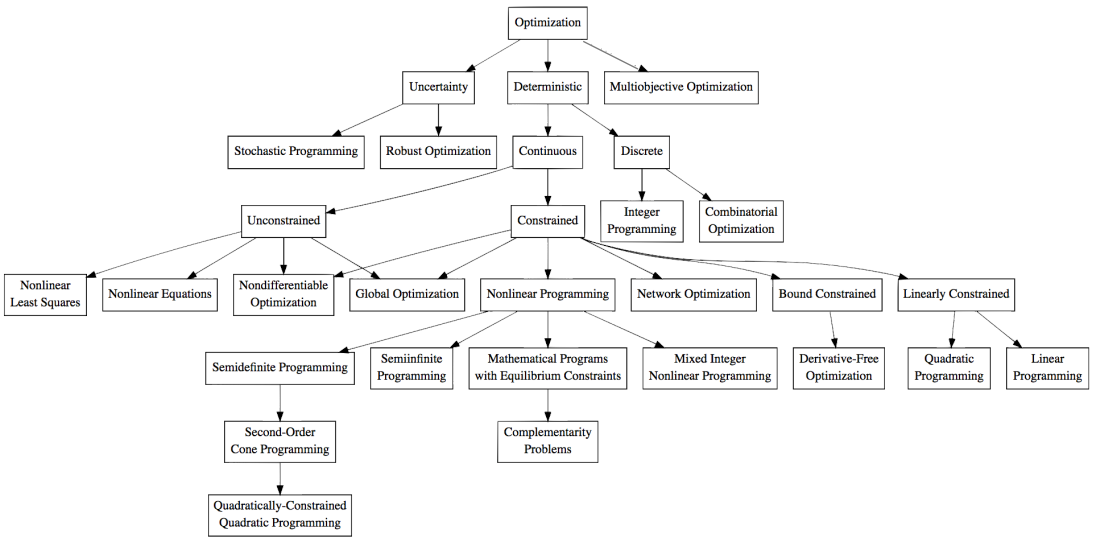


FIGURE 1.2: Type of optimization problems (source: https://neos-guide.org/content/optimization-taxonomy)

Bi-level problems are optimization problems in which a decision maker attempts to discover the optimal solution of some defined utility function with respect to a second decision maker with a conflicting utility function. In fact, bi-level optimization problems generalize the notion of constrained optimization problems in which

constraints are not only equalities/inequalities expressions but also whole optimization problems. Such kind of nested modeling often occurs in Game Theory.

1.2.2 Game theory

According to Roger B. Myerson, Game Theory can be described as "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers". In other words, games are mathematical problems in which multiple decision makers are involved. Each decision maker, i.e., player has its own utility function that defines the desirability of a given strategy. In addition, a player is said "rational" if he attempts to play the strategy that maximizes his own utility function. Strategies represent all the possible actions of a given player. A mixed strategy is a randomization scheme of the employed actions with regards to a probability distribution. In games, players seek for equilibrium states maximizing their own utility function. The so-called Nash equilibrium describes a state in which no player can unilaterally modify its strategy to improve its current utility value. As for the Optimization Theory, games have received a great deal of attention. In 1973, J. Waldegrave established the well-known minimax mixed strategy for zero-sum games with two persons. Later on, A. Cournot published the book "Researches into the Mathematical Principles of the Theory of Wealth" in which chapter 7 introduced competition of producers. In 1913, chess is proved strictly determined by Zermelo's Theorem, i.e., chess has only one individually rational payoff profile in pure strategies. Emile Borel, a french mathematician, proposed a first formal description of Game Theory in 1921 which has been strengthened by John Von Neumann in 1928. Game Theory became officially a field of study after the release of the book "Theory of Games and Economic" by Oskar Morgenstern and John Von Neumann in 1944. However, this is certainly the demonstration of John Nash that finite games have always an equilibrium state which plays a central role in non-cooperative game theory. Due to the complex interactions between players, several types of games can be highlighted as depicted in Figure 1.3. Game Theory has a major role in many domains such Economics, Resource Allocation, Networking and Artificial Intelligence.

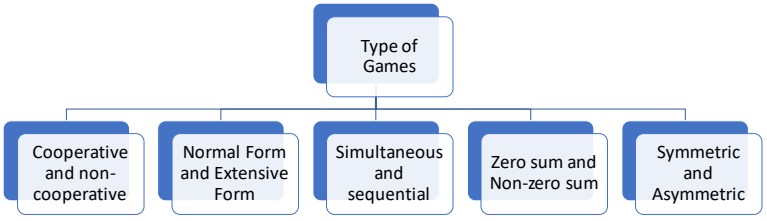


FIGURE 1.3: Type of Games

Bi-level problems also generalize the notion of Stackelberg games [346] clearly depicting an adversarial situation between two players. Compared to a Cournot games, Stackelberg games give advantage to the leader and are perfect information games.

1.3 Thesis roadmap

1.3.1 Research problematic

Beyond their wide scope of applications that we are going to describe in chapter 2, bi-level problems are extremely challenging from a mathematical point of view. The nested structure makes bi-level optimization problem $\mathcal{NP}$-hard even if all components (e.g. objective functions, constraints) are linear and continuous. The bi-level feasible search space is not known apriori even if both upper-level and lower-level search spaces are clearly defined. In fact, a bi-level feasible solution is necessary lower-level optimal. Consequently, an $\mathcal{NP}$-hard lower-level problem makes bi-level feasibility $\mathcal{NP}$-hard as well.

As it will be shown in chapter 2, many approaches have been proposed in the literature to solve mostly small-scale and continuous bi-level optimization problems. Assuming convexity and regularity conditions, they generally reformulate the original bi-level into a single-level problem by replacing the lower-level problem with its first-order conditions. Such a strategy enables the use of classical optimization techniques to solve the resulting single-level problem. On the contrary, very few algorithms have been designed to cope with combinatorial versions even if they are the most encountered problems in real applications. While classical approaches are very restricted to some well-posed bi-level problems, bi-level metaheuristics have a wider application scope but suffer from the time-consuming nested optimization scheme that has been mostly utilized in the literature. Indeed, they repeatedly solve one level after the other which justifies the limited numbers of contributions that attempt to solve large-scale and combinatorial bi-level problems.

In this PhD thesis, we intend to investigate metaheuristic approaches and strategies that extend the scope of bi-level metaheuristics to large-scale and combinatorial bi-level optimization problems.

1.3.2 Research questions

We identified two main directions to develop a research plan. The first one assumed that nested optimization is unavoidable and that explicit lower-level optimizations should be minimized using approximation strategies. The second one would defend the idea of breaking the inherent nested structure. As a consequence, we define 3 main research questions:

- **P1**: What are the newest strategies in the literature to cope with the nested structure ?
- **P2**: How can we decrease the number of explicitly nested optimization or at least reduce its computational cost ?
- **P3**: What kind of combinatorial bi-level problems could benefit from a bi-level modeling ?

While the first two questions are directly related to resolutions approaches, the third one is dedicated to the definition of large-scale applications that could benefit from a novel bi-level modeling.

1.3.3 Research objectives

In order to answer the research questions, we first study the theoretical properties of bi-level optimization problems. A deep study of the advantages and limitations of the existing bi-level metaheuristics provide an interesting starting point to avoid some unpromising directions. These tasks constitute the first objective O1.

Directly related to the question P3, we study large-scale and combinatorial applications that can benefit from an alternative bi-level modeling. This represents the objective O2.

Our third objective O3 explores methodologies reducing the lower-level optimization cost. Indeed, the nested optimization cost is directly related to the complexity of the lower-level problem. By focusing on more efficient strategies to handle it, we could drastically improve bi-level metaheuristics.

Objective O4 investigates a decentralized approach such as coevolution. It is particularly well-suited for large-scale applications.

Finally the last objective O5 identifies the key characteristics and concepts of bi-level metaheuristics to provide an **hybrid** and **decentralized algorithm** able to tackle **large-scale bi-level combinatorial problems**.

- **O1**: Survey the literature on bi-level optimization and study the advantages and limitations of existing bi-level metaheuristics
- **O2**: Propose novel bi-level and combinatorial models for real-world applications
- **O3**: Investigate approaches to reduce lower-level optimization cost for bi-level problems
- **O4**: Investigate decentralized approaches such as co-evolution
- **O5**: Propose an hybrid co-evolutionary algorithm for bi-level optimization: **CARBON**

1.3.4 Thesis methodology and outline

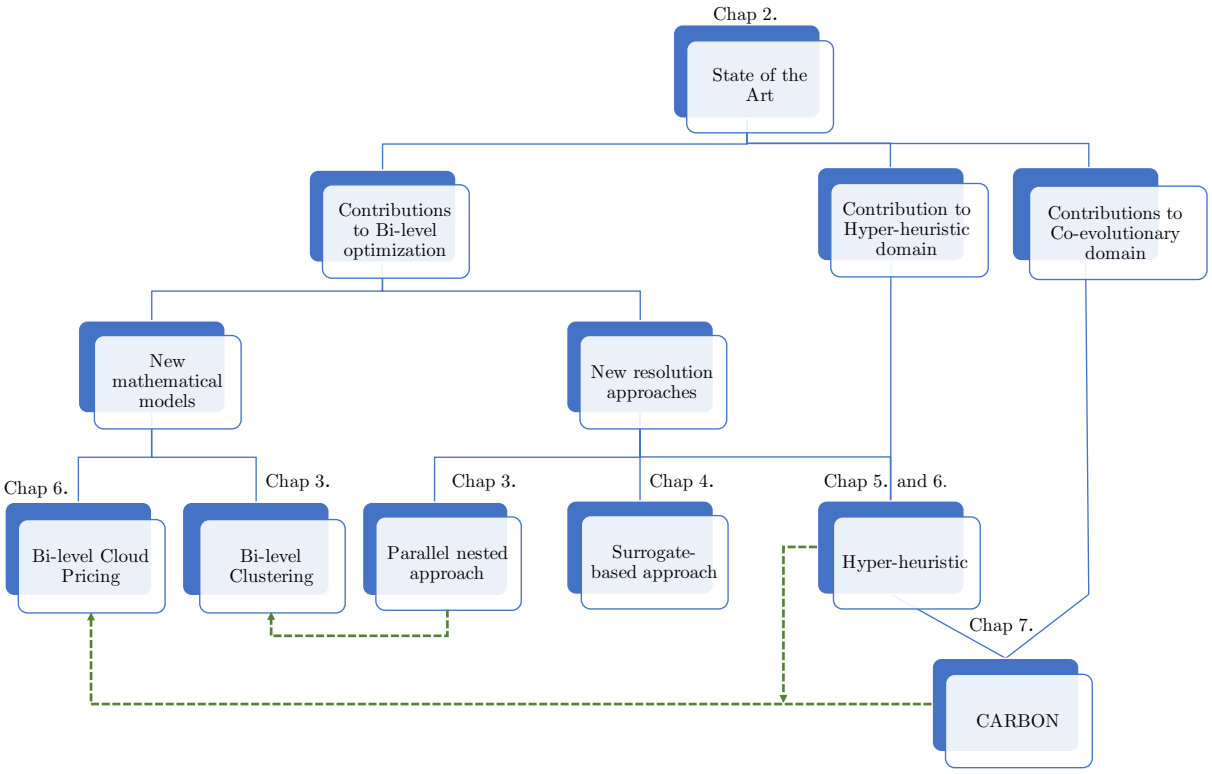


FIGURE 1.4: Thesis workflow

1.3.4.1 Methodology and contributions

Figure 1.4 illustrates the different paths followed by this thesis to answer the research questions stated in section 1.3.2. Among the contributions made to the bi-level optimization field, **chapters 3** and **6** propose and introduce formally two novel mathematical models related to real applications in which bi-level modeling can be naturally beneficial.

The model designed in **chapter 3** considers the challenging problem of extracting knowledge from biomedical repositories while **chapter 6** describes a novel two-level pricing model for cloud service computing. New Resolution approaches are developed for each model in order to tackle their large-scale instances. Furthermore in **chapter 4**, we adapt a well-known black-box optimization algorithms to solve general but continuous bi-level problems through a series of official benchmarks.

Besides the contributions to bi-level optimization field, we also contribute to the hyper-heuristic field of study by extending its scope of application. We demonstrate that constructive hyper-heuristics are efficient tool to deal with complex problems. **Chapters 5 and 6** focus on Genetic Programming Hyper-Heuristics, a remarkable tool to generate automatically heuristics.

Last but not least, we show the relevance of co-evolutionary approaches to cope with large-scale bi-level problems by addressing the issue of strong epistatic links occurring between both level of decision variables. Using the knowledge gained through **chapters 5 and 6**, we develop a hybrid co-evolutionary algorithm for bi-level optimization.

1.3.4.2 Outline

Chapter 2 lays the theoretical foundations of bi-level optimization problems. After a description of the different bi-level applications found in the literature, we discuss the existing bi-level variants and their associated mathematical formulations. Bi-level optimization is a field of study that overlaps many others. We show the tight relationship with other classes of problems that can be found in the literature. Finally we will conclude this chapter with a survey on classical and advanced resolution approaches while studying their strength and limitations.

Chapter 3 introduces a first contribution to the bi-level optimization field by proposing a novel clustering model in order to perform bi-level clustering optimization. Extending the well-know "uncapacitated k-median problem", we propose a new clustering model relying on a two-level optimization problem obtained using decomposition techniques. This resulting two-level problem can be turned into a bi-level optimization problem enabling a novel way to combine distance metrics. In this chapter, we also exhibit some useful properties allowing the efficient application of classical nested optimization strategies through a hybrid and parallel genetic algorithm. In order to examine the added value brought by the proposed bi-level clustering, numerical experiments have been performed in cooperation with the Luxembourg Centre for Systems Biomedicine (LCSB) on real datasets such as disease maps (e.g. Parkinson, Alzheimer).

Chapter 4 dives into more complex and general bi-level problems that do not possess specific properties to alleviate the computational cost of the lower-level problem. Therefore, we investigate metaheuristics with approximation techniques, also known as "surrogate-based or model-based approaches". Particularly well-adapted to time-consuming and black-box optimization problems, we adapt Bayesian Optimization to tackle general bi-level problems. After a campaign of experiments on official bi-level benchmarks, we compare our results and methodology against the bi-level evolutionary algorithm based on quadratic approximations (BLEAQ), i.e., a

reference in the bi-level evolutionary field. Although these surrogate-based approaches are very promising and efficient, they are restricted to small-scale and continuous bi-level problems.

Chapter 5 is an intermediary chapter focusing on the "learning to optimize" paradigm. Like surrogate-based approaches for continuous problems, we develop a machine learning strategy to generate automatically heuristics in order to solve efficiently combinatorial problems. In this chapter, we turn our attention to GP Hyper-heuristics approaches and experiment them on the multidimensional knapsack problem. Although this problem is a single-level problem, it allows us to validate the concept of "trained heuristics" on a well-known combinatorial problem.

Chapter 6 is the extension of chapter 5 to large-scale and combinatorial bi-level optimization problems. At this occasion, we introduce a second modeling contribution through a novel large-scale and mixed-integer bi-level problem dealing with pricing in the cloud, i.e., the Bi-level Cloud Pricing Optimization Problem (BCPOP). The lower-level problem is a parametric optimization problem which strongly depends on the upper-level decision variables. Consequently, the different lower-level instances can be considered as family of instances with a common and parametrized part. Instead of learning the lower-level objective function, we could automatically learn more efficient and dedicated heuristics to tackle this family of instances. Using heuristics trained to solve the lower-level covering problem in a first phase, we tackle the BCPOP using a hybrid genetic algorithm during a second phase. In order to evaluate the potential of the proposed methodology, we compare our results against a state-of-the-art algorithm.

Chapter 7 investigates a decentralized approach based on the co-evolutionary paradigm. Our goal is to break the nested structure and solve both levels during the same optimization process contrary to chapter 6 where a two-phase approach has been developed. We show that the strong epistatic links between the upper-level and lower-level decision variables are a real drawback for classical bi-level co-evolutionary metaheuristics. To cope with this issue, we design a competitive co-evolutionary algorithm, i.e., CARBON that optimize the upper-level problem while evolving efficient heuristics for the lower-level problem. Such an approach allows us to work with two independent populations: one evolving the upper-level solutions while the second one evolves strategies (heuristics).

Finally, the **Conclusion** chapter summarizes all the contributions and outcomes that have been successfully obtained during this PhD work. We also provide some obtained contributions to other related topics that are directly following the knowledge gained during this PhD thesis. Future works and perspectives are discussed at the very end of this chapter. We also demonstrate that the "learning to optimize" paradigm can lead to novel promising research areas.

Chapter 2

Bi-level Optimization: state of the art

Contents

2.1 Introduction

The literature contains numerous real-world problems that have been modeled as bi-level programs. Their current mathematical formulation has been introduced by the seminal work of Bracken and McGill in [51] who designated them in their works as "mathematical programs with optimization problems in the constraints". First applications have also been provided by Bracken et al. in [52] and were related to defense applications. Many other application domains can be found with suitable bi-level representation. Transportation [272], planning [66] and management [27] are domains in which bi-level optimization problems can be found in practice since they naturally involve hierarchical levels of decisions. Famous problems like the "Toll setting problems" have been intensively studied in the literature [53, 359, 190]. Some bi-levels models have been proposed to tackle environment economics problems [374],[48]. Their main objectives are to determine a tax policy that offers a compromise between revenues and environmental impacts. We can also find bi-level problems when designing the conditions of chemical reactions [306, 158]. Indeed, the amount of reactants transformed into products strongly depends on the initial conditions and the precise moment where the system reached an equilibrium state. The optimal design of structures can be also considered using bi-level modeling [167]. Structural optimization generally requires to minimize the weight or cost of building a structure knowing that some mechanical constraints (e.g., forces) are defined in terms of variational equalities. Bi-level modeling has also been employed for control theory applications which are bi-level by nature [347], [185]. Finally, bi-level optimization is particularly well-adapted for meta-optimization. For instance, evolutionary algorithms and machine learning techniques often require a large number of parameters. The determination of those parameters can be achieved through bi-level optimization [240], [338]. Section 2.2 proposes a classification of the existing bi-level problems that have been investigated in the literature.

In section 2.3, we formalize bi-level optimization problems and describes some theoretical key aspects that should be taken into account when considering bi-level mathematical models. Computational complexity and optimality conditions are discussed in details as well. As aforementioned, bi-level optimization stems form two well-know and studied theories. Bi-level problems have strong relationship with some other domains (e.g., game theory, multi-stage problems, semi-infinite programming). These relationships are demonstrated at the end of the section.

Classical resolution approaches are investigated in section 2.4. Although most of the existing approaches have been designed to tackle bi-level problems with convex levels, we also dedicate time to discuss non-convex bi-level programming and more precisely problems with combinatorial levels. Classical approaches can only cope with very small-scale instances of bi-level problems. This is the reason why section 2.5 is devoted to metaheuristics for bi-level optimization. Although these approaches cannot guarantee optimal solutions, they made possible to tackle real bi-level applications with significant higher number of decision variables. As we will discussed it, large-scale and combinatorial bi-level problems are still a real challenge for metaheuristics. In this PhD work, we investigate and discuss the properties of the 5 main categories of bi-level metaheuristics existing so far in the literature.

2.2 Bi-level optimization problems and their applications

To the best of our knowledge, 8 main categories of applications have drawn our attention since they contain very up-to-date problems. Figure 2.1 illustrates existing applications which have been modeled as bi-level problems. All these categories have one specificity in common: they are naturally **hierarchical**. Hereafter, each of them is described in details and accompanied with real example cases found in the literature. Obviously,

one should not believe that we provide an exhaustive view of all the existing applications even though we took a particular attention to obtain the widest possible overview. Finally, Table 2.1 summarizes the different bi-level applications described in this section.

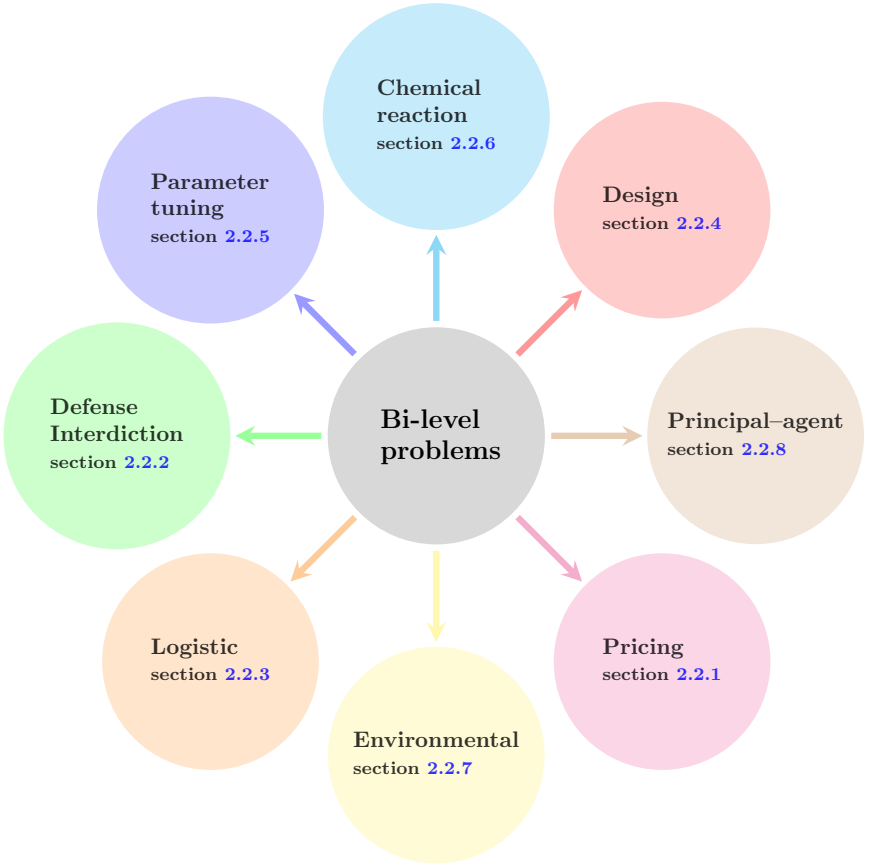


FIGURE 2.1: Bi-level application domains

2.2.1 Pricing problems

Pricing problems often occurs when a first entity influences the objective value of a second one. This very specific category of bi-level problems occurs naturally in economic, social and management problems. The literature is replete with many versions of the "Toll setting problem" which has been the first bi-level pricing problem introduced by Brotcorne et al. in [53]. In this problem, a network of roads is operated by an authority which set toll prices. Naturally, this authority would like to determine the optimal prices knowing that customers will try to minimize their travel cost. High toll prices would lead drivers to take secondary roads which could saturate them and reduce the authorities income. This authority wishes to determine the optimal threshold that should not be overcome. Toll pricing is highly relevant in the economic context and has been therefore intensively investigated in [272, 83, 229, 263, 189, 123, 359, 190, 295, 105]. In addition, multi-objective versions have been studied in [390, 366, 334]

2.2.2 Defense/Interdiction problems

Defense applications are historically the first applications modeled as bi-level problems by J. Bracken and J. McGill [52]. In the context of the Cold War, defense applications were key of major importance especially at the Nuclear Era. Most of these problems were designed to determine an optimal defense policy in case of attack.

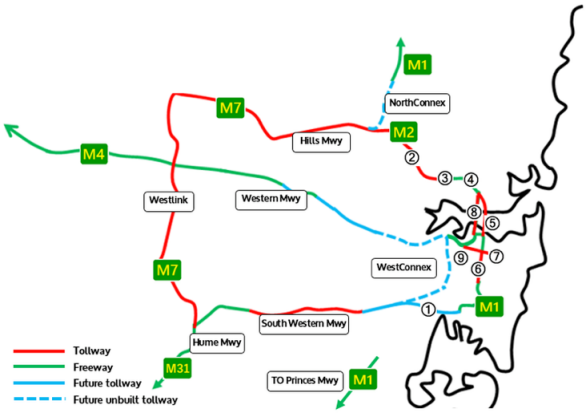


FIGURE 2.2: Toll settings and road pricing problems

Also known as "Interdiction" problems", bi-level models for defense applications have found today many other application domains from interdicting Nuclear Weapons [54, 55] to preventing Network Intrusion [379, 196]. In these models, the defender controls the upper-level problem and aims at minimizing the potential damages caused by an attacker. Therefore, the defender needs to model his problem using two nested optimization levels in order to take into account the potential reactions of the attacker to interdict damages. From a mathematical point of view, interdiction problems arises when the upper-level decision maker directly acts on the lower-level constraints to reduce its operating margin. Further works on interdiction problems and defense applications have been illustrated in [5, 55, 321, 56].

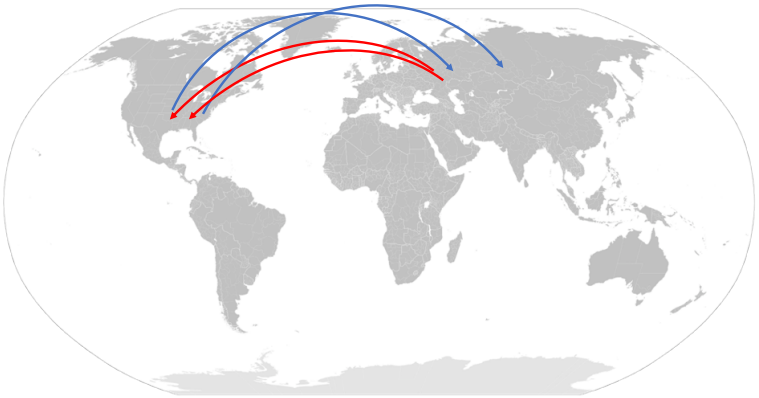


FIGURE 2.3: Strategic defense application

2.2.3 Logistic problems

Facility location problem are a famous kind of optimization problems modeling the optimal placement of facilities while taking into account external factors or constraints. As example, we can consider a company wishing to open new warehouses in a country to expand its activities. For this purpose, this company knows several locations to install the future facilities but wants to minimize the total installation cost and shipping costs. Despite its single-level modeling, facility location problems often involve several decision makers (e.g. competitor's facilities, customers and public authorities). A classical facility location problem only focus on the company's problem while a bi-level model could also take into account the others actors. For example, a public authority providing subsidies at some locations or the impact of the presence of existing competitors. Bi-level modeling and associated resolution approaches have been undertaken in many scientific works such as [299, 257, 68, 264, 64, 344].

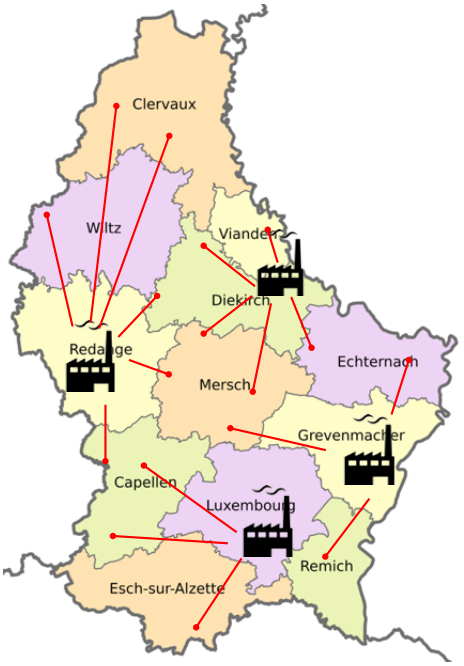


FIGURE 2.4: Logistic problems

2.2.4 Design problems

Designing structures, networks or complex objects is a really challenge for private companies and public authorities. These problems are generally strongly constrained due to the necessity of guaranteeing security, reliability and trust. For example when designing a new road network layout, public authorities want to minimize the cost while ensuring optimal road use for users. In fact all problems where the designer of a solution is not the only user of these solutions can be considered as a bi-level problem. The designer needs to take into account the actions undertaken by the other users. As additional example, we could cite the design of any industrial product which is tested under many extreme conditions of use to measure its reliability. Generally, these problems are modeled as "Mathematical programming with equilibrium constraint" and are equivalent to bi-level problems. The equilibrium constraints can represent contact forces, minimum state energy or user optimal decisions. The scope of this category is very wide and encompass numerous real-world applications. Nevertheless, most of the existing contributions deal with Network Design Problems [234, 260, 388, 389, 385, 75, 142, 124, 271, 382, 311, 65], Structural Optimization [167, 79, 401] and Topology Optimization [214, 215].

2.2.5 Parameter tuning problems

Modern optimization algorithms and machine learning approaches contain a certain number of required parameters to be set before execution. These parameters have key influences and require time-consuming efforts to be optimized. Classical approaches to tune parameters often rely on brute force strategies and random searches. They are not suitable for large scale problems and suffer therefore from a lack of scalability. Recently, bi-level modeling has been considered as valid alternative for this purpose. Indeed, parameter tuning can be seen as a two-step process in which a decision maker would like to determine a set of optimal parameters according to a given algorithm. Consequently, the upper-level problem consists in discovering these parameters while minimizing the total computing time. The lower-level problem is represented by the algorithm to be tuned which implements its own objective function. As an example, we could imagine a researcher attempting to

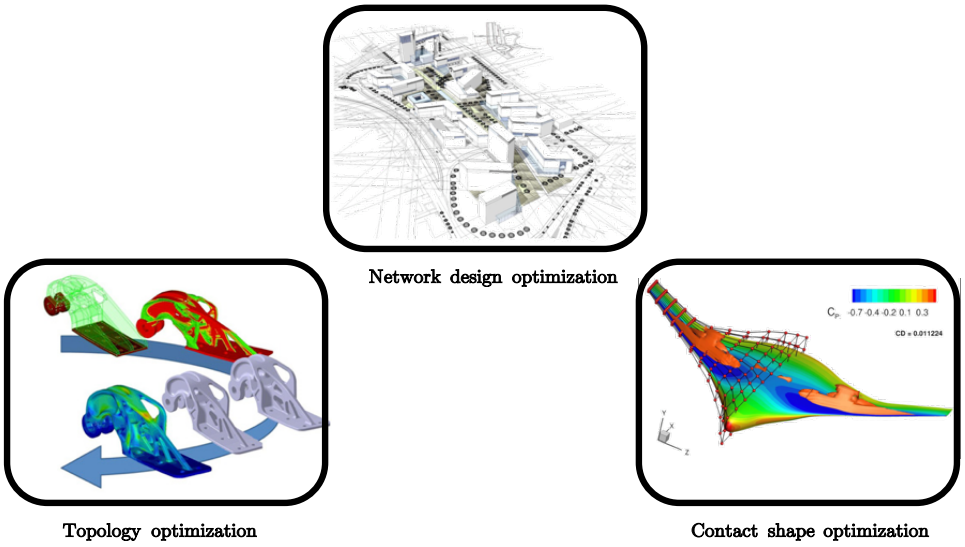


FIGURE 2.5: Structure design optimization problems

discover the best parameters of an evolutionary algorithm. The retro-action which is specific to bi-level optimization is clearly related to the computing time. Indeed, good parameters for the algorithms could lead to high computing time. Therefore, a trade-off has to be defined. In Machine learning, parameter tuning is also refereed to as hyper-parameter optimization when dealing with parameters that cannot be estimated with the same criteria as the one used for the model. Therefore an additional level of optimization has to be added and is generally denoted as "Meta-optimization". Such a situation clearly acknowledges the bi-level nature of parameter tuning. Details on the different applications of bi-level optimization for parameter tuning can be found in [38, 39, 338, 240, 12, 220, 277, 223]

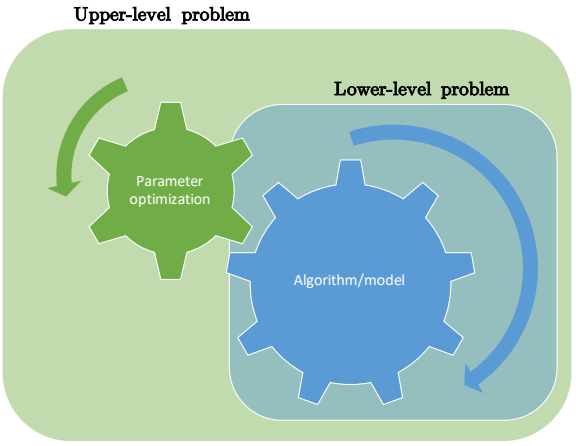


FIGURE 2.6: Parameter tuning as bi-level optimization problem

2.2.6 Chemical reaction problems

Chemical Reaction Optimization (CRO) problems are very similar to the "Parameter tuning" category described in section 2.2.6. They are generally modeled as mathematical problems with thermodynamical equilibrium constraints describing the non-linear chemical reactions . As shown later in section 2, mathematical problems with equilibrium in the constraints (MPEC) can be easily reformulate as bi-level optimization problems. In this very specific case, the upper-level decision maker could be considered as the chemical reaction designer who wishes to determine the best experiments parameters (e.g. amount of reactants, temperature) to ensure a

maximum of effectiveness. The lower-level decision maker is none other than the chemical system considered for the reaction. The upper-level decision maker indirectly controls the system by setting the experimental parameters. However, the thermodynamical equilibrium is solely dependent on the system. At this point, we would like to refer the readers for more details to [81, 306, 158].

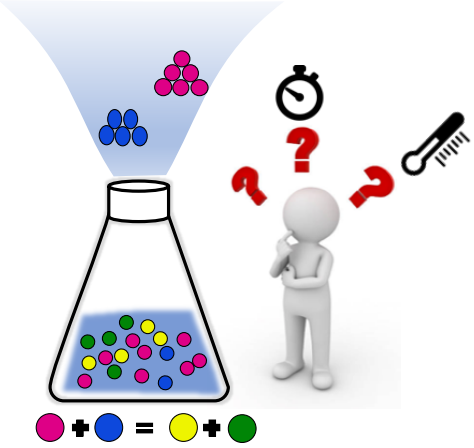


FIGURE 2.7: Determining parameters for chemical processes

2.2.7 Environmental economic problems

In such real-world applications, a regulating authority has to determine an environment policy maximizing its tax revenues while minimizing the environmental damages caused by companies. The regulating authority acts as upper-level decision maker while companies represent lower-level decision makers. A too restrictive policy could lead companies to stop their activities and would be negative for the economy. In contrast, a too permissive policy could cause environmental issues. Indeed, the aim of these companies is to solely maximize their profit. Therefore, the authority has to figure out a compromise leading companies to continue their activities why satisfying a fair policy. These kind of environmental economic problems are really up to date and not trivial to cope with. Once again, bi-level programming allows researchers to model them. Their resolution is a key of major importance especially in the context of globalization. Bi-level models and resolution approaches have been developed for different environmental economic domains such as agriculture [49, 48, 374] , mining [337] and hazardous waste recycling [10].

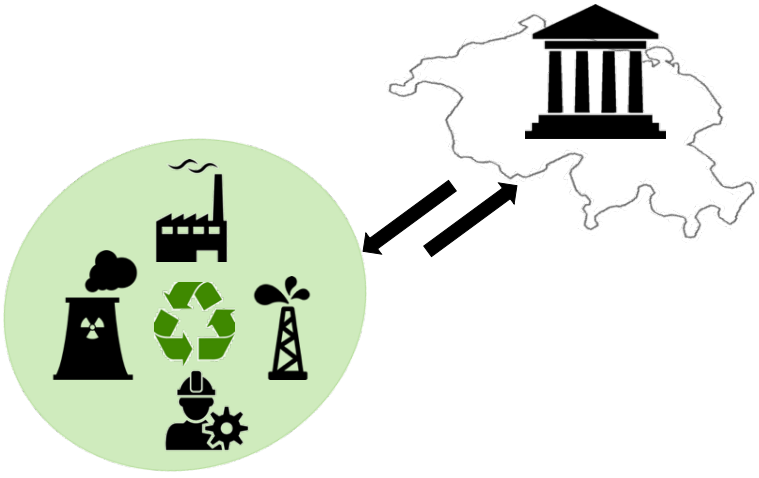


FIGURE 2.8: Environmental economic questions for an authority

2.2.8 Principal–agent problems

The "Principal-agent dilemma" also known as "agency dilemma" reflects the situation in which an institution or simply a person (the "Principal") gives authority to another institution or person (the "agent") to take decisions on its behalf. However the principal cannot fully observe nor directly control the agent. Closely related to political science and economics, these problems represent the potential competition between the ownership entity and the management entity which have generally different objectives. Theories and characterizations of the "Principal-agent" problem can be found in [354, 145]. The principal objective is generally to maximize his utility (e.g. profit) while the agent attempts to minimize the risk or the effort to complete the delegated tasks. Generally, the decisions in such problems should be defined via a contract where the principal and the agent found a common agreement in terms of remuneration and efforts to realize the tasks. consequently, bi-level modeling is particularly well-suited for these problems. The upper-level decision maker plays the role of the "principal" while the lower-level decision maker is the agent. The main goal for the principal is to minimize the remuneration offer. According to the offer, the agent will then decide the effort that he is going to provide. The agent would like to maximize its revenue while minimizing risk and effort. Related works on bi-level modeling and optimization of "Principal-agent" problems can be found in [72, 396, 403].

FIGURE 2.9: The Principal-agent dilemma

TABLE 2.1: Summary table of bi-level applications

References	Applications	section n°
[53, 272, 83, 229, 263, 189] [123, 359, 190, 295, 105, 390, 366, 334]	Pricing problems	2.2.1
[52, 54, 55, 379, 196, 5, 55, 321, 56]	Defense/Interdiction problems	2.2.2
[299, 257, 68, 264, 64, 344]	Logistic problems	2.2.3
[234, 260, 388, 389, 385, 75, 142, 124, 271, 382] [311, 65, 167, 79, 401, 214, 215]	Design problems	2.2.4
[38, 39, 338, 240, 12, 220, 277, 223]	Parameter tuning problems	2.2.5
[81, 306, 158]	Chemical reaction problems	2.2.6
[49, 48, 374, 337, 10]	Environmental economic problems	2.2.7
[354, 145, 72, 396, 403]	Principal–agent problems	2.2.8

2.3 Theoretical aspects and complexity

2.3.1 Mathematical formulation

Bi-level modeling arises when a decision maker would like to solve a problem which depends on another one. Unfortunately, it does not fully control the decision vector of this second problem but can influence it. Therefore, he needs to "forecast" the possible outcomes based on this influence. This second problem can be related to

another decision maker (e.g. [205, 53, 265, 45]) or a physical/chemical process (e.g. [84, 126, 162]) defining some equilibrium states. Program 2.1 illustrates the general mathematical formulation of bi-level optimization problems. This formulation has been first introduced by the seminal work of Bracken and McGill in [51] who designated them in their works as "mathematical programs with optimization problems in the constraints". The obvious nested structure presupposes that a bi-level feasible solution should be a lower-level optimal solution.

$$\begin{aligned} \min_{x \in \mathbf{X}} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \preceq 0 \\ & \min_{y \in \mathbf{Y}} \quad f(x,y) \\ & \text{s.t.} \quad g(x,y) \preceq 0 \end{aligned}$$

PROGRAM 2.1: General Bi-level Optimization Program

where $x \in \mathbf{X} \subset \mathbf{R}^n$, $y \in \mathbf{Y} \subset \mathbf{R}^m$ with F, f : $\mathbf{X} \times \mathbf{Y} \to \mathbf{R}$, $G : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^p$ and $g : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^q$.

Program 2.1 describes a general bi-level problem where x is the decision vector controlled by the upper-level decision maker while y is the decision vector provided by the lower-level decision maker. Each decision maker possesses his own objective function, i.e., $F(x,y)$ for the upper-level decision maker and $f(x,y)$ for the lower-level decision maker. The vector (x,y) represents the bi-level decision vector. Notice that each objective function depends on the two levels of variables, i.e., on x and y. Therefore, the upper-level decision maker must know the decision of the lower-level decision maker before computing $F(x,y)$. Each level can be restricted with constraints: $G(x,y) \preceq 0$ for the upper-level and $g(x,y) \preceq 0$ for the lower-level where the symbol $\preceq$ represents the element-wise lower or equal operator. Such a nested problem can be designated as bi-level only if each level has an impact on each other. Else we could solve both problems separately. The degree and the location of dependences between both levels also permit to categorize bi-level problems into three recurrent classes. The first one is certainly the most general one and has been formally described in Program 2.1. The upper-level decision x is part of the lower-level objective function $f(x,y)$ as well as the lower-level constraint set. The same observation can be made for the lower-level which has an impact on the upper-level constraints and the upper-level objective function. Such general formulation can be found in [200, 333, 332, 269]. Many investigations have been devoted to "Pricing" models where the upper-level decision x only interacts with the lower-level objective value. Those models can embed upper-level constraints which may also depend on the lower-level decision variables (see Program 2.2a). A very rich literature on some bi-level pricing problems can be found in [53, 295, 366, 402, 205]. Finally, Program 2.2b depicts "Interdiction " models. In this case, only the lower-level constraint set is impacted by the upper-level decision variables. The literature contains several works investigating interdiction models [318, 6, 17, 55, 177].

$$\begin{aligned} \min_{x \in \mathbf{X}} \quad & F(x,y) \\ \text{s.t.} \quad & \text{“}G(x,y) \preceq 0'' \\ & \min_{y \in \mathbf{Y}} \quad f(x,y) \\ & \text{s.t.} \quad g(y) \preceq 0 \end{aligned}$$

(A) Bi-level Pricing program

$$\begin{aligned} \min_{x \in \mathbf{X}} \quad & F(x,y) \\ \text{s.t.} \quad & \text{“}G(x,y) \preceq 0'' \\ & \min_{y \in \mathbf{Y}} \quad f(y) \\ & \text{s.t.} \quad g(x,y) \preceq 0 \end{aligned}$$

(B) Bi-level Interdiction program

PROGRAM 2.2: General bi-level classes

TABLE 2.2: Description of the definitions

Definition	Description
(x, y)	Bi-level decision vector
x	Upper-level decision vector
y	Lower-level decision vector
$F(x, y)$	Upper-level objective function
$f(x, y)$	Lower-level objective function
$G(x, y) \leq 0$	Upper-level set of constraints
$g(x, y) \leq 0$	Lower-level set of constraints
$\theta(x) := \min\{f(x, y) \quad s.t. \quad g(x, y) \leq 0\}$	Lower-level optimal value for a given x
$\Psi(x) = \{y \in Y : g(x, y) \leq 0, f(x, y) \leq \theta(x)\}$	Lower-level rational decision set solution set mapping
$\mathcal{IR} = \{(x, y), y \in \Psi(x)\}$	Inducible Region

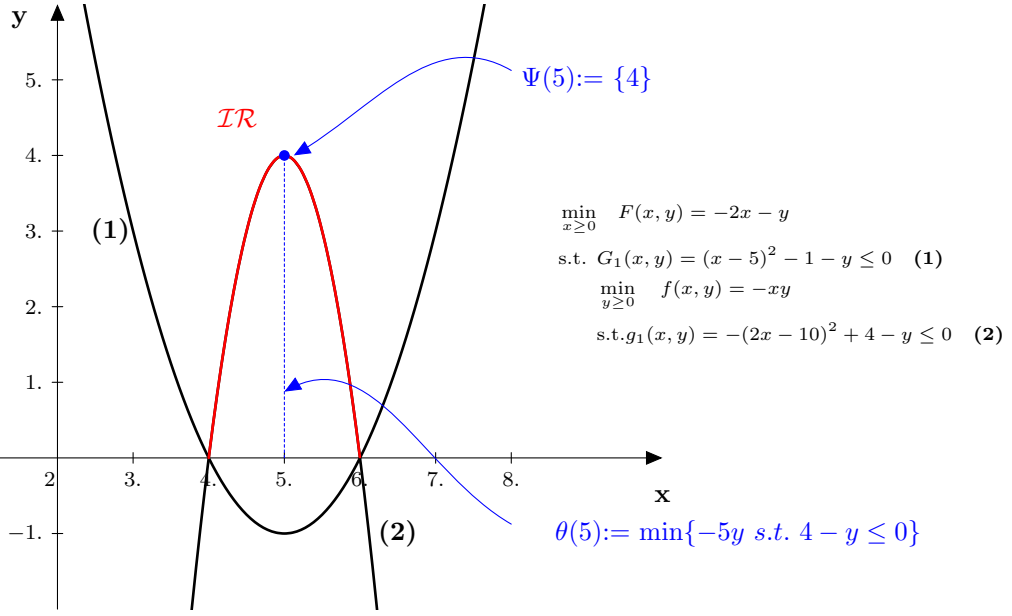
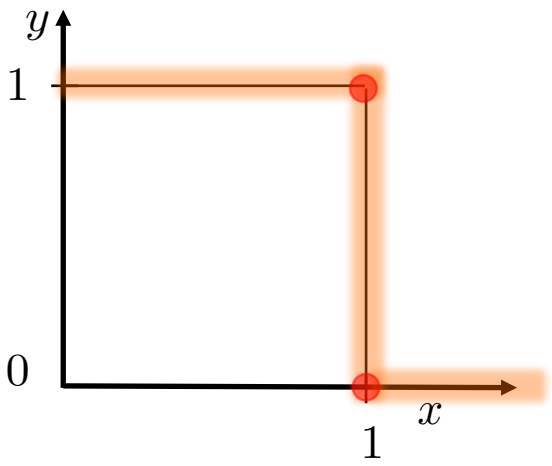


FIGURE 2.10: Example of a general bi-level problem

Table 6.1 describes the main definitions and properties of a bi-level program. The lower-level problem is a parametric optimization problem $\theta(.)$ whose parameters are the upper-level decision variables x. Its is very important for the upper-level decision maker to determine the consequences of his decision. For this purpose, he wishes to forecast the lower-level rational set, i.e., $\Psi(x)$ which is a set representing all the optimal lower-level solutions according to the upper-level decision x. Such a general bi-level program has been pictured in Figure 4.2. The red curve segment represents the inducible region $\mathcal{IR}$ which is essentially the bi-level feasible search space. Only solution $(x, y) \in \mathcal{IR}$ can be assimilated and considered as bi-level feasible solutions. As an example, let us consider an upper-level decision $x = 5$ on the example of Figure 4.2. In this case, the lower-level problem becomes $\theta(5) := \min\{-5y \ s.t. \ 4 - y \leq 0\}$ which sparks the lower-level rational set $\Psi(5) = \{4\}$.

2.3.2 Lower-level rational reaction set $\Psi(x)$

As aforementioned, it may happen that the lower-level rational decision $\Psi(x)$ for a given $x \in \mathbf{X}$ contains more than a single decision $y \in \mathbf{Y}$. These decisions are all lower-level optimal but have different impact on the upper-level objective value. Figure 2.11 depicts such a situation with the corresponding bi-level program on the right hand side of the figure. To an undiscerning eye, the bi-level optimal solution is $(\hat{x}, \hat{y}) = (1, 1)$. However, if we turn our attention to $\Psi(x)$, we can easily observe that for $x = 1$, $\Psi(x) = \{y : 0 \leq y \leq 1\}$. Indeed at $x = 1$, the optimal solution value for the lower-level is 0 for any $0 \leq y \leq 1$. In the case of a fairplay lower-level decision maker returning $y = 1$, the upper-level can reach the optimal solution $(x = 1, y = 1)$. On the contrary, a competitor choosing $y = 0$ definitely prevents the upper-level decision maker to obtain the optimal solution. In fact for $y = 0$, the upper-level decision maker realizes the worst outcome. Consequently, it would be more reasonable for the upper-level decision maker to consider $0 \leq x < 1$. The literature is replete with many examples describing such an issue. The community and more specifically Dempe et al. in [98] illustrated two possible situations to face this problem of non-uniqueness: *the optimistic case* and *the pessimistic case*. In order to study these both cases, the general Program 2.1 needs to be rewritten to Program 2.3.



$$\begin{aligned} \max_{x} \quad & xy \\ \text{s.t.} \quad & 0 \leq x \leq 1 \\ & \max_{y} \quad (1-x)y \\ & \text{s.t.} \quad 0 \leq y \leq 1 \end{aligned}$$

FIGURE 2.11: Example of Bi-level program where $|\Psi(x = 1)| > 1$ (source: CO workshop Aussois 2017)

$$\begin{aligned} \min_{x \in \mathbf{X}} \quad & F(x, \hat{y}) \\ \text{s.t.} \quad & G(x, \hat{y}) \preceq 0 \\ & \hat{y} = \phi(x) \end{aligned}$$

$$\text{where } \phi(x) = \begin{cases} \phi^o(x) & \text{in the optimistic case} \\ \phi^p(x) & \text{in the pessimistic case.} \end{cases}$$

PROGRAM 2.3: Reformulation of the general bi-level optimization program

2.3.2.1 Optimistic rational reaction: $\phi^o(x)$

In *the optimistic approach*, it is assumed that the lower-level rational reaction is the most favorable for the upper-level decision maker. This situation suggests that both decision makers can cooperate together. The optimistic case is generally referred as to *Weak Bi-level Problem* and is illustrated by Figure. 2.12. In order to model it, we have to consider the best scenario which implies that $\phi^o(x) = \arg\min_{y \in \mathbf{Y}} \{F(x, y) : y \in \Psi(x)\}$

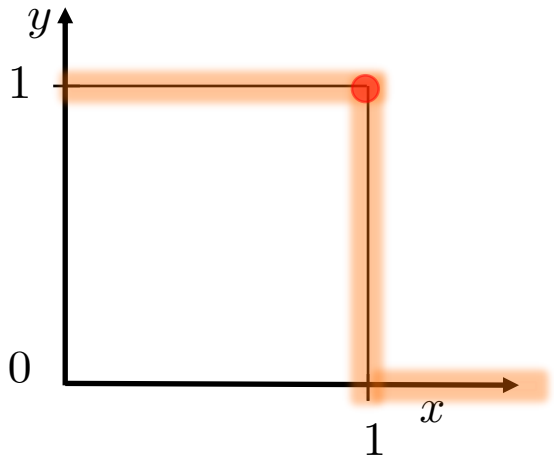


FIGURE 2.12: The optimistic case leads to the optimal bi-level solution $(\hat{x}, \hat{y} = (1, 1))$

Definition 2.1 (Local optimality)**.** A feasible point $(\hat{x}, \hat{y}) \in \mathcal{IR}$, $G(\hat{x}, \hat{y}) \leq 0, \hat{y} = \phi^o(\hat{x})$ is local optimal if there exists $\epsilon > 0$ with $F(x, y) \geq F(\hat{x}, \hat{y}) \; \forall \; (x, y) \in \mathcal{IR}$, $G(x, y) \preceq 0, y = \phi^o(x)$, $\|(\hat{x}, \hat{y}) - (x, y)\| \leq \epsilon$.

It is global optimal if ϵ can be taken arbitrarily large.

Theorem 2.2 (Optimality conditions)**.** *We first assume that F, $f : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}$, $G : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^p$ and $g : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^q$ are sufficiently smooth. If the set $\{(x, y) : g(x, y) \preceq 0\}$ is not empty and compact and the Mangasarian-Fromovitz constraint qualification [258] are satisfied at each solution $(x, y) \in \mathcal{IR}$ with $G(x, y) \preceq 0$ (not empty and compact) such that $y = \phi^o(x)$, then the optimistic bi-level problem has a (global) optimal solution.*

Further investigations on the optimistic optimality conditions can be found in the literature. Interested readers can refer to [162, 292, 242, 98, 96, 100, 217, 29].

2.3.2.2 Pessimistic rational reaction: $\phi^p(x)$

In *the pessimistic case*, the collaboration between upper-level and lower-level is no longer possible. The upper-level decision maker has to consider the worst situation. He needs to bound damages resulting from a *bad* selection. Such a situation is clearly the most realistic one when dealing with strong competition between decision makers. The pessimistic case is generally referred to as *Strong Bi-level Problem* and can be observed on Figure 2.13. The worst situation can be modeled by $\phi^p(x) = \underset{y \in \mathbf{Y}}{\arg\max}\{F(x, y) : y \in \Psi(x)\}$.

Definition 2.3 (Local optimality)**.** A feasible point $(\hat{x}, \hat{y}) \in \mathcal{IR}$, $G(\hat{x}, \hat{y}) \leq 0, \hat{y} = \phi^p(\hat{x})$ is local optimal if there exists $\epsilon > 0$ with $F(x, y) \geq F(\hat{x}, \hat{y}) \; \forall \; (x, y) \in \mathcal{IR}$, $G(x, y) \preceq 0, y = \phi^p(x), \|(\hat{x}, \hat{y}) - (x, y)\| \leq \epsilon$.

It is global optimal if ϵ can be taken arbitrarily large.

Theorem 2.4 (Optimality conditions)**.** *We first assume that F, $f : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}, G : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^p$ and $g : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^q$ are sufficiently smooth. If the set $\{(x, y) : G(x, y) \preceq 0, g(x, y) \preceq 0\}$ is not empty and compact and the point to set mapping Ψ is **lower** semicontinuous , then the pessimistic bi-level problem has an optimal solution.*

Corollary 2.5 (Weak optimality conditions)**.** *We first assume that F, $f : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}, G : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^p$ and $g : \mathbf{X} \times \mathbf{Y} \to \mathbf{R}^q$ are sufficiently smooth. If the set $\{(x, y) : G(x, y) \preceq 0, g(x, y) \preceq 0\}$ is not empty and*

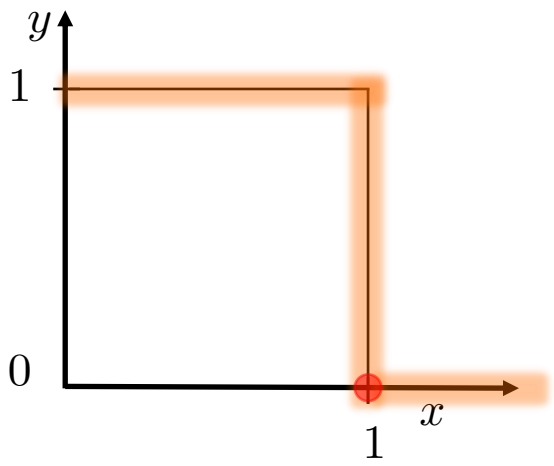


FIGURE 2.13: The pessimistic case leads to the worst possible bi-level solution $(\hat{x}, \hat{y} = (1, 0))$

compact and the point to set mapping Ψ *is* ***upper*** *semicontinuous , then the pessimistic bi-level problem has a weak optimal solution.*

Further investigations on the pessimistic optimality conditions can be found in the literature. Interested readers can refer to [375, 98, 97, 252, 248].

To summarize this section on "Optimality conditions", we distinguish two cases when the lower-level rational set $\Psi(x)$ have multiple optimal solutions for a same upper-level decision x. In the *optimistic* case, the upper-level decision maker is "able" to influence the lower-level decision maker to obtain the most favorable solution. An optimistic bi-level problem is solvable if its bi-level feasible region is nonempty and compact, and if the Mangasarian-Fromowitz constraint qualification holds at all feasible bi-level solutions. Pessimistic bi-level problems are much more tedious and difficult. They can only be solved if the feasible region is nonempty and compact and if the set of lower-level optimal solutions is lower semicontinuous for all upper-level decisions.

Remark

Wiesemann et al. introduced an additional condition on the upper-level dependence. They call ''independent" a bi-level problem where the lower-level rational reaction is independent from the upper-level decision x, though: $\Psi(x) = \Psi(x') \quad \forall\, x, x' \in \mathbf{X}$. Here, we only considered the ''dependent" version where two upper-level decision x and x' may lead to two different lower-level rational set, i.e., $\Psi(x) \neq \Psi(x')$. For more precision on ''independent" bi-level problems and their associated optimality conditions, the reader can refer to [375].

2.3.3 The presence of upper-level constraints

We can distinguish two sort of upper-level constraints:

1. Non-connecting upper-level constraints: $G(x) \preceq 0 \ \forall\ x \in \mathbf{X}$
2. Connecting upper-level constraints: $G(x, y) \preceq 0 \ \forall\ x \in \mathbf{X},\ y \in \Psi(x)$

In the first case, the lower-level decision y is absent and thus has no impact on the feasibility of the upper-level problem. On the contrary, the second case is referred to as "connecting constraints" and has to take into

account y. As described above, y is obtained by solving the lower-level problem for a given x. Nonetheless, y has been computed without taking into account $G(x, y) \preceq 0$. The existence of connecting upper-level constraints have raised many questions in the bi-level community. Some works in the literature [330, 331] considered that such constraints could be simply shifted to the lower-level problem. Mersha et al in [269] showed that shifting constraints violates the essential idea of the modeling concept that is behind bi-level programming. As we explained it in the chapter Introduction , Bi-level programming represents essentially the needs of a decision maker to model its problem while taking into account the impact of a third party, i.e, the lower-level decision maker. The last one is represented by an additional problem in the constraint set of the first one which adds another level of complexity in the strict sens of the term.

In order to stick to reality and therefore to the hierarchical process, the lower-level problem has to be *indifferent* to the upper-level constraints. Indeed, nothing forces the lower-level decision maker to satisfy those constraints since bi-level decision making is not a simultaneous but sequential. The upper-level constraints play no role into the lower-level problem. In fact, it is quite conceivable that the lower-level decision maker is not aware of the existence of the upper-level problem. This is the reason why the upper-level decision maker may not end with a feasible solution if the lower-level rational reaction violates one of the upper-level constraints. This issue can arise for *optimistic* and *pessimistic* cases. Mersha et al in [269] provided a very interesting example of linear bi-level problems with upper-levels constraints as illustrated on Figure 2.14.

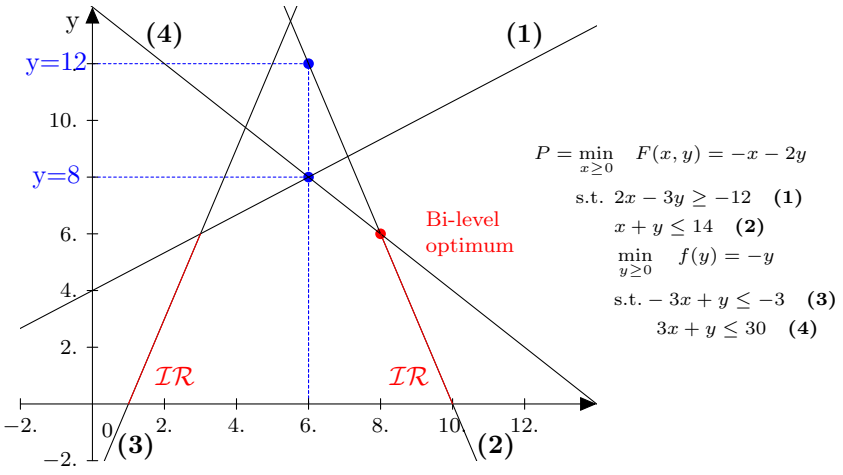


FIGURE 2.14: Example of bi-level problem with upper-level constraints

Figure. 2.14 depicts a scenario where two such connecting constraints restrained the upper-level search space decision. If the upper-level decision maker chooses the decision $x = 6$ and by means of optimization obtains for example $y = 8$, the upper-level decision maker will wrongly suppose that $x = 6$ is the best upper-level decision. Unfortunately such a decision inevitability leads the lower-level decision maker to choose $y = 12$ since it is the optimal rational reaction with respects to $x = 6$. In this case, the upper-level decision maker does not even obtain a feasible solution. As a result, it is highly important to obtain the best forecast of the possible lower-level reactions.

2.3.4 Computational complexity

Bi-level problems are very challenging to solve even if the following parts are convex: F, f, X, Y and $\Psi(x)$. Unfortunately, this situation arises also for two linear levels. Many complexity studies have been undertaken for linear bi-level problems. In [35, 26], the authors conclude that it is $\mathcal{NP}$-hard. Further investigations [102, 161, 181] strengthened the common observations and conclusions that the linear bi-level problem is **strongly**

$\mathcal{NP}$-hard. Using a reduction from 3-SAT, it has also been proven in [261, 361] that local optimality of a given solution is $\mathcal{NP}$-hard as well. Finally, the number of local optimal solutions can be exponential as mentioned in [30]. When levels loose their convexity property, the only task consisting to check bi-level feasibility become strongly $\mathcal{NP}$-hard.

Additionnaly, we can show that linear bi-level problems are equivalent to combinatorial problems. Let us first demonstrate that combinatorial problems,i.e., $\mathcal{NP}$-hard optimization problems have an equivalent bi-level formulation.

2.3.4.1 Bi-level equivalent form of a linear combinatorial problem

Let us consider the linear mixed-integer program illustrated by Program 2.4:

$$\begin{aligned} \min_{x,u} \quad & c^T x + d^T u \\ \text{s.t.} \quad & Ax + Bu \preceq b \\ & x \in \mathbf{R}^n_+,\ u \in \{0,1\}^m \end{aligned}$$

PROGRAM 2.4: A linear mixed-integer problem

where $c \in \mathbf{R}^n$, $d \in \mathbf{R}^m$, $A \in \mathbf{R}^{p \times n}$, $B \in \mathbf{R}^{p \times m}$, $b \in \mathbf{R}^b$.

Note that the binary requirement on the decision variables u can be replaced by the equality constraint: "min"$\{u, \mathbb{1} - u\}$ with $\mathbb{1}$ representing all-ones vector and relaxing the decision variables u such that $u \in [0,1]$ This constraints is neither linear nor convex. We can introduce an additional level which could ensure the binary requirement of u. For this purpose, let us reformulate the non-convex constraint as a mathematical program. The lower-level decision variables y have been introduced to linearize the "min" operator. Finally, we have to link both problems (levels) such that the u or $\mathbb{1} - u$ reach the bounds, i.e 0 or 1.

$$\begin{aligned} \min_{x,y,u} \quad & c^T x + d^T u \\ \text{s.t.} \quad & Ax + Bu \leq b \\ & \mathbb{1}^T y = 0 \\ & \max_{y} \quad \mathbb{1}^T y \\ & \quad \text{s.t.} \quad y \preceq u \\ & \qquad\quad y \preceq \mathbb{1} - u \\ & x, y \in \mathbf{R}^n_+,\ u \in [0,1]^m \end{aligned}$$

PROGRAM 2.5: Resulting linear bi-level problem

Program 2.5 is the resulting bi-level program after linking both levels. As it can be observed, the link is made through the constraint $y = 0$ which ensures indirectly that u will be a binary vector. The lower-level objective should be maximized to guaranty that $\mathbb{1}^T y = 0$ if and only if $u = 0$ or $(\mathbb{1} - u) = 0$.

2.3.4.2 Linear combinatorial equivalent form of a linear bi-level problem

Let us consider the linear bi-level program illustrated by Program 2.6:

where $c_1, c_2 \in \mathbf{R}^n$, $d_1, d_2 \in \mathbf{R}^m$, $b_2 \in \mathbf{R}^q$, $A_1 \in \mathbf{R}^{p \times m}$, $A_2 \in \mathbf{R}^{q \times n}$, $B_2 \in \mathbf{R}^{q \times m}$.

$$
\begin{aligned}
\min_{x} \quad & c_1^T x + d_1^T y \\
\text{s.t.} \quad & A_1 x + B_1 y \preceq b_1 \\
& \min_{y} \quad c_2^T x + d_2^T y \\
& \text{s.t.} \quad A_2 x + B_2 y \preceq b_2 \\
& \qquad x \in \mathbf{R}^n_+, y \in \mathbf{R}^m_+
\end{aligned}
$$

PROGRAM 2.6: A linear bi-level problem

Problems (P) and dual (D) are respectively the primal and dual representations of the lower-level problem (see Program 2.7).

$$
\begin{aligned}
(\mathsf{P}) = \min_{y \geq 0} \quad & c_2^T x + d_2^T y \\
\text{s.t.} \quad & B_2 y \preceq b_2 - A_2 x
\end{aligned}
$$

(A) Primal lower-level problem

$$
\begin{aligned}
(\mathsf{D}) = \max_{u \geq 0} \quad & u^T (A_2 x - b_2) \\
\text{s.t.} \quad & u^T B_2 \succeq -d_2
\end{aligned}
$$

(B) Dual lower-level problem

PROGRAM 2.7: Primal and dual representations of the lower-level problem

The complementary slackness conditions implies that $(b_2 - B_2 y - A_2 x)^T u = (-d_2 - u B_2)^T y$ and can be used to replace the lower-level problem with a set of constraints. Notice that the resulting program has only a single level. In addition, the complementary slackness conditions link the primal and dual lower-level variables.

$$
\begin{aligned}
\min_{x} \quad & c_1^T x + d_1^T y \\
\text{s.t.} \quad & A_1 x + B_1 y \preceq b_1 \\
& A_2 x + B_2 y \preceq b_2 \\
& (b_2 - B_2 y - A_2 x)^T u + (d_2 + u B_2)^T y = 0 \\
& - u B_2 - d_2 \preceq 0 \\
& y \in \mathbf{R}^m_+, u \in \mathbf{R}^m_+, x \in \mathbf{R}^n_+
\end{aligned}
$$

PROGRAM 2.8: Single-level reformulation

Despite the fact that there is only a single level, the introduction of the complementary slackness constraints does not facilitate its resolution. To bypass this problem, notice that $b_2 - B_2 y - A_2 x \geq 0$ and $d_2 + u B_2 \geq 0$. Therefore the complementary slackness constraints are true if and only if $(b_2 - B_2 y - A_2 x) u = 0$ and $(d_2 + u B_2) y = 0$. Both equalities can be linearized by introducing additional binary decision variables. As a result, a linear mixed-integer program is obtained (see Program 2.9).

Finally, we have shown the equivalence between linear combinatorial (mixed-integer) and linear bi-level problems. Many equivalences between single-level programs and bi-level programs can be demonstrated. Some resolution approaches are directly based on them to reduce complex bi-level programs to single-level programs enabling the usage of classical resolution approaches. The next section describes the relationship existing in the literature with different classes of problems.

$$
\begin{aligned}
\min_{x} \quad & c_1^T x + d_1^T y \\
\text{s.t.} \quad & A_1 x + B_1 y \preceq b_1 \\
& A_2 x + B_2 y \preceq b_2 \\
& (b_2 - B_2 y - A_2 x) \preceq Mz \\
& u \preceq M(1 - z) \\
& d_2 + uB_2 \preceq Mt \\
& y \preceq M(1 - t) \\
& -uB_2 - d_2 \preceq 0 \\
& y \in \mathbf{R}_+^m, u \in \mathbf{R}_+^m, x \in \mathbf{R}_+^n, z \in \{0,1\}^q, t \in \{0,1\}^m
\end{aligned}
$$

PROGRAM 2.9: The resulting combinatorial problem

2.3.5 Relationship with others domains

Bi-level optimization does not only concern a specific branch of Optimization. Many relationships can be established with other domains. Some of these problems are just a matter of reformulation or specific bi-level versions. The next sections will describe some of them and their correlation to bi-level programming. This is not an attempt to list exhaustively all related problems but it should explain the reason why bi-level optimization has been approached from different perspectives. Hereafter, we discuss various domains that share many common concepts with bi-level programming and allow us to switch from one to the other just by matter of reformulation. Finally, we summarize our discussion with Table 2.3.

2.3.5.1 Stackelberg games

Stackelberg duopoly models, also referred as Stackelberg games [346], are non-cooperative games and are essentially sequential (not simultaneous as in Cournot duopoly) non-zero sum games involving two players. The first player called "leader" is aware of the problem of the second player called "follower". Taking the point of view of the leader, his problem depends on the reaction of the follower. In order to take the right decision, it has to compute the best response of the follower called " the follower rational decision". Bi-level Optimization are based on the similar concept except that the follower is an equilibrium problem. To some extend, bi-level optimization generalizes Stackelberg games. For example, two companies F_1 and F_2 competing on the marked would like to determine their production levels maximizing their respective profits. One of this company, say F_1, set its production level first. Since its production level will depend on the market reaction, i.e., F_2, it wishes to solve the model illustrated in Program 2.10 where $P(q_l, q_f)$ is the unit price of the production. C_l is the unit cost for the leader production. C_f is the unit cost for the follower production. Many investigations have been done on Stackelberg games. Detailed information can be obtained in [134, 241, 248, 252, 283].

$$
\begin{aligned}
\max_{q_l, q_f} \quad & P(q_l, q_f) q_l - C_l q_l \\
\text{s.t.} \quad & q_f \in \arg\max_{q_f} P(q_l, q_f) q_f - C_f q_f \\
& q_l, q_f \geq 0
\end{aligned}
$$

PROGRAM 2.10: Stackelberg competition between two companies on the market

2.3.5.2 Maxmin Problems

Maxmin problems [112, 313] are special kind of bi-level problems where $F(x, y) = -f(x, y)$. In fact, zero-sum games can be modeled as maxmin problems. They naturally emerge from optimization under uncertainty. In such case, the player attempting to solve the maxmin problem wishes to react to the worst scenario and not necessarily to the most probable one. Maxmin optimization is directly related to some kind of robust optimization. Audet in [18] established the link between linear maxmin problems and linear bi-level problems. Let us consider the linear maxmin problem formulated for the first time by Falk [120, 121], $\mathcal{P} = \max_{x \in \mathbf{X}} \min_{y \in \mathbf{Y}} \{cx + dy : \mathbf{A}x + \mathbf{B}y \preceq \mathbf{b}\}$. This model can be easily reformulate as follows $\mathcal{P} = \max_{x \in \mathbf{X}} \{cx + \min_{y \in \mathbf{Y}} \{dy : \mathbf{B}y \preceq \mathbf{b} - \mathbf{A}x\}\}$. Finally, we obtain the two-stage program depicted by Program 2.11 by introducing $\hat{y}$ at the upper-level objective function. When the coefficient vector d associated to the second stage decision variables are different for the two stages, Program 2.11 can be assimilated as a bi-level program. Additional information on max-min problems can be found in [352, 353, 362].

$$
\begin{aligned}
\min_{x} \quad & c^T x + d^T \hat{y} \\
\text{s.t.} \quad & x \in \mathbf{X} \\
& \hat{y} \in \arg\min_{y} d^T y \\
& \text{s.t. } \mathbf{B}y \preceq \mathbf{b} - \mathbf{A}x \\
& \quad y \in \mathbf{Y}
\end{aligned}
$$

PROGRAM 2.11: A bi-level reformulation of the linear maxmin problem

2.3.5.3 Generalized semi-infinite problems

When an optimization problem contains a formulation involving either an infinite number of constraints or an infinite number of variables, it is then classified as a semi-infinite problem. Bi-level and semi-infinite problems are closely related to each other. Let us consider the two following cases:

- The optimistic case ;
- The lower-level rational set $\Psi(x)$ is a singleton.

In both situations, we can reformulate Program 2.1 as a semi-infinite program by returning the lower-level optimal value $\theta(x) = \min\{f(x, y) \quad s.t. \quad g(x, y) \preceq 0\}$ instead of the optimal lower-level decision variables. However, this value function reformulation only works for the two aforementioned situations. The added value of such a transformation is the decomposition of both levels while keeping the dependence through a set of infinite constraints as depicted by Program 2.12. Semi-infinite programming has been extensively investigated in the literature [41, 40, 130, 275, 358, 353].

2.3.5.4 Set-valued optimization problems

According to [195],"*Set-valued optimization is a vibrant and expanding branch of mathematics that deals with optimization problems where the objective map and/or the constraints maps are set-valued maps acting between certain spaces*". Bi-level optimization problems are strongly related to set-valued optimization problems. In

$$\begin{aligned} \min_{x\in\mathbf{X},y\in\mathbf{Y}} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \preceq 0 \\ & g(x,y) \preceq 0 \\ & f(x,y) - \theta(x) \preceq 0 \end{aligned} \qquad \Leftrightarrow \qquad \begin{aligned} \min_{x\in\mathbf{X},y\in\mathbf{Y}} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \preceq 0 \\ & g(x,y) \preceq 0 \\ & f(x,y) - f(x,y*) \preceq 0 \quad \forall\, y* \in \{y* : g(x,y*) \leq 0\} \end{aligned}$$

(A) Reformulation of the general bi-level program (B) Equivalence with semi-infinite programming

PROGRAM 2.12: Reformulation of the general bi-level program to a semi-infinite program

fact, Program 2.1 can be rewritten as Program 2.13 using the set-value mapping Ψ: $\mathbf{R}^n \to \mathbf{R}^m$. This new formulation is ambiguous as discussed in section 2.3.2. When $\Psi(x)$ is not a singleton for every $x \in X$, a choice between the *optimistic* or teh *pessimistic* situation has to be made. Details on set-valued optimization can be found in [178, 195, 76, 219, 95].

$$\begin{aligned} \min_{x\in\mathbf{X},\hat{y}\in\Psi(x)} \quad & F(x,\hat{y}) \\ \text{s.t.} \quad & G(x,\hat{y}) \preceq 0 \end{aligned}$$

PROGRAM 2.13: Reformulation of the general bi-level problem to a set-valued problem

2.3.5.5 Mathematical problems with equilibrium constraints

An optimization problem with equilibrium constraints (see Program 2.14) can be written as a mathematical program with some constraints defined as variational equalities that should satisfy equilibrium conditions.

$$\begin{aligned} \min_{x\in\mathbf{X},y\in\mathbf{Y}} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \preceq 0 \\ & \nabla_y f(x,y) = 0 \end{aligned}$$

PROGRAM 2.14: A general mathematical program with equilibrium constraints

where F, f : $\mathbf{X}\times\mathbf{Y} \to \mathbf{R}, G : \mathbf{X}\times\mathbf{Y} \to \mathbf{R}^p$ and $\nabla_y f$ denotes the partial derivatives of f with respect to y. In the case of $f(x,y)$ is convex, Program 2.14 can be expressed equivalently as a bi-level problem. Assuming that $\Psi(x)$ is a singleton for every x, $\nabla_y f(x,y) = 0$ can be replaced by $y = \arg\min_{y\in\mathbf{Y}}\{f(x,y) \quad s.t. \quad g(x,y) \preceq 0\}$. Generally, bi-level programs are transformed into mathematical programs with equilibrium constraints when convexity can be assumed. Such a reformulation is more convenient and has been extensively exploited in many works [31, 253, 115, 161]. Dempe in [99] established a complete annotated bibliography on bi-level programming and mathematical programs with equilibrium constraints. Deep investigations on the related works and optimality conditions can be found in his bibliography.

2.3.5.6 Multi-stage problems with recourse

Multi-level programming (e.g. bi-level programming) should not be confused with multi-stage problems. While a multi-level problem involves several decision makers, a multi-stage problem possesses several levels but only a

single decision maker. Multi-stage problems with recourse depict situations in which a risk function models some uncertainties. These uncertainties are due to some external factors that the decision maker cannot controlled. However, its decision has an indirect influences on those factors. Ralph [307] has shown the link between the linear bi-level interdiction problem and the linear two-stage problem. In section 2.3.1, we introduced interdiction models through Program 2.2b. These models generally describe a situation in which the upper-level decision maker only influences the lower-level feasible search space. The linear interdiction problem can be modeled as follows: $\mathcal{I} = \min_{x\in\mathbf{X}}\{c^T x + d1^T y : A^1 x \preceq b^1, y \in \min_{y\in\mathbf{Y}}\{d2^T y : B^2 y \preceq (b^2 - A^2 x)\}\}$. We assume, hereafter, that we only have a single lower-level reaction for every upper-level decision x. When $d^1 = d^2$, the bi-level interdiction problem becomes a two-stage problem with recourse. Indeed, we can easily see:

$$
\begin{aligned}
\mathcal{I} &= \min_{x\in\mathbf{X}}\{c^T x + d_1^T y : A^1 x \preceq b^1, y \in \min_{y\in\mathbf{Y}}\{d_2^T y : B^2 y \preceq (b^2 - A^2 x)\}\} \quad \text{with} \quad d_1 = d_2 \\
&= \min_{x\in\mathbf{X}}\{c^T x + d_1^T y : A^1 x \preceq b^1, y \in \min_{y\in\mathbf{Y}}\{d_1^T y : B^2 y \preceq (b^2 - A^2 x)\}\} \\
&= \min_{x\in\mathbf{X}}\{c^T x + \min_{y\in\mathbf{Y}}\{d_1^T y : B^2 y \preceq (b^2 - A^2 x)\} : A^1 x \preceq b^1\} \\
&= \min_{x\in\mathbf{X}}\{c^T x + Q(x) : A^1 x \preceq b^1\} \quad \text{with} \quad Q(x) = \min_{y\in\mathbf{Y}}\{d_1^T y : B^2 y \preceq (b^2 - A^2 x)\}
\end{aligned}
$$

$Q(x)$ is the risk function, also called "The value function of the recourse problem". In two stage stochastic problem, the uncertainties is modeled by random variables. The recourse is a statistical indicator, i.e, mean on the value function obtained at the second stage: $\min_{x\in\mathbf{X}, y\in\mathbf{Y}}\{c^T x + \mathbb{E}_\xi(Q_\xi(x)) : A^1 x \preceq b^1\}$ with $Q_\xi(x) = \min_{y\in\mathbf{Y}}\{d_1^T y : W_\xi y \leq (h_\xi - T_\xi x)\}$ where ξ describes the realization of uncertain data. The concept is very close to the strategy adopted by the upper-level decision maker in a bi-level program. The decision maker in a two-stage stochastic problem makes a "here and now " decision x to observe the average realization outcome at the second stage. Then according to the gained information, he solves an appropriate optimization problem. The concept of multi-stage programming relies strongly on decomposition techniques and especially on Bender decomposition approach [37].

TABLE 2.3: Some problems with relationship to bi-level programming

References	**Relationship with**	**section n°**
[346, 134, 241, 248, 252, 283]	Stackelberg games	2.3.5.1
[112, 313, 120, 121, 352, 353, 362]	Maxmin problems	2.3.5.2
[41, 40, 130, 275, 358, 353]	Generalized semi-infinite problems	2.3.5.3
[195, 178, 195, 76, 219, 95]	Set-valued problems	2.3.5.4
[31, 253, 115, 161, 99]	Mathematical problems with equilibrium constraints	2.3.5.5
[307]	Multi-stage problems with recourse	2.3.5.6

2.4 Classical resolution approaches

Many algorithms have been proposed since the mid-1970. Most of them focus on specific classes of bi-level problems. For example, the literature is replete with approaches dedicated to solve linear and convex levels. Very often strong properties have been assumed like continuous differentiability, convexity and lower semi-continuity. The introduction of a second-level increases significantly the number of existing bi-level categories. Some of them have been extensively studied while others have been seldom tackled due to their difficult nature.

Therefore, we adopt a taxonomy which focuses more on the different class of algorithms designed to cope with bi-level problem. According to the existing literature, we divided this section between two categories: convex and non convex bi-level problems. All approaches and related works found in the literature are summarized in Table 2.4.

2.4.1 Approaches for bi-level problems with convex levels

The literature often describes bi-level problems as "convex" when each level is a convex sub-problem. Nevertheless, it is incorrect to conclude that a hierarchy of convex levels lead to a convex global problem. As shown in section 2.3.4, linear bi-level problems are equivalent to combinatorial problems. All the classical approaches described in the further sections struggled to cope with problems that would be "easy" to solve in single-level optimization.

2.4.1.1 Extreme point search

The main property motivating this section is described by Theorem 2.7 and detailed in [26].

Theorem 2.6. *An optimal solution* $(\hat{x}, \hat{y})$ *of a linear bi-level problem occurs at an extreme point of* $\mathbf{S} = \{(x, y) : G(x, y) \preceq 0, g(x, y) \preceq 0\}$.

Candler and Townsley [67] developed an approach for linear bi-level problems with no upper-level constraints and assumed that for every x, $\Psi(x)$ is a singleton, i.e., $|\Psi(x)| = 1$. The algorithm enumerates extreme points of the lower-level problem using necessary conditions. Unfortunately, this algorithm has to explore a large number of extreme points which tend to make it very slow.

The $\mathcal{K}$th-Best Algorithm is an extreme point search procedure which has been designed by Bialas and Karwan in [43] for linear bi-level problems. It enumerates extreme points of $\mathbf{S}$ until the current one belongs to $\mathcal{IR}$. The enumeration is ordered according to the upper-level objective values. Obviously $\mathcal{IR}$ has to be bounded and $\Psi(x)$ must be single-valued.

In [94], Dempe proposed a modified simplex procedure for linear bi-level problem with no upper-level constraints. This procedure relies on specialized rules for updating the basis. Dempe did not provide numerical results but illustrated his approach on an example.

In [296], Papavassilopoulos adopted a similar enumeration procedure except that all enumerated extreme points belonged to the inducible region, i.e, $\mathcal{IR}$.

Although the approach proposed by Jaumard and Savard in [161] does not explicitly rely on vertex enumeration, they proposed to determine which lower-level constraints are binding at optimality. Indeed, a solution is bi-level feasible if it is not possible to improve the lower-level objective value without violating one of the lower-level constraints. For this purpose, they proposed a Branch & Bound procedure based on active set enumeration. Artificial variable α_i are assigned to each constraint $g_i(x, y)\ \forall\ i \in \{1, ..., q\}$. The branching procedure is done by setting α_i to 1 or 0. Jaumard and Savard extended the algorithm to the non-linear bi-level problem where $F(x, y)$ and $G(x, y)$ are convex, $f(x, y)$ is quadratic and $g(x, y)$ is affine.

2.4.1.2 Reformulation to single-level problem

As discussed in sections 2.3.5 and 2.3.4, reformulation techniques can be retained to create equivalent single-level problems. Many works rely on this methodology even if strong assumptions have sometimes to be taken. Hereafter, we suppose that the lower-level problem has convex and regular properties. In such a situation, it can be replaced by the corresponding Karush-Kuhn-Tucker (KKT) conditions described in Program 2.15b. The KKT conditions consist of four mains conditions:

1. Stationarity: $\nabla_y \mathcal{L}(x, y, \lambda) = 0$ with the Lagrangean function $\mathcal{L}(x, y, \lambda) = f(x, y) + \sum_{i=1}^{q} \lambda_i g_i(x, y)$
2. Complementary slackness: $\lambda_i g_i(x, y) = 0 \quad \forall\, i \in \{1, ..., q\}$
3. Primal feasibility: $(x, y) \in \{G(x, y) \preceq 0, g(x, y) \preceq 0\}$
4. Dual feasibility: $\lambda_i \geq 0 \, \forall\, i \in \{1, ..., q\}$

The reformulation does not change the complexity of the resulting problem. Note that despite the convexity assumptions made for both levels, the complementary slackness conditions are combinatorial by nature. This result should not be surprising since in section 2.3.4, we have shown the equivalence between combinatorial and bi-level problems.

All the contributions listed in this section are based on this reformulation and attempt to efficiently solve the resulting combinatorial problem.

Fortuny-Amat and McCarl [132] designed a procedure to tackle the linear bi-level problem. They assigned binary variables for each complementary slackness conditions. Each binary variable indicates whether the multiplier $\lambda_i = 0$ or the corresponding lower-level constraint g_i is binding. A B&B algorithm is then used to solve the problem.

Bard and Falk [31] chose an alternative approach to handle the complementary slackness conditions. They replaced each equality: $\lambda_i g_i(x, y) = 0 \quad \forall\, i \in \{1, ..., q\}$ by the two following constraints: $\sum_{i=1}^{q} \min\{0, w_i\} + \lambda_i = 0$ and $w_i - g_i + \lambda_i = 0 \, \forall\, i \in \{1, ..., q\}$. Although the new formulation remains non-convex, it has the particularity to be separable. Bard in [30] explained that the complementary term has been replaced by a piecewise and separable term.

$$
\begin{aligned}
\min_{x \in \mathbf{X}} \quad & F(x, y) \\
\text{s.t.} \quad & G(x, y) \preceq 0 \\
& \min_{y \in \mathbf{Y}} \quad f(x, y) \\
& \text{s.t.} \quad g(x, y) \preceq 0
\end{aligned}
$$

(A) Bi-level program with convex lower-level

$$
\begin{aligned}
\min_{x \in \mathbf{X}, y \in \mathbf{Y}, \lambda} \quad & F(x, y) \\
\text{s.t.} \quad & G(x, y) \preceq 0 \\
& g(x, y) \preceq 0 \\
& \lambda_i g_i(x, y) = 0 \quad \forall\, i \in \{1, ..., q\} \\
& \nabla_y \mathcal{L}(x, y, \lambda) = 0 \\
& \lambda_i \geq 0 \qquad \forall\, i \in \{1, ..., q\}
\end{aligned}
$$

(B) Single-level program

PROGRAM 2.15: Single-level reformulation relying on the KKT conditions

In order to remove the complementarity condition terms, Bard and Moore [33] developed an alternative strategy consisting in giving all control of the lower-level variables to the upper-level problem. At each step, their procedure checks if the complementary terms are satisfied. If this is the case, the current solution is a valid bi-level feasible solution and a potential bi-level optimal candidate.

Bialas and Karwan in [44] proposed the "Parametric Complementary Pivot " algorithm. The latter is closely related to the KKT reformulation and sequentially computes a feasible point (x, y) with the upper-level objective value $F(x, y)$ bounded by a parameter α. This parameter is updated after each step and the procedure stops when no new feasible solution is found. However as explained in [30], their algorithm must be considered as an *heuristic* as shown by [36]. Following the proposed approach, Judice and Faustino in [187] modified the algorithm to provide guaranty of convergence. It is important to note that the complementary approach has not been designed for bi-level problems having upper-level constraints.

Further investigations and algorithms based on the ones presented in this section can be found in [7, 161, 330]

2.4.1.3 Descent approaches

Descent methodologies in bi-level optimization aim at decreasing the upper-level objective value while keeping solutions bi-level feasible, i.e, lower-level optimal. A feasible descent direction $d \in \mathbf{R}^n$ should allow the upper-level objective value to decrease by computing a new feasible candidate as follows $x + \alpha d$ with $\alpha > 0$. Assuming that $\Psi(x)$ is a singleton for every upper-level decision x, we can define y as function of x, i.e, y: $\mathbf{R}^n \to \mathbf{R}^m$. Therefore the gradient $F(x, y(x))$ is defined as $\nabla_x F(x, y) = \nabla_x F(x, y) + \nabla_y F(x, y)\nabla_x y(x)$. Nevertheless, we cannot guaranty its availability for every feasible solution (x, y).

In order to tackle this issue, Kolstad and Lasdon [218] attempted to approximate this gradient. Vicente et al. in [361] proposed a descent algorithm for convex quadratic bilevel programs with linear constraints. The algorithm moves along the inducible region by using complementary pivoting approach. Bard in [30] reported that this approach cannot guaranty local optimality unless the upper-level objective function is concave. In [320], Savard and Gauvin designed a descent algorithm which employ an auxiliary program to compute descent direction. They basically solve an other linear-quadratic bi-level program to compute d for which exact algorithm exists. Finally, a bundle approach has been developed by Falk and Liu [122] where the decrease of the upper-level objective function is controlled according to subdifferential data gained from the lower-level.

2.4.1.4 Penalty approaches

The main concept behind this important class of bi-level resolution approaches relies on penalizations of solutions which do not satisfy lower-level optimality. Although they have been extensively used in "non-linear programming", it is still a real challenge to measure the extend of violation of the lower-level problem. In [2, 3], Shimizu and Aiyoshi replaced the lower-level problem by an unconstrained problem with penalization as follows $\min_y \{f(x, y) + C\sigma(g(x, y))\}$ where C is a positive real value and $\sigma(.)$ a continuous penalty function. This function has the following properties with $S(x) = \{y \in Y : g(x, y) \preceq 0\}$

- $\sigma(g(x, y)) > 0$ if $y \in relint\, S(x)$
- $\sigma(g(x, y)) \to +\infty$ if $y \to bd\, S(x)$

$relint\, S(x)$ and $bd\, S(x)$ denotes respectively the relative interior and the boundary of $S(x)$. As aforementioned, the new formulation is not easier to cope with.

In [174], Ishizuka and Aiyoshi adopted a new approach penalizing both objective functions $F(x, y)$ and $f(x, y)$. The lower-level problem is this time replaced by its stationary conditions to obtain a single-level problem. We can consider this strategy as an hybrid between single-reformulation and penalty approaches.

In the same vein, Case in [71] replaced the lower-level problem by its KKT conditions and penalized the upper-level objective $F(x, y)$. Contrary to previous works, he designed a penalty function relying on the ℓ_1 norm.

Anandalingam and White proposed in [11, 373] a penalty approach for linear bi-level problems. Their algorithm is based on the fact that a bi-level feasible solution should have a lower-level duality gap equals to 0. Therefore, the problem can be reformulated as a single-level problem with primal and dual lower-level constraints. The upper-level objective function is penalized by a function evaluating the lower-level duality gap, i.e., the difference between the dual and the primal objective values of the lower-level problem.

2.4.1.5 Trust-region approaches

Trust-region algorithms are conceived to approximate regions of the objective function with respect to a given model. When the approximation is accurate enough, these algorithms expand the promising regions otherwise regions are contracted. These trust-regions are generally modeled as the vicinity around a given point p^k such as a ball with radius r^k. The solution s^k at iteration k is obtained according the optimization of a model m^k over the trust-region as depicted in Program 2.16 .

$$\begin{aligned} \min_{s} \quad & m_k(p_k + s) \\ \text{s.t.} \quad & \|s\| \leq r^k \end{aligned}$$

PROGRAM 2.16: Optimization of a trust-region subproblem

Trust-region algorithms have been developed by [245] for non-linear bi-level problems with convex lower-level objective function and linear lower-level constraints. In [259], another trust-region approach has been proposed by Marcotte et al. in combination with line search techniques. They considered bi-level problems with linear upper-level and linear variational equality at lower-level. Colson et al. in [82] coped with non-linear bi-level programs. Their iterative approach is based on a linear-quadratic trust-region subproblem that can be approached globally with dedicated algorithms such as the one designed by Jaumard et al in [261].

2.4.2 Approaches for bi-level problems with non-convex levels

Given the difficulty of addressing bi-level problems with convex levels, it is not a surprise to see very few approaches to tackle bi-level problems with non-convex levels. Most of the contributions detailed in this section is dedicated to (mixed-) Integer linear bi-level problems. To the best of our knowledge, non-linear and non-convex problems have been approached by:

- Mitsos et al in [274] with a bounding algorithm that terminates with a solution satisfying ϵ-optimality at both levels.
- Tsoukalas et al. in [353] with an algorithm relying on a "oracle" deciding whether or not a target objective value is reachable. If the oracle concludes that the value can be attained, then a feasible solution reaching this value is returned. A binary search is performed for all target values to detect the global optimal solution.
- Kleniati and Adjiman in [212, 210] developed a new approach that explores both level spaces using a single B&B tree. Lower and upper bounds are computed for both levels. In [211], Kleniati and Adjiman extended their approach for mixed-integer non-linear bi-level problems.

2.4.2.1 Mixed-integer Bi-level problems: properties

As in single-level optimization, most bi-level optimization problems are mixed-integer problems. The location of integer variables may have serious consequences on the properties of the inducible region, i.e., $\mathcal{IR}$. Figure 2.15 describes 4 versions of the same linear bi-level problem with no upper-level constraints. Discrete variables can occur at each level.

Figure 2.15a represents the bi-level problem with only continuous variables, i.e., $x \in \mathbf{R}$ and $y \in \mathbf{R}$. Even if it is not a mixed-integer problem, we added it in Figure 2.15 for the sake of exhaustivity. Such bi-level problems have been discussed in the previous section. The inducible region is a piecewise linear function. The optimal solution occurs at solution $(\hat{x}, \hat{y}) = (8, 1)$. In the remainder, such bi-problems will be referred to as **CCLBP**.

Figure 2.15b depicts a bi-level problem where $x \in \mathbf{Z}$ and $y \in \mathbf{R}$. Such a configuration will be denoted as **CDLBP**. The inducible region is basically a finite set of points (red on Figure 2.15b). Notice that it is very specific example, the optimal bi-level solution is the same as for the **CCLBP**, i.e. $(\hat{x}, \hat{y}) = (8, 1)$.

Figure 2.15c is a pure integer bi-level problem, i.e. $x \in \mathbf{Z}$ and $y \in \mathbf{Z}$ This category will be designated as **DDLBP**. The inducible region is a finite set of points. The optimal solution occurs at $(\hat{x}, \hat{y}) = (2, 2)$. As observed in this example, the optimal solution may be an interior point of the set $\{(x, y) \in \mathbf{Z} \times \mathbf{Z} : g(x, y) \leq 0\}$.

The last example is illustrated by Figure 2.15d with $x \in \mathbf{R}$ and $y \in \mathbf{Z}$. **CDLBP** has an inducible region made by a finite union of quasi-polyhedral sets. If $\mathcal{IR} \neq \emptyset$, it does not mean necessarily that $\mathcal{IR}$ is compact. In this example, there is no optimal solution but only a bound occurring at $(2.5, 2.0)$. Indeed the upper-level decision maker would realize the best results for $x = 2.5$ and $y = 2$. However when x reached $x = 2.5$, the lower-level reaction y drops to $y = 1$. The upper-level decision maker should consider ϵ-optimality to get as close as possible to this bound while avoiding to reach it.

Some interesting properties concerning the different classes of mixed-integer bi-level problems have been listed by J. Bard in his book [30]. We have decided to present two of them which links the different classes between each others.

Property 2.1. *The inducible regions of* **DCLBP** *and* **DDLBP** *are respectively included in the inducible regions of* **CCLBP** *and* **CDLBP**.

The proof of Property 2.1 is straightforward and can be easily observed through the different example in Figure 2.15.

Note **CCLBP** is the relaxed version of **DDLBP**. The optimal solution value for **CCLBP** is -18 while the one for **DDLBP** is -22. Therefore an additional property can be stated.

Property 2.2. *The continuous relaxation an integer bi-level problem does not necessarily provide a valid bound.*

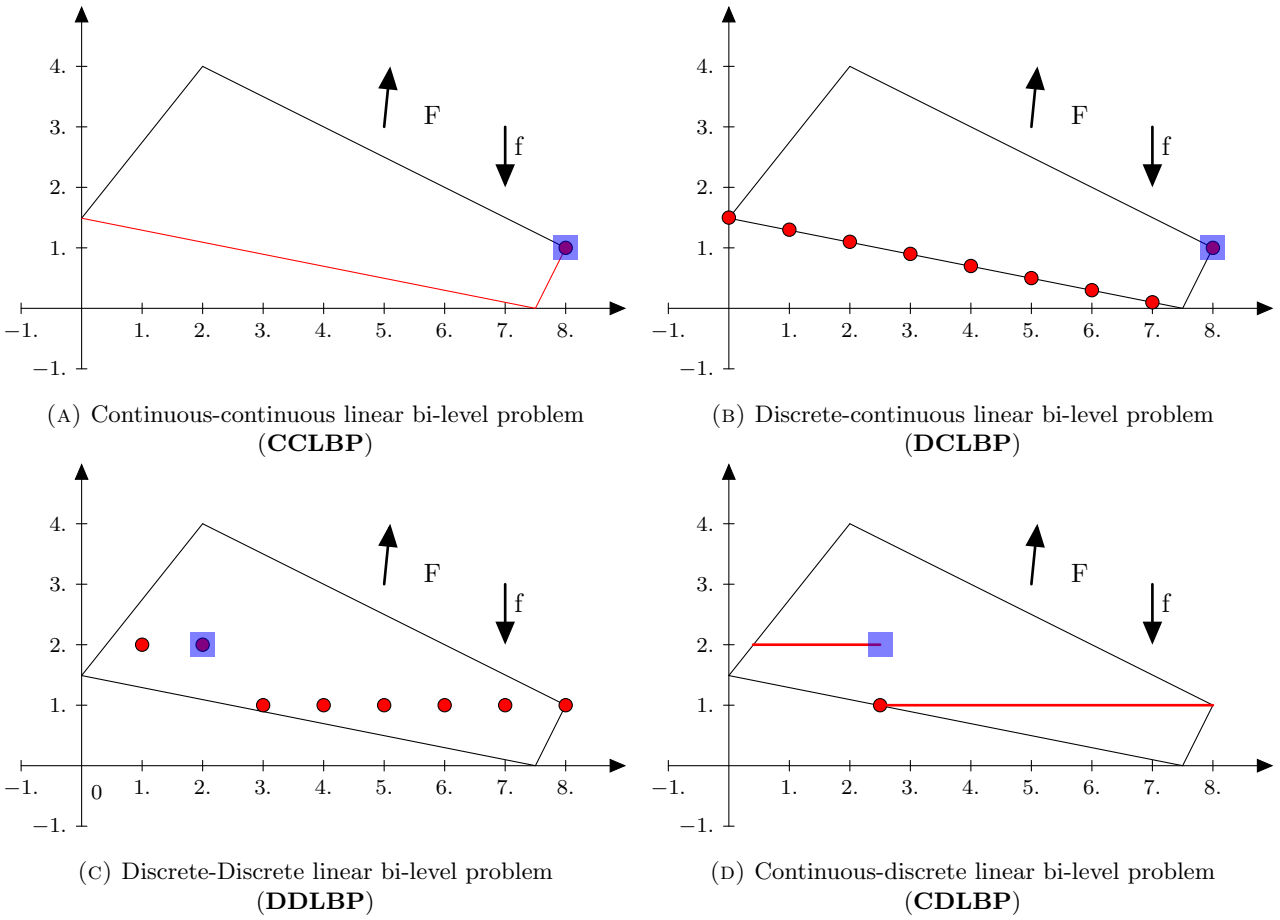
For more properties on mixed-integer linear bi-level problems, the reader can refer to [30, 101].

2.4.2.2 Reduction to linear problems

Let's consider a single-level problem P where $x \in \{0, 1\}^n$ and $y \subseteq \mathcal{R}^m$:

$$
\begin{aligned}
&\min_{x\geq 0} \quad F(x,y) = -x - 10y \\
&\text{s.t.} \ \min_{y\geq 0} \quad f(y) = y \\
&\qquad \text{s.t.} -25x + 20y \leq 30 \\
&\qquad\qquad x + 2y \leq 10 \\
&\qquad\qquad 3x + y \leq 30 \\
&\qquad\qquad 2x - y \leq 15 \\
&\qquad\qquad 2x + 10y \geq 15 \\
&\qquad\qquad x, y \geq 0 \quad \text{(continuous or integers)}
\end{aligned}
$$

PROGRAM 2.17: Program described by the 4 examples in Figure 2.15



(A) Continuous-continuous linear bi-level problem (**CCLBP**)

(B) Discrete-continuous linear bi-level problem (**DCLBP**)

(C) Discrete-Discrete linear bi-level problem (**DDLBP**)

(D) Continuous-discrete linear bi-level problem (**CDLBP**)

FIGURE 2.15: Categories of bi-level problems with possible discrete variables

$$
\begin{aligned}
&\min_{x\geq 0} \quad c_1^T x + d_1^T y && (2.1)\\
&\text{s.t.} \ A_1 x + B_1 y \preceq b_1 && (2.2)\\
&\qquad x \in \{0,1\}^n, y \subseteq \mathcal{R}^m && (2.3)
\end{aligned}
$$

Without loss of generality, consider $c_1 \leq 0$ and $d_1 \leq 0$. Let $\Theta : \mathcal{R}^n \to \mathcal{R}$ be a continuous function so that $\Theta(x) \geq 0 \quad \forall \quad 0 < x < \mathbb{1}^n$ and $\Theta(x) = 0 \quad iff \quad x \in \{0,1\}^n$.

Theorem 2.7. *If Θ is a concave function there exists a positive real number M such that(2.7) and the following problem have the same optimal solution:*

$$\min_{x \geq 0} \quad c_1^T x + d_1^T y + M\Theta(x) \tag{2.4}$$

$$s.t. \ A_1 x + B_1 y \preceq b_1 \tag{2.5}$$

$$0 \preceq x_1 \preceq \mathbb{1}^n \tag{2.6}$$

$$\tag{2.7}$$

In [30], Bard shows that Theorem 2.7 may be used to transform (DCLBP) in a bi-level linear optimization problem and (DDLBP) as three-level linear optimization problem. However there is no linear equivalent for (CDLBP). These results are only applicable for binary decision variables and such reduction implies variable transformation if the binary assumption cannot be made.

2.4.2.3 Resolution approaches for mixed-integer bi-level problems

Like single-level algorithms for integer(mixed-integer) problems, algorithms for discrete bi-level problems should rely on the generation of bounds. As it has been shown in section 2.4.2.1, the relaxation consisting in solving the continuous version of a integer bi-level problem may not be valid. Contrary to the convex-linear bi-level problems, (mixed-)integer bi-level problems have less results in the literature not because they have not been studied, but because they are strongly hard. Most of the developed algorithms are bi-level version of Branch and Bound algorithms.

Since relaxing a discrete bi-level problem does not provide necessarily valid bounds, Wen and Yang in [372] shifted the lower-level constraints to the upper-level problem and removed the lower-level objective function. The optimal solution value of the resulting single-level problem constitutes a valid lower-bound (upper-bound) when F is minimized(maximized) and may be used in a Branch & Bound procedure. The authors have shown that the computational time grows exponentially when the number of upper-level decision variables increases linearly. Thus, they proposed a heuristic based on a judgment index for the weighted estimated optimal solution of the problem while neglecting the lower-level objective value. They also applied it for the weighted estimated optimal solution of the problem while neglecting the upper-level objective function. However this methodology can only solve problems where the upper-level problem has no constraints and variable are all integers. Furthermore, the lower-level objective function should not contains any integer variables.

Bard and More in [33] developed a solution algorithm for pure binary bi-level problems with constraints at lower-level. This algorithm is a based on branch & bound approach branching on the upper-level decision variables. In addition, the upper-level objective is replaced by the constraints $F(x, y) \leq \alpha$ where α is a parameter which is successively increased. The authors claimed that this algorithm may be modified to handle integer decision variables. In [278], Bard and More presented a Branch & Bound algorithm to solve mixed-integer bi-level optimization problem. They called High point: the optimal solution of the continuous and relaxed bi-level problem where the lower-level objective function has been removed. The hight point provides valid bounds used to prune non-promising nodes. They showed that if all variables controlled by the leader are discrete, the algorithm finds an optimal bi-level solution. They also demonstrated that it is no longer possible to prune a

node which provides an integer solution. Indeed, the integer solution may not be a bi-level feasible solution forcing to keep exploring the resolution tree.

A Branch and Cut version for bi-level problems has been introduced by DeNegre and Ralph in [101]. Based on the branch and bound algorithm provided by Bard and More in [32], they added cuts instead of branching whenever a non-feasible integer bi-level solution is found. By doing so, they no longer need to deal with non-feasible bi-level solutions. The authors claimed that the advantage of their procedure rely on standard pruning and branching rule applied in single-level integer optimization.

A decomposition approach based on Bender Decomposition [37] has been proposed by Saharidis and Ierapetritou in [316]. The only restriction of this algorithm concerns the integer decision variables which should be controlled by the upper-level decision maker. They defined the Restricted Master Problem (RMP) as a bi-level problem without lower-level objective and constraints that involve both continuous and integer variables. The auxiliary problem is the restriction of the original bi-level problem where all integer variables have been set. This auxiliary problem (AP) is thus a linear bi-level optimization problem. The authors used the KKT conditions and the active set strategy [154] to reformulate the AP in order to obtain a linear single-level problem. They use then the methodology applied in single-level optimization consisting in adding feasibility and optimality cuts until the RMP becomes optimal. Another Bender Decomposition has been considered by Fontaine and Minner in [131] who first replaced the continuous lower-level problem with its KKT conditions and obtained a non-linear mixed-integer problem. This resulting single-level problem is thus non-linear because of the complementary constraints. To avoid the non-linearity, they added binary values indicating if the dual variables or the complementary slack variables are equal to 0. By doing so, they can use the standard Benders decomposition developed for the linear case.

In [236], Li and Guo considered mixed-integer bi-level problems with integer variables at lower-level. Using a separation approach between the lower-level continuous and integer variables, they were able to transform the mixed-integer bi-level program into a mathematical program with complementarity constraints. The authors reported global and local minimizers that are equivalent for both programs.

Hemmati and Smith in [166] proposed a cutting algorithm for the competitive prioritized set covering problem. This problem represents two decision makers who select items one after another. The authors relied on equalities that support the convex hull of feasible solutions to apply cuts.

Fischetti et al. in [127] considers mixed-integer bi-level problems where integer variables can occur at both levels. They developed a new Branch & Cut approach relying on a new class of inequalities.

Camaria et in [69] described two exact algorithms for discrete variables at both levels. The first one is a cutting algorithm with non-linear cuts. These cuts can be reformulated as bi-level problems with integer upper-level variables and continuous lower-level variable. Therefore each time a cut is added, a new lower-level is added to the original bi-level problem. The second algorithm proposed by the authors is Branch & Cut algorithm taking advantage of the geometrical properties of bi-level problems. The rational behind this idea is the introduction of inequalities that cut off the largest possible set of integer but not bi-level feasible solutions. Results reported by the authors have shown that both approaches are faster than existing algorithms applied on the same benchmark set.

TABLE 2.4: Summary table of classical approaches for bi-level optimization

References	Type of levels	Type of approach	section n°
[67, 43, 94, 296, 161]	convex	Extreme point search	2.4.1.1
[132, 31, 30, 33] [44, 36, 187, 7, 161, 330]	convex	Reformulation to single-level problem	2.4.1.2
[218, 361, 320, 122]	convex	Descent approaches	2.4.1.3
[2, 3, 174, 71, 11, 373]	convex	Penalty approaches	2.4.1.4
[245, 259, 82, 261]	convex	Trust-region approaches	2.4.1.5
[274, 353, 212, 210, 211]	general non-convex	enumeration, Branch & Bound	2.4.2
[372, 33, 278, 101]	mixed-integers	Branch & Bound	2.4.2
[316, 131, 236]	mixed-integers	Decomposition techniques, separation	2.4.2
[166, 127, 69]	mixed-intergers	cutting algorithms, Branch & cut	2.4.2

2.5 Metaheuristics for bi-level optimization

Bi-level optimization is a complex and challenging task that requires often strong assumptions to be efficiently performed. The simplest versions of bi-level problems are already $\mathcal{NP}$-hard and classical resolution approaches that we discussed in Section 2.4 do not scale when the number of decision variables increases at both levels. The complexity induced by multiple levels makes classical approaches non-efficient even though very important theoretical results have been reached. For that reason, researchers turned towards metaheuristics.

Metaheuristics (e.g. genetic algorithm [169], ant colony optimization [107], particle swarm optimization [194]) are inspired by nature and have been widely used in single-level optimization cases to tackle $\mathcal{NP}$-hard problems. Gathering most of the main researchers in the field (e.g. K. Deb, El-G. Talbi), they are the spearhead of the methodologies challenging large and complex optimization systems. Indeed for $\mathcal{NP}$-hard problems and under the assumption that $\mathcal{P} \neq \mathcal{NP}$ [163], it does not exist algorithms with polynomial complexity solving each instance to optimality. The goal is thus to determine near-optimal solutions while guaranteeing a polynomial time complexity. Since convex bi-level problems are $\mathcal{NP}$-hard contrary to their single-level equivalents, the scope of metaheuristics has been extended.

Talbi in [350] proposed a first taxonomy to classify the different metaheuristics for bi-level optimization that is illustrated in Figure 2.16. We extend this taxonomy with a new category: "*Lower-level approximation approaches*" which relies on machine learning and approximation techniques.

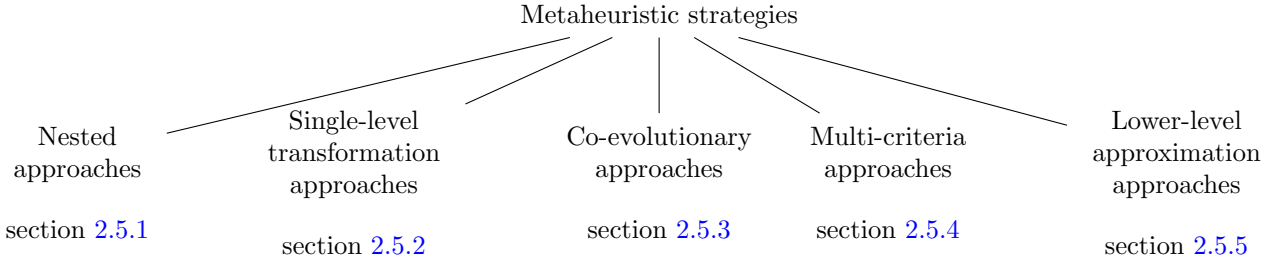


FIGURE 2.16: Extended bi-level metaheuristics taxonomy

2.5.1 Nested approaches

2.5.1.1 Description

Nested optimization approaches (see Figure 2.17) sequentially solve the upper-level and the lower-level problems. Different metaheuristics and exact optimization algorithms may be combined to give birth to a large set of hybrid algorithms. Generally, the choice of an algorithm for each level strongly depends on the nature of this same level. The first classification proposed by Prof. E-G Talbi in [350], the nested approach can be divided into 2 main categories of algorithms: *Repairing strategy* and *Constructing strategy*. In *Repairing strategy*, the lower-level problem is considered as a constraint which can be violated. The goal is thus to improve the objective value of the upper-level while trying to repair each time solutions being not optimal for the lower-level problem. The *Constructing strategy* sequentially solves the upper-level problem then the lower-level problem to increase both objectives. Both strategies are very equivalent since each upper-level decision x should be accompanied by the resolution of the lower-level problem.

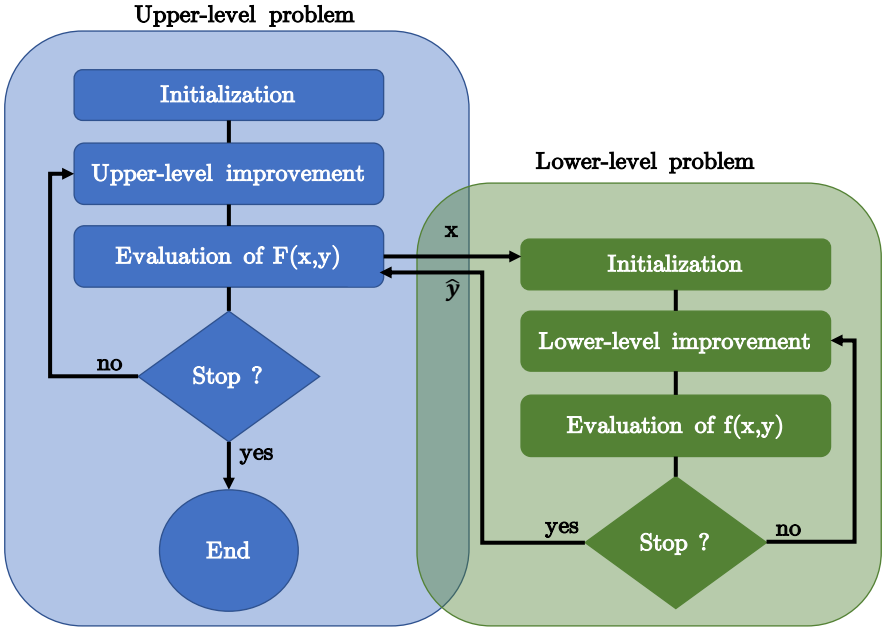


FIGURE 2.17: Repairing strategy

Wang et al in [273] proposed a nested bi-level metaheuristic which is fundamentally an hybrid version to solve bi-level linear programming problems. Improvements at upper-level are realized with a genetic algorithm while a standard linear programming solver is in charge of the lower-level. In [365], the same authors extends their approach to deal with linear-quadratic bi-level programming problems.

In [62], Calvete et al. tackled a continuous-linear bi-level problem by using genetic algorithm (GA). Each encoded solution is an extreme point of $\mathcal{S} = \{(x, y) \in \mathbf{X} \times \mathbf{Y} : G(x, y) \preceq 0, g(x, y) \preceq 0\}$. They applied two different crossovers: the first one may generate offsprings which are not "basis"(extreme points) while the second keeps the basis property of the parent. The upper-level objective function is also penalized by an additional term indicating whether or not the solution is optimal for the lower-level problem.

In order to solve transit scheduling and toll-setting problems, Shepher and Sumalee in [329] and Zhang in [398] chose a genetic algorithm at upper-level to determine prices while solving the lower-level assignment problem with classical approaches.

Dimitriou et al. in [103] coped with the problem of road network designing and pricing. The upper-level decision maker is typically the system operator of the roads and the lower-level decision maker represents a set of customers. The authors applied a genetic algorithm to the upper-level problem while solving the lower-level using a path enumeration algorithm.

Particle swarm algorithms [143, 225, 239] have been often employed to tackle bi-level problems. Their particular structure (global and local search) is particularly suitable for solving such problems.

In [238], Li and Wang investigated nonlinear bi-level programming problems in which the lower-level objective function is affine. Using a specific decomposition scheme for the lower-level problem, the authors tackled these problems by combining a genetic algorithms and non-linear optimization techniques.

Küçukaydin et al. in [228] developed an hybrid tabu search algorithm in order to solve a competitive bi-level facility location model. The modified tabu search algorithm is combined with a gradient ascent algorithm to solve the upper-level problem while the lower-level problem is solved with a branch & bound algorithm.

In [367], Wang and Dang proposed a nested optimization approach to tackle nonlinear bi-level programs where the upper-level problem is non-differentiable and the lower-level problem is non-convex. The upper-level algorithm is mainly a genetic algorithm with modified evolutionary operators. A deterministic approach is utilized at lower-level.

In [63], Calvete et al. considered a production-distribution problem in which the upper-level problem is a Mutiple Depot Vehicule Routing Problem (MDVRP) and the lower-level problem is a transportation problem. They applied an Ant Colony Optimization (ACO) algorithm to solve the upper-level problem while solving the transportation problem at lower-level. Ant Colony Optimization algorithms are efficient for solving network problems, several works [14, 61] considered combinations of ACO and specialized metaheuristic to tackle production bi-level bi-level production-planning problems. In [342], Scrivastava et al. considered a hybrid metaheuristics at upper-level while considering Petri Net at lower-level to minimize the waiting time of vehicle at traffic signal to improve road network traffic.

Recently, Islam et al. in [175] proposed a nested evolutionary algorithm using differential evolution at both levels. This approach has been mainly designed to tackle multi-objective bi-level problems.

2.5.1.2 Limitations

Despite the numerous works in the literature, most of the nested approaches described in this section are not suitable for large scale problems. Nested optimization is time-consuming especially when two population-based metaheuristics [349] are sequentially applied on the two levels. The efficiency of nested optimization approaches strongly relies on the difficulty to approach the lower-level problem. For linear and convex lower-level problems, fast and accurate resolution approaches exist. Nevertheless when no assumptions can be made, the repetitive lower-level optimizations quickly become unsuitable in practice.

2.5.2 Single-level transformation approaches

2.5.2.1 Description

The concept behind single-level transformation is closely related to the one addressed by classical methods and discussed in Section 2.4.1.2. The reformulation of bi-level to single-level optimization problems strongly relies

on the properties of the lower-level problem. When strong assumptions can be made such as convexity and twice-differentiability, the approaches described in section 2.4.1.2 can be applied. Therefore, several works made use of the KKT conditions (see Figure 2.18) to obtain a problem with a single-level. This resulting problem is generally non-convex and difficult to tackle. This is the reason why the applicaton of metaheuristics is relevant in this context. However such transformations limit the scope of bi-level problems that can be considered. For example, discrete and combinatorial bi-level problems stay out of this scope. Despite this downside, several works in the literature have taken advantage of reformulation techniques.

Hejazi et al in [165] reduced the linear bi-level to a single-level linear problem. They solved the resulting problem by applying a genetic algorithm that encodes extreme points.

KKT reformulation has been taken into consideration by Wang et al in [367] before applying a dedicated constraint handling scheme to solve the resulting single-level optimization problem. The same authors extend their approach in [369] to handle directly non-convex lower-level problem.

Jiang et al. employed particle swarm algorithms in [183] after reformulating bi-level problems into non-linear single-level problems with complementary constraints.

Similarly, Li and Wang in [237] turned their attention to fractional linear bi-level programming. As usual, they first utilized the lower-level optimality conditions to cope with the nested bi-level structure.

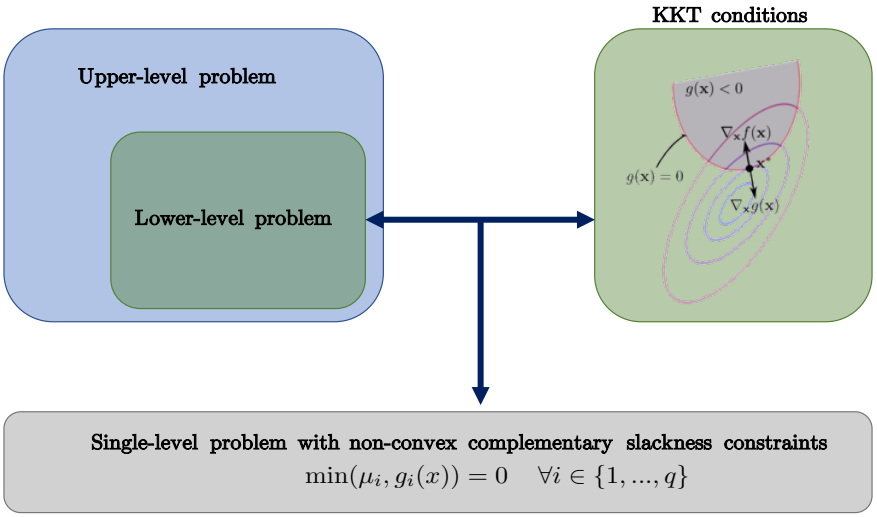


FIGURE 2.18: Reformulation of the lower-level problem using KKT conditions

In [283], the authors faced the same issue as for a classical approach. They reformulated the linear bi-level problem as a linear combinatorial problem with single level. In order to get rid of the binary variables used to linearize the complementary constraints, they evolved a population of binary individuals intended to replace these variables in the linear combinatorial problem. Evaluation of each individual is performed by assigning the binary values to the variables and by solving the resulting linear and continuous model with a linear solver.

A dual temperature simulated annealing approach has been considered by Sahin and Ciric in [317]. The authors demonstrated their approach with different categories of bi-level problems (e.g. linear, nonlinear, mixed integer nonlinear bilevel programming problems). In [137], Friesz et al tackled the equilibrium network design problem using a simulated annealing algorithm as well.

Recently, Sinha et al in [339] reduced bi-level into single-level problems using approximate KKT conditions. Based on the work of Dutta et al. in [114], the authors relied on the KKT proximity measure obtained

by relaxing the complementary slackness and equilibrium equations of the KKT conditions. The authors successfully embedded this measure inside an evolutionary algorithm.

Along similar lines of using KKT conditions, Ren and Wang in [310] solved bi-level linear programming with fuzzy coefficients in both objective functions. After KKT reformulation, the single-level optimization problem is modified into a multi-objective optimization problem which is then optimized with global criteria method.

2.5.2.2 Limitations

As aforementioned, reformulation approaches requires strong assumptions that cannot be made for many practical and realistic problems. In addition, the resulting problems are still very challenging even if they contain only a single level. One should also not forget that KKT conditions may induce some non-convex constraints/conditions to handle inequalities in the original problem. Most of the works in the literature applied an additional mechanism to replace these non-convex constraints (see Figure 2.18) by adding binary variables. Therefore, this category of approaches is only suitable when the number of constraints are limited.

2.5.3 Co-evolutionary approaches

2.5.3.1 Description

Coevolutionary algorithms are special kind of decentralized algorithms based on independent populations evolving apart. They may appear similar to the approaches described in the previous section but they do not have any nested design. Multiple populations explore their own decision space and exchange information (see Figure 2.19). This exchange can be realized by combining full or partial solutions. Co-evolution occurs when intimate species influence each other's evolution. Several co-evolutionary models have inspired the evolutionary computing field. The predator/prey model is a famous example. We generally distinguish two classes of co-evolutionary algorithms: competitive and cooperative. Competitive co-evolution has been first proposed by Hillis in [168] for sorting networks. Then several applications stemmed from his results and notably in game theory where players often compete as in bi-level optimization. Competition focuses on the abilities of species to evolve and develop new skills to outperform the other species during a so called "armed race". On the contrary, cooperative co-evolution relies on the skills emerging from species tending to collectively work to solve a common problem. The seminal work of Potter and De Jong in [303] has shown how cooperative co-evolution can be employed to optimize multi-dimensional functions. Few works in the literature tried to tackle bi-level optimization problems by using coevolutionary metaheuristics. The algorithm BIGA (Bi-level Genetic algorithm), designed by Oduguwa and Roy and described in [288], is the first coevolutionary algorithm developed to handle nested optimization problems. Although its name does not put the emphasis on the coevolutionary aspect, BIGA uses a coevolutionary operator aiming at synchronizing both populations, one for each level. The authors described it as a nested optimization algorithm since both problems are iteratively solved. However the presence of the coevolutionary operator shows that BIGA operates an exchange of information between both populations. BIGA can be considered as the first coevolutionary bi-level algorithm. Contrary to BIGA, the algorithm COBRA introduced in [235] by Legillon et al. does not work iteratively and follow the scheme depicted in Figure 2.19. Two independent populations represent respectively the upper-level and the lower-level problems which evolve independently from each other. The coevolutionary operator ensures that the bi-level hierarchical structure is satisfied. Finally, a more recent approach, CODBA, proposed in [73], aims at generating from the upper-level solutions many lower-level populations. The authors then evaluate in parallel each sub-populations. Each individual of these lower-level populations mates using crossover with the best archived

lower-level solutions until no more improvement occurs at lower-level. Although the authors categorized their algorithm as a bi-level co-evolutionary approach, the fact that it relies on a single thread reduce it to a simple nested optimization algorithm.

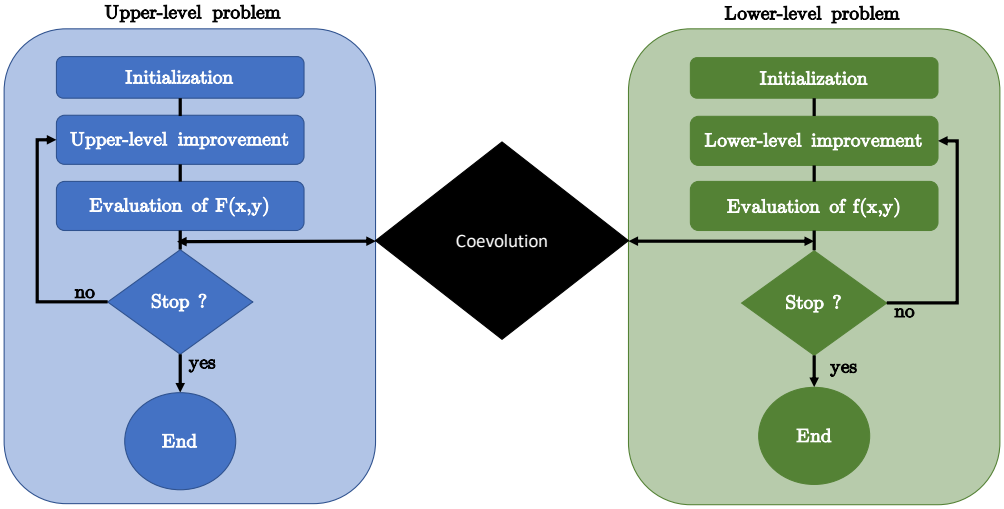


FIGURE 2.19: Coevolutionary strategy

2.5.3.2 Limitations

To the best of our knowledge, all bi-level co-evolutionary algorithms considered in the literature rely on two populations. Each population attempts to evolve independently the upper-level and lower-level decision variables. Trying to optimize independently both levels can raise many issues. For example, it is possible that pairing two level decisions x and y provides non-feasible solutions. In the light of the few identified works on bi-level co-evolution, we need to temper these limitations since it exists many kind of co-evolution. This work for example demonstrates how we can cope with the strong dependent epistatic links between both levels.

2.5.4 Multi-criteria approaches

2.5.4.1 Description

A bi-level program cannot be directly reformulated as a bi-objective program. Indeed as depicted in Figure 2.20, the optimal bi-level solution $(x = 4, y = 4)$ is dominated by any point at the intersection between the constraint set and the polar cone formed by $-F$ and $-f$. Marcotte and Savard [262] have demonstrated that there is no guaranty of Pareto optimality for a bi-level optimal solution except if both objective functions are collinear. To cope with this issue, Fliege and Vicente in [128] employed a different strategy. They have shown that there exists an order relation such that bi-level optimal solutions become non-dominated points with regards to this order relation. For this purpose, they assume that the lower-level problem is convex and that f,g are twice-differentiable. Let us define two bi-level solutions $u = (u_x, u_y)$ and $v = (v_x, v_y)$. $u_x(v_x)$ represents upper-level variables while $u_y(v_y)$ stands for the lower-level variables. The order of relation proposed in [128] is described in Equations 2.8. The authors affirmed that this new order relation is hard to work with and proposed to take advantage of the convexity and differentiability assumptions made before. Therefore, they replaced $u_y \in \arg\min f(u_x, .)$ by $||\mathcal{D}^2 f(u_x, u_y)|| < 0$ where $\mathcal{D}^2 f(u_x, u_y)$ denotes the gradient of f with respect to u_y. Further inquieries on the relationship between bi-level and multi-criteria problems can be found in [301, 28, 371, 315].

$$u \prec v :\Leftrightarrow [u_x = v_x \quad \text{and} \quad f(u) \leq f(v)] \tag{2.8}$$
$$\text{or}$$
$$[u_y \in \arg\min f(u_x, .) \quad \text{and} \quad F(u) \leq F(v)]$$

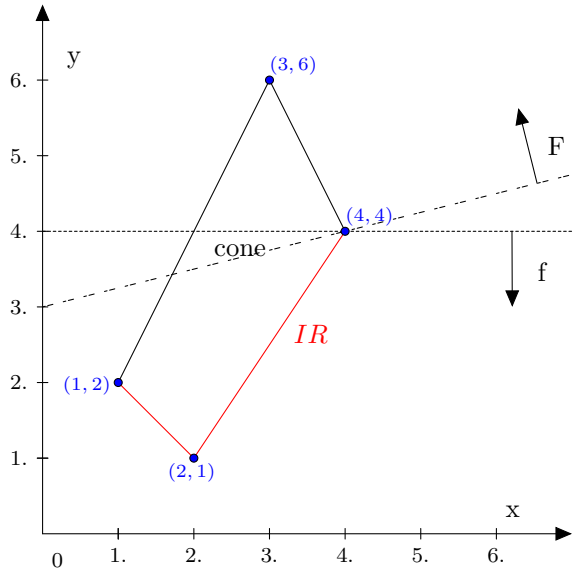


FIGURE 2.20: Example of bi-level optimal but not Pareto optimal solution

The application of multi-objective metaheuristics on bi-level problems derives directly from the ability to find a new relation of order. Russka et al. investigated in [314] the relation between bi-level optimization and multi-criteria optimization in order to construct multi-objective evolutionary bi-level algorithms. Their approach does not require any regularity assumptions and relies on some partial order compatible with bi-level optimization.

2.5.4.2 Limitations

As it can be observed, few approaches have been designed to reformulate a bi-level problem into an equivalent multi-criteria problem. The success of such an approach strongly depends on the capacity to ensure that the optimal bi-level solutions belongs to the optimal Pareto front of the resulting multi-criteria problem. If this occurs, any multi-objective evolutionary algorithms could be taken into consideration. For additional details on single-level and multi-objective evolutionary algorithms, the readers may refer to [397, 400].

2.5.5 Lower-level approximation approaches

2.5.5.1 Description

Most of the approaches discussed so far attempted to tackle the nested structure of bi-level problems by re-formulating the lower-level or separating both levels. This section introduces a recent category of approaches adopting a new strategy. Instead of questioning the structure, they attempt to reduce the evaluation cost of lower-level instances generated during optimization. For this purpose, they focus on meta-modeling approaches [364] where evaluation functions are very expensive. These approaches rely on surrogate models which aim at approximating either the lower-level variables ($\mathcal{IR}$) or directly the upper-level solution value for solutions

belonging to $\mathcal{IR}$. Indeed, it can be valuable to avoid explicit optimization of a lower-level instance for a given upper-level decision. Approximation of the $\Psi(x)$ could drastically reduce the expensive lower-level optimizations. Under Lipschitz continuity assumption, two close upper-level decisions x and x' can be assumed to lead to two close rational lower-level decisions $\hat{y}$ and $\hat{y}'$. This methodology has been considered by Sinha and Deb in [332] with the recent bi-level evolutionary algorithm based on quadratic approximations (BLEAQ). This algorithm tries to approximate the lower-level decision variables with regards to the upper-level decision variables in order to reduce the number of lower-level optimizations. In [200] and chapter 4, we describe another approach proposed in the context of this PhD work. It has been designed to approximate directly the upper-level objective value instead of each lower-level decision variable separately. Later on, Sinha et al. proposed an improved version of their BLEAQ algorithm [335, 336] to approximate the lower-level objective function as well. Finding the most accurate surrogate model is generally a real challenge. Therefore, Islam et al. in [176] investigated the use of multiple surrogate models. Their work is based on Kriging techniques [305] and Response Surface methods [156]. In [15], Angelo et al. adapted a nested optimization approach based on two differential evolution algorithms to embed a surrogate model. This hybrid approach has been designed to reduce the number of lower-level optimizations and have shown promising results.

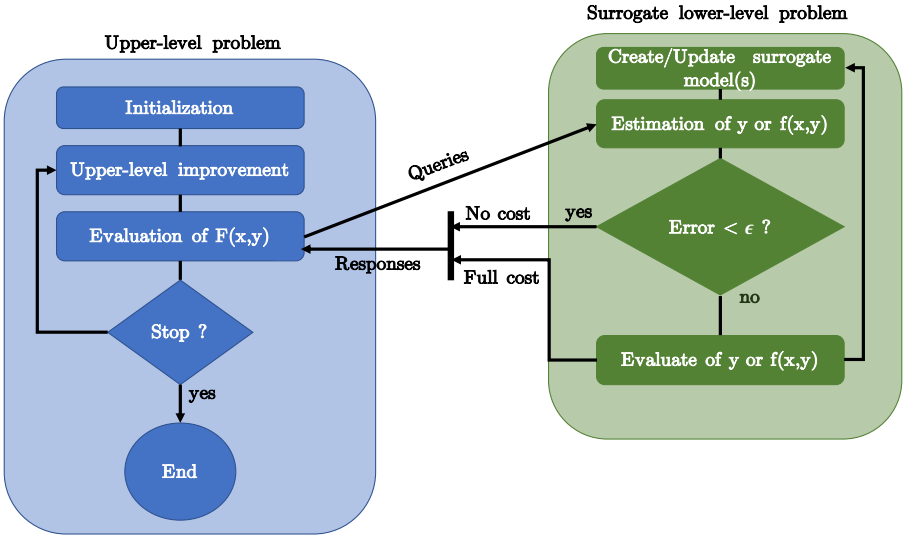


FIGURE 2.21: Approximation of the lower-level with surrogate(s)

2.5.5.2 Limitations

Despite their interesting work-flow and results, methods based on lower-level approximation have only been designed to cope with continuous bi-level optimization problems. The approximation of the lower-level decision variables lead to a multivalued function which is not trivial to deal with. An alternative consists in constructing as many surrogate functions as lower-level decision variables. Sinha et al. [332] applied this option and independently estimated each lower-level variable. This independence assumption can be inefficient when dealing with large-scale bi-level problems.

TABLE 2.5: Summary table of the existing Bi-level metaheuristics

References	Type of approach	section n°
[273, 62, 329, 398, 103, 143, 225] [239, 238, 228, 367, 63, 14, 61, 342, 175]	Nested approaches	2.5.1
[165, 367, 183, 237, 283, 317, 137, 339, 114, 310]	Single-level transformation approaches	2.5.2
[288, 235]	Co-evolutionary approaches	2.5.3
[262, 128, 301, 28, 371, 315]	Multi-criteria approaches	2.5.4
[364, 332, 200, 335, 336, 176, 305, 156, 15]	Lower-level approximation approaches	2.5.5

2.6 Conclusion

In this chapter, we have introduced the main properties of bi-level problems and their challenging nested structure. Deep studies on these problems have shown strong $\mathcal{NP}$-hardness even for the simplest version, i.e., linear levels. We also highlighted the most relevant relationships with other well-known domains (e.g. game theory, set-valued optimization, ...). Several strategies have been designed to solve small scale and mostly continuous bi-level problems. Most of the time, they rely on some kind of reformulation when strong properties can be assumed. Since the last decade, numerous metaheuristics have been introduced to tackle complex bi-level problems. Nevertheless, the nested structure has some disastrous effect on the performance of evolutionary algorithms. Therefore, researchers conceived new hybrid concepts such as meta-modeling to avoid expensive computations. Unfortunately, meta-modeling literature offers solutions only for continuous bi-level optimization.

Part II

Tackling bi-level problems with metaheuristics

Chapter 3

The Bi-level Clustering Optimization Problem

Contents

3.1 Introduction

Biomedical knowledge grows in complexity, and becomes encoded in network-based repositories, which include focused, expert-drawn diagrams, networks of evidence-based associations and established ontologies. Combining these structured information sources is an important computational challenge, as large graphs are difficult to analyze visually. Visual exploration of biomedical knowledge repositories is important for the expert to handle the increasingly complex content. A significant amount of this content is encoded as graphs, representing known or inferred associations between bioentities of various types. Canonical pathway databases like KEGG [191], Reactome [119] or Wikipathways [227] provide small-scale, manually drawn diagrams of molecular mechanisms. Another type of repositories, like STRING [348], NDex [304] or SIGNOR [300], relies on large databases of associations, which are queried and visualized as graphs. These graphs are procedurally generated and rely on automated layout algorithms.

An important kind of knowledge repository combines the properties of pathway databases and association repositories. These are middle to large size molecular interaction diagrams, established in the context of systems biomedicine projects. Such diagrams are in fact knowledge maps, covering different areas, from basic molecular biology [287, 286, 70, 188, 282] to various diseases [276, 267, 139, 226]. Especially in the area of human diseases, they offer contextualized insight into interactions between numerous convoluted factors, like genetic profile, environmental influences or effects of medications.

In order to efficiently support health research, these knowledge maps have to be useful and interpretable for domain experts, like life scientists or medical doctors. This is a challenge, as the knowledge mapped into such diagrams is difficult to explore because of their size and complexity. This is well reflected by the fact that they need dedicated software to be used efficiently [129, 47, 147]. Recently proposed solutions suggest coloring of entire modules in such diagrams using experimental datasets [47, 319]. However, they rely on existing definitions of modules, introduced when the maps were drawn. New solutions for aggregating information are needed to enable the discovery of new knowledge from these established repositories.

In this chapter, we investigate knowledge discovery in manually curated and annotated molecular interaction diagrams. To evaluate similarity of content we use: i) Euclidean distance based on expert-drawn diagrams, ii) shortest path distance using the underlying network and iii) ontology-based distance. We employ clustering with the above mentioned metrics used separately and in pairwise combinations. We propose a novel bi-level clustering approach together with an evolutionary algorithm for informative combination of distance metrics. We compare the enrichment of the obtained clusters between the solutions and with expert knowledge. Moreover, we calculate the number of Gene and Disease Ontology terms discovered by different solutions, as an independent measure of cluster quality and assess the relevance of this novel clustering model.

The remainder of this chapter is organized as follows. Section 2 defines the notion of cluster analysis and provides an overview of the existing clustering approaches in the literature. Then section 3 introduces the novel bi-level clustering optimization model and its mathematical formulation. A hybrid and parallel evolutionary algorithm is proposed to solve this new model in section 4. Sections 5 introduces metrics and data considered in the numerical experiments performed in section 6. Finally, the last section concludes this work and propose future investigations.

3.2 Cluster Analysis

3.2.1 Definition

With the emergence of online visual repositories like disease maps [139, 226] or metabolic maps [284], it becomes important to provide users with high-order interpretation of the content. As these repositories are large and densely networked diagrams, their visual examination, especially for discovery and data interpretation purposes, is a challenging task. Clustering approaches are a plausible methodology to address the challenge of visual exploration and understanding of large, complex networks. **Clustering** or **Cluster Analysis** (CA) is an unsupervised learning approach that consists in regrouping data into clusters based on a similarity measure. Extensively employed in Data Mining, clustering is a challenging knowledge discovery approach that does no rely on any apriori information. This is the reason why it is considered as an "unsupervised" approach. The literature lists 3 main categories of clustering : hard, soft and fuzzy. *Hard clustering* of a n-dimensional dataset $D = \{d_1, d_2, ..., d_n\}$ is a collection $C = \{C_1, C_2, ..., C_k\}$ of k subsets where $\bigcap_i C_i = \emptyset$ and $C_i \cap C_j = \emptyset \ \forall i \neq j$. In other words, hard clustering is a partitioning of the dataset. In a soft clustering, an object d_i may belong to several subsets of the collection D. It is basically a covering of all objects where each object belongs at least to one subset. Contrary to the two previous clustering , a fuzzy clustering describes the degree of memberships in all subsets of D.

3.2.2 Classical approaches

CA enables to discover relations between data points by grouping them according to a similarity metric. It is a very important tool in biomedical data interpretation, as it allows to explore and mine high-dimensional datasets. As a number of CA methods are summarized and compared in a recent review [377], here we would like to focus on an important aspect of the problem, which is the application of multiple similarity measures, in particular for graphs.

The literature is rich with clustering algorithms [381]. Since planar clustering is NP-hard [256], the use of exact optimization solvers is clearly not suitable for large datasets. Thus, most clustering approaches are based on heuristics, including broadly recognized methods like k-means [151], PAM [380] and hierarchical clustering [186]. These, and more sophisticated approaches, rely on the notion of similarity between clustered objects, obtained using various distance metrics [173]. It is worth mentioning that although different similarity metrics in clustering were evaluated on the same datasets [148, 246], their combination for improved clustering accuracy was proposed only recently [89].

Distance functions can be used to define a grid in the data space used by grid clustering algorithms [387], detecting cluster shapes with a significant reduction of the computational complexity when considering large data sets. In turn, distribution models [384] estimate density for each cluster based on the distance between data points, allowing statistical inference of the clustering. An interesting approach is the Formal Concept Analysis [343]. Here, a concept is an encoding extending the definition of distance or similarity. Generally, concepts allow to represent clusters with a set of satisfied properties, extending the criterion beyond distance. For instance, its application to disease similarity analysis [155] introduced a bipartite graph of disease-gene associations to define clusters of similar diseases.

3.2.3 Evolutionary Clustering (EC)

As these heuristics may be trapped in local optima, alternatives based on evolutionary computing emerged recently. Evolutionary algorithms have shown their abilities to overcome the drawbacks encountered in classical clustering algorithms that can be trapped in local optimal solutions.

Figure 3.1 depicts a general taxonomy of EC algorithms. 3 mains approaches emerge from the literature. Numerous contributions have been done in hard clustering whether the number of cluster k is known apriori or not. From binary to real encoding, different representations have been adopted to model clustering solutions. Medoid-based [328, 224], centroid-based [24, 270] and label-based representations [251, 280] are certainly the most encountered in the literature since they do not need extra computations and can be considered as *direct encoding* for the clustering problem. Nonetheless, more complex representations have been developed when dealing with a variable number of clusters. For example, in [254], the authors designed an evolution algorithm called EvoCluster in which each gene represents a cluster with all necessary information to discover the elements belonging to it. From the authors point of view, such an encoding has a real advantage over the classical encoding that can be mostly found in many works. However, its specificity requires the definition of new evolutionary operators. Several works [85, 171, 170] adopted this encoding and suggested to store the number of clusters in the genotype. Additionally to the EvoCluster encoding scheme, a graph-based representation has been proposed in [160] that is particularly well-suited for multi-objective clustering.

Concerning overlapping clustering, many investigations have focused on approach evolving fuzzy partitions. As for hard clustering, some contributions [233, 394] assume that the number of clusters has been already determined while more recent publications [293, 298] strengthened the determination of the optimal number of clusters. Regardless of the fixed or variable number of clusters, evolutionary fuzzy clustering algorithms are mostly adaptations of the approaches designed for hard clustering. The fact that each element belongs to each cluster with a certain affinity enforces the choice of a matrix-based representation. Evolutionary fuzzy clustering has a broad scope of application going from pattern classification to dynamic system identification.

Last but not least, multi-objective evolutionary algorithms extends the approaches described previously. They can be selected for hard and overlapping clustering. It turns out that clustering is naturally bi-objective. Indeed, minimizing the number of clusters and the distance inside each element to their cluster representative is conflicting. This intra-class distance is minimal when the number of clusters is equal to the number of data while it reaches a maximum when $k = 1$. Therefore, a multi-objective evolutionary clustering algorithm [25] does not provide a single clustering but a set of alternative clusterings that does not "dominate" each other in terms of size and intra-class distance. In [160], the authors used as fitness vector the compactness and the connectedness of clusters. While compactness is closely related to the notion of intra-class distances, connectedness measures the degree of neighbor objects placed in the same cluster. Both objectives are conflicting and the authors justified their use by claiming that they permit to obtain a stable number of clusters. This property avoids the convergence to trivial solutions such as $k = 1$ or $k = N$ with N the number of data. On a similar note, intra-class and inter-class distances have been proposed as objectives in [312]. Notice that the inter-class distance is maximal when the intra-class is null.

At this point, we would like to refer the readers to the following surveys [172, 383, 182, 179] for more details on the EC approaches that has been described in this section. Finally, let us point out some other bio-inspired approaches and, more broadly, metaheuristics have been successfully taken into account to tackle clustering problems. This is the case for Particle Swarm Algorithms [355, 88], Ant Colony Algorithms [327, 399] and Variable Neighborhood Search [8, 232].

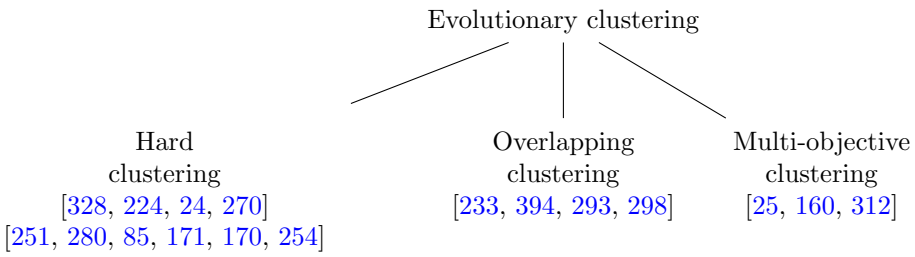


FIGURE 3.1: Evolutionary clustering taxonomy

3.2.4 Graph clustering in biomedicine

In biomedical research, disease mechanisms are often represented as networks of interactions on different scales - from molecular to physiological. These networks are in fact graphs, which can reach substantial size and complexity, as our knowledge on disease mechanisms expands. In order to make accurate interpretations using this interconnected body of knowledge, new approaches are needed to visualize meaningful areas and interactions in large biomedical networks.

Visual exploration of complex graphs requires certain aggregation of information about their content and structure, providing the user with an overview of dense areas of the graph, and their relationships. This task can be facilitated by means of graph clustering. Graph clustering groups vertices or edges into clusters that are homogeneous in agreement with a certain predefined distance function. An example is the application of local neighborhood measures to identify densely connected clusters in protein-protein interaction networks [20, 209]. Another approach is to construct clusters based directly on the global connectivity of the graph to identify strongly connected subgraphs [322, 164]. In these approaches, however, the visualization component of graph exploration is outside of the scope of analysis. Moreover, focusing on graph structure alone does not benefit from additional information on edges and vertices, available via various bioinformatics annotations. For instance, *eXamine* [104] uses annotations to improve the grouping of network elements for better visualization, while MONGKIE [180] bases on clustering graph-associated 'omics' data to improve the visual layout. Another interesting method, *Network2Canvas* proposes a novel lattice-based approach to visualize network clusters enriched with gene-set or drug-set information. Importantly, the approaches discussed above focus either on large networks without a visual layout (protein-protein interaction networks) or on small-scale molecular diagrams. However, to the best of our knowledge, the challenge of clustering of large, manually curated molecular interaction diagrams [139], remains unaddressed.

In this chapter, we focus on graph clustering of large repositories of molecular interaction networks. As these carry not only the information about their graph structure, but also information about manual layout and annotation of the elements, we decided to explore the simultaneous use of multiple distance functions to create the clusters. For this purpose, we propose a novel bi-level clustering optimization problem that is a reformulation of the well-know uncapacitated k-median problem to a two-levels nested problem.

3.3 A novel bi-level clustering optimization model

3.3.1 A classical mathematical formulation

As discussed in the previous sections, it exits many clustering approaches. In this chapter, we consider medoid-based clustering where medoids are cluster representatives and clusters are built around them. Medoids provide

important information concerning a cluster. For instance, they allow to obtain a median element that can highlight the differences between clusters. Generally, clustering based on k medoids can be modeled as an uncapacitated k-median problem (UKMP) that can be formalized as follows: *"Given a metric space M, a set of clients $C \in M$, and a set of facilities $F \in M$ such that $|M| = |F \cup C| = n$, open k facilities in F that minimize the sum of distances from each client to its nearest open facility"* . Clients play the role of elements to be clustered while medoids can be assimilated to facilities. The only difference lies in the fact that medoids can be both elements and cluster representatives. Closely related to the well-known "facility location problem", the general mathematical model of the UKMP is depicted by Program 3.1. We distinguishes two types of decision variables:

$$x_{jj} = \begin{cases} 1 & \text{if element j becomes a cluster representative, i.e, a medoid} \\ 0 & \text{else.} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if element i is assigned to cluster represented by medoid j} \\ 0 & \text{else.} \end{cases}$$

The objective function F represents the total distance from elements to their respective medoids: $\sum_i \sum_j d_{ij} x_{ij}$ with regards to a distance matrix $\mathbf{D}$. d_{ij} is the distance between element i and element j. The set of constraint (3.2) expresses that each element should be solely assigned to a single cluster. The set of constraints (3.3) ensures that an element j becomes a cluster representative (medoid) if some elements are assigned to it. Constraint sets (3.2) and (3.3) are specific to the "facility location problem". The last constraint really characterizes the UKMP. It states that the number of medoids and thus clusters should be equal to k.

$$\textbf{(UKMP)} \quad \min \quad \sum_i \sum_j d_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1 \quad \forall i \in \{1, ..., N\} \tag{3.2}$$

$$x_{ij} \leq x_{jj} \quad \forall i \in \{1, ..., N\} \quad \forall j \in \{1, ..., N\} \tag{3.3}$$

$$\sum_j x_{jj} = k \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \tag{3.5}$$

Program 3.1: The uncapacitated k-median problem (UKMP)

The k-median problem, a fortiori the UKMP, has been proven $\mathcal{NP}$-hard by Kariv and Hakim in [192] using a reduction of the dominating set problem.

3.3.2 Decomposition of the UKMP into a two-level problem

Medoid-based clustering algorithms have generally two sequential phases that update medoids and then assign each elements to the cluster with the closest medoid. These two phases are generally repeated until the algorithm convergence and the update of the medoids does not conduct to a drop of the fitness function value. As an example, we can cite the k-medoid algorithm in [297] which is the counterpart of k-means when cluster representatives should be elements from the dataset. Its pseudo-code is depicted by Algorithm 1. Even though this k-medoid algorithm is an heuristic, it highlights a very interesting property: clustering models can be

naturally decomposed into two nested optimization problems. The question would be now to find k optimal medoids knowing that the assignments to these medoids should be optimal in terms of total intra-class distance.

Algorithm 1 Pseudo-code for the k-medoids version implemented in [297]

```
Initialize medoids by randomly peaking k elements in the dataset
Assign each element to the closest medoid
while Total intra-class distance decreases do
    for each cluster do
        Select a new medoid minimizing the total intra-class distance
    end for
    Assign each element to the closest medoid
end while
return population
```

Mathematically speaking, we can reformulate Program 3.1 as a two-level program by first dividing the decision variables into two sets as follows:

1. $\{x_{11}, x_{jj}, ..., x_{NN}\}$ representing the selection variables for the medoid candidates
2. $\{x_{ij} : i \neq j, i \in \{1, ..., N\}, j \in \{1, ..., N\}\}$ representing the assignment variables for the elements to be clustered

Then Program 3.1 can be reformulated following the next steps:

$$
(UKMP) \begin{cases} \min & \sum_i \sum_j d_{ij} x_{ij} \\ \text{s.t.} & \sum_j x_{ij} = 1 \quad \forall i \in \{1, ..., N\} \\ & x_{ij} \leq x_{jj} \quad \forall i \in \{1, ..., N\} \quad \forall j \in \{1, ..., N\} \\ & \sum_j x_{jj} = k \\ & x_{ij} \in \{0, 1\} \end{cases}
$$

$$
\equiv (UKMP)' \begin{cases} \min & \sum_j d_{jj} x_{jj} + \sum_i \sum_{j:i \neq j} d_{ij} x_{ij} \\ \text{s.t.} & \sum_j x_{ij} = 1 \quad \forall i \in \{1, ..., N\} \\ & x_{ij} \leq x_{jj} \quad \forall i \in \{1, ..., N\} \quad \forall j \in \{1, ..., N\} \\ & \sum_j x_{jj} = k \\ & x_{ij} \in \{0, 1\} \end{cases}
$$

$$
\equiv (UKMP)'' \begin{cases} \min & \sum_j d_{jj} x_{jj} + \begin{cases} \min & \sum_i \sum_{j:i \neq j} d_{ij} x_{ij} \\ \text{s.t.} & \\ & \sum_j x_{ij} = 1 \quad \forall i \in \{1, ..., N\} \\ & x_{ij} \leq x_{jj} \quad \forall i \in \{1, ..., N\} \quad \forall j \in \{1, ..., N\} \\ & x_{ij} \in \{0, 1\} \end{cases} \\ \text{s.t.} & \\ & \sum_j x_{jj} = k \\ & x_{jj} \in \{0, 1\} \end{cases}
$$

We can easily observed that the original $(UKMP)$ can be divided into two problems. In fact such a decomposition is clearly related to Bender's decomposition [37] and has been discussed in section 2.3.5.6. This is a kind of "divide & conquer" strategy that permits to separate the original problem into smaller but nested sub-problems. The $(UKMP)''$formulation can be further improved to obtain a more general two-level model as illustrated by $(UKMP)'''$.

$$(UKMP)''' \equiv \begin{cases} \min \quad \sum_j d_{jj}x_{jj} + \sum_i \sum_{j:i\neq j} d_{ij}\hat{x}_{ij} \\ \text{s.t.} \\ \quad \{\hat{x}_{ij} : i \neq j\} \in \begin{cases} \text{argmin} \quad \sum_i \sum_{j:i\neq j} d_{ij}x_{ij} \\ \quad \text{s.t.} \\ \quad\quad \sum_j x_{ij} = 1 \quad \forall i \in \{1,...,N\} \\ \quad\quad x_{ij} \leq x_{jj} \quad \forall i \in \{1,...,N\} \quad \forall j \in \{1,...,N\} \\ \quad\quad x_{ij} \in \{0,1\} \end{cases} \\ \quad \sum_j x_{jj} = k \\ \quad x_{jj} \in \{0,1\} \end{cases}$$

Since $d_{jj} = 0 \; \forall j \in \{1,...,N\}$, $(UKMP)'''$ can be finally written as follows:

$$(UKMP)''' \equiv \begin{cases} \min \quad \sum_i \sum_{j:i\neq j} d_{ij}\hat{x}_{ij} \\ \text{s.t.} \\ \quad \{\hat{x}_{ij} : i \neq j\} \in \begin{cases} \text{argmin} \quad \sum_i \sum_{j:i\neq j} d_{ij}x_{ij} \\ \quad \text{s.t.} \\ \quad\quad \sum_j x_{ij} = 1 \quad \forall i \in \{1,...,N\} \\ \quad\quad x_{ij} \leq x_{jj} \quad \forall i \in \{1,...,N\} \quad \forall j \in \{1,...,N\} \\ \quad\quad x_{ij} \in \{0,1\} \end{cases} \\ \quad \sum_j x_{jj} = k \\ \quad x_{jj} \in \{0,1\} \end{cases} \tag{3.6}$$

(3.7)

$(UKMP)'''$ is strictly equivalent to the original version that has not been decomposed. We have now two nested problems, i.e., a mathematical problem constrained by another one. The outer problem determines first the medoids by setting all variables x_{jj} that are now **parameters** for the inner problem. The latter, based on the provided medoids, determines the optimal assignment of elements to clusters by setting the variables $\hat{x}_{ij}$. Notice that the structure of the inner problem looks very familiar. Indeed, it is a "facility location problem" except that $x_{jj} \; \forall \; j \in \{1,...,N\}$ are not variables but parameters. In such a case, constraint (3.6) can be removed because it is either redundant if $x_{jj} = 1$, i.e, $x_{ij} \leq 1$ or it forces some assignment variables to become 0, i.e, $x_{ij} \leq 0$. Let us take an example in order to illustrate our last remark. Suppose without lost of generality that $k = 2$ with $x_{11} = 1$, $x_{NN} = 1$ and $x_{jj} = 0 \; \forall \; j \notin \{1, N\}$ and let us inject this information into the inner problem.

$$\{\hat{x}_{ij} : i \neq j\} \in \begin{cases} \text{argmin} & \sum_{i:i\neq 1} d_{i1}x_{i,1} + \sum_{i:i\neq N} d_{iN}x_{i,N} \\ \text{s.t.} & \\ & x_{i1} + x_{iN} = 1 \quad \forall i \in \{1,...,N\} \\ & x_{i1} \leq 1 \quad \forall i \in \{1,...,N\} \\ & x_{iN} \leq 1 \quad \forall i \in \{1,...,N\} \\ & x_{ij} \leq 0 \quad \forall i \in \{1,...,N\}, \forall j \notin \{1,N\} \\ & x_{ij} \in \{0,1\} \end{cases}$$

$$\equiv \begin{cases} \text{argmin} & \sum_{i:i\neq 1} d_{i1}x_{i,1} + \sum_{i:i\neq N} d_{iN}x_{i,N} \\ \text{s.t.} & \\ & x_{i1} + x_{iN} = 1 \quad \forall i \in \{1,...,N\} \\ & x_{ij} \in \{0,1\} \quad \forall i \in \{1,...,N\}, j \in \{1,N\} \end{cases}$$

PROGRAM 3.2: Inner problem parametrized with $x_{11} = 1$, $x_{NN} = 1$ and $x_{jj} = 0 \; \forall \; j \notin \{1, N\}$

Let us focus on Program 3.2 who describes the inner problem after setting the example. Although the inner problem looks originally like a "facility location problem", the fact that all variables $x_{jj} \; \forall \; j \in \{1, ..., N\}$ are parameters simplifies greatly the problem. Indeed the search space is drastically reduced going from $N^2 + N$ to $k \times N$ which represents $\frac{k}{N+1} * 100$ percent of the original number of variables. Imagine that the dataset to be clustered has 100 elements, i.e., $N = 100$ and that we wish two clusters, i.e.,$k = 2$. After setting the medoids, the inner problem search space only contains 1.98 % of all the initial variables. Notice also that the parametrization of the inner problem with the medoids changes its type. Indeed, we are not dealing anymore with a "facility location problem" but with a simple assignment problem. This should not be a surprise since our first assumption was to separate medoid choices from element assignment. In fact, the parametrized inner problem is an assymetric assignment problem since it restricts all elements from the dataset to belong to a single cluster while making no restriction on the size of the clusters. This decomposition scheme leverages a very interesting property. Even if an optimization problem such as the $(UKMP)$ is $\mathcal{NP}$-hard, it maybe possible to decompose it into several sub-problems. Some of these sub-problems may be solved in polynomial time. Indeed, the parametrized inner problem of $(UKMP)$ is clearly polynomial. To prove it, we will show that its coefficient constraint matrix is "totally unimodular".

Definition 3.1. (Unimodularity)
A square matrix whose determinant is 0, 1, or -1 is called unimodular. A matrix M is totally unimodular (TU) if the determinant of every square submatrix of M has value 0, 1, or -1.

Theorem 3.2. *(Sufficient Condition for TU)[323]*
A $m \times n$ matrix $\boldsymbol{M}$ is totally unimodular if the following conditions hold:

1. *Every element of M is 0, 1, or - 1.*
2. *Each column of M contains at most two nonzero elements.*
3. *The m rows of M can be partitioned into two mutually exclusive subsets M_1 and M_2 such that:*
   - *If any column contains two nonzero elements of the same sign, one element can be placed in M_1 and the other in M_2.*

- *If any column contains two non zero elements of opposite signs, both elements can be placed in the same subset.*

Suppose the inner problem is parametrized with k medoids. The relations defining the coeffcient constraint matrix between elements from the dataset and the medoids can be modeled as a bipartite graph $\mathcal{G}$ whose vertices can be partitioned into two disjoint and independent sets such that all edges only occurs between these two sets. In the current context, these two sets are clearly the dataset and the medoid set as shown in Figure 3.2.

Corollary 3.3. *The incidence matrix of a bipartite graph is totally unimodular*

By Theorem 3.2 and Corollary 3.3, the coefficient constraint matrix of the inner problem is clearly totally unimodular since it corresponds to the incidence matrix of the bipartite graph $\mathcal{G}$.

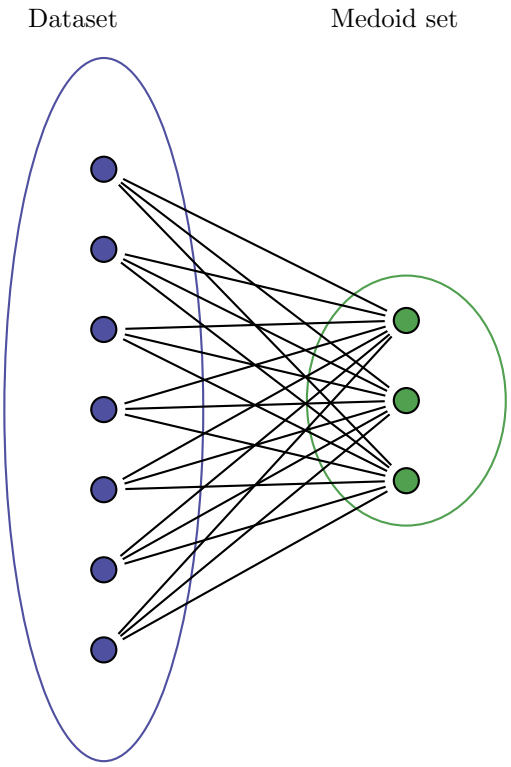


FIGURE 3.2: Assignment problem as a bi-partite graph

To conclude this section, we have shown that the decomposition of UKMP leads to a two-level problem that has an inner sub-problem that can be solved with a linear solver due to the total unimodularity of its coefficient constraint matrix. This is a very precious information that will be considered when tackling the resolution of the bi-level version presented in the following section.

3.3.3 From a two-level UKMP to the Bi-level Clustering Optimization Problem

The new two-level formulation representation by $(UKMP)'''$ show us that both sub-problems have the same objective functions. Both attempt to minimize the total intra-class distance between elements and their respective medoids using the same distance metric. Nevertheless, we could easily imagine that both objective

functions may rely on different distance metrics. The medoids could be chosen with regards to a first metric d_1. while the assignment of element to clusters could be performed according to a second metric d_2 while

A questions then arises: "What would be the benefit of having different metrics ?". Similarities in large and heterogeneous dataset can be evaluated in many different ways. For instance, disease maps (e.g. Parkinson, Alzheimer) can be described using multiple criteria (e.g. network, euclidean, gene ontology). One cannot consider that a single metric can capture all the knowledge contained in such complex maps. Therefore, some works [138] focused on metrics combinations to improve clustering. Multi-objective approaches could be also consider but they add another level of uncertainties. Indeed, such approaches would provide a set of non-dominated clusterings solutions which would not help to determine which of the objective metrics are the most relevant. An alternative could be to envisage weighted- sum approaches and try to determine the weights. This could be very useful to rank the distance metrics. Unfortunately, these approaches are very sensitive and are strongly dependent on the shape of the Pareto front [116]. Indeed, weighted-sum approaches searches for supported solutions, i.e., the ones that can be obtained by linear combinations of the multiple objectives. If the Pareto front is not regular and some area of the objective space are empty, weights may lost their signification. Figure 3.3 depicts such an example where two distances metrics d_1 and d_2 are believed to contribute identically to the combination, i.e., with weight $w_1 = 0.5$ and $w_2 = 0.5$. In this example, the Pareto front is non-regular with a large area having no solutions in the middle part of the front. Thus, one can wrongly conclude that the obtained Pareto solution reflects truly a situation where both distance metrics have the same impact. In fact, the solution obtained in the example of Figure 3.3 is biased toward the second metric, i.e., d_2.

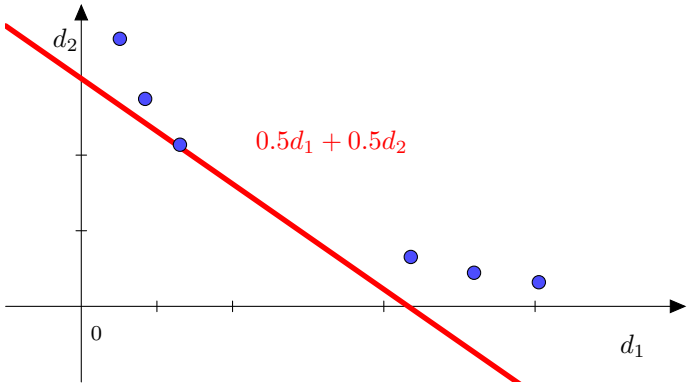


FIGURE 3.3: Example of weighted-sum on non-regular Pareto front

Contrary to classical approaches, combining distance metrics at different levels can have a real advantage. First it does not require any kind of weights and does not have to deal with several non-dominated clustering as for multi-objective approaches. It also provide a way to compare a distance metric relatively to another. Indeed as it can be observed for $(UKMP)'''$, the two nested sub-problems depends strongly on each other. The outer sub-problem is subject to the optimality of the inner sub-problem. This implies that the objective value of the outer problem strictly depends on the objective value of the inner problem. Therefore, the distance metric related to the inner problem has clearly higher priority compared to the one defined in the outer problem. In the case fo the $(UKMP)'''$, both distance metrics are the same. However, if we have two different metrics, say d_1 and d_2, the two-level problem will take into account a hierarchy of metrics to produce an new clustering. The two formulations (3.8) and (3.9) represent the bi-level clustering optimization model that has two different objective functions at each level. While medoids are chosen to minimize the total intra-class distance with regards to d_1, elements are assigned to medoids and thus clusters that minimize the total intra-class distance with regards to d_2.

$$
(BCOP) \equiv \begin{cases} \min \quad \displaystyle\sum_i \sum_{j:i\neq j} d^1_{ij} \hat{x}_{ij} \\ \text{s.t.} \\ \qquad \hat{x}_{ij} \in \begin{cases} \text{argmin} \quad \displaystyle\sum_i \sum_{j:i\neq j} d^2_{ij} x_{ij} \\ \qquad \text{s.t.} \\ \qquad\qquad \displaystyle\sum_j x_{ij} = 1 \quad \forall i \in \{1,...,N\} \\ \qquad\qquad x_{ij} \leq x_{jj} \quad \forall i \in \{1,...,N\} \quad \forall j \in \{1,...,N\} \\ \qquad\qquad x_{ij} \in \{0,1\} \end{cases} \\ \qquad \displaystyle\sum_j x_{jj} = k \\ \qquad x_{jj} \in \{0,1\} \end{cases} \tag{3.8}
$$

$$
\equiv \begin{cases} \min \quad \displaystyle\sum_i \sum_{j:i\neq j} d^1_{ij} x_{ij} \\ \text{s.t.} \quad \displaystyle\sum_j x_{jj} = k \\ \qquad \min f = \displaystyle\sum_i \sum_{j:i\neq j} d^2_{ij} x_{ij} \\ \qquad \text{s.t.} \quad \displaystyle\sum_j x_{ij} = 1 \quad \forall i \in \{1,...,N\} \\ \qquad\qquad x_{ij} - x_{jj} \leq 0 \quad \forall i \in \{1,...,N\} \quad \forall j \in \{1,...,N\} \\ \qquad x_{jj}, x_{ij} \in \{0,1\} \end{cases} \tag{3.9}
$$

We will now take the notations adopted in the bi-level optimization field. This means that the outer problem will be referred to as the upper-level problem while the inner problem will be named the lower-level problem. x_{jj} are the upper-level decision variables and all x_{ij} with $i \neq j$ are the lower-level decision variables. As discussed in chapter 2, the two formulations (3.8) and (3.9) are equivalent even if the description in (3.8) using the "argmin" notation seems more explicit. Indeed, it explicitly mentions that the upper-level objective function can only be evaluated once the "optimal" lower-level variables are known. In the formulation (3.9), the "argmin" operator is replaced by the "min" operator hiding the lower-level optimality of the lower-level variables. Nevertheless, both formulations are commonly found in the literature and it was worth discussing them in a real application cases such as the Bi-level Clustering Optimization Problem.

3.4 A nested hybrid and parallel evolutionary algorithm

3.4.1 A hierarchical discovery of new knowledge

In complex data repositories and notably in biomedical networks, a hierarchical view (see Figure 3.4) is very informative. Researchers do not only want to determine clusters but also would like to discover the mechanisms behind the similarities of aggregated data. For this purpose, a common practice is to produce multiple clusterings with varying k in order to obtain multiple layers of visualization. Hierarchical clustering approaches are

particularly well-suited for such analyses since the dendrogram generated by these approaches permits obtain all clustering with $1 \leq k \leq N$. One main issue is that the hierarchical views are discovered with a mechanism that is hierarchical by nature. Regarding of the validity or quality of the clustering, hierarchical approaches will always output a results for any k. We would like to obtain a data-driven approach that would not produce a clustering with k clusters if it is not relevant. Consequently, we adopt an alternative methodology relying on a bi-objective reformulation of (3.9) where the constraint setting the number of clusters is added as a second upper-level objective.

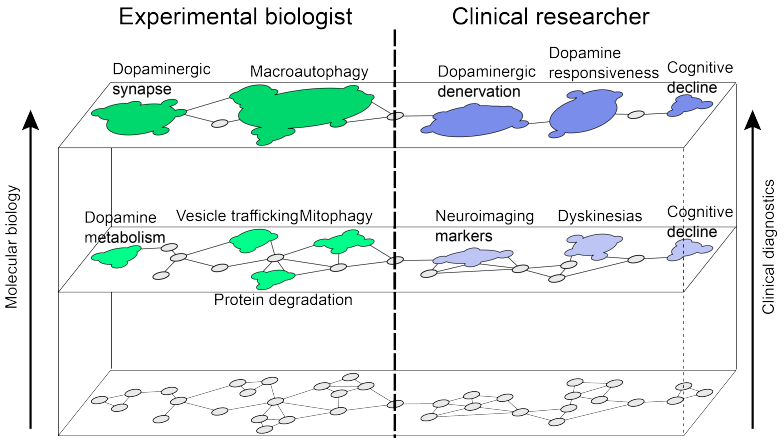


FIGURE 3.4: Visual interpretation requires knowledge aggregation

$$
\begin{aligned}
\textbf{(P)} \quad \min \quad & \Big(\sum_i \sum_j d^1_{ij} x_{ij}, \sum_j x_{jj}\Big) \\
\text{s.t.} \quad & \min f = \sum_i \sum_j d^2_{ij} x_{ij} \\
& \text{s.t.} \sum_j x_{ij} = 1 \quad \forall i \in \{1, ..., N\} \\
& \quad x_{ij} - x_{jj} \leq 0 \quad \forall i \in \{1, ..., N\} \quad \forall j \in \{1, ..., N\} \\
& \quad x_{ij} \in \{0, 1\}
\end{aligned}
$$

PROGRAM 3.3: Bi-objective bi-level clustering model

The two objectives aim to minimize both the intra-class inertia and the number of clusters respectively. These both objectives are negatively correlated since the intra-class inertia is minimal when the number of cluster equals the number of data points, while a single cluster generates a maximal intra-class inertia. Thus, optimizing Program 3.3 results in a set of clusterings, which are alternatives or non-dominating solutions.

For this purpose, a hybrid and parallel multi-objective evolutionary algorithm is considered to determine the best centroids at upper-level with regards to the bi-objective vector $\min F = (\sum_i \sum_j d^1_{ij} x_{ij}, \sum_j x_{jj})$ while an embedded exact optimization algorithm optimizes the lower-level problem $\min\{f = \sum_i \sum_j d^2_{ij} x_{ij} : \sum_j x_{ij} = 1 \quad \forall i \in \{1, ..., N\}, x_{ij} - x_{jj} \leq 0 \quad \forall i \in \{1, ..., N\} \quad \forall j \in \{1, ..., N\}\}$ where $x_{ij}, x_{jj} \in \{0, 1\}$.

3.4.2 Encoding a clustering

The encoding retained for solving the bi-objective BCOP is the binary medoid-based representation as illustrated in Figure 3.5. Let us recall that a solution in Evolutionary Computing is called an "*individual*" which is composed of a chromosome field (e.g. list, matrix) and a fitness value field (e.g. scalar, vector). Each chromosome is an aggregation of genes that represents an instantiation of the abstract decision variables. This instantiation depends naturally on the type (e.g. binary, integer, floating values). Binary medoid-based representation means that the chromosome of each individual will be a binary vector. For each individual and thus medoid vector, the optimal assignment of the remaining elements from the dataset to the clusters needs to be performed. Therefore, this encoding cannot be considered as a direct encoding since it requires some extra-computations to obtain a complete clustering. Nonetheless, we consider here that each medoid vector is assumed to have an unique corresponding assignment which corresponds to the optimal lower-level solution.

3.4.3 Evaluation of the encoded clustering

In order to compute the total intra-class inertia, the assignment of each element to a cluster (medoid) is required. As aforementioned, the medoid-based representation implemented in this work requires the optimal resolution of the lower-level problem, i.e, assignment problem. Each evaluation of an individual , i.e, medoid vector requires a lower-level optimization to obtain a complete clustering (see Figure 3.5). According to the taxonomy on bi-level metaheuristics described in chapter 2, the proposed approach can be qualified as "Nested" and one could believe that it may suffer from intensive lower-level optimization calls. Fortunately, we demonstrated in section 3.3.2 that the lower level problem is polynomial due to "total unimodularity" of its constraint coefficient matrix.

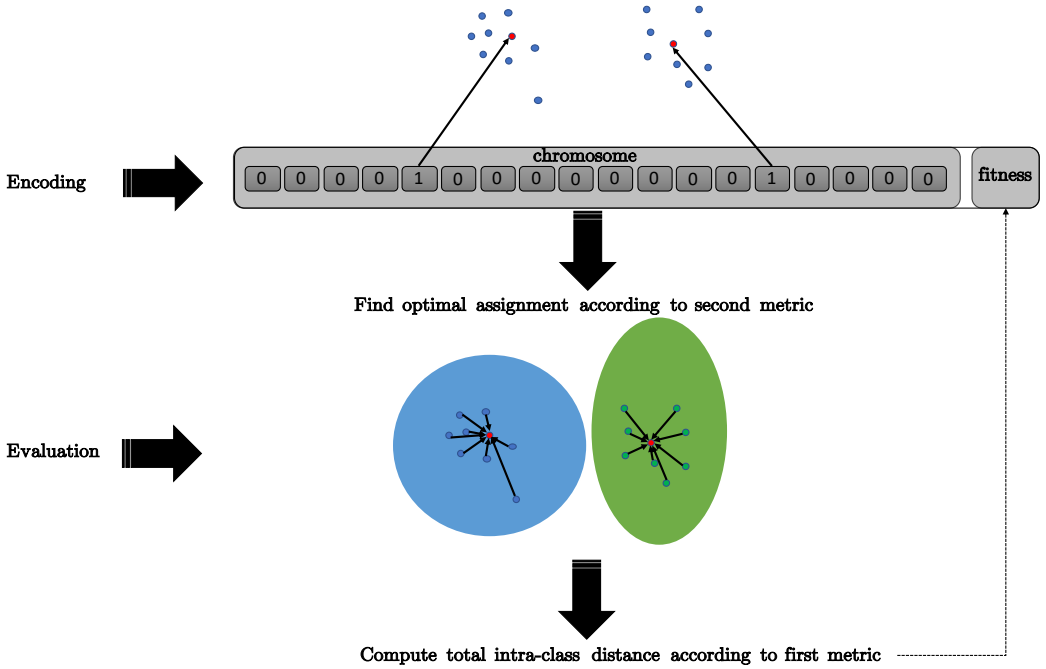


FIGURE 3.5: Evaluation of a solution

3.4.4 Workflow

At the time of this work, the version 3 of the *Non-dominated Sorting Genetic Algorithms* developed by Deb was not released yet. Thus, we turned our attention into NSGA-II [93], i.e., the previous version that has been extensively exploited in numerous works. NSGA-II is undoubtedly the most well-known and referenced algorithm in the multi-objective literature. It is a genetic algorithm with a panmictic (i.e., non-structured) population which is evolved by applying the typical genetic operators (selection, crossover, and mutation); then, the individuals are sorted according to their rank, and the best solutions are chosen to create a new population. In the case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals is used to improve diversity.

The application of NSGA-II is straightforward and does not require any modification of the algorithm (see Algorithm 2). Nonetheless, NSGA-II requires the full clustering, i.e., medoids and labels in order to compute the fitness value of an individual. Algorithm 3 details the process that permits to evaluate an individual. Line 3 generates an linear assignment program minimizing the total intra-class inertia with regards to the distance metric d_2. In order to obtain the labels indicating the reference cluster of each element from the dataset, line 4 calls a linear solver. In this work, we selected the IBM ILOG CPLEX optimizer which is the most widely used large-scale solver. Finally, line 5 computes the total intra-class distance according to the distance metric d_1. This distance becomes the fitness value of an individual.

Algorithm 2 Pseudo-code of NSGA-II for the BCOP

```
1:  Input: n // population size
2:  P ← random_population_medoid_vectors(n) // P is the population of medoid vectors
3:  while not termination_criteria() do do
4:     for i ← 1 to n do do
5:        parents ← select_parents(P,2)
6:        offspring ← mate(parents)
7:        offspring ← mutate(offspring)
8:        evaluate_fitness(offspring)
9:        P ← P ∪ offspring
10:    end for
11:    R ← P ∪ Q
12:    ranking_crowding(R)
13:    P ← select_best(P,n)
14: end while
15: return P
```

Algorithm 3 evaluate_fitness(offspring)

```
1: Input: offspring, i.e., a medoid vector
2: Parameters: d1 as upper-level metric and d2 as lower-level metric
3: LP_problem ← generate_assignment_problem(offspring,d2)
4: assigned_labels ← call_linear_solver(LP_problem)
5: intra_distance ← compute_total_distance(offspring,assigned_labels,d1)
6: number_of_clusters ← count_clusters(assigned_labels)
7: return ( intra_distance ,number_of_clusters )
```

Figure 3.6 summarizes graphically the bi-level clustering optimization problem as well as the nested multi-objective bi-level evolutionary algorithm proposed to tackle it. As illustrated in this figure, evolutionary computing is performed at upper-level while exact optimization is applied at lower-level. The next section introduces now more precisely the biomedical repositories used as benchmarks to assess the model and the resolution approaches described so far.

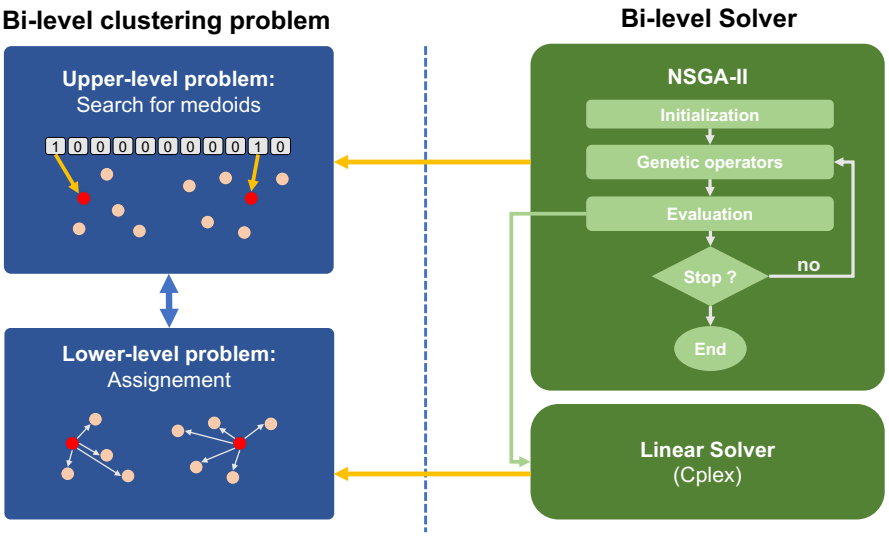


FIGURE 3.6: Graphical recap of the problem and the implemented resolution approaches

3.5 Tackling visual knowledge exploration in complex biomedical repositories

We used two separate disease map repositories as the evaluation datasets: the Parkinson's disease map (PD map, pdmap.uni.lu) and the AlzPathway map (AlzPathway, alzpathway.org).

The PD map is a manually-curated repository about Parkinson's disease, where all interactions are supported by evidence, either from literature or bioinformatic databases [139]. Similarly, the AlzPathway [276] is a map drawn manually on the basis of an extensive literature review about Alzheimer's disease. Both diagrams are human-drawn, so the use of Euclidean distance is reasonable, as the clusters will reflect the curators' knowledge. In turn, network and ontology-based distances will represent relationships difficult to comprehend by eye.

The PD map version from December'15 contains 2006 reactions connecting 4866 elements. Of these we selected 3056 elements of type *gene*, *mRNA* and *protein*. The AlzPathway (see Figure 3.8) contains 1015 reactions connecting 2203 elements, 1404 of which of type *gene*, *mRNA* and *protein*. For these elements, we extracted graphic coordinates for Euclidean distance and graph structure for network distance. For ontology-based distance, Entrez identifiers ncbi.nlm.nih.gov/gene are needed. For the PD map, HGNC symbols (www.genenames.org) were used to obtain Entrez ids. For the AlzPathway, Entrez ids were obtained from the Uniprot identifiers uniprot.org.

To test the robustness of our approaches in the situation, when the content of a molecular interaction network changes, we prepared a reorganized version of AlzPathway (AlzPathway Reorg). The CellDesigner file for this new version is provided in Figure 3.9. The AlzPathway Reorg is rearranged in such a way that a number of nodes is duplicated, edge lengths are shortened and the content is grouped together locally. Overall, 225 new elements were added, 140 of which of type *gene*, *mRNA* and *protein*, and 16 reactions were removed as redundant. The resulting map in comparison to AlzPathway has an overall smaller Euclidean distance (0.372 ± 0.183 vs 0.378 ± 0.182) and bigger network distance (0.890 ± 0.278 vs 0.601 ± 0.420).

We propose to combine different distance functions to improve the clustering results of large molecular interaction maps. We approach the problem by applying three distinct distance functions to the Parkinson's disease map as our use case. We then optimize the bi-level clustering approach to obtain clusterings from pairwise

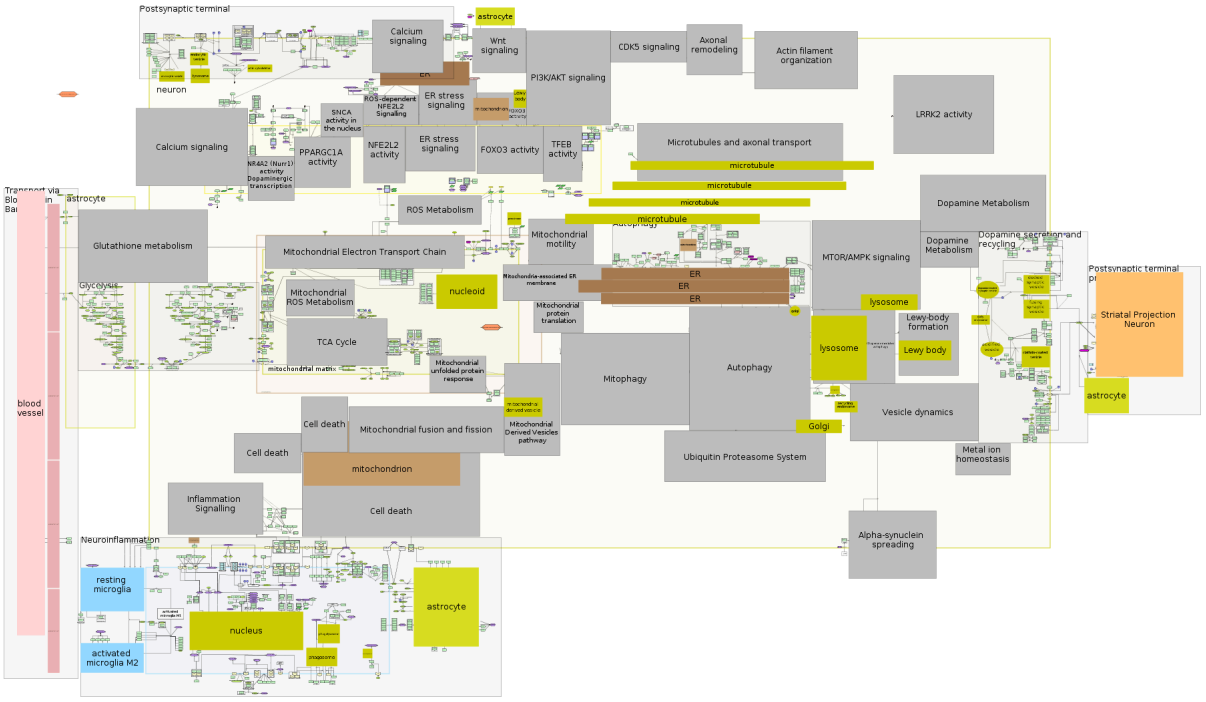


FIGURE 3.7: The Parkinson's disease map: a knowledge repository describing molecular mechanisms of Parkinson's disease

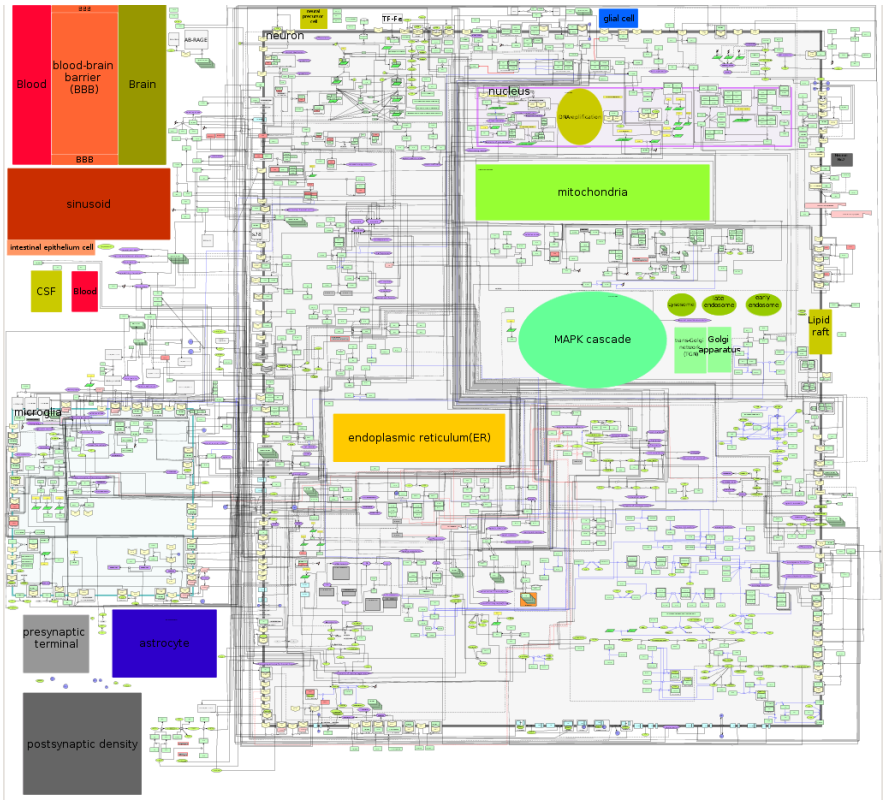


FIGURE 3.8: The Alzheimer's disease map: a knowledge repository describing molecular mechanisms of Alzheimer's disease

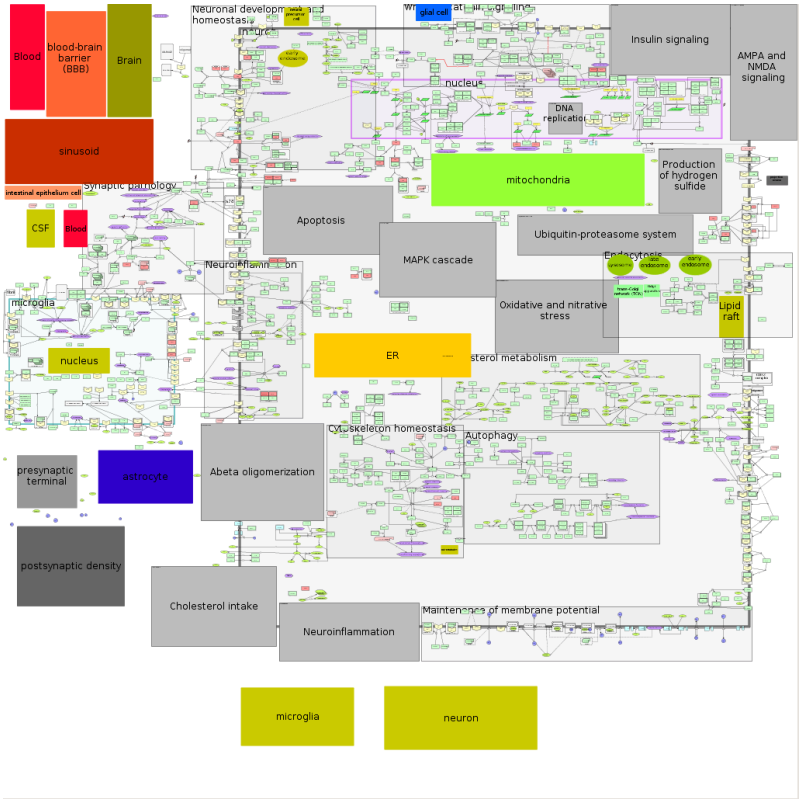


FIGURE 3.9: The Alzheimer's disease map after reorganizations

combinations of these metrics. We compare our algorithm against hierarchical clustering applied for the same set of distance functions. We evaluate the solutions by comparing against expert-provided groupings of the considered map contents, and by enrichment analysis of the obtained clusters.

Different distance functions can be applied to manually curate molecular interaction networks, reflecting distinct aspects of their contents. When clustering the contents of these disease maps, we considered the three following distances: Euclidean, network distance and ontology-based.

3.5.1 Euclidean distance

We calculated the Euclidean distance between elements of the maps by obtaining absolute values of (x, y) coordinates of elements of type *gene*, *mRNA* and *protein*. The rationale behind this distance function is that the distance between manually drawn elements reflects expert's knowledge about their similarity.

3.5.2 Network distance

We calculated the network distance between elements of the maps by constructing a graph from the interactions of the elements of type *gene*, *mRNA* and *protein*. PD map and AlzPathway are encoded in SBGN [285], which is essentially a hypergraph - interactions with elements are allowed. We transformed such a hypergraph into a graph by replacing each multi-element interaction by a clique of pairwise interactions between all elements in this interaction. The network distance over the resulting graph is the set of pairwise shortest paths between all elements in the graph. For unconnected elements, we set the distance to $2 * max(shortest\ path)$.

3.5.3 Ontology-based distance

Ontologies are restricted sets of defined terms allowing the representation and description of features in specific and complex system of knowledge. The Gene Ontology (GO) project has been initiated to provide shared vocabularies in order to characterize specific gene product properties. It enables the analysis of relationships between gene products. We used the GOSemSim [391] method to calculate pairwise similarity between the elements of the considered disease maps. The distance (d) was calculated as $d = 1/(1 + similarity)$. Three versions of the distance matrix were calculated, for Biological Process, Cellular Compartment and Molecular Function.

3.6 Clustering evaluation

3.6.1 Expert-based evaluation

In order to evaluate the performance of the considered clustering approaches we applied an external evaluation criterion. Both disease maps have already been compartmented by experts. These compartments constitute a form of expert-based clustering. Therefore, we would like to measure how similar are the clustering generated with the proposed approach to this expert-based knowledge. For this purpose, we have to rely on an supervised evaluation criterion. This measure relies on a "confusion matrix" (see Figure 3.10) that can only be considered for supervised cases where the problem is divided into classes that are known and fixed. In this work, the obtained clusterings will have different sizes, i.e., number of cluster k. In addition, confusion matrices focus on labels although two clusterings can be identical even if they do not have the same labels.

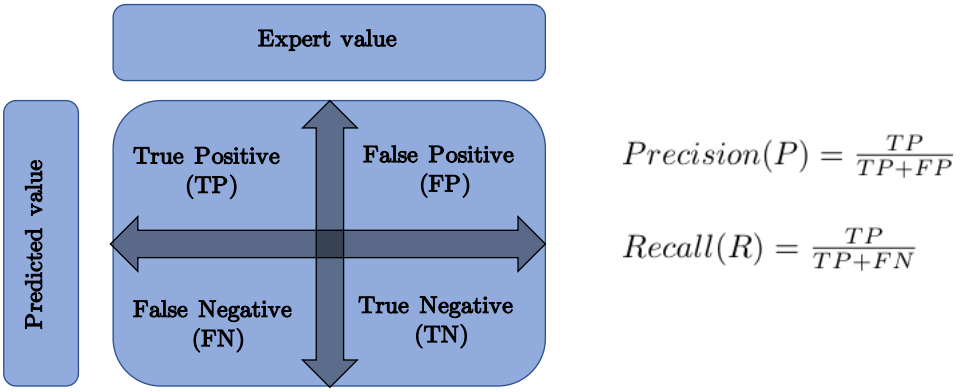


FIGURE 3.10: Confusion matrix

To cope with these issues and being able to apply classical supervised criterion, the confusion matrix can be created on pairs of data indicating whether or not two elements are in the same cluster. For example, a True Positive case will mean that two elements which are in the same cluster are also in the same compartment defined by the expert.

Now that a supervised criterion can be selected. We decide to take the F-measure which allows to assess how well the clustering is reflecting previously defined classes of data points [133]. It combines "*precision*" (P) and "*recall*"(R) as follows:

$$F_\beta = (1 + \beta^2).\frac{P.R}{P + R} \tag{3.10}$$

We calculate the F-measure with $\beta = 5$, also called F5 measure, using as target classes the annotation areas, e.g. "Mitophagy" or "Glycolysis", available in the PD map and both versions of AlzPathway.

The F-measure evaluates the performance of clustering in recreating previously defined groups, but is not capable of indicating how well a given set of clusters captures new knowledge. Therefore, we adopt an additional evaluation criteria which should inform us on the abilities of the created clusters to discover novel knowledge.

3.6.2 Discovery-based evaluation

To evaluate the discovery potential of a given clustering solution we performed an enrichment analysis for Gene Ontology [392] and Disease Ontology terms [393].

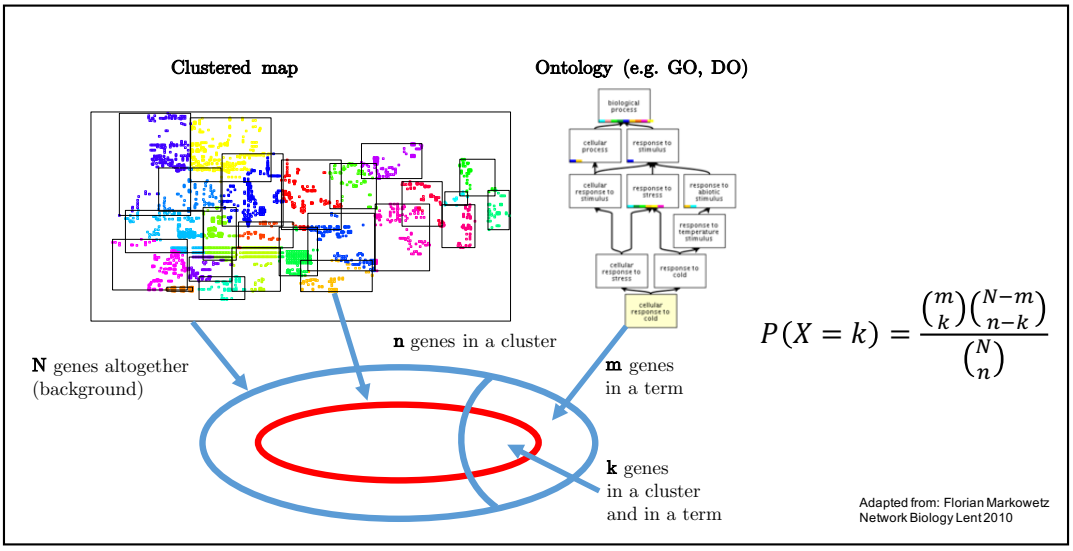


FIGURE 3.11: Enrichment analysis for a clustering

Enrichment analysis is a statistical approach and well-know procedure that is performed on gene sets. Given a set of genes, enrichment analysis will determine the ontology terms that are over-represented using the existing annotations for the given gene set.

Figure 3.11 describes how enrichment analysis is performed. It mostly relies on hyper-geometric tests. Indeed suppose that a cluster represents a sample of n genes from a total population of N genes. It is know that the considered GO/DO term contains m genes. What is the probability to have the same k genes in our cluster and in the considered GO/DO term ? By evaluating each cluster on all terms, we obtain an enrichment analysis that allows us to evaluate the over-representation of terms. This provide us additional knowledge and a measure to evaluate clusters with regards to biological functions. An adjusted p-value cutoff has been set to 0.05, 0.01 and 0.001 for enrichment analyses for both Gene and Disease Ontology.

3.7 Numerical experiments

3.7.1 Parameters

3.7.1.1 Bi-level Clustering Optimization

We considered the Non-dominated Sorting Genetic Algorithm (NSGA-II) [93], a well-known multi-objective genetic algorithm. Such algorithms proved their efficiency on various numerical and combinatorial problems. As linear exact solver, we use the IBM ILOG CPLEX Optimizer's mathematical programming technology [230] which is one of the most efficient mathematical programming solver on the market according to experiments realized in [60] on commercial and open-source solvers. Each generation of the algorithm involves standard evolutionary operators (e.g. selection, crossover and mutation. The evolutionary algorithm iterated for 300 generations in 30 independent runs in order to obtain good statistical confidence. Binary tournament was chosen as a selection method. We set the probability of a single-point crossover to 0.8, and the probability of a bit-flip mutation to $\frac{1.0}{Numberofdata}$. Concerning the CPLEX solver, no specific parameters have been selected. The stopping condition is the optimality of the solution. This is not an issue since the resulting assignment problem can be solved in polynomial time. Each of the 30 independent runs returns a set of non-dominated solutions called Pareto front. Once the 30 runs have been performed, all fronts are merged together and the F-measure is computed for each solution. Since we are only interested in solutions with different clustering sizes and the merge operation can introduce duplicates, we filtered the solutions according to the best F-measure. According to Program 3.3, different combinations of distance metrics are experimented for the distance functions: Euclidean (Eu), network (Net), Gene Ontology for Biological Process (GO BP), Gene Ontology for Molecular Function (GO MF) and Gene Ontology fo Cellular Compartment (GO CC). For the sake of simplicity and avoid rewriting the bi-level model for each distance combination, bi-level clustering experiments will be stated as follows: $d_1 > d_2$ where d_1 is the upper-level distance metric and d_2 the lower-level distance metric. For example, GO BP > Eu will describe a bi-level clustering optimization where d_1 is the GO BP metric and d_2 the Eu metric.

3.7.1.2 Comparison with Hierarchical Clustering

Hierarchical Clustering is a set of multi-level approaches to group or separate data. While classical clustering approaches discover a single partition, hierarchical clustering generates a cluster tree called "dendrogram" that represents a series of partitions from an unique cluster to as many clusters as data. There are two types of hierarchical clustering, Divisive and Agglomerative. While the divisive approach is a top-down procedure starting from a cluster containing all data, the agglomerative approach is a bottom-up alternative that consider first each data as a cluster. In order to compare the results obtained after solving the bi-level clustering optimization model using the proposed hybrid and parallel evolutionary algorithm, the agglomerative approach using the proven Ward grouping (HCW) [370] has been performed on :

- Each distance metric **separately** (e.g. Eu, Net, GO BP, GO MF, GO CC)
- **Pairwise products** between the distance metrics normalized to the $[-1, 1]$ range (e.g. EU*Net, Eu*GO BP, Net*GO BP)

3.7.1.3 Comparison with expert-based clustering

Evaluations are also performed for annotation areas already existing on the PD map and both versions of AlzPathway. Thus it gives us a baseline for comparing expert-based organization of knowledge with different clustering approaches.

3.7.2 Computing platform

Experiments have been conducted on the High Performance Computing (HPC) platform of the University of Luxembourg [357]. The algorithm has been implemented in Python with the DEAP library [141]. Hierarchical Clustering and enrichment analysis have been realized with the R language. Tables 3.1 describes all parameters defined for the parallel and hybrid approach. In order to tackle the large scale of these maps, we decided to evaluate all individuals of the population in parallel using a master-slave mechanism.

Parameters	
Iterations	30000
Independent runs	30
Selection	Binary tournament
Crossover operator	Single-Point
Crossover probability	0.8
Mutation operator	bit flit
Mutation probability	$\frac{1}{\#elements}$
population size	100
Parallelism type	Master-slave
Parallelized method	evaluation of Individuals

TABLE 3.1: Experimental parameters

3.7.3 Results

3.7.3.1 Hierarchical clustering

We compared the quality of hierarchical clustering with Ward grouping (HCW) on the contents of the PD map and two versions of AlzPathway (the original and the reorganized). For this purpose, we applied expert-based evaluation to assess how well the clusters reflect the areas drawn in the maps to annotate groups of elements and interactions with a similar role. The results of our comparisons are illustrated in Figures 3.12 and 3.13 , with Figure 3.12 showing the particular F-measure scores for each map and distance metric. Figure 3.13 illustrates the ranking of particular distance metrics, constructed using F-measure summed for all three maps. Of three HCW with single distance functions, the Euclidean offers superior results over the other two for small cluster sets, while the network distance function is superior for larger sets. Pairwise combinations of distance metrics improve overall quality of clustering. Interestingly, Gene Ontology-based distance, while having the worst quality of clustering, when in combination with the Euclidean distance improves the quality of smaller sets of clusters. Reorganization of the content, seen in comparison of two versions of AlzPathway, has a moderate effect on the quality of the clustering, with a small improvement for cases with small number of clusters.

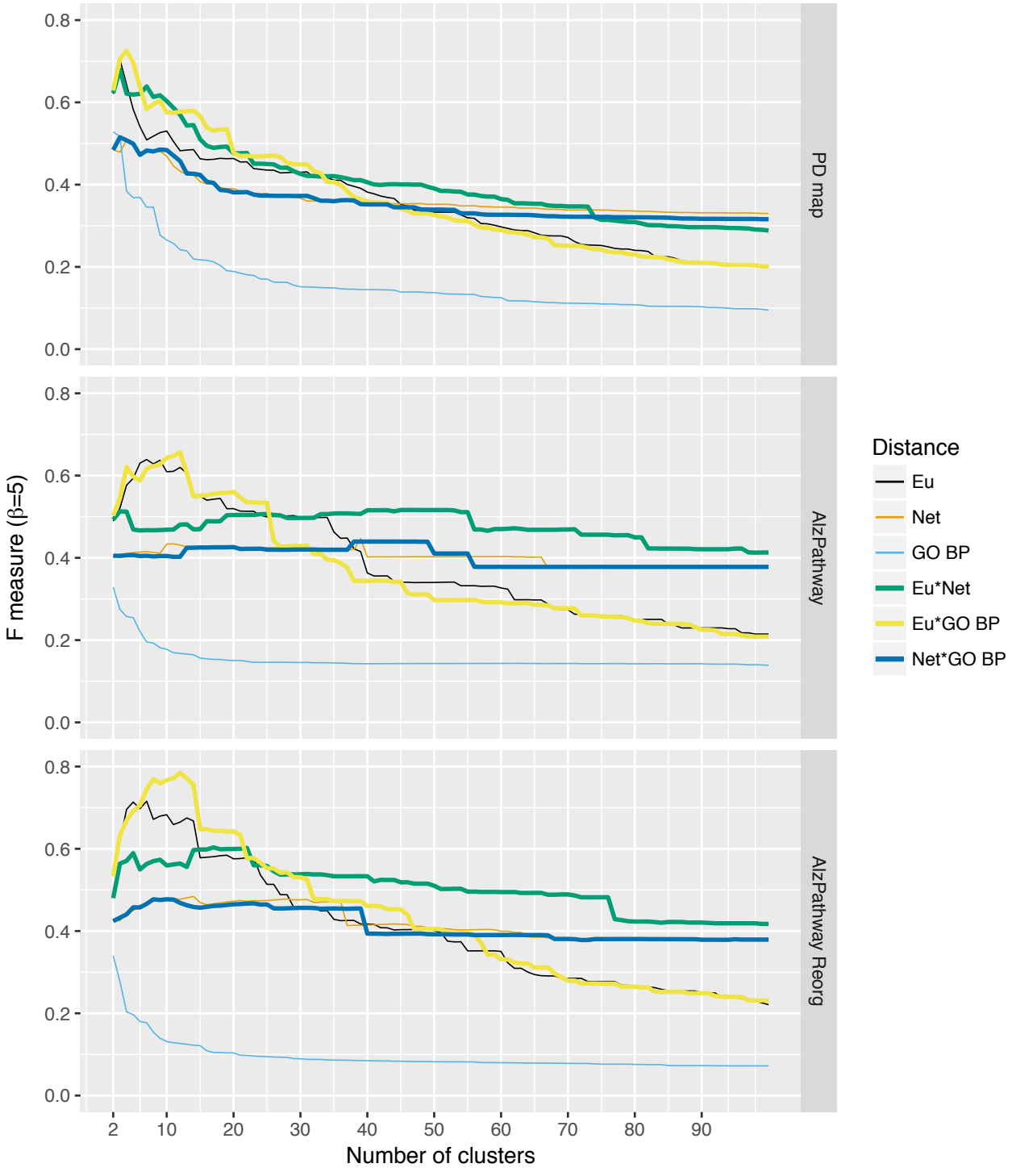


FIGURE 3.12: Hierarchical clustering (HCW) quality for different distance metrics. The values of F-measure ($\beta = 5$) for hierarchical clustering are based on different distance functions and their pairwise combinations. Eu: Euclidean distance, Net: Network distance, GO BP: Gene Ontology-based (Biological Process) distance

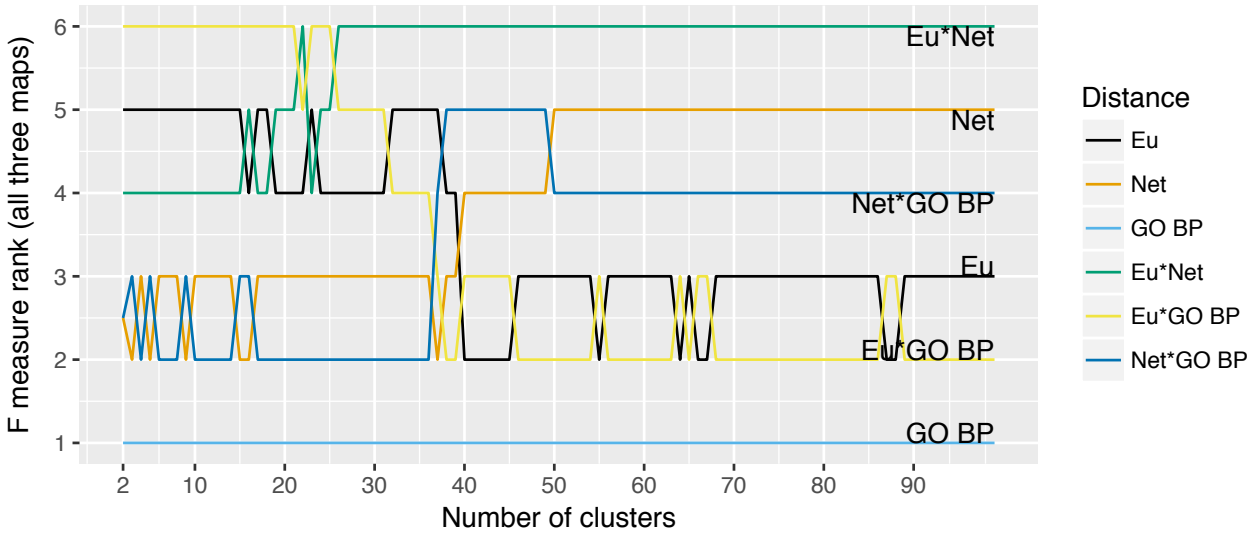


FIGURE 3.13: Ranking of different distance functions by summed F-measure

3.7.3.2 Bi-level clustering

Similarly, we calculated the F-measure for the results of bi-level clustering optimization. The results are presented in Figures 3.14 and 3.15. A comparison of the quality of different clusterings across the three maps shows grouping according to the "lower-level" distance function, with Gene Ontology-based metric being the worst-performing, and Euclidean being the best performing. As different combinations of distance functions yield varying number of clusterings, these pairings are the best observable in the PD map. For both instances of the AlzPathway there is either a small number, or no clusterings produced with GO BP as a lower-level distance metric. Reorganization of the content, seen in comparison of two versions of AlzPathway, has a bigger impact on moderate the quality of the clustering than in the case of hierarchical clustering, where both combinations of GO BP and network distance no longer yield a viable clustering.

A direct comparison of the best performing clustering schemes, as seen in Fig 3.16, shows that HCW with the combined metrics offers the best F-measure values for the solutions with small and large number of clusters. The middle part of the clustering range is covered by the bi-level clustering. All Fmeasure results are provided in Annexe B.2.

3.7.3.3 Bi-level clustering improves knowledge discovery

Next, we evaluated the impact of the bi-level clustering on discovery of new knowledge, in comparison to HCW with combined distance functions. We performed an enrichment analysis for each set of clusters generated by each solution in the three maps. Each cluster was considered as a separate group of genes. We looked for enriched terms in Gene Ontology and Disease Ontology, with the cutoff threshold for adjusted p-value=0.001. Figures 3.17 and 3.18 illustrate the results of our comparison for five best-performing approaches per map. With the same cutoff we calculated the enrichment of expert-provided annotation areas ('expert") in the considered maps as a reference point to the performance of our clustering approaches.

The majority of the proposed clustering approaches discovers more unique terms than the expert-provided annotation for larger number of clusters. Notably, for the PD map, both hierarchical (Ward) and bi-level clustering approaches discovered more terms in the Disease Ontology than expert annotation for any number

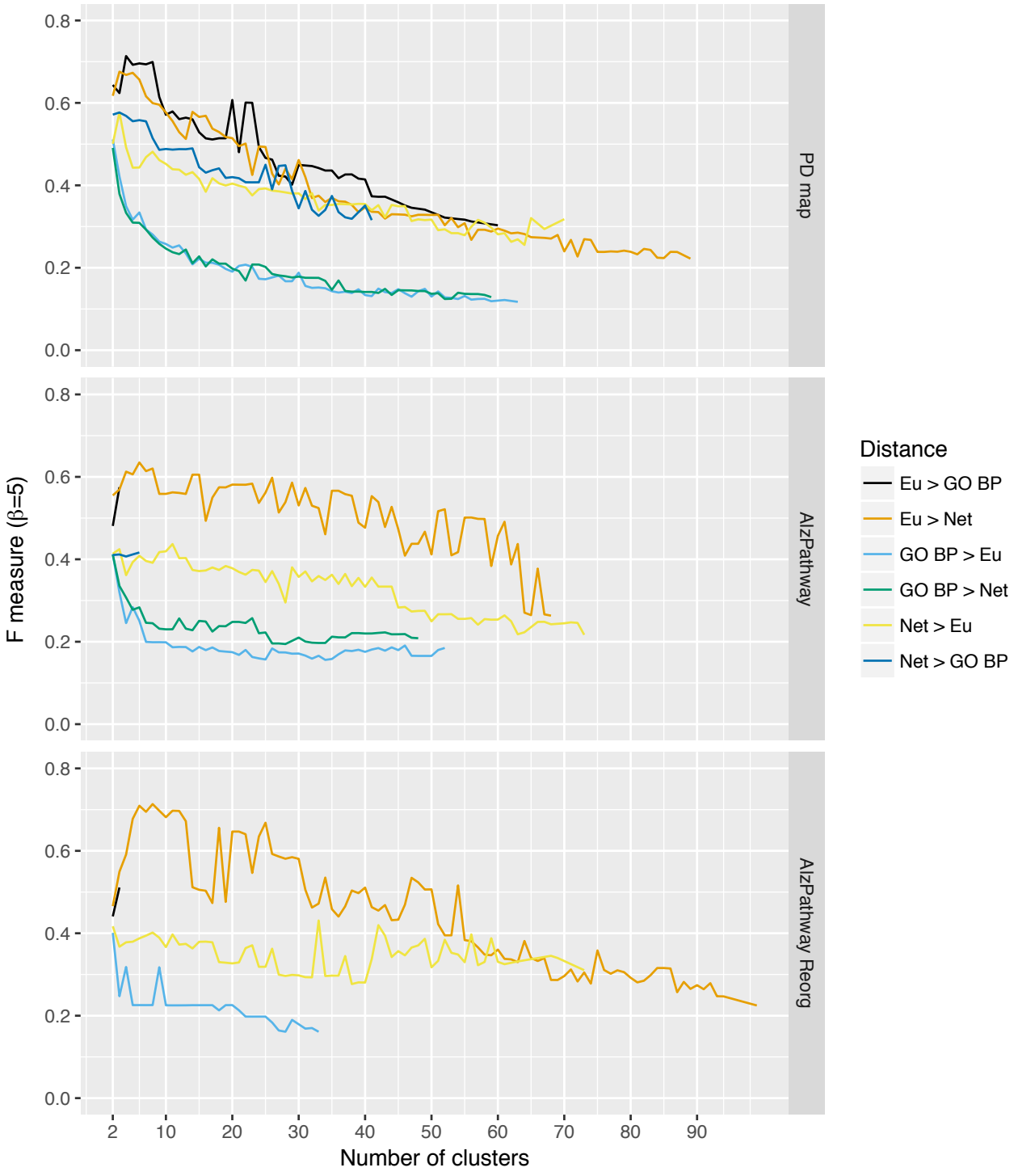


FIGURE 3.14: Bi-level clustering quality for different distance functions.The values of F-measure ($\beta = 5$) for bi-level clustering based on pairwise combinations of distance metrics, arranged as $d_1 > d_2$ distance functions, with Eu: Euclidean distance, Net: Network distance, GO BP: Gene Ontology-based (Biological Process) distance

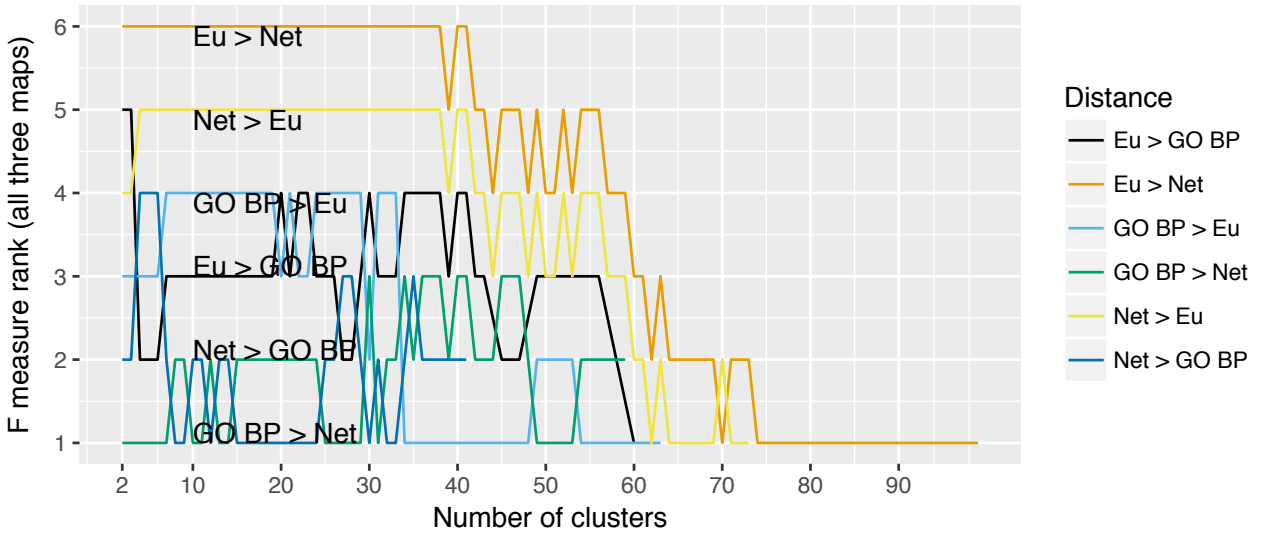


FIGURE 3.15: Ranking of the distance functions by F-measure summed across three maps

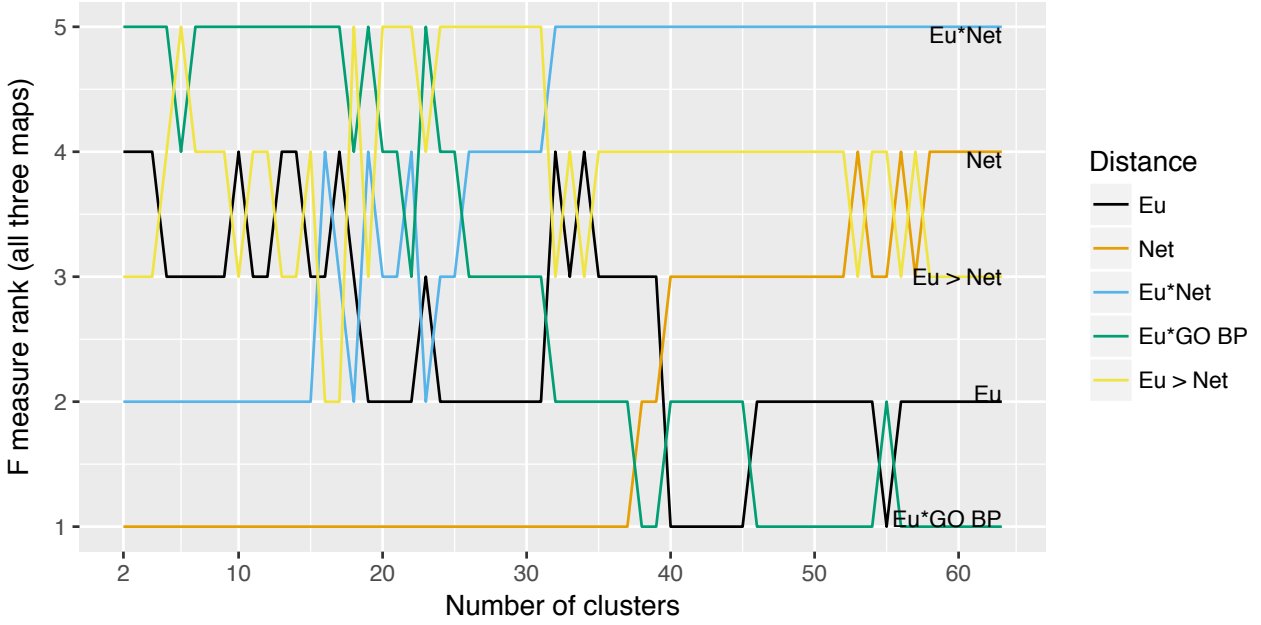


FIGURE 3.16: Ranking of HCW and Bi-level clustering approaches for selected distance metrics. A combined ranking of the best performing distance functions (for hierarchical and bi-level clustering) by F-measure summed across three maps.

of clusters (Figure 3.17). This also holds true for AlzPathway and AlzPathway Reorg, but given that only one DO term was discovered for expert annotation.

When comparing the performance of hierarchical and bi-level approaches, for larger number of clusters the bi-level clustering provides clusters enriched for more terms, both for Disease and Gene Ontology. Table 3.2 summarizes the highest scores for the selected clustering approaches. The table of complete results can be found in Annexe B.1. For the PD map and AlzPathway maps, four out of five best distance metrics are bi-level solutions.

Interestingly, the bi-level clustering provides smaller number of clustering. This is due to the criterion in the evolutionary algorithm that stops further exploration of the search space if subsequent iterations offer no gain

in the objective functions. These results may suggest, which distance functions offer better exploration of the search space and clustering properties.

When comparing AlzPathway and AlzPathway Reorg, one can notice that the restructuring of the map changed significantly the numbers of unique terms discovered, as well as ordering of the best performing combinations of metrics. However, bi-level clustering GO BP > Eu and GO BP > Net remained relatively stable with their amounts of discovered terms. Interestingly, the reorganization moderately reduced the amount of Disease Ontology terms, while significantly increasing the amount of Gene Ontology discovered terms.

We performed the enrichment analysis for higher adjusted p-value cutoffs : $p - adj < 0.05$ and $p - adj < 0.1$ (data not shown). We observed that the numbers of enriched terms for all clustering solutions as well as the expert-based one converge to the same levels.

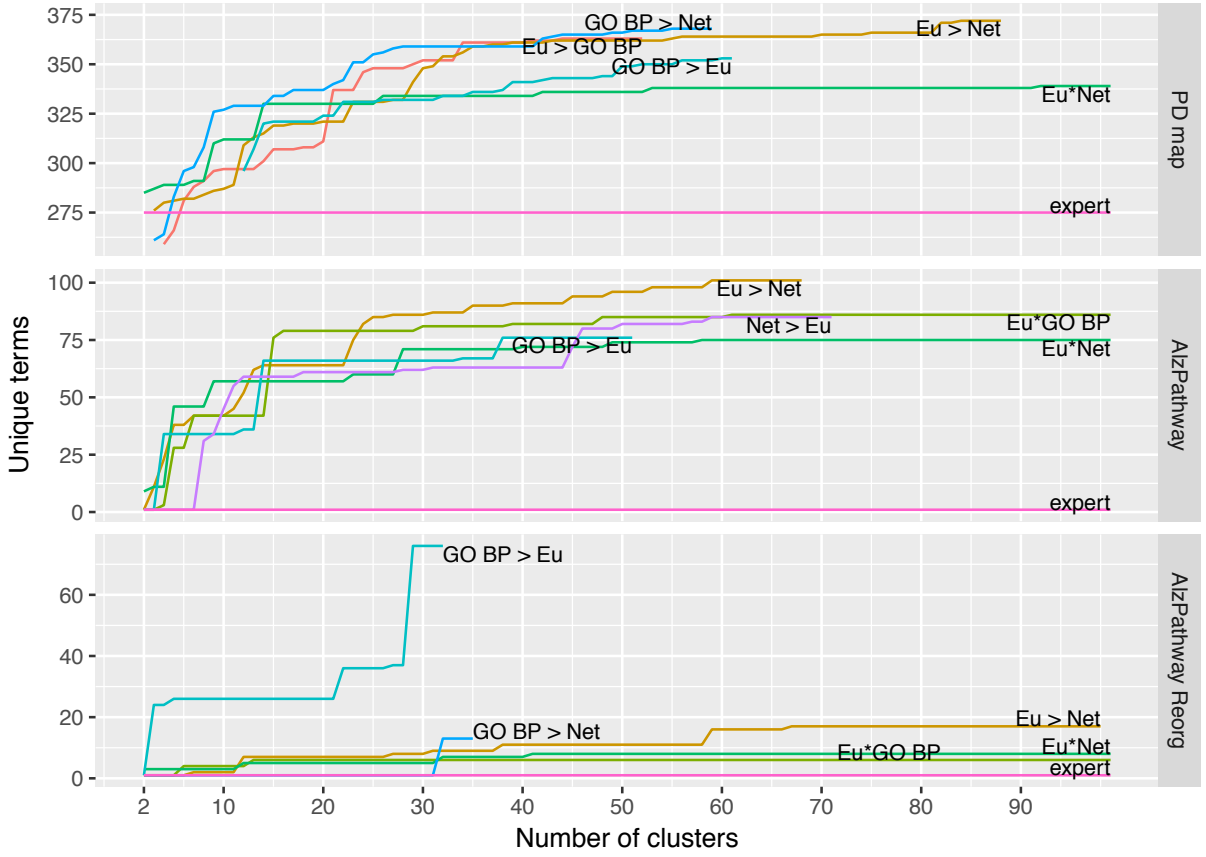


FIGURE 3.17: Discovered Disease Ontology terms by best performing bi-level and hierarchical clustering approaches. The curves represent the cumulative amount of unique terms enriched in all clusters in a given clustering. The adjusted p-value= 0.001 was used as a cutoff threshold for the significance of an enriched term. For bi-level clustering, the distance functions are arranged $d_1 > d_2$, with Euclidean: Euclidean distance, Net: Network distance, GO: Gene Ontology-based (Biological Process) distance

3.7.3.4 Examples of the discovered clusters

Here, we discuss two examples of clustering results. Both examples come from bi-level clustering of the contents of the Parkinson's disease map. Even though these distance pairs did not score high F-measures, their results reflect properly the content of the map and reveal new knowledge. To additionally validate the content of the clusters, we compared their content with the transcriptome of the brain area specific to Parkinson's disease - the substantia nigra [149].

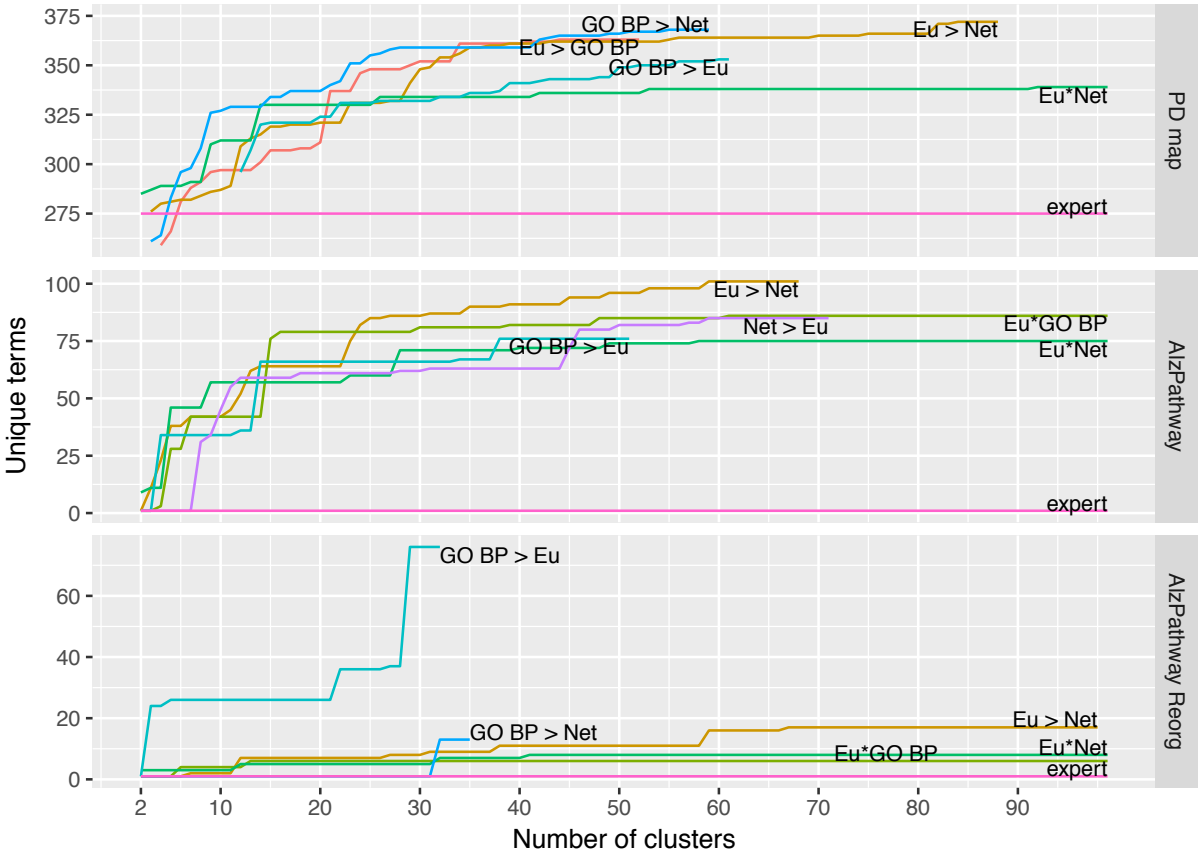
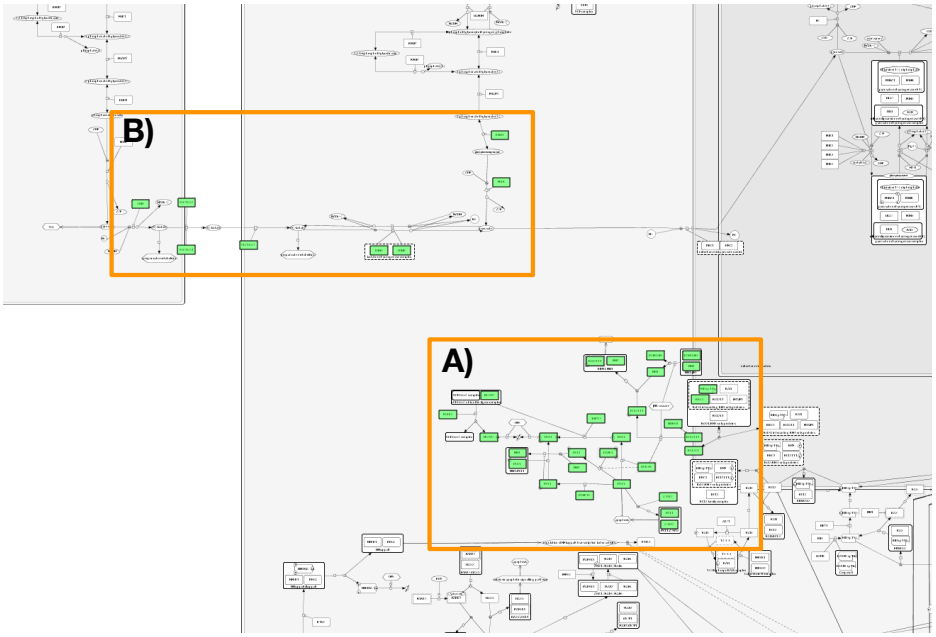


FIGURE 3.18: Discovered Gene Ontology terms by best performing bi-level and hierarchical clustering approaches. The curves represent the cumulative amount of unique terms enriched in all clusters in a given clustering. The adjusted p-value= 0.001 was used as a cutoff threshold for the significance of an enriched term. For bi-level clustering, the distance functions are arranged as $d_1 > d_2$, with Eu: Euclidean distance, Net: Network distance, GO: Gene Ontology-based (Biological Process) distance



Elements belonging to the cluster (unique names, 38 total):
Box A) ATG12, BBC3, BCL2L1, BCL2L11, BID (p15), FBXW7, GSK3B, MAPK1, MAPK10, MAPK8IP3, MCL1, NDRG1, PARK2, PIN1, TRIM17
Box B) ENO1, LDH, LDHA, LDHB, PKLR, SLC16A1, SLC16A4, SLC16A7

FIGURE 3.19: Bi-level clustering optimization for Eu > Net configuration

TABLE 3.2: Number of unique terms enriched for different clusterings. Best values for each map are marked in bold.

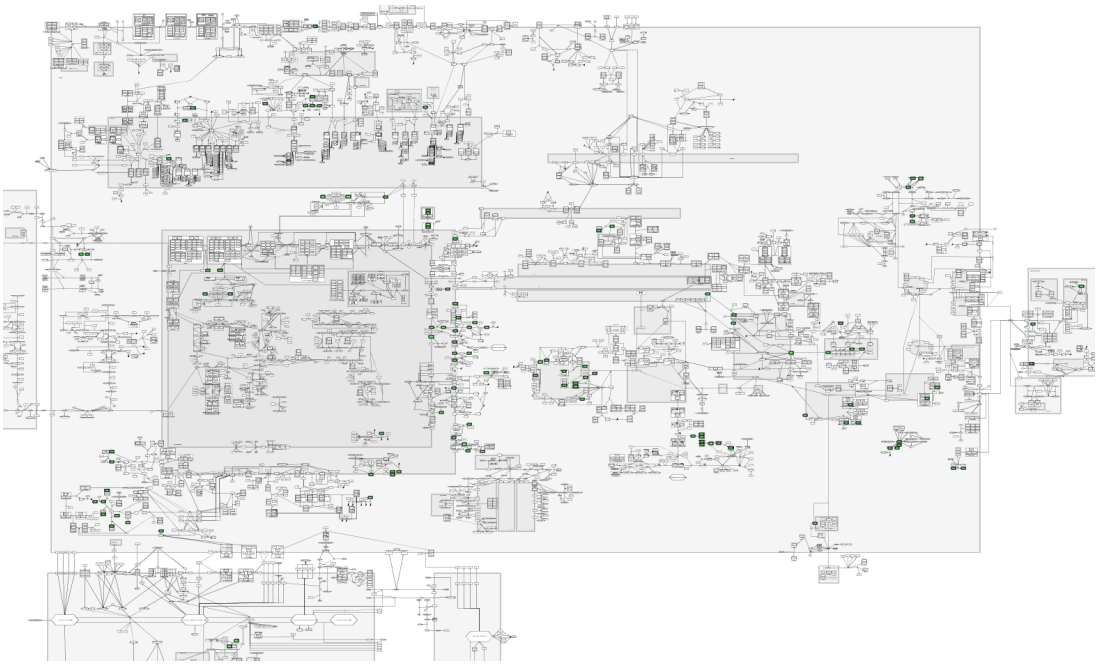
Enrichment with	PD map		AlzPathway		AlzPathway Reorg	
	DO clusters/terms	GO clusters/terms	DO clusters/terms	GO clusters/terms	DO clusters/terms	GO clusters/terms
Expert-based	36 / 275	36 / 1449	20 / 1	20 / 43	20 / 1	20 / 70
GO BP >Eu	**61 / 353**	**61 / 2203**	**51 / 76**	**51 / 196**	**32 / 76**	**32 / 720**
GO BP >Net	**59 / 368**	**59 / 2211**	48 / 72	**48 / 196**	**35 / 13**	**35 / 409**
Eu >GO BP	**52 / 363**	**52 / 1860**	3 / 5	3 / 19	3 / 1	3/1
Eu >Net	**88 / 372**	**88 / 1929**	**68 / 101**	**68 / 238**	**98 / 17**	**98 / 18**
Net >Eu	67 / 158	67 / 1463	**71 / 85**	**71 / 343**	65 / 1	**65 / 23**
Eu*Net	**93 / 339**	98 / 1641	**58 / 75**	**90 / 201**	**41 / 8**	2 / 1
Eu*GO BP	89 / 334	**97 / 1669**	**61 / 86**	97 / 179	**13 / 6**	90 / 14
Net*GO BP	81 / 289	86 / 1563	49 / 47	55 / 182	2 / 1	**97 / 136**

Example of Figure 3.19 is based on Euclidean-Network distances, scoring the highest for enrichment of the Disease Ontology terms. The cluster contains elements classified by experts as "Apoptosis" (Box A), but also elements that by the original classification of the PD map belong to the "Glycolysis" area (Box B). Interestingly, elements of Box B are known regulators of apoptosis in various contexts, including the neuronal environment with ENO1 [386] and SLC16A4 [243], and different types of cancer [1, 16, 140]. This can be considered as a novel regrouping of the content in the PD map, which would be difficult to discover optically, as the network distance between the elements of Box A and B cannot be immediately discerned by eye. When compared to the Parkinson's disease transcriptome dataset, 19 out of 38 cluster elements were down-regulated, suggesting the importance of the contained mechanisms for the pathology of the disease.

Example of Figure 3.20 is based on Gene Ontology-Network distances, scoring the highest for enrichment of the Gene Ontology terms. When this cluster is displayed in the Parkinson's disease map, it becomes evident that Euclidean distance was not used for its construction, as its elements are dispersed across the map. Nevertheless, the majority of the cluster contents are connected to the processes of response to oxidative stress and maintenance of mitochondrial homeostasis. There are, however, a number of elements that extend this picture. One of them is KCNN3, member of potassium calcium-activated channel family. Though originally curated in the map in the context of pathology of alpha-synuclein, its appearance in this cluster is supported by literature evidence [106]. Similarly, evidence supports inclusion of ATP13A2 in the mechanisms regulating oxidative stress [157]. On the other hand, the presence of GSK3A, another novel element, may be questionable. Even though its role in nerve regeneration was recently demonstrated [152], its association, together with PRKCD, may be due to the GO Biological Process annotation with cardiac myocyte function [395]. Still, when compared to the Parkinson's disease transcriptome dataset, 94 out of 117 cluster elements were down-regulated, which gives confidence in its contents and corresponds well to the fact, that reactive oxygen species play a major role in Parkinson's disease [139].

3.7.3.5 Robustness of distance metrics in the evaluated scenarios

Three classification concepts are available in Gene Ontology: Biological Process (BP), Cellular Compartment (CC) and Molecular Function (MF). Thus, the ontology-based distance calculated according to these criteria may yield different results and, potentially, has different impact on the clustering results. Our metric of choice was Biological Process, as conceptually the closest to the nature of disease maps, describing processes of health and disease. To clarify the potential impact of the remaining concepts on the clustering quality, we compared clustering quality and enrichment of both hierarchical and bi-level approaches for all three. Annexes B.1, B.2, B.3 and B.4 contain all the results of this comparison.



Elements belonging to the cluster (unique names, 117 total):
ATP13A2, BAG5, FBXW7, GBA, GSK3A, HAX1, KCNN3, MUL1, PARK7, PARL, PINK1, PRKCD, TOMM6, TXN, VPS35

FIGURE 3.20: Bi-level clustering optimization for GO BP > Net configuration

F-measure values for hierarchical clustering are similar to each other, with GO BP having the highest impact on the clustering of the PD map, and GO CC on the AlzPathway Reorg. Nevertheless, this effect is rather moderate. Interestingly, the bi-level clustering results indicate that PD map and AlzPathway (original) could benefit from GO MF as the upper-level distance metric. Still, inclusion of these results would not alter the ranking of the distance metrics.

The number of enriched terms for Disease and Gene Ontology is also the highest for the BP-based ontology distance for PD map and AlzPahway Reorg. In case of the original AlzPathway, GO CC and MF as upper-level distance metrics offer improvement in the discovered GO terms, but only for GO MF > Eu combination this improvement is significant. Overall, GO BP remains the most robust metric considered in our clustering analysis.

3.8 Conclusions

Large diagrams representing biomedical knowledge become an important part of workflows for interpretation of experimental data and generation of new hypotheses. Clustering approaches may provide a high-level overview of this complex content by grouping together similar elements. Different distance functions may be applied for this purpose. Here we investigated their impact on the clustering of the Parkinson's disease (PD map) and Alzheimer's disease (AlzPathway) maps.

In this chapter, a medoid-based clustering optimization is proposed through the decomposition of the "uncapacitated k-median problem". Even if the latter is $\mathcal{NP}$-hard, it does mean that some part of the problem cannot be exactly and efficiently solved. We have shown that the two-level clustering problem obtained after decomposition possesses an inner level playing the role of an assignment problem. Indeed, the decomposition

separates the medoid selection from the assignment of elements to the medoids (clusters). We also discussed the relation between this two-level and bi-level problems. We conclude that, by opting for different distance metrics at both level, a bi-level clustering problem is obtained. Additionally, we demonstrated the utility of combining different distance metrics to meaningfully cluster the contents of a complex visual repository on human disease (e.g. Parkinson disease and Alzheimer map). We proposed a bi-level clustering approach as a solution for combining two distance functions and exploring their relationship.

In order to investigate the relationships between different distance functions we performed a bi-level clustering for the pairwise combinations of different distance metrics (e.g. Euclidean, network distance and ontology-based). The results are clearly grouped by the lower-level distance metric, with the Euclidean distance scoring the highest. Additionally, because of the stopping criterion in the evolutionary algorithm, the upper-level Gene Ontology-distance provides smaller sets of clusters. This is understandable, as the Gene Ontology-based distance describes the conceptual similarity between the contents of the map and has no reflection of the actual structure of the diagram. In turn, the expert-based annotations reflect visual areas of the PD map. Therefore, Gene Ontology-based distance will not perform well to define meaningful cluster medoids in the PD map.

We also evaluated the impact of combined distance functions on knowledge discovery in the maps. For each set of clusters from both HCW and bi-level clustering, we performed an enrichment analysis for Disease Ontology and Gene Ontology terms. Our results showed that the number of unique terms for both ontologies grows with growing size of cluster sets, and surpasses the expert-provided annotation areas. Notably, for the cluster set size = 36 (the number of expert-provided areas) all selected clustering solutions - one hierarchical (ward) and four bi-level clusterings - provide more unique terms for both ontologies. This result, in combination with F-measure results, suggests that the clusters may offer an improvement to the existing annotation of the maps.

Bi-level clustering in direct comparison with HCW produces cluster sets with the higher number of enriched terms. It is worth noticing that HCW based on combination of Eu*Net distance functions performs better than Net > Euclidean bi-level clustering, but worse than Eu > Net clustering. This suggests a hierarchy of distance functions, and their importance in the exploration of the PD map.

The proposed bi-level clustering optimization problem can be considered as a “weak” bi-level problem since the lower-level problem is polynomial. This is the reason why there was no restrictions to consider a nested metaheuristic that iteratively solve both levels. Nevertheless, many bi-level optimization problems have $\mathcal{NP}$-hard levels which implies that nested optimization becomes unsuitable. The next chapter will be dedicated to the resolution of general bi-level optimization problems. The mainstream will be to propose a viable alternative to the state-of-the art algorithms that minimize the time-consuming evaluation of the lower-level optimizations.

Chapter 4

A Surrogate-based approach to tackle General Bi-level Problems

Contents

4.1 Introduction

In chapter 2, theoretical complexity of bi-level optimization problems has been discussed in details. Numerous investigations have shown that these problems are $\mathcal{NP}$-hard even if both levels are separately convex. We recall that the upper-level problem is constrained by the lower-level problem. Feasibility of the upper-level problem, and thus bi-level feasibility, is strongly dependent on the optimality of the lower-level problem.

In chapter 3, we pointed out that the bi-level formulation of the uncapacitated k-median problem can lead to a novel clustering model where different metrics at each level can help to capture more knowledge by taking into account the hierarchy that exists between these metrics. We have shown that the lower-level problem of this bi-level clustering model is similar to an assignment problem once cluster representatives are set. In turn, such a formulation can be solved with nested optimization approaches'. For this purpose, we took advantages of the properties of the uncapacitated k-median problem to decompose it into two-levels with different complexity.

This allowed us to solve exactly and efficiently lower-level instances. Nonetheless, the global complexity of a bi-level problem is as hard as its strongest sub-problems (level). This means that it remains $\mathcal{NP}$-hard even if it can be considered as a "weak " bi-level problem.

Now we are interesting to more general problems in which the lower-level problem is not necessarily solvable in polynomial time and no advantageous properties can guide us to reformulate them. The last decade has seen a renewed interest for bi-level optimization, especially in the field of evolutionary computing. This is probably due to the new optimization needs to cope with problems having multiple decision makers. As a matter of fact, we can clearly observe that standard evolutionary computing could be challenged by "strong" bi-level problems, i.e., where both levels are $\mathcal{NP}$-hard. First, bi-level feasibility implies lower-level optimality which cannot be guaranty which such approaches. Then, the computations time can be repellent in case of population-based approaches. Indeed, each upper-level solution would require a lower-level optimization. The number of lower-level function evaluations would be definitely voluminous.

In chapter 2 and section 2.5, a state-of-the-art on bi-level metaheuristics has been provided. Among the numerous works that can be listed in the literature, some interesting approaches attempted to approximate lower-level decision variables in order to decrease dramatically the number of evaluations. The reader may refer to section 2.5.5 to obtain an overview of them.

Among the listed approaches, the Bilevel Evolutionary Algorithm based on Quadratic Approximations (BLEAQ) is a reference in evolutionary bi-level optimization due to its capacity to tackle not only single but multi-objective bi-level problems. Hereafter, we are only interested to the single objective bi-level problems that are still a real challenge for decision makers. Sinha *et al.* in [332] developed the BLEAQ algorithm in order to approximate the lower-level optimal solutions using the upper-level ones while reducing the number of lower-level optimizations. Although BLEAQ is very innovative, a genetic algorithm is still employed at upper-level which necessitates numerous of fitness evaluations. In addition, the authors are required to perform multi-output approximations of the lower-level decision variables. To tackle this issue, they considered m independent single-output regressions by building as many approximate functions as lower-level variables.

The drawbacks of BLEAQ lie in the fact that it requires the creation of multiple and independent "surrogate functions" to estimate each lower-level decision variable which is not suitable for large-scale lower-level problems. In addition, the independence assumption between the surrogate functions is not reflecting the existing links between the lower-level variables. BLEAQ restricts the surrogates to quadratic approximation functions while the mapping between upper-level and lower-level variables can be highly not convex.

In this work, we propose to tackle these issues by considering Bayesian optimization. Instead of generating several approximate functions for each lower-level variable, we make use of gaussian processes to predict directly the upper-level fitness value. Bayesian optimization also embeds an acquisition function which permits to determine the most promising search areas during optimization. Therefore, this approach attempts to minimize the number of fitness evaluations by nature. Generally in the literature, acquisition functions are optimized using local search approaches, i.e., L-BFGS-B [279] which tend to be trapped in local optimum. This is the reason why we propose to use an evolutionary algorithm and more precisely "Differential evolution" to detect better acquisition points and thus better search areas. Bayesian optimization stemmed from the need to optimize black-box problems with expensive objective functions. It has been categorized as a hyper-parameter optimization algorithm and has been widely employed for the optimization of parameters in Machine Learning models.

The remainder of this chapter is organized as follows. Section 2 introduces bayesian optimization. Section 3 formalizes the problem and then states explicitly how Bayesian optimization can be adapted to solve bi-level

problems. General bi-level benchmarks are introduced in section 4 while section 5 focuses on the experimentation protocol and parameters. Section 6 provides comparison results with a state-of-the-art algorithm, namely the Bi-level Evolutionary Algorithm based on Quadratic approximations (BLEAQ). Finally, section 7 concludes this work and propose future investigations.

4.2 Bayesian optimization (BO)

Since discovering $\mathcal{IR}$ could be a hard task, it is therefore complex to evaluate the upper-level objective function on it.

Bayesian Optimization algorithm is a Kriging approach [87] that relies on Gaussian Processes to generate a surrogate function approximating the true objective function of a complex problem. A initial set of random points are chosen in the feasible space and are evaluated with the true objective function. Gaussian processes are then fitted to them to obtain a first estimation of this true objective function. From that point, the algorithm will consider the surrogate function to start the search by determining a next and promising point that could be sampled. For this purpose, an auxiliary function, i.e., the acquisition function is build and then solved. One could wonder why the next point is not directly determined by optimizing directly the surrogate function. The answer lies in the fact that the surrogate function is only a rough estimation that does not capture all the topology of true objective function. The acquisition function adjusts the surrogate function to maximize the chance to discover a new promising point. The bayesian algorithm converges once the acquisition function is unable to generate a new improving point given a certain tolerance threshold.

Bayesian optimization has been mostly employed in machine learning [78, 308] as an alternative to the grid search and random search algorithms. They are well-suited for multi-level optimization problems as they take into account the evaluation cost. Bayesian optimization is a model-based approach which aims at solving problems with time consuming evaluation functions. It is often assimilated as a black-box optimization algorithm where the formal expression of the objective function may be unknown or very difficult to obtain. To overcome this issue and reduce computation cost, bayesian optimization generates a surrogate model of the unknown function using gaussian processes[46]. It samples promising zones in the feasible region by computing a distribution of the objective function. This distribution gives us a prior knowledge on location of the optimal solution. Bayesian optimization is thus characterized by two important mechanisms:

- A probability measure on F describing our prior beliefs on it ;
- The acquisition function which allows to gain information on the location of the minimum value of the objective function.

Considering a cost function $F(x)$, gaussian processes determine the probability distribution of the function $F(x)$ at each x. These distribution are Gaussian and thus characterized by a mean value $\mu(x)$ and a variance $\sigma^2(x)$. Hence a probability distribution over functions can be defined as follows:

$$P(F(\mathbf{x})|\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})) \tag{4.1}$$

Obviously, the parameters $\mu(x)$ and $\sigma^2(x)$ have to be estimated. This is done by fitting the Gaussian processes to the data. Using several observations, we obtain a sample of a multivariate Gaussian distribution [42], determined by a mean vector and a covariance matrix. In fact, Gaussian processes generalize the notion of

multivariate Gaussian distribution. For complex non-linear functions, the covariance matrix can be defined using a kernel function $k(x, x')$. This covariance matrix defines the correlation between data. Two distant data x and x' should not influence each other while two close data are strongly correlated.

$$F(x) \sim \mathcal{GP}(\mu(x), k(x, x')) \tag{4.2}$$

where $\mathcal{GP}$ stands for Gaussian processes. The squared exponential kernel is often used and defined as follows:

$$k(x, x') = l \cdot \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \text{with parameters } l \text{ and } \sigma^2 \tag{4.3}$$

To fit the Gaussian process to the data, the likelihood is optimized from the evaluations of each observations. Each time a new point is added to the model, a re-optimization is performed to maximize the likelihood. The question is now :'How should we determine a new point ?'. This is achieved by optimizing an acquisition function which statistically models our confidence to find the location of the optimal value. Several acquisition functions exist such as the Maximum Probability of Improvement (MPI), the Expected Improvement (EI), or the Lower-Confidence Bounds (LCB) and are computed as follows:

- $\text{acq}_{MPI}(x) = \Phi(\gamma(x))$.
- $\text{acq}_{EI}(x) = \sigma(x)(\gamma(x)\,\Phi(\gamma(x)) + \phi(\gamma(x)))$.
- $\text{acq}_{LCB}(x) = \mu(x) - k\sigma(x)$.

where $\gamma(x) = \frac{F(x_{best} - \mu(x))}{\sigma(x)}$, ϕ is the standard cumulative distribution function, ϕ the standard normal probability density function and k is a parameter allowing to balance exploration-exploitation.

Finally, Algorithm 4 depicts the different steps of the standard bayesian optimization algorithm.

Algorithm 4 Bayesian Optimization

```
1: X =initRandom(n);
2: Y =problem.evaluate(X)
3: model=GP(X, Y)
4: model.update()
5: while not has_converged() do
6:    acq = getAcquisition(k);
7:    x_new =acq.optimize();
8:    y_new =problem.evaluate(x_new);
9:    model.update(x_new,y_new);
10: end while
11: return model.best;
```

4.3 Adaption to general bi-level problems

4.3.1 The upper-level objective function as a black-blox

The contribution of this work is two-fold. First, we adapt the bayesian algorithm (see algorithm 4 to bi-level problems). And finally, we improve the optimization of the acquisition function by considering a differential evolution algorithm. As discussed in the introduction, it is very time consuming to adopt a nested strategy

approach which sequentially solves both levels. In [332], the authors of BLEAQ focused on localization of $\mathcal{IR}$. Under the assumption that $\Psi(x) = \{\hat{y}\}$, i.e., singleton, for each x, they tried to establish a relation between x and $\hat{y}$ to avoid repetitive lower-level optimizations. They assume that under Lipschian continuity 2 close solutions x and x' will lead to close rational solutions $\hat{y}$ and $\hat{y}'$. Consequently, they attempt to estimate $\hat{y}$ from x. Unfortunately, to generate these approximate functions, they have to deal with multi-output regression to determine m functions minimizing the approximation error. In addition, these approximate functions are restricted to be quadratic functions. Contrary to BLEAQ, we suggest to estimate the upper-level objective function $F(x, \hat{y})$ instead of all $\hat{y}$ separately. Under the same assumption that $\Psi(x)$ is a singleton, the optimal solution $\hat{y}$ is unique and we can conclude that it may exist a surrogate function $F' : \mathbf{R}^n \mapsto \mathbf{R}$ with $F'(x) = F(x, \hat{y})$. Notice that if $\Psi(x)$ is not a singleton, two alternatives exist:

- The optimist case: we choose $\hat{y} = \arg\min\{F(x, y) : y \in \Psi(x)\}$.
- The pessimist case: we choose $\hat{y} = \arg\max\{F(x, y) : y \in \Psi(x)\}$.

According to this new representation, we only focus on x. By doing so, we obtain a black-box function and we do not need to deal explicitly with the lower-level decision variables y at the UL. Therefore, it allows us to decouple x and y implying that a complex model is replaced by a function modeling its effects. Gaussian processes require no assumption to approximate the true objective function contrary to the work proposed in [332] which only uses quadratic approximations.

Bi-level problems are evaluated in two-steps in order to compute the upper-level function $F(x, y)$. Indeed, the upper-level decision maker has no control on y. He can only observe the lower-level rational decision $\hat{y} \in \Psi(x)$ which strongly depends on x. In some ways, $F(x, y)$ can be rewritten by $F(x, \hat{y})$. The upper-level objective function F is in a way a function that only dependents on x. So we will now consider $F'(x) = F(x, \hat{y} = \arg\min\{F(x, y) : y \in \Psi(x)\}$ in the singleton or the optimistic case while the pessimistic case would be $F'(x) = F(x, \hat{y} = \arg\max\{F(x, y) : y \in \Psi(x)\}$. Figure 4.1 depicts a surrogate function obtained from the example described in Figure 4.2. The updated model allows us to plot at each upper-level decision variables x, the mean fitness value and a confidence interval around the mean value . We can distinguishes two types of upper-level solutions x:

- The first one represents the upper-level solutions which have been evaluated to compute $F'(x)$ according to the lower-level problem. This is the reason why the variance at these points is null. They have already been selected by the acquisition solver during the optimization.
- The second one represents the upper-level solutions which have not been explicitly evaluated. This is the reason why we only have a confidence interval around $F'(x)$. They are potential points to be selected by the acquisition solver.

In order to evaluate explicitly an upper-level solution x and obtain $F'(x)$, we have to perform a lower-level optimization to determine $\hat{y}$. This optimization is realized using the Sequential Least Squares Programming (SLSQP) algorithm developed by Kraft [294]. SLSQP is a gradient-based procedure for non-linear optimization problems supporting inequality and equality constraints. This algorithm requires a initial guess to start the optimization. Instead of using a random initial guess, we select as starting point the lower-level optimal solution obtained by the closest upper-level solution x'. Figure 4.2 depicts this situation where the lower-level problem is optimized according to the upper-level decision $x = 3$. The upper-level decision $x' = 2$ is the closest that has been explicitly evaluated so far. The lower-level rational decision set according to x' is $\Psi(2) = \{3\}$. This strategy is based on the idea that two close upper-level decision variables should have closed optimal lower-level solutions. In this case, the number of evaluations required by the SLSQP algorithm is drastically reduced.

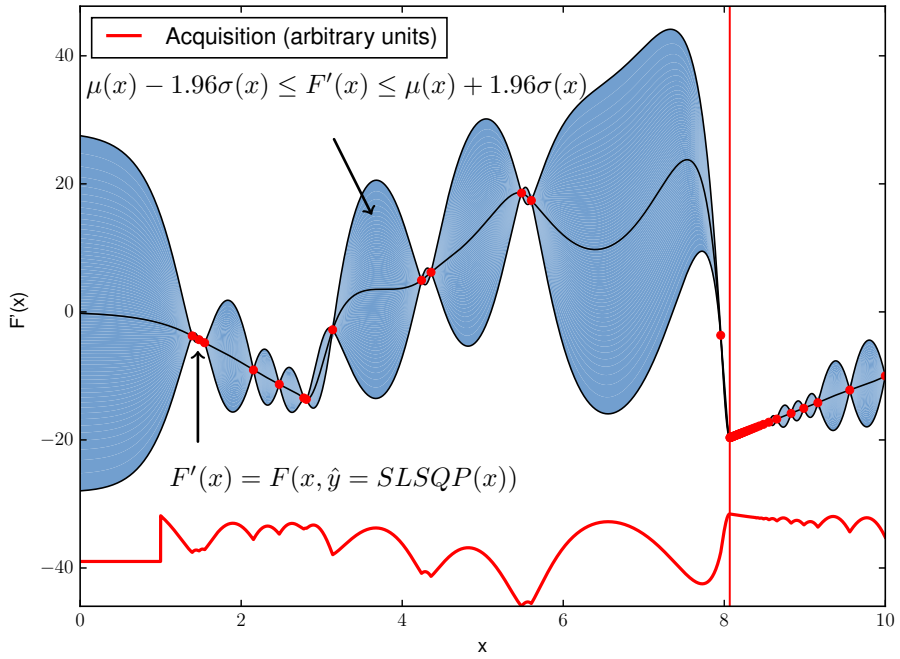


FIGURE 4.1: Surrogate function and its acquisition function (in red)

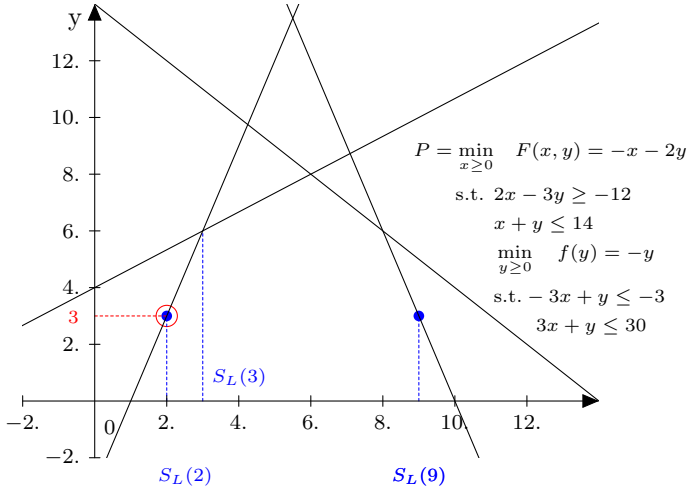


FIGURE 4.2: Select initial guess to solve $P(3) = \{\hat{y} \in Y : \hat{y} \in \arg\min[f(3, y) : y \in S_L(3)]\}$

4.3.2 Improving the acquisition function using differential evolution

As described in the previous section, bayesian optimization is well-adapted to problem with time-consuming function evaluations. For instance, it has been used with success to optimize machine learning algorithms[368]. The main advantage is a clever search in order to limit the number of evaluations contrary to some other metaheuristics. As its name suggests it, the acquisition function allows to find the next point to be evaluated according to some information gain obtained from the previous iterations. This is clearly a very efficient way of learning promising areas of the true objective function. The acquisition function should be globally optimized since it is generally a function with multiple local maxima. Therefore and contrary to most of the Bayesian optimization implementation we have seen so far, we considered a differential evolutionary algorithm to search its global optimal solution becoming the next acquisition point. Such global algorithms have been successfully applied inside Bayesian optimization algorithms in [244] . Since we need to spare a maximum number of function evaluations, it is really important to find the most promising areas. Different strategies can be considered:

- Select the best acquisition point
- Select a set of promising acquisition point

In the second case, the number of function evaluations will increase but we are more likely to escape local optimal solutions. In this work, we only test the first case where only a single acquisition point is selected. Future investigations will focus on different strategies of optimizing the acquisition function.

In this work, the LCB acquisition function will be preferred because it allows to detect the most promising solutions in terms of mean and deviation. Large k values put the emphasis on exploration while small k values focus on the best expected performance.

4.4 General Bi-level Benchmarks

As general bi-level problems, we considered the benchmarks introduced by Sinha et al in [332]. Table 4.1 describes the first five benchmarks, i.e., TP1 - TP5 while Table 4.2 presents the remaining five benchmarks going from TP6 to TP10. Both tables detail the mathematical formulation of each bi-level problems as well as their size and best known solutions. Although these benchmarks have small size in terms of number of upper-level and lower-level variables, their structure and properties make them difficult to solve using classical bi-level approaches. To the best of our knowledge, they are the only reported benchmarks with associated best known solutions found in the literature. They embeds linear and non-linear objective functions and constraints at both levels. Therefore, we can qualified them as general bi-level problems. One exception can be noticed: they take only into account continuous decision variables. This will be the subject of a discussion at the end of this chapter and constitutes one of the main drawback of the approaches based on the approximation of the lower-level problem.

4.5 Experimental setups

4.5.1 Comparison with BLEAQ on Bi-level Benchmarks

The 10 bi-levels problems, previously introduced and proposed in [332], have been selected to evaluate the potential of bayesian optimization to solve bi-level problems. This set of benchmarks including linear and non-linear levels will afford us to compare the Bayesian approach with the BLEAQ algorithm.

BLEAQ is an evolutionary algorithm that focuses on quadratic approximations of the set-value mapping Ψ: $\mathbf{R}^n \rightarrow \mathbf{R}^m$ that characterize bi-level problems (see Figure 4.3). As starting point, BLEAQ creates an initial and random population of upper-level individuals that encodes the original upper-level variables. For the first generation, a full and global lower-level optimization using a second evolutionary algorithm is undertaken for all individual of the initial population. Once all lower-level solutions have been computed, a multiple quadratic approximation models between the upper-level variables and each lower-level variables are established. Depending on whether or not the approximation error is below a predefined threshold, the model is used to predict further optimal lower-level variables for any new upper-level variables, i.e., any new generated individual. This reduces the number of call to the second evolutionary algorithm to determine the new lower-level optimal variables. However the authors of BLEAQ reported that caution should be taken while accepting solutions from the approximation model. Indeed, poor solutions might drive the algorithm to a wrong bi-level optimal

TABLE 4.1: Benchmark problems TP1-TP5 with x as upper-level variables and y as lower-level variables

Problem	**Formulation**	**Best known sol. value**
TP1 n=2 m=2	$\min_x \quad (x_1-30)^2+(x_2-20)^2-20y_1+20y_2$ s.t. $\quad x_1+2x_2 \geq 30$ $\quad x_1+x_2 \leq 25$ $\quad x_2 \leq 15$ $\min_y (x_1-y_1)^2+(x_2-y_2)^2$ s.t. $\quad 0 \leq y_i \leq 10, \quad i=1,2$	upper= 225.0 lower= 100.0
TP2 n=2 m=2	$\min_x \quad 2x_1+2x_2-3y_1-3y_2-60$ s.t. $\quad x_1+x_2+y_1-2y_2 \leq 40$ $\quad 0 \leq x_i \leq 50 \quad i=1,2$ $\min_y (y_1-x_1+20)^2+(y_2-x_2+20)^2$ s.t. $\quad x_1-2y_1 \geq 10$ $\quad x_2-2y_2 \geq 10$ $\quad -10 \leq y_i \leq 20 \quad i=1,2$	upper= 0.0 lower= 100.0
TP3 n=2 m=2	$\min_x \quad -(x_1)^2-3(x_2)^2-4y_1+(y_2)$ s.t. $\quad (x_1)^2+2x_2 \leq 4$ $\quad x_i \geq 0 \quad i=1,2$ $\min_y 2(x_1)^2+(y_1)^2-5y_2$ s.t. $\quad (x_1)^2-2x_1+(x_2)^2-2y_1+y_2 \geq -3$ $\quad x_2+3y_1-4y_2 \geq 4$ $\quad y_i \geq 0 \quad i=1,2$	upper= -18.6787 lower= -1.0156
TP4 n=2 m=3	$\min_x \quad -8x_1-4x_2+4y_1-40y_2-4y_3$ s.t. $\quad x_i \geq 0 \quad i=1,2$ $\min_y x_1+2x_2+y_1+y_2+2y_3$ s.t. $\quad y_2+y_3-y_1 \leq 1$ $\quad 2x_1-y_1+2y_2-0.5y_3 \leq 1$ $\quad 2x_2+2y_1-y_2-0.5y_3 \leq 1$ $\quad y_i \geq 0 \quad i=1,2,3$	upper= -29.2 lower= 3.2
TP5 n=2 m=2	$\min_x \quad rt(x)x-3y_1-4y_2+0.5t(y)y$ s.t. $\quad x_i \geq 0 \quad i=1,2$ $\min_y 0.5t(y)hy-t(b(x))y-0.333y_1+y_2-2 \leq 0$ s.t. $\quad y_1-0.333y_2-2 \leq 0$ $\quad y_i \geq 0 \quad i=1,2$ where $t(.)$ denotes the transpose of vector $h=\begin{bmatrix}1 & 3\\ 3 & 10\end{bmatrix}, \quad b(x)=\begin{bmatrix}-1 & 2\\ 3 & -3\end{bmatrix}x, r=0.1$	upper= -3.6 lower= -2.0

TABLE 4.2: Benchmark problems TP6-TP10 with x as upper-level variables and y as lower-level variables

Problem	**Formulation**	**Best known sol. value**
TP6 n=1 m=2	$\min_x \quad (x_1-1)^2+2y_1-2x_1$ s.t. $\quad x_1 \geq 0$ $\min_y (2y_1-4)^2+(2y_2-1)^2+x_1y_1$ $4x_1+5y_1+4y_2 \leq 12$ $4y_2-4x_1-5y_1 \leq -4$ $4x_1-4y_1+5y_2 \leq 4$ $4y_1-4x_1+5y_2 \leq 4$ s.t. $\quad y_i \geq 0, \quad i=1,2$	upper= -1.2091 lower= 7.6145
TP7 n=2 m=2	$\min_x \quad -\frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}$ s.t. $\quad (x_1)^2+(x_2)^2 \leq 100$ $x_1-x_2 \leq 0$ $x_i \geq 0 \quad i=1,2$ $\min_y \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}$ $0 \leq y_i \leq x_i \quad i=1,2$	upper= -1.96 lower= 1.96
TP8 n=2 m=2	$\min_x \quad \lvert 2x_1+2x_2-3y_1-3y_2-60\rvert$ s.t. $\quad x_1+x_2+y_1-2y_2 \leq 40$ $0 \leq x_i \leq 50 \quad i=1,2$ $\min_y (y_1-x_1+20)^2+(y_2-x_2+20)^2$ s.t. $\quad 2y_1-x_1+10 \leq 0$ $2y_2-x_2+10 \leq 0$ $-10 \leq y_i \leq 20 \quad i=1,2$	upper= 0.0 lower= 100.0 or 200.0
TP9 n=10 m=10	$\min_x \quad \sum_{i=1}^{10}(\lvert x_i-1\rvert+\lvert y_i\rvert)$ s.t. $\min_y e^{(1+\frac{1}{4000}\sum_{i=1}^{10}(y_i)^2-\prod_{i=1}^{10}\cos(\frac{y_i}{\sqrt{i}}))\sum_{i=1}^{10}(x_i)^2}$ $-\pi \leq y_i \leq \pi \quad i=1,2,3,...,10$	upper= 0.0 lower= 1.0
TP10 n=10 m=10	$\min_x \quad \sum_{i=1}^{10}(\lvert x_i-1\rvert+\lvert y_i\rvert)$ s.t. $\min_y e^{(1+\frac{1}{4000}\sum_{i=1}^{10}(y_ix_i)^2-\prod_{i=1}^{10}\cos(\frac{y_ix_i}{\sqrt{i}}))\sum_{i=1}^{10}(x_i)^2}$ $-\pi \leq y_i \leq \pi \quad i=1,2,3,...,10$	upper= -3.6 lower=-2.0

solution. For every new generation, the approximation model is improved until convergence of the algorithm to the optimal solution. The authors highlighted the fact that, at termination, BLEAQ provides not only the optimal solution but also an approximation of the mapping between upper-level and lower-level variables.

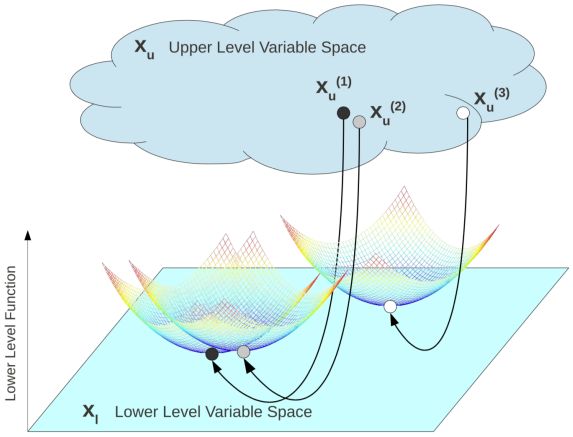


FIGURE 4.3: Approximation of the set-value mapping Ψ: $\mathbf{R}^n \rightarrow \mathbf{R}^m$ between upper-level and lower-level decision variable applied in BLEAQ (source: Sinha et al. [332])

4.5.2 Experimental protocol and parameters

For both approaches, experiments have been conducted on the High Performance Computing (HPC) platform of the university of Luxembourg[357]. Each run was completed on a single core of an Intel Xeon E3-1284L v3 @ 1,8 GHz, 32Gb of RAM server, which was dedicated to this task. The GPyOpt python library [184] has been selected to apply bayesian optimization. The LCB acquisition function has been considered to determine the next promising point with parameter $k = 2$. The differential algorithm implemented to optimize the acquisition function is the one proposed by the python scipy library with default parameters: strategy='best1bin', maxiter=1000, popsize=15,mutation=(0.5, 1), recombination=0.7. The scipy library implements SLSQP as well. Concerning the BLEAQ algorithm, the authors provided a MATLAB code located at http://www.bilevel.org in the resources section. We kept the same parameters described in [332] and performed 30 runs on each benchmark using the provided code.

The stopping criteria for both algorithms is the convergence. The bayesian algorithm proposed in this chapter uses the same criteria as BLEAQ except that the criteria is not applied on a population but on the last acquired points.

The next section will first describe the results obtained after applying the adapted bayesian algorithm proposed in this chapter on the 10 benchmarks. A comparison between the results obtained with BLEAQ are discussed to show the main advantages brought by the bayesian approach.

4.6 Experimental results

4.6.1 Results and discussion

Numerical results have been summarized in Tables 4.3, 4.4 and 4.5. Table 4.3 represents the average best fitnesses obtained over all runs. A Wilcoxon rank-sum test [376] has been realized to determine if the fitness

differences are statistically significant for both levels. P-values have been provided in Table 4.4. In addition, Table 4.5 illustrates the average number of evaluations for both levels over all runs. The "ULcalls" column represents the average number of objective function calls at upper-level while the "LLcalls" column indicates the average aggregated number of objective function calls at lower-level. We can easily observe that the number of evaluations obtained with the Bayesian algorithm is much lower than for BLEAQ. This is the reason why no statistical tests have been computed in this case. In general, BLEAQ achieves better accuracy even if results for both approach are very close. Indeed, BLEAQ makes use of an evolutionary algorithm to perform lower-level optimization while the bayesian algorithm implemented to tackle these bi-level benchmarks only invokes a local search, i.e SLSQP. Nevertheless on TP9 benchmark, the fitness at upper-level is better than the best known solution. For TP9 and TP10, the average fitnesses are better for the Bayesian algorithm despite the fact that these both benchmarks have been generated from complex single-level benchmarks with multiple local optimal solutions (see http://www.bilevel.org). We also noticed the ability of the Bayesian algorithm to face multi-modal problems. Indeed, the Bayesian algorithm provides two solutions with fitnesses $(F, f) = (0, 100)$ and $(F, f) = (0, 200)$ which are both upper-level optimal solutions. The same observation has been made for the TP8 benchmark. On the contrary, BLEAQ never detects the solution providing the fitness values $(0, 200)$. In a competitive and real-life application, such solution could be preferred by the upper-level decision maker since the fitness associated to the lower-level problem is definitely higher while still optimal for the upper-level decision maker. These promising results show the advantage of such surrogate optimization algorithms over traditional evolutionary computing techniques when function evaluations are complex and time-consuming.

	Bayesian		BLEAQ	
Average	UL fitness	LL fitness	UL fitness	LL fitness
TP1	253.6155	70.3817	224.9989	99.9994
TP2	0.0007	183.871	2.4352	93.5484
TP3	-18.5579	-0.9493	-18.6787	-1.0156
TP4	-27.6225	3.3012	-29.2	3.2
TP5	-3.8516	-2.2314	-3.4861	-2.569
TP6	-1.2097	7.6168	-1.2099	7.6173
TP7	-1.6747	1.6747	-1.9538	1.9538
TP8	0.0008	180.6452	1.1463	132.5594
TP9	0.0012	1.0	1.2642	1.0
TP10	0.0049	1.0	0.0001	1.0

TABLE 4.3: Average best fitnesses for both levels

p-values	UL fitnesses	LL fitnesses
TP1	5.09354939843e-08	6.62154466203e-08
TP2	1.99180208303e-05	0.000455639542651
TP3	1.33535344389e-11	1.33535344389e-11
TP4	1.33535344389e-11	0.827259346563
TP5	4.88497305946e-08	9.352489315e-05
TP6	1.33535344389e-11	3.21862967172e-09
TP7	1.06973503522e-10	1.06973503522e-10
TP8	1.91908665211e-09	0.00204782236
TP9	2.58028430416e-08	7.11788655392e-06
TP10	1.97034447118e-11	1.97034447118e-11

TABLE 4.4: Wilcoxon Rank-Sum Test for both levels

	Bayesian		BLEAQ	
Average	UL calls	LL calls	UL calls	LL calls
TP1	211.1333	1558.8667	588.6129	1543.6129
TP2	35.2581	383.0645	366.8387	1396.1935
TP3	89.6774	1128.7097	290.6452	973.0
TP4	16.9677	334.6774	560.6452	2937.3871
TP5	57.2258	319.7742	403.6452	1605.9355
TP6	12.1935	182.3871	555.3226	1689.5484
TP7	72.9615	320.2308	494.6129	26682.4194
TP8	37.7097	413.7742	372.3226	1418.1935
TP9	16.6875	396.3125	1512.5161	141303.7097
TP10	21.3226	974.0	1847.1	245157.9

TABLE 4.5: Average number of function evaluations for both levels

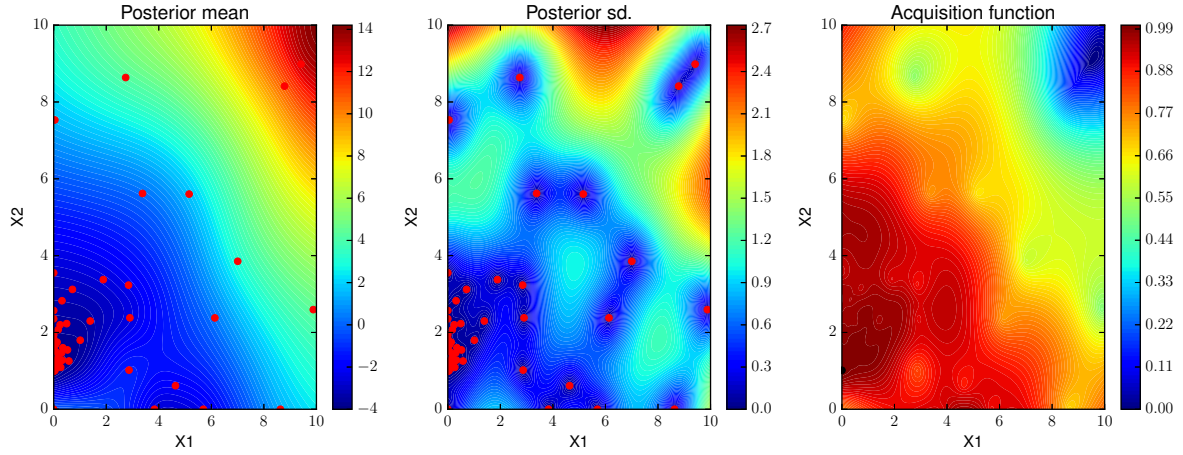


FIGURE 4.4: Gaussian process state after TP5 optimization

4.6.2 Example of landscape obtained after bayesian optimization

Figure 4.4 depicts the posterior mean, the standard deviation and the current acquisition function after applying the bayesian algorithm on the benchmark TP5. Red dots indicates data points that have been sampled so far by the algorithm. The posterior mean shows that a large number of points concentrate around the best solution. Furthermore, the standard deviation is also very low in this area indicating a strong confidence on the location of the optimal solution. Finally, the landscape of the acquisition function suggests that the next acquired point would be probably in this area confirming the convergence and therefore termination of the algorithm. Additional landscapes for the remaining benchmarks are provided in Annex C.

4.7 Conclusions

Bi-level optimization problems are special kind of optimization problems involving two decisions makers. These hierarchical problems require to optimize iteratively two problems called the upper-level and the lower-level because the decision variables are partially controlled by each of them. This nested optimization scheme is time-consuming when both levels are $\mathcal{NP}$-hard. In this chapter, a surrogate-based algorithm has been proposed to tackle these complex problems. Based on gaussian processes, bayesian optimization is a global and free-derivative optimization algorithm which has been intensively employed to discover optimal machine learning hyper-parameters (e.g. neural network, support vector machine) by creating a surrogate model which is fine-tune until the convergence occurs. Bayesian optimization makes no assumption on the characteristic

of the optimized function to approximate it. It also embeds a mechanism to detect new promising points after refinement of the model at each iteration. These properties make bayesian optimization a very good candidate algorithm to tackle hierarchical problems such as bi-level problems. Indeed, the upper-level objective value is generally non-convex over the inducible region $\mathcal{IR}$. Numerical experiments on 10 bi-level benchmarks confirmed that bayesian optimization can dramatically reduce the number of evaluations and thus the number of lower-level optimizations while guaranteeing very good and accurate results. Future works could consider different algorithms to perform lower-level optimizations other than SLSQP to improve accuracy. In fact, the combination of multiple and different lower-level optimization solvers would certainly be the best approach to do such optimizations. In addition, new optimization techniques could be proposed to solve the acquisition function. For example, multi-objective optimization could use to consider several acquisition functions instead of optimizing only one at a time.

Even if surrogate-based approaches can dramatically reduce the number of evaluation required to compute the bi-level optimal solution, they have some limitations that reduce their scope of applications. In the case of BLEAQ, the approximations of the set-value mapping Ψ: $\mathbf{R}^n \rightrightarrows \mathbf{R}^m$ would be very time consuming due to the number of lower-level decision variables. For bayesian optimization, a large number of decision variables would require to train a large dataset which could be very time consuming due to the gaussian processes. Indeed, the computational bottleneck in using gaussian processes is the inversion of the covariance matrix that can grow very rapidly, especially for large scale problems. An additional issue with surrogate-based approaches is the fact that they cannot cope with combinatorial problems although most real applications are generally combinatorial or discrete. Consequently, tackling efficiently large scale combinatorial bi-level problems remains a real challenge. In the next chapter, we investigate this issue and propose to rely on the concept of "learning to optimize" in order to offer an efficient solution to tackle any large scale combinatorial bi-level problem.

Part III

Tackling large scale and combinatorial bi-level problems under the "learning to optimize" paradigm

Chapter 5

"Learning to optimize" paradigm using GP Hyper-Heuristic

Contents

5.1 Introduction

In chapter 2, we listed several strategies to tackle bi-level optimization problems. Due to their high complexity, metaheuristics for bi-level optimization are the most suitable techniques when assumptions such as convexity and regularity cannot be applied. Through chapter 3, we have shown the relevance of a bi-level clustering model obtained after decomposition into two-levels, of which one has polynomial complexity. A nested optimization approach was therefore efficient tackle. Therefore, we proposed a parallel and hybrid evolutionary approach to tackle this novel bi-level clustering problem. Unfortunately, general bi-level problems can have more complex properties and often have $\mathcal{NP}$-hard lower-level. Chapter 4 has been dedicated to such general problems. We studied the possibility to employ surrogate-based approaches such as bayesian optimization to replace the costly and numerous evaluations of the lower-level problem. Despite their real advantage for continuous and small-scale problems, surrogate-based approaches are hold in check by large-scale and combinatorial bi-level problems.

Although there is no extension for combinatorial problems, the concept of "learning knowledge" from the problem to improve the resolution could be also investigated in the discrete/combinatorial case. Through the last two parts and four chapters, we have shown that the key issue to solve bi-level problems depends strongly

on our capacity to solve the various instances generated at the lower-level. Therefore our efforts should be dedicated to tackle efficiently every lower-level instances that can be generated during bi-level optimization. For this purpose, there already exists many metaheuristics and bio-inspired approaches that have been designed to tackle general $\mathcal{NP}$-hard problems. Nonetheless, the "No free Lunch Theorem" [378] states that there is no universal metaheuristic that can perform better across all existing problems. Therefore the number of publications in the metaheuristics field escalated during the last two decades and went from 100 in 1995 to 10500 published papers per year in 2017 (see Figure 5.1). This current plethora of metaheuristic algorithms requires also to determine the optimal parameters which is not trivial. Instead of manually searching for and fine-tuning metaheuristics, we could develop a methodology to learn optimization algorithms for dedicated problems. In this situation, we do not try to approximate the objective function as for continuous problem but the path that leads to an algorithm producing a good approximation.

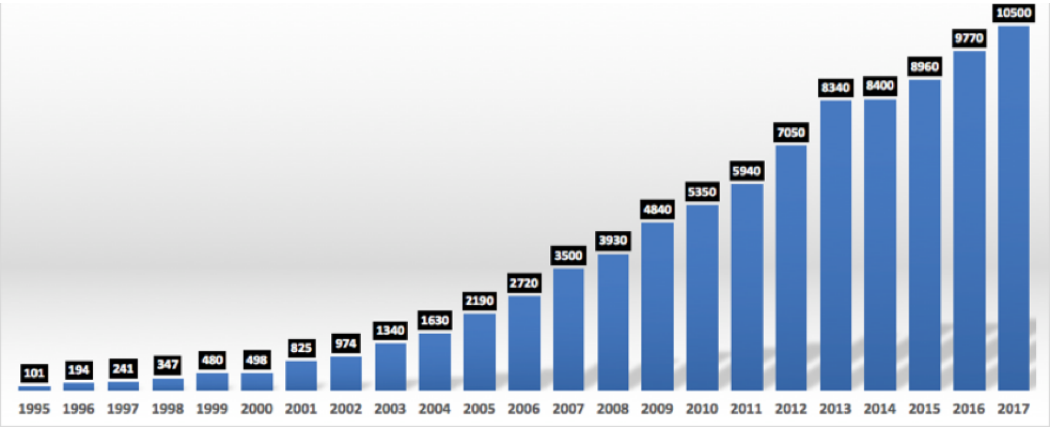


FIGURE 5.1: Number of publication in the metaheuristic field

In this part, we introduce the "learning to optimize" paradigm based on training algorithms on a set of instances in order to tackle efficiently a problem. This is largely motivated by the fact that we need to be able to tackle any lower-level instances encountered during bi-level optimization. It is perfectly suitable for discrete/combinatorial problems where no strong properties help us to characterize optimal solutions. One solutions is therefore to learn knowledge from a large set of known instances. We solely believe that we can determine automatically rules that can guide us to find a path to near optimal solutions. Before applying this paradigm on bi-level optimization, we focus first on single-level combinatorial problems.

Consequently, this chapter is a transition allowing us to define and explore more in details the "learning to optimize" paradigm. Our aim is now to develop an automatic procedure guiding us to generate dedicated and efficient heuristics to cope with $\mathcal{NP}$-hard problems and by extension $\mathcal{NP}$-hard lower-level problem. We suggest to learn automatically the extraction of this knowledge using a machine learning mechanism producing efficient heuristics as outputs. Here, we attempt to learn full or partial heuristics to obtain fast but efficient solvers. For this purpose, the recent approach referred to as GP Hyper-Heuristics could serve as learning model. Based on Genetic Programming, GP hyper-heuristics are constructive hyper-heuristic algorithms in which a population of self-made heuristics is evolved until a certain convergence criterion is met. Introduced by Koza in [221], GP makes use of a tree-based encoding representation and was initially designed to evolve functions and programs. Recent works [21, 110] have shown that GP can be extended and considered as a learning model for code generation. Terminal and function sets are key elements for GP. They are the nodes and leaves of each solution represented as a Tree. The concept behind GP hyper-heuristics is very similar to a machine learning model. In order to obtain a final population of heuristics with high abilities of approximation, the algorithm will learn from a set of various instances, i.e., the training set and validate the heuristics on a test set containing unseen instances.

Hyper-heuristics have been originally designed to determine the best selection of heuristics, *"heuristics to choose heuristics"*, for solving optimization problems. Despite their very good results, they were constrained by the existing knowledge of a problem. It means that for a new problem with few existing heuristics, the chance to produce an efficient hyper-heuristic is low. GP hyper-heuristic can be a very good alternative in this case. Successful attempts in the literature (e.g. [109], [9],[21]) relied on this new class of hyper-heuristics that search through the space of heuristics instead of the space of solutions and use specific instances data and properties (e.g., columns for a mathematical problem) as inputs for the design of heuristics.

Designing efficient heuristics requires a deep understanding of the problem and can be time consuming. This is the reason why a learning mechanism could both ease and improve the problem analysis and resolution. Previous works have shown that automatic generation of heuristics can challenge human-based heuristics. Nevertheless, existing approaches rely on a training mechanism which generates size-dependent heuristics, contrary to human-based heuristics, which could mitigate their utility.

In this chapter, we propose to test this approach by generating automatically new competitive and general heuristics for a well-known single-level problem: the Multi-dimensional Knapsack problem (MKP). Our contribution relies on a new definition of the terminal and function sets enabling the generation of efficient and size-independent heuristics. Additionally, we argue that this new terminal set should contain all information helping a decision maker to discriminate good items from bad ones. Such information can be found in the literature or using other resolution approaches. For instance, the LP relaxed solution is an additional knowledge which enhance the design of new heuristics. Experiments have been conducted on instances from the OR-Library and results show that this improved approach outperforms state-of-the-art approaches comprising human-based heuristics and original heuristic generation approaches.

This chapter is organized as follows. An overview of hyper-heuristics is provided in section 2 and the MKP is introduced in section 3. Then the improved GP hyper-heuristic is described in details in section 4. The experimental setup and results are discussed in section 5. Finally, the last section provides our conclusions and proposes some possible perspectives.

5.2 Hyper-heuristic: overview

Described as "heuristics to choose heuristics" by Cowling et al. [86], the term "hyper-heuristics" has been first introduced by Denziger et al. in [150] and referred to approaches combining artificial intelligence (A.I.) methods. Contrary to algorithms searching in the space of solutions, hyper-heuristic algorithms search in the space of algorithms, i.e., heuristics in order to determine the best heuristics combination to solve a problem. Burke et al. in [57] compared hyper-heuristics as "*off-the-pegs*" methods which are generic approaches providing solutions of acceptable quality as opposed to *"made-to-measure"* techniques. This need of generalization is clearly related to machine learning approaches. Therefore, hyper-heuristics can be classified as learning algorithms and have been motivated by the following factors: the difficulty of maintaining problem-specific algorithms and the need of automating the design of algorithms. Two methodologies of hyper-heuristics rose from the literature: the first one is referred to as **heuristic selection** while the second one is described as **heuristic generation**.

Heuristic selection is the legacy approach which consists in determining the best subset of heuristics solving a problem. Among these approaches, we can distinguish constructive and perturbative methods. Constructive methods generate a solution step by step, starting from a partial or empty solution. The construction of a full solution is achieved through the selection and application of the heuristic to this partial solution. For this purpose, a pre-existing set of constructive heuristics should be provided in order to determine the best

heuristics to apply at a given state of the search. The resolution then stops when the solution is complete. Constructive methods have been applied for instance on vehicle routing [146], 2D packing [247], constraint satisfaction [290] and scheduling [144]. On the contrary, perturbative methods start from a valid solution and attempt to modify it using a pre-existing set of perturbative heuristics. At each step, one heuristic is selected from this set and applied to the solution. According to a specific acceptance strategy factor (e.g., deterministic or non-deterministic), the new solution is accepted or rejected. It is also possible to perturbate multiple solutions at once but it has been seldom used in the literature. Scheduling [193], space allocation [22] and packing [23] are problems where such pertubative methods have been exploited.

More recently, a growing interest has been devoted to heuristic generation. The motivation behind this approach is the automatic generation mechanism which does not rely on a possible set of pre-existing heuristics. Instead of searching in the space of heuristics, the hyper-heuristic searches in the space of components, i.e., instances data. Building a complete heuristic is not a trivial task but can be performed using Genetic Programming (GP) [13]. In contrast to Genetic Algorithms (GA) where solution vectors are improved via genetic operators, GP evolves a population of programs until a certain stopping criterion is satisfied. Programs are expressed as tree structures which means that their length is not defined a priori contrary to GA solutions. The suitability of GP to produce heuristics has been outlined in a survey published by Burke et al. [58]. The major advantage brought by GP is the possibility to automatize the assembly of building blocks, i.e., terminal sets and function sets emerging from knowledge gained on the problem. For the MKP, this knowledge can be easily retrieve in the literature. The dynamic length of the tree encoding is an advantage if some size limitations are implemented. Indeed, large programs will tend to have over-fitting symptoms meaning that the generated heuristics will be very efficient on the instances trained by the GP but not on unseen ones. These are typically the same issues faced by machine learning models. GP hyper-heuristics, i.e., heuristic generation, encountered real success in cutting and packing [59], function optimization [289] and other additional domains [118],[356],[281].

It is also worth mentioning the recent approaches such as Cartesian GP and Grammar-based GP that are improvements of the classical GP. Cartesian GP is an alternative form of GP that encodes a graph representation of a computer program. Cartesian GP defines explicitly a size preventing bloat but can be very sensitive to parameters. In Grammar-based GPs [50, 341, 340], a grammar in Backus-Naur Form (BNF) is considered to map linear genotypes to phenotype trees and have less structural difficulties than a classical GP.

This work relies on hyper-heuristics using classical GP and on a more general definition of the terminal sets in order to remove the size limitation problem mentioned in [109].

5.3 Multi-dimensional 0-1 Knapsack

The Multi-dimensional 0-1 Knapsack (MKP) is a $\mathcal{NP}$-hard combinatorial problem [163] which extends the well-know 0-1 Knapsack problem for multiple sacks. The objective is to find a subset of all items maximizing the total profit and fitting into the m sacks. Each item j gives a profit p_j and occupies some space a_{ij} in the sack i. Each sack i has a maximum capacity b_i which should not be exceeded. The Multi-dimensional 0-1 Knapsack can be formally expressed as a 0-1 Integer Linear Program (ILP) as presented in Program 5.1.

More practically, the MKP is a resource allocation problem which has been first employed as a capital budgeting model ([249, 266]). Like the standard knapsack problem, this problem received wide attention from many communities, including operation research and evolutionary computing). Multiple heuristics and metaheuristics have been designed to tackle the MKP in addition to the existing exact approaches which can only handle small instances. Among the existing heuristics for MKP, *greedy* heuristics are designed to be fast, i.e., polynomial

$$\text{maximize} \quad \sum_{j=1}^{n} p_j x_j \tag{5.1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i \in \{1, ..., m\} \tag{5.2}$$

$$x_j \in \{0, 1\} \qquad \forall j \in \{1, ..., n\} \tag{5.3}$$

PROGRAM 5.1: 0-1 ILP for the multi-dimensional knapsack

time complexity. Generally, they are constructive methods which can be primal or dual heuristics. A primal heuristic starts from a feasible solution and tries to improve the objective value while keeping the solution feasible. On the contrary, a dual heuristic starts from an upper-bound (in case of maximization) solution, i.e., not feasible and attempts to make it feasible while minimizing the impact on the objective value. For example, the authors in [324] used a dual heuristic with a starting solution consisting in taking all items. The heuristic aims at removing items according to an increasing ratio until feasibility is reached. The ratio or score of each item j is computed as follows: $r_j = \frac{p_j}{\sum_i^m w_i a_{ij}}$. Weights are sometimes omitted since they add a new level of complexity and are specific to the considered instances. Using Lagrangian relaxation, the authors in [255] improved the dual heuristic published in [324]. Concerning primal heuristics, they attempt to add items until all constraints are satisfied. In this case, items with the largest ratios have the priority ([213],[250],[351]). These new heuristics using dual multipliers give insights about the variables to set. The work proposed by Magazine and Oguz has been then extended by [363]. Further improvements based on surrogate constraints [302], bound tightness [136], threshold acceptance [113] and noising approaches [74] have contributed to improve such heuristics. The interested reader may refer to the survey on MKP heuristics from Fréville [135]. Metaheuristics have been also considered to tackle MKP instances. They are stochastic algorithms which successfully tackled many combinatorial optimization problems. The MKP is one of those problems which have been diligently investigated. A simulated annealing (SA) algorithm has been first employed in [360] in which a specific random move should maintain feasibility during the search. Then a Tabu Search (TS) algorithm has been designed in [90] with a dynamic management of the tabu list managed to outperform the SA algorithm. This dynamic management is ensured by maintaining feasibility through DROP-ADD moves. After that, diverse metaheuristics have been considered to solve the MKP during the last decade including Ant Colony Optimization [125], Genetic algorithms [231], Memetic algorithms [80], Particle Swarm algorithms [77], Fish Swarm Algorithms [19] and Bee Colony algorithms [345]. Finally, machine learning algorithms such as Augmented Neural Networks [92] have also been studied to provide approximate solutions for MKP.

5.4 Automatic Heuristic Generation for the Multi-dimensional 0-1 Knapsack

5.4.1 GP Hyper-Heuristic workflow

We extend the approach developed in [109] and also based on GP hyper-heuristics. Referred to as bio-inspired algorithms, GPs belongs to the class of evolutionary algorithms inspired by Darwin's theory of evolution. Contrary to GAs, GPs makes use of tree encodings to represent solutions. Proposed by Koza [13], a GP uses a dynamic-length encoding due to the tree representation and are thus good candidates for evolving programs or mathematical expressions. According to [309], the key choices to use a GP are the definition of the *terminal*

set and *function set*. Indeed, the *terminal set* consists in all external inputs, constants or functions without parameters which are leaf nodes of the tree solution while the *Function set* determines all operators (e.g., arithmetic, logic, etc.), i.e., the intermediate nodes.

A GP hyper-heuristic starts with a random population of heuristics which are evolved in order to obtain heuristics with high resolution abilities. The GP trains the population of heuristics to solve a set of instances, i.e., the *training set*. The obtained heuristics are also wished to be able to tackle efficiently new encountered instances by using the knowledge gained during the training phase. This work-flow is very similar to a supervised learning algorithms which learns on a training data and validates the training on test data. Nonetheless to qualify GP hyper-heuristics as supervised learning methods, we should normally know the optimal solution of the training instances. For that, two possible choices exist. One can either consider small training instances and solve them exactly in order to have a strict supervised knowledge or consider another approach which optimizes while learning. Drake in [109] chose this second approach which does not rely on any a priori knowledge about the optimal solution. In this work, we also consider this second approach as well while trying to obtain more general heuristics.

Generating full and valid programs requires a lot of computation. It is also very difficult to obtain valid programs even for simple tasks. Nevertheless, they have been recently employed and are referred to as to *Grammar-based GP* [50]. Terminal and functions sets are modeled using a specific Grammar which ensure that the closure property will be satisfied. Such Grammar-based GP is dedicated to small and simple tasks. Indeed, the advantage provided by GP hyper-heuristics is to reduce the combinatorial explosion by searching in the heuristic space. However, the use of a large grammar would not go in this direction. As an alternative, it is more efficient to identify the dynamic part of an heuristics, i.e., parts which should evolve and have a direct impact on the heuristic results. For example, on the MKP, a constructive and greedy heuristic will sort all items according to a **scoring function** (1st phase) and then try to add items (2nd phase) to the sacks while following the order induced by the scoring function. It is clear that the second phase will be common to all generated heuristics whereas the first phase has a key element: **the scoring function**. Therefore, an efficient way to train heuristics is to train scoring functions. A GP hyper-heuristic is thus devoted to the generation of mathematical functions able to score the available items which can be added to the sacks (see Algorithm 8). [109] were the first to introduce such strategy for the MKP problem. Each solution, i.e. scoring function is represented as a syntax tree depending on the definition of the terminal and function set (see Table 5.1). Each scoring function in the population is then inserted to the greedy heuristic template (see Algorithm 10) in order to evaluate the relevance of the function to obtain an efficient insertion order. In order to drive the algorithm to produce efficient heuristics optimizing the MKP, the fitness of a scoring function is obtained as the sum of each profit computed (see Algorithm 10) on the training instances (see Algorithm 9).

5.4.2 Generation of size-independent heuristics

The previous section introduced the concept of GP hyper-heuristics and how they have been implemented to tackle the MKP. The original approach proposed by [109] has the disadvantage to yield heuristics which depend on the size of the training instances. Despite their good approximation results, the trained heuristics cannot be employed to solve instance of different size.

The contribution of this work lies in the fact that we focus on learning generic heuristics applicable on different instances size. For this purpose, we update the terminal and function sets. They are crucial since they contain

Algorithm 5 Genetic programming hyper-heuristic

```
hall_of_fame ← create_empty_HOF(size=1)
population ← gen_ramped_half_and_half(NPOP,min,max)
for ind in population do
    ind.fitness ← evaluate(ind,training_instances)
end for
hall_of_fame ←update_HOF(hall_of_fame)
while gen ≤ NGEN do
    parents← selection(population)
    offsprings ← ∅
    for ind in parents do
        if random() ≤ CXPB then
            mate ← sample(parents,1)
            offspring1,offspring2 ← crossover(ind,mate)
            offsprings ← offsprings ∪ {offspring1,offspring2}
        else if random() ≤ CXPB+MUTPB then
            mutant ← mutation(ind)
            offsprings ← offsprings ∪ {mutant}
        else
            repro_ind ← copy(ind)
            offsprings ← offsprings ∪ {repro_ind}
        end if
    end for
    for ind in offsprings do
        ind.fitness ← evaluate(ind,training_instances)
    end for
    hall_of_fame ←update_HOF(hall_of_fame)
    population ← offsprings
end while
return hall_of_fame
```

Algorithm 6 evaluate(ind,training_instances)

```
profits←0
for instance in training_instances do
    func ← compile(ind)
    value ← greedy_heuristic(instance,func).solve()
    profits ← profits + value
end for
return profits
```

Algorithm 7 greedy_heuristic_template(instance,function)

```
value ← 0
solution ← [0,0,...,0]
sacks ← [0,0,...,0]
indexes ← sort(items,func)
while indexes ≠ ∅ do
    index ← indexes.pop_head()
    if sacks_i + A_{i,index} ≤ rhs_i ∀i ∈ {1,...,m} then
        solution_index ← 1
        value ← value + p_index
        for i ∈ {1,...,m} do
            sacks_i ← sacks_i + A_{i,index}
        end for
    end if
end while
return value
```

the building blocks for the construction of the scoring functions. The sorting mechanism of the greedy/constructive heuristic relies on these functions and **discriminate** interesting items from unattractive ones. This notion of discrimination is again very close to the one defined in machine learning which often relies on feature selection. The latter, also known as attribute selection, aims at selecting a subset of relevant attributes to build a model. Feature selection is essential for interpretation purpose, shortening training time, reducing over-fitting and avoiding the so called curse of dimensionality. For GP hyper-heuristics, terminals and functions set should be chosen to achieve the same process. Here, both sets represent the features of the learning model.

In [109], the authors chose the sets described in Table 5.1 which consist of all elements defining the properties of an item (e.g., price, resource consumption, etc.) namely a column of the mathematical program defining the MKP. The drawback of such an approach is the dependence between the training instances and the test instances. Indeed both should have the same number of constraints. Despite the very good results reported in their work, they cannot apply their generated heuristics to instances having a different number of constraints. We propose to reconsider another terminal set which should be able to reach a better generalization level.

According to Table 7.1, we propose a new set where none of the terminal elements are depend on the instance size. In order to discriminate **good** items from **bad** ones, we also add the solution of the LP relaxation as prior knowledge which is clearly a very good feature for our learning purpose. We keep the average difference which was originally introduced by [109]. In fact, good indicators can be found due to the numerous works existing in the literature. Many of these indicators have been already proposed to generate efficient approximation of the MKP. The sets shown in Table 7.1 are not exhaustive and could be easily completed with new features which would permit to discriminate more accurately profitable instances from valueless ones.

The next two sections describe the conducted experiments which attest the better generalization results of our approach. We first detail the parameterization and the benchmarks used in both Drake's work [109] and this one. Then a detailed analysis of the results is provided.

TABLE 5.1: Functions and terminal sets implemented in [109]

Name	Description
Operators	
+	Add two inputs
-	Substract two inputs
*	Multiply two inputs
%	Divide two inputs with protection
Terminal sets/ Arguments	
p_j	Profit of the current item j
$avgDiff_j = \frac{\sum_i b_i - a_{ij}}{m}$	Average difference between the capacity and the resource consumption for item j
a_{1j}	Ressource consumption of item j for sack 1
a_{2j}	Ressource consumption of item j for sack 1
$a_{\dots,j}$	...
a_{mj}	Ressource consumption of item j for sack m

TABLE 5.2: Functions and terminal sets implemented in this work

Name	Description
Operators	
+	Add two inputs
-	Substract two inputs
*	Multiply two inputs
%	Divide two inputs with protection
mod	Modulo b.t.w. two inputs with protection
Terminal sets/ Arguments	
p_j	Profit of the current item j
$avgDiff_j = \frac{\sum_i b_i - a_{ij}}{m}$	Average difference between the capacity and the resource consumption for item j
$\sum_i a_{ij}$	Total ressource consumption of item j for sack i
$\max_i a_{ij}$	Max ressource consumption of item j for sack i
$\bar{x}_j$	Solution value for item j after LP relaxation

5.5 Numerical experiments

5.5.1 Setups and parameters

As for the experiments realized in [109], the OR-Library instances have been considered. They have been originally introduced in [80] and consist of 270 instances (see Table 5.3). These are classified according to the number of variables $n \in \{100, 250, 500\}$, number of constraints $m \in \{5, 10, 30\}$ and the tightness ratio $r \in \{0.25, 0.50, 0.75\}$.

In order to evaluate the efficiency of the heuristics on these instances, we adopt as performance measure the %-gap (see equation 7.2) between a lower-bound and an upper-bound. Lower bounds are provided by the heuristics, i.e. $\underline{v}^h$ while the continuous LP relaxation, i.e. $\overline{v}_{lp}$ will be the reference upper bound. In addition, we multiply by 100 all gaps such that results are comparable with the work described in [109].

TABLE 5.3: OR-Library benchmarks

	Tightness ratio			
Instance set	0.25	0.50	0.75	Total
OR5x100	10	10	10	***30***
OR5x250	10	10	10	***30***
OR5x500	10	10	10	***30***
OR10x100	10	10	10	***30***
OR10x250	10	10	10	***30***
OR10x500	10	10	10	***30***
OR30x100	10	10	10	***30***
OR30x250	10	10	10	***30***
OR30x500	10	10	10	***30***
All instances	***90***	***90***	***90***	***270***

TABLE 5.4: GP parameters

Generations	50
Population size	100
Crossover Probability (CXPB)	0.85
Mutation Probability (MUTPB)	0.1
Reproduction Probability	0.05
Tree initialization method	Ramped half-and-half
Selection Method	Tournament selection with size=7
Depth limitation	17
Crossover Operator	One point crossover
Mutation Operator	Grow

$$\%\text{-}gap = 100 * \frac{\overline{v}_{lp} - \underline{v}^h}{\overline{v}_{lp}} \tag{5.4}$$

Concerning the GP hyper-heuristic, Table 5.4 describes all the GP parameters and GP operators considered for these experiments. The GP algorithm will perform 50 generations with a population size of 100 syntax trees that represent each a possible scoring function. Contrary to GA, GP makes a different use of the evolutionary operators. First of all, their probabilities should sum to 1. For example, in the case of Table 5.4, 85% of the solutions will mate with another one, 10% will face mutations and only 5% will be kept without any modifications for the next generation. In order to keep control of the size of each syntax tree, we added a limitation operator which guaranty us that a solution will not have a depth greater than 17 nodes. The crossover operator and mutation operators are the same as utilized in [109].

Finally, we adopt the same protocol as described in [109] to train heuristics. All instances have been divided into groups depending on the number of variables, the number of constraints and the tightness ratio as illustrated in Table 5.3. The GP hyper-heuristic has been applied on all of these groups containing each ten instances. Five random instances have been selected as training instances while the remaining five instances have been considered as test instances. The fitness value of a heuristic is thus the sum of the fitness value on the five training instances. However, the reported %-gaps are only computed on the test instances. For each group, five GP hyper-heuristic runs have been performed in order to obtain an average %-gap and the best heuristic has been recorded.

Experiments have been conducted on the High Performance Computing (HPC) platform of the University of Luxembourg [357]. Each run was completed on a single core of an Intel Xeon E5-2680 v3 @ 2.5 GHz, 32Gb of RAM server, which was dedicated to this task. The python library DEAP [108] has been considered for the GP implementation.

TABLE 5.5: Performance of the best found heuristics on the ORlib instances based on average gap (%)

	Original approach				**Improved approach (AHG)**			
Instance set	0.25	0.50	0.75	Average	0.25	0.50	0.75	Average
OR5x100	4.98	2.05	1.36	***2.80***	1.22	0.94	2.00	***1.39***
OR5x250	3.08	1.66	0.77	***1.84***	0.55	0.36	0.73	***0.55***
OR5x500	2.38	1.64	0.71	***1.58***	0.23	0.13	0.33	***0.23***
OR10x100	7.39	3.54	2.26	***4.40***	1.74	1.65	2.97	***2.17***
OR10x250	4.43	2.78	1.15	***2.79***	0.81	0.41	0.99	***0.74***
OR10x500	3.77	1.97	0.99	***2.24***	0.44	0.26	0.43	***0.38***
OR30x100	8.67	4.70	2.43	***5.27***	2.81	1.55	5.83	***3.39***
OR30x250	5.73	3.25	1.70	***3.56***	1.62	1.01	1.73	***1.45***
OR30x500	4.80	2.54	1.40	***2.91***	0.71	0.46	0.95	***0.71***
All instances	***5.03***	***2.68***	***1.42***	***3.04***	***1.12***	***0.75***	***1.77***	***1.22***

5.5.2 Results

The average %-gap obtained after 5 runs is provided in Table 5.5. The left part of this table represents the results from [109] while the right part corresponds to the improved approach (AHG) that we have proposed in this work.

Each row depicts a specific instance set **ORnXm** divided into groups of different tightness ratios. For instance, the average %-gap obtained for the instance set **OR5x100** with tightness ratio 0.25 is **4.98** in the original approach. Gray shaded cells indicate that the average %-gap is better for the considered approach. For example, the average %-gap obtained for the instance set **OR5x100** with tightness ratio 0.25 is lower and then better for the heuristics obtained with the AHG approach.

Table 5.5 shows us that each instance set **ORnXm** has a better average %-gap when solved with the AHG approach. When considering tightness ratios, we can observe that AHG outperforms all instances with $r = 0.25$ and $r = 0.5$ while this is not the case for $r = 0.75$. The tightness ratio defines the scarcity of capacities. The closer to 0 the tightness ratio the more constrained the instance. Indeed, a ratio $r = 0.25$ implies that about 25% of the items can be packed contrary to a ratio $r = 0.75$ where 75% of the items can be packed. These results show that the proposed AHG approach is able to handle more efficiently highly constrained instances.

We can find a simple reason for these differences. The learning proposed in [109] depends on the size of the instance which means that the cardinality of the terminal set increases with the number of constraints. As a consequence, the heuristic space search is enlarged which increases the possibility of combinations of terminal elements. Another reason could be due to over-fitting but the %-gap on the training instances have not been provided by the authors. In the case of over-fitting, the GP hyper-heuristic would probably perform very well during the training phase but the resulting heuristics would be specific to the training instances impacting the %-gap on test instances. On instances with $r = 0.75$, the problem is less constrained and a general heuristic has less difficulties to find to pack items efficiently which explains the small differences between both approaches. Table 5.6 confirms that the absolute difference increases with the tightness ratio. The dark grey shaded cells represents highest deviation while the white ones depict the smallest deviation for each instance set.

Table 5.7 presents the %-gap obtained by different existing methods found in the literature on the same benchmarks. The approach proposed in this work, i.e. AHG, obtains a good rank, i.e., 5th position which show that the automatic generation of heuristic is a promising domain. This table is not exhaustive and we are sure that while writing these lines, new challenging methods have been proposed.

TABLE 5.6: Absolute deviation between both approach based on average gap (%)

	Tightness ratio		
Instance set	0.25	0.50	0.75
OR5x100	3.76	1.11	0.64
OR5x250	2.53	1.30	0.04
OR5x500	2.15	1.51	0.38
OR10x100	5.65	1.89	0.71
OR10x250	3.62	2.37	0.16
OR10x500	3.33	1.71	0.56
OR30x100	5.86	3.15	3.40
OR30x250	4.11	2.08	0.45
OR30x500	4.09	1.93	0.35

TABLE 5.7: Comparisons with multiple existing approaches over all ORlib instances in terms of gap (%)

Type	**Reference**	**%-gap**
MIP	[111] (CPLEX 12.2)	0.52
MA	[80]	0.54
Selection HH	[111]	0.70
MA	[404]	0.92
AHG	**this work**	**1.22**
Heuristic	[302]	1.37
Heuristic	[136]	1.91
Metaheuristic	[159]	2.28
GHH	[109]	3.04
MIP	[80](CPLEX 4.0)	3.14
Heuristic	[4]	3.46
Heuristic	[363]	6.98
Heuristic	[255]	7.69

Automatic generation of heuristics are general approaches which have been proposed to facilitate the generation of **good performing** and **fast** algorithms to solve problems. Table 5.7 shows that training heuristics can provide better results than human-based algorithms.

The generated heuristics obtained from the previous experiments are claimed to be applicable on any MKP instances. The question is now to determine if they are efficient enough to solve other unknown instances. Therefore, we applied them on all the other groups to observe whether or not the learning process was able to reach a good level of generalization.

Figure 5.2 depicts multiple boxplots representing the distribution of the %-gap obtained after applying each heuristic on the 27 groups. On the Y axis, each row represents a heuristic trained on one specific group (n, m, r) with n the number of variables, m the number of constraints and r the tightness ratio. On the X axis, the scale provides the average %-gap obtained after applying the heuristic on the 10 instances of each groups. In other words, the red dots represents the average %-gap obtained one each group.

We can easily observe that all groups and thus heuristics reach nearly the same performance. We applied the non-parametric Kruskal-Wallis test in order to evaluate if the results are statistically different. The obtained statistic is **36.28** and the p-value is **0.08**. According to an α**-level** set to **0.05**, we can conclude that the null hypothesis is accepted and thus the heuristics have similar performance even if they were trained of different instance size. Tables D.1,D.2 and D.3 in the Annexes provide all the numeric values summarized in 5.2. These 3 tables represent respectively the application of the generated heuristics on instances with $n = 100$, $n = 250$ and $n = 500$.

In this section, we observed that we can automatically generate efficient algorithm to cope with an optimization problem like the MKP. In addition, we proposed better features to be embedded in the terminal and function sets. Results confirmed that the generated heuristics are better than the ones obtained in [109] and some human-based heuristics. Last but not least, we were able to generate more general heuristics which are not dependent on the training instances. A Kruskal-Wallis test confirmed that the generated heuristic have similar median performance on all instances.

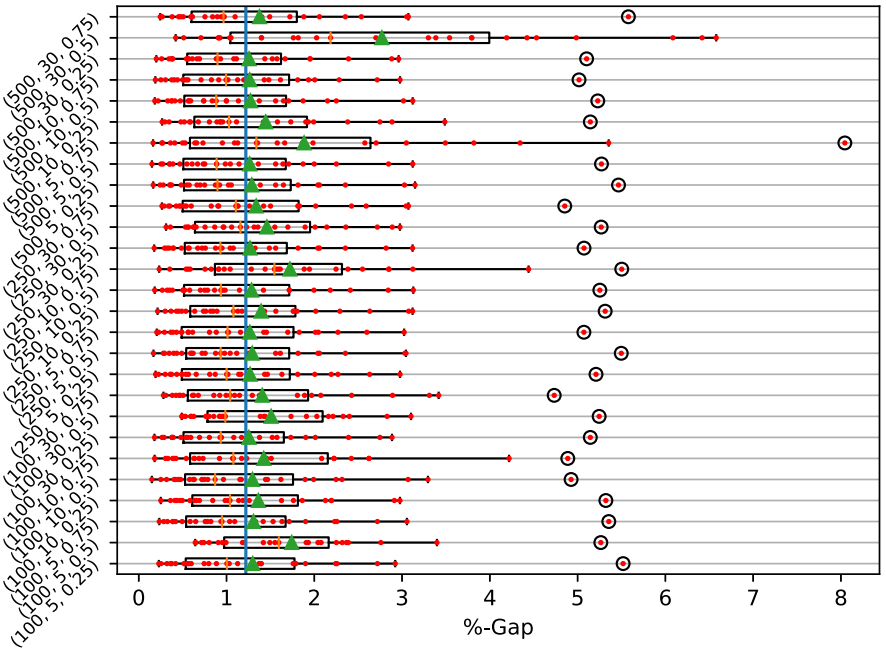


FIGURE 5.2: Performance of the generated heuristics on all 270 instances

5.6 Conclusion and Perspectives

Through this chapter, we investigate a new alternative inspired by machine learning and relying on the paradigm "Learning to optimize". The mainstream consists in generating automatically heuristics using GP hyper-heuristic to automatically solve combinatorial problems. We retained Multi-dimensional 0-1 knapsack instances as benchmarks to test the proposed approach. GP hyper-heuristics are recent hyper-heuristic approaches based on heuristic generation contrary to heuristic selection. Relying on Genetic Programming, they make possible to train heuristics or parts of heuristics. In this chapter, only a part of a greedy/constructive heuristic is learned, namely the scoring functions that allow to find an inserting order in the sacks and are the major component of such greedy heuristics. By defining new terminal sets and functions sets, a genetic programming algorithm has been applied on training instances in order to find efficient and discriminative scoring functions. After the training phase, all functions have been embedded into a greedy heuristic template to become real heuristics.

The proposed GP hyper-heuristic is based on [109]. Despite the novelty of this work, we observed that the author's approach yields heuristics working only on instances having the same number of constraints as the ones used for training. It would be more valuable to automatically create heuristics for any instance size of the MKP. In order to enable the creation of such heuristics, more general features should be provided to the training algorithm, i.e., GP. For example, an LP relaxed solution provides very relevant information and could be used as additional input data in the GP.

Here, our main concern was to adapt GP hyper-heuristic to tackle any instance size by redefining more general terminal and function sets. We applied strictly the same experiment protocol defined in [109] and showed that the new terminal set enable to tackle any MKP instance size efficiently.

To measure their performance, test instances which have been not provided to the GP hyper-heuristics during training have served to compute a performance measure, i.e., %-gap. Results have shown that the new terminal sets and function sets improved the %-gap and outperformed the state-of-the-art approach. In addition, we proved empirically that it is possible to train heuristics on small instances size and then apply them on larger instances while obtaining the same performance. Experiments described in this work have shown that the trained heuristics captured enough information to solve efficiently instances of various size even if they never encountered them during the training phase. These promising results indicate us the necessity to perform some kind of feature detection in optimization problems in order to define appropriate terminal sets and improve further this automatic mechanism for generating heuristics.

In the next chapter, we will apply the strategy of "learning heuristics" to a large scale and combinatorial bi-level problem, i.e., the Bi-level Cloud Pricing Problem. The mainstream is to exploit this strategy on the combinatorial lower-level problem in order to obtain a fast but efficient heuristics set able to tackle any of its lower-level instances that can be yield during bi-level optimization.

Chapter 6

The Cloud Bi-level Pricing: a large scale and combinatorial bi-level problem

Contents

6.1 Introduction

In chapter 4, we have observed that surrogate-based or model-based approaches are very good candidates to reduce dramatically the numerous lower-level optimizations that makes bi-level optimization so time consuming. We contributed to this strategy by demonstrating that bayesian optimization could be extended to solve general but continuous bi-level optimization problems. Indeed, such algorithms are generally employed for black-blox problems where the evaluation of a solution can be very time consuming. In fact, bi-level problems can be considered as an indirect black-blox problem since the Inducible Region ($\mathcal{IR}$), i.e., the bi-level feasible search space is not know and has to be discovered during optimization. In spite of their promising results, surrogate-based optimization approaches have some drawbacks that restrict their use to small and continuous bi-level problems. Large scale and combinatorial bi-level problems such as the Bi-level Cloud Pricing Optimization Problem (BCPOP) introduced in this chapter are inaccessible for surrogate-based algorithms.

The BCPOP is a mixed-integer bi-level problem that belongs to the category of Pricing problems (see chapter 2). It has been designed in the context of this thesis to help cloud service providers to find the best selling prices according to the market. The advent of Cloud Computing [268] has radically changed IT services market where nearly everything can be now considered as a service [153]. Cloud Computing definitely contributes to shift the market from monopolistic to perfect competition between cloud service providers and should theoretically be a benefit for cloud service consumers. However, this shift reduces the differentiation between service prices increasing the complexity of the market. In order to stand out of the crowd in a very competitive market, cloud service providers introduced new services in the form of bundles. Although bundles allow cloud service providers to create new distinctive offers, it renders cloud service decision-making more difficult. Indeed to cover their need at minimal cost, cloud service customers have to make combinations of bundles to cover all their IT needs. On the opposite side, cloud service providers would like to determine the best prices for the proposed bundles. They should be competitive to other offers while maximizing profits. Generally, these two optimization problems are solved independently from each other whereas they belong in fact to the same global problem. The latter can be modeled as a bi-level optimization problem, i.e., the BCPOP. In the context of cloud pricing, a cloud service provider would like to determine the best prices for its bundles on the market. It assumes that cloud service customers are rational and wish to minimize their cost while satisfying all their needs. Setting high prices will discourage cloud service customers to select the proposed offers. However, low prices may not be sufficient to maximize the profits. In such a situation, prices are strongly dependent on customers choices. Hence, cloud service providers can only determine the best prices if they are able to *measure* or *forecast* the reactions of the customers to the prices.

A Cloud service provider wants to find the highest prices while being sure that customers will still be attracted by the bundle offers. The BCPOP models accurately this situation and is definitely more realistic. The counterpart is that the problem resolution is more difficult. Indeed, the profits depends on the customer rational reactions that should be forecast/computed by the provider in order to determine the relevance of its proposed prices. For this purpose, the provider not only solves its own problem but also has to solve the customer problem as many times as it wishes to evaluate the impact of the prices. It has to put itself in the "shoes of the customers" by solving the same problem to obtain a certain degree of confidence. The BCPOP is an example of bi-level pricing problem that often occurs when a first entity influences the objective value of a second one. This very specific category of bi-level problems occurs naturally in economic, social and management problems.

Numerous approaches have been proposed in the literature to solve mostly small-scale and continuous bi-level optimization problems. On the contrary, very few algorithms have been designed to cope with combinatorial versions even if they are the most encountered in real applications. The last decade has seen a renewed interest for bi-level optimization, especially in the field of evolutionary computing. Indeed, the new IT technologies creates a real need to tackle multi-level optimization. The complexity of a bi-level optimization problem is strongly related to the complexity of its lower-level problem. In the case of an $\mathcal{NP}$-hard lower-level problem, only heuristics and meta-heuristics are generally suitable. In some cases, they also reach some limitations.

For example, let us consider population-based meta-heuristics to solve both levels. A trivial way to apply these meta-heuristics is to consider two populations, i.e., one at each level. Each time, an upper-level solution has to be evaluated, the meta-heuristic at lower-level is called to determine the lower-level rational reaction. Suppose that the upper-level population contains 1000 individuals, 1000 lower-level instances have to be solved for each generation. This implies to call 1000x the meta-heuristic at lower-level to solve each of them. This configuration is time consuming and should be definitely avoided.

It is unlikely that an existing heuristic or metaheuristic obtains the same efficiency on all lower-level instances that can be encountered during bi-level optimization. Some of them can be very hard to optimize. As a results,

the estimation of the optimal lower-level solution value can be very inaccurate leading to some illusory results when the upper-level objective value is computed. Due to the numerous lower-level optimizations to perform, we cannot guarantee that a heuristic nor a metaheuristic solve with the same efficiency each lower-level instances. Therefore, we propose to utilize Machine Learning concepts to train heuristics with the aim of obtaining high generalization abilities. We do not want an algorithm performing very well on some instances and poorly on some others. We would like to be on average as "efficient" as possible.

Since nested optimization can be very time-consuming, we decide to train greedy heuristics using a GP hyper-heuristic algorithm. Greedy heuristics are easy to train since their main mechanism relies on a scoring function that permit to rank decision variables. As an example, we can cite the scoring function of the knapsack problem that ranks items according to a ratio computed between the item value and its weight. Hereafter, we automatically generate heuristics for the lower-level instances as done in chapter 5. Bi-level optimization is a very specific task which requires to have some knowledge about the lower-level problem. Therefore, we try to extract automatically this knowledge using a learning mechanism. These trained heuristics will then serve to create a more robust and stable solver for lower-level instances. The contributions of this work are two-fold. We first investigate the automatic generation of heuristics (AGH) for the lower-level associated with the BCPOP. Once generated, we compare the new heuristics with human-based ones to check their relevance. Once this first task has been performed, we take advantage of these new heuristics to create an hybrid genetic algorithm (GA+AGH) that can solve efficiently BCPOP instances. To validate our methodology, we propose to compare the performance of the trained heuristics against two human-based heuristics and one of the rare bi-level metaheuristic that can tackle large-scale and combinatorial bi-level problems, i.e., COBRA. The latter is a state-of-the-art algorithm that adopts a promising decentralized strategy to tackle bi-level problems. We choose COBRA since it is also an alternative to the classical nested optimization.

The remainder of this chapter is organized as follows. The next section introduces formally the BCPOP and a remainder on GP hyper-heuristic is provided in Section 3. Section 4 details the proposed methodology to train lower-level heuristics for large-scale bi-level problems. Experiment designs are discussed in section 5 while section 6 analyzes the results. Finally, we conclude this chapter and presents some perspectives.

6.2 BCPOP: mathematical formulation and properties

Kieffer et al. in [206] introduced this bi-level problem in the context of Cloud Pricing. Indeed, determining the right pricing in the Cloud is a delicate task. Nevertheless, pricing problems can be easily modeled as bi-level optimization problems as done for the toll setting Problem in [53]. In this problem, a network of roads is operated by an authority who set toll prices. Naturally, this authority would like to determine the optimal prices knowing that customers will try to minimize their travel cost. High toll prices would lead drivers to take secondary roads which could be then saturated them. This authority should determine the optimal threshold which should not be overcome. The decision vector is not fully controlled by any part. Kieffer et al. relied on such a bi-level modeling for the BCPOP.

In the BCPOP, a cloud service provider is searching for an optimal pricing leading cloud service consumers to buy its bundles while maximizing its own profit. Since the market is very competitive, it has to take into account the different competitors. It definitely knows that setting prices too high will lead cloud service consumers to buy bundles from its competitors while setting prices too low will not provide maximum profits. In order to model this problem into a bi-level optimization problem, we first need to set the two-level decision makers. The upper-level decision maker is obviously the cloud service provider looking for an optimal pricing. The lower-level decision makers will be represented by cloud service customers who want to satisfy their IT

needs while minimizing their total cost. For the sake of simplicity, we consider a single rational cloud service customer. Indeed we assume independent cloud service customers whose purchase have no consequences on the others (e.g. no service limitations, no collective purchases). In this work, the single lower-level decision maker wishes to select a subset of bundles on the market at minimal cost. This problem can be modeled as a covering optimization problem (see Program 6.1).

$$\begin{aligned} \min \quad & f = \sum_{j=1}^{M} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{M} q_j^k x_j \geq b^k \quad \forall k \in \{1, ..., N\} \\ & x_j \in \{0, 1\} \quad \forall j \in \{1, ..., M\} \end{aligned}$$

PROGRAM 6.1: A covering optimization problem

Variables $x_j \in \{0, 1\}$, $\forall j \in \{1, ..., M\}$ represent the lower-level decision variable indicating whether or not the bundle j is bought by the cloud service customer. c_j represents the price of bundle j. b^k, $\forall k \in \{1, ..., N\}$ represent the service requirements. They indicate the number of services k which should be covered by the solution, i.e., a set of chosen bundles. q_j^k represent the number of services k occurring in the bundle j. In addition, q_j^k, $\forall k \in \{1, ..., N\}$ is a column-vector of size N corresponding to the distribution of all services in bundle j. If the cloud service provider wants to determine whether its prices are bi-level optimal, it must forecast the rational cloud service customer reaction which is determined by the optimal solution obtained from the resolution of Model 6.1. Without loss of generality, suppose that the first L bundles belong to the cloud service provider and the remaining bundles on the market are part of the competitors' offer. The upper-level decision maker wishes to determine all c_j, $\forall j \in \{1, ..., L\}$ in order to maximize its own profit but its also has to solve the cloud service customer related instance (see Model 6.1) in order to determine $x_j \in \{0, 1\}$, $\forall j \in \{1, ..., M\}$ and evaluate its objective function $F = \sum_{j=1}^{L} c_j x_j$. Model 6.2 depicts the resulting bi-level model for the BCPOP and Table 6.1 summarizes its components.

$$\begin{aligned} \max_{c_j, j \in \{1,...,L\}} \quad & F = \sum_{j=1}^{L} c_j x_j \\ \text{s.t.} \quad & c_j \geq 0 \qquad \forall j \in \{1, ..., L\} \\ \textbf{lower - level} \quad & \min_{x_j, j \in \{1,...,M\}} \quad f = \sum_{j=1}^{M} c_j x_j \\ & \text{s.t.} \quad \sum_{j=1}^{M} q_j^k x_j \geq b^k \qquad \forall k \in \{1, ..., N\} \\ & x_j \in \{0, 1\} \qquad \forall j \in \{1, ..., M\} \end{aligned}$$

PROGRAM 6.2: Bi-level Cloud Pricing Model

It is a bi-level mixed-integer program where the upper-level decision variables are continuous values while the lower-level decision variables are strictly binary values. These two heterogeneous levels imply that the resulting Inducible Region ($\mathcal{IR}$) will be made up of discontinuous and non-monotonic piecewise hyperplanes. Even if the upper-level problem consists in finding the optimal prices, i.e., continuous variables, the lower-level problem is a combinatorial problem.

TABLE 6.1: Description of Model 6.2

Definition	**Description**
$x_j \in \{0,1\}$	Lower-level decision variables
$c_j,\ \forall j \in \{1,...,L\}$	Upper-level decision variables
$c_j,\ \forall j \in \{L,...,M\}$	Prices of competitor products
$b^k,\ \forall k \in \{1,...,N\}$	Service requirements
q_j^k	Number of services k occurring in the bundle j
$F = \sum_{j=1}^{L} c_j x_j$	Upper-level objective function
$f = \sum_{j=1}^{M} c_j x_j$	Lower-level objective function
$\sum_{j=1}^{M} q_j^k x_j \geq b^k$	Lower-level covering constraint

6.3 GP Hyper-heuristics

Hereafter, we consider heuristic generation using genetic programming and referred to as "GP hyper-heuristics". A GP hyper-heuristic starts with a random population of heuristics which are evolved in order to obtain heuristics with high resolution abilities (see Algorithm 8). The GP evolves the population of heuristics to solve a predefined set of instances, i.e., the *training set*. The obtained heuristics are also wished to be able to tackle efficiently new encountered instances by using the knowledge gained during the training phase. This work-flow is very similar to supervised learning algorithm which learns on a training data and validates this same training on test data. Nonetheless to qualify GP hyper-heuristics as supervised learning methods, we should normally know the optimal solution of the training instances. For that, two possible choices exist. One can either consider small training instances and solve them to optimality in order to have a strict supervised knowledge or consider another approach which optimizes while learning. Drake in [109] chose this second approach which does not rely on any a priori knowledge about the optimal solution. In this work, we consider this second approach as well.

Algorithm 8 Genetic programming hyper-heuristic

```
population ← gen_ramped_half_and_half(NPOP,min,max)
for ind in population do
    ind.fitness ← evaluate(ind,training_instances)
end for
while gen ≤ NGEN do
    parents← selection(population)
    offsprings ← ∅
    for ind in parents do
        if random() ≤ CXPB then
            mate ← sample(parents,1)
            offspring1,offspring2 ← crossover(ind,mate)
            offsprings ← offsprings ∪ {offspring1,offspring2}
        else if random() ≤ CXPB+MUTPB then
            mutant ← mutation(ind)
            offsprings ← offsprings ∪ {mutant}
        else
            repro_ind ← copy(ind)
            offsprings ← offsprings ∪ {repro_ind}
        end if
    end for
    for ind in offsprings do
        ind.fitness ← evaluate(ind,training_instances)
    end for
    population ← offsprings
end while
return population
```

6.4 Proposed methodology

Our solution considers a very promising approach relying on hyper-heuristics and more precisely on heuristic generation. Instead of continuously solving from scratch the lower-level instances parametrized by the prices. We would like to train heuristics on a set of lower-level instances to improve their performance and maximize their abilities to solve unseen lower-level instances. We chose heuristics due to the numerous and promising works on heuristic generation for combinatorial problems. We claim that it is possible to solve approximately this lower-level problem quickly and more efficiently by training dedicated heuristics for it. The number of possible lower-level instances is infinite here. Indeed, the lower-level problem is parametrized by the prices obtained at upper-level and these prices are continuous values. Therefore, we train heuristics on a subset of lower-level instances.

6.4.1 Training heuristics

To conceive a set of efficient heuristics, GP hyper-heuristic will serve as learning model. Introduced by Koza [13], Genetic Programming makes use of a tree-based encoding representation and was initially designed to evolve functions and programs. *Terminal set* and *Function set* are very important components of GPs. They are nodes and leaves of each solution represented as a syntax tree. The concept behind GP hyper-heuristic is similar to a Machine Learning model. To obtain a final population of heuristics with high abilities of *approximation*, the algorithm learns from a set of various instances, i.e., the *training set* and validate the heuristics on a *test set*. In this work, we also rely on a *non-supervised* approach in which training and lower-level optimizations are done simultaneously because the optimal solutions are unknown.

Instead of generating heuristics from scratch, a specific greedy heuristic template is selected and only relevant sub-parts are evolved through GP, i.e., the ones impacting heuristics' results. For the lower-level covering problem, *a greedy heuristic template consisting in sorting all decision according to a scoring function*. This greedy heuristic template starts from a solution where all bundles have been selected. Then the heuristic tries to remove bundles with regards to the ranking induced by the scoring function. Two phases can be therefore distinguished: the ranking phase and the deletion phase.

While the deletion phase is common to all generated heuristics, the ranking phase has a key and specific element: **the scoring function**. In this case, an easier and more efficient way to train heuristics is to train scoring functions instead.

6.4.2 Scoring functions: training and evaluation

According to the GP hyper-heuristic mechanism (see Section 6.3), an initial population of scoring functions is generated. The encoding of each scoring function is represented by a syntax tree (see Figure 6.1) where the nodes and leaves represent elements that characterize the instances (e.g., objective coefficient, matrix coefficient, right-hand side members). These elements are described in Table 7.1 and are referred to as "Terminals". In order to combine these elements together, we need mathematical operators (e.g., addition, subtraction, division). They basically represent the intermediate nodes of syntax trees while terminals represents the key elements of the considered problem.

The population is then evolved using traditional GP evolutionary operators (see Figure 6.2). As it can be observed in (see Algorithm 10), the heuristic template takes a scoring function as parameters. Therefore, the

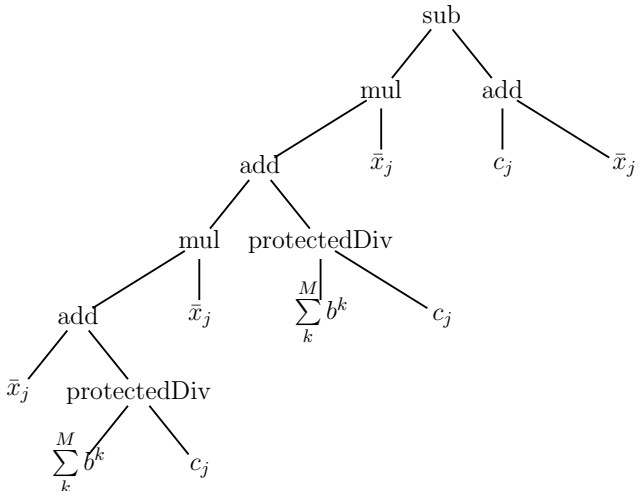


FIGURE 6.1: Example of a scoring function obtained in this work

TABLE 6.2: Functions and terminal sets

Name	Description
Operators	
add	Add two inputs
sub	Subtract two inputs
mul	Multiply two inputs
protectedDiv	Divide two inputs with protection
mod	Modulo b.t.w. two inputs with protection
Terminal sets/ Arguments	
c_j	Cost of the current bundle j
$avgDiff_j = \frac{\sum_k b^k - q_j^k}{M}$	Average difference between the capacity and the resource consumption for item j
$\sum_k q_j^k$	Total services for bundle j
$\max_k q_j^k$	Max services for bundle j
$\sum_k b^k$	total services requirement
$\bar{x}_j$	Solution value for bundle j after LP relaxation

fitness value of a specific scoring function can then be obtained by solving all instances using the template parametrized by this same scoring function. The sum of all objective values obtained on the training set (see Algorithm 9) becomes the fitness value of the scoring function and measure its abilities to solve the training instances.

As described above, the heuristic template solves a training instance using the scoring function that ranks the bundles (lower-level binary decision variables). This ranking mechanism separates interesting bundles from unattractive ones. After the first phase, the template removes the bundles with regards to the ranking while minimizing the objective value.

It is noteworthy that a large number of terminals could penalize the training in terms of processing time while leading also to over-fitting. Terminal selection is therefore a critical task which should shorten training time and reduce over-fitting. Additionally, it can be interesting to study other related works in the literature to determine the key elements (see Figure 6.2). As a consequence, we added a prior knowledge on the location of the optimal solution by providing the solution of the LP relaxation which is clearly a key element for our training purpose. The LP relaxation in this work consists in replacing the lower-level binary decision variables by continuous ones while bounding all of them in the interval $[0, 1]$. The relaxed problem is then solved using a linear solver. This is the solution of this relaxed problem that is added to the Terminal set described in Table 7.1. The latter could be further enhanced by finding new elements leading to a better discrimination of bundles. For example, we could consider the solution of the dual variables after LP relaxation.

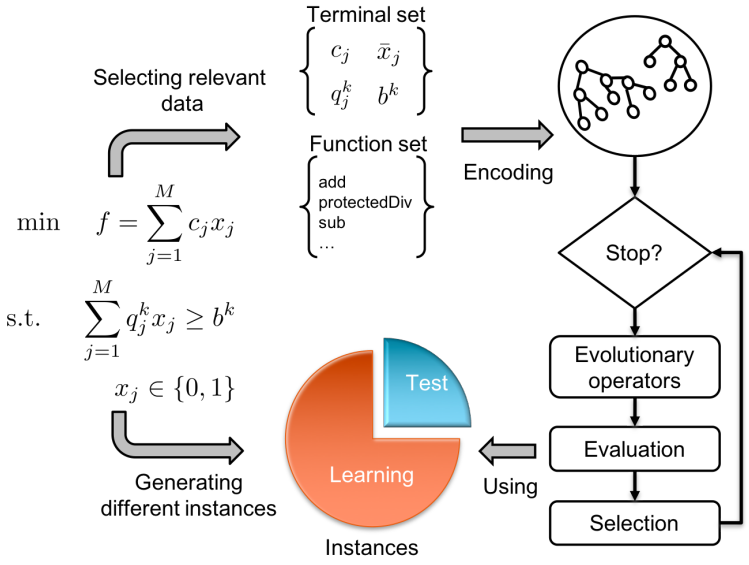


FIGURE 6.2: GP hyper-heuristic workflow

Algorithm 9 evaluate(scoring_function,training_instances)

```
1: total_costs←0
2: for all instance in training_instances do
3:     heuristic ← heuristic_template(scoring_function)
4:     heuristic_solution_value ← heuristic(instance)
5:     total_costs ← total_costs + heuristic_solution_value
6: end for
7: return total_costs
```

Algorithm 10 heuristic_template(scoring_function)

```
1: Take all existing bundles into solution
2: Sort all bundles according to scoring_function
3: while true do
4:     Try to remove a bundle according to the pre-computed order
5:     if If no bundle has been removed then
6:         Break;
7:     end if
8: end while
9: return solution value
```

In order to validate the proposed approach on BCPOP instances, the next section is dedicated to the description of the experiments that have been performed in this work.

6.5 Experimental design

6.5.1 Training

We first train a set of heuristics using a classical GP hyper-heuristic algorithm on lower-level instances. The lower-level problem is a parametric covering problem with non-binary coefficient matrix. Despite investigations, we did not find any library proposing instances for such covering problems. Instead of generating them from scratch, we turned our attention to the OR-library [34]. This library provides various instances for different combinatorial problems. The closest problem with such non-binary matrix coefficients and binary decision variables is the Multi-dimensional Knapsack Problem (MKP). We therefore modified the MKP instances found at the OR-library (http://people.brunel.ac.uk/ mastjjb/jeb/info.html) such that all $\leq$-constraints becomes $\geq$-constraints. We also ensure that each modified instance has a non-empty search space. All instances used in this work can be found at https://gitlab.uni.lu/ekieffer/instances.

We considered 9 different instance size with M constraints and N lower-level variables (N, M): (100;5), (100;10), (100;30), (250;5), (250;10), (250;30), (500;5), (500;10), (500;30). Each class contains 30 instances. To obtain a robust learning and avoid over-fitting, we decided to use the cross-validation method as validation technique [216]. For each training, the cross-validation approach generates a new training set and a new test set. The advantage of the cross-validation approach lies in the fact that each instance belongs at least once to the training and the test sets.

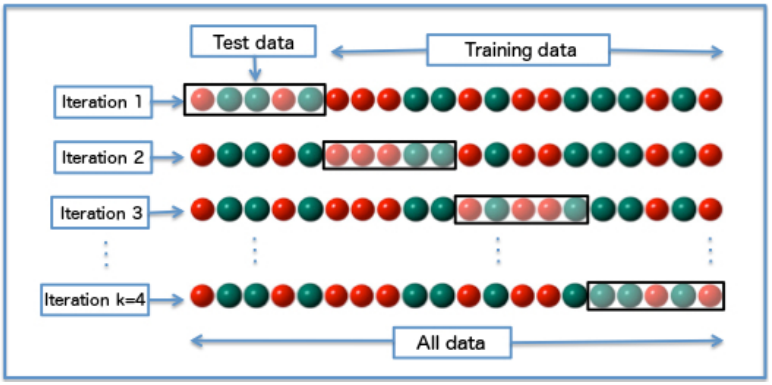


FIGURE 6.3: The cross-validation technique (source: https://www.wikipedia.org)

Cross-validation relies on the number of folds (K) to consider, we experimentally set this parameter. Using $K = 5$ means that 5 different trainings are performed for each class of instances. For each training, one fold is considered as a test set and %-gaps are only computed on test folds to reflect the abilities of the new generated heuristics to solve new instances. To summarize, we have 9 classes of instances and each class experiences 5 trainings with different training and test sets. Therefore, we should obtain 5 heuristics per class of instances, i.e., one per fold. In the end, we only retained the best one among the 5 folds.

Concerning the training algorithm, i.e., GP hyper-heuristic, the population size has been set to 100. 85% of the programs will be subject to crossover while 10% will be mutated. Finally, 5% of the program is not modified by evolutionary operations at each generation. Tree initialization method implements the well-known "ramped half-half" approach and we set a depth limit of 17. One point crossover has been chosen while the "grow" approach has been retained as mutation operator. Table 6.3 sums up all the GP parameters and GP operators considered for these experiments.

TABLE 6.3: GP parameters and operators

Generations	50
Population size	100
Crossover Probability (CXPB)	0.85
Mutation Probability (MUTPB)	0.1
Reproduction Probability	0.05
Tree initialization method	Ramped half-and-half
Selection Method	Tournament selection with size=7
Depth limitation	17
Crossover Operator	One crossover point
Mutation Operator	Grow

6.5.2 Application of the new heuristic onto the BCPOP

In this section, we focus on the description of BCPOP instances and their optimization. To create these instances, 9 random lower-level instances, i.e., one per classes have been generated from scratch. These instances

have only one level and only represent the lower-level problem. We recall that the lower-level problem consists in buying a sub-set of bundles on the market while minimizing its own costs.

To obtain BCPOP instances, we have to consider an additional level related to the upper-level decision maker who wishes to maximize its own profit. The upper-level decision maker would like to determine the optimal pricing for its bundles. According to the problem description in Section 6.2, we only need an upper-level fitness function which should maximize the profit obtained by the cloud service provider. Therefore, we first have to select a subset of bundles from the lower-level instances that belong to the upper-level decision maker. In this work, we randomly selected 25% of the bundles for each considered instance. The remaining bundles, i.e., 75% belongs to the competitors.

In bi-level optimization, we face multiple lower-level optimizations. We do not want to find an algorithm that is particularity good on a single instance but an algorithm that is "robust" for very different lower-level instances. Indeed, an algorithm with wide performance variations on different lower-level instances would be disastrous for the optimization and could lead to illusory results at upper-level.

For this purpose, we employ a hybrid algorithm that combines a genetic algorithm (GA) for the upper-level problem and the trained heuristics for solving lower-level instances. This global and hybrid algorithm is referred to as GA+AGH. The genetic algorithm aims at determining the best prices for the bundles for a lambda cloud service provider that wishes to maximize its profits. This GA therefore implements a population of individuals representing the prices of the bundles brought on the market. In order to evaluate each individual, the GA calls first a heuristic on the individual to obtain the reaction of a cloud service customer to the prices, i.e., all x_j decision with $j \in \{1, ..., L\}$. Once this reaction is known, the individual can be evaluated. The GA repeats this scheme for all individuals while targeting maximum profits.

For the sake of completeness and in order to measure the efficiently of the trained heuristics, we propose also to use 4 other algorithms (heuristics and metaheuristics): (GA+ H1), (GA+H2), (GA+H1H2), (GA+H1H2) and COBRA.

- GA+H1 combines the same classical GA with the H1 heuristic found in the literature. H1 is a rounding heuristic which requires to solve the continuous LP relaxation first. Then, the method iterates over the solution and set $x_j = 1$ if $\hat{x}_j \geq \frac{1.0}{\max_k \sum_j q_j^k}$ else 0.
- GA+H2 combines the same classical GA with the H2 heuristic also found in the literature. H2 is a greedy heuristic which sorts all bundles according to the following and static scoring function $s(x_j) = \frac{c_j}{\sum_k q_j^k}$. Then bundles are selected according to this order until all constraints are satisfied.
- GA+H1H2 uses the same classical GA for the upper-level problem and a combination of H1 and H2 for the lower-level problem. H1H2 can be considered as an hyper-heuristic relying on heuristic selection.
- COBRA [235] attempts to solve bi-level optimization problems by separating the upper-level from the lower-level problem. The co-evolution mechanism implemented in COBRA (see Algorithm 11) exchanges individuals from both populations, namely the upper-level population and the lower-level population. An archiving approach is embedded as well to keep track of the best individuals generated along the evolution at both levels. Its pseudo-code is described by Algorithm 11.

We adopted the parameters described in Table 6.4 for the implementation of (GA+AGH), (GA+H1), (GA+H2), (GA+H1H2) and COBRA . Despite their differences, we set the total number of upper-level fitness evaluations to 50000 for all algorithms at both levels. These algorithms perform the same number of fitness evaluations at

Algorithm 11 COBRA algorithm [235]

```
pop ← create_initial_pop()
pop_upper ← copy_upper(pop)
pop_lower ← copy_lower(pop)
while stopping criterion is not met do
    upper improvement (pop_upper) and lower improvement (pop_lower)
    upper archiving (pop_upper) and lower archiving (pop_lower)
    selection (pop_upper) and selection (pop_lower
    coevolution(pop_upper,pop_lower)
    adding from upper archive (pop_upper) and from lower archive (pop_lower)
end while
return lower archive
```

upper-level and at lower-level. COBRA requires a second population to solve the lower-level problem. Thus, the encoding differs from the upper-level.

TABLE 6.4: Parameters used for both Bi-level evolutionary approaches

	GA+(AGH;H1;H2;H1H2)	COBRA
Independent runs	30	30
UL encoding	real values	real values
UL Population size	100	100
UL fitness evaluations	50000	50000
UL Archive size	100	100
UL Fitness evaluations	50000	50000
UL Selection	Binary Tournament	Binary Tournament
UL Crossover operator	Simulated binary	Simulated binary
UL Crossover probability	0.85	0.85
UL Mutation operator	Polynomial	Polynomial
UL Mutation probability	0.01	0.01
LL encoding	—	binary values
LL fitness evaluations	50000	50000
LL Archive size	–	100
LL Selection	–	Binary Tournament
LL Crossover operator	–	Two-points
LL Crossover probability	–	0.85
LL Mutation operator	–	swap
LL Mutation probability	–	$\frac{1}{\#variables}$

It is not trivial to compare two bi-level evolutionary algorithms since results depend on how each algorithm solves lower-level instances. According to the bi-level definition (see Chapter 2), two bi-level solutions $S_1 = (x_1, y_1)$ and $S_2 = (x_2, y_2)$ can be compared if and only if both solutions are bi-level feasible, i.e. they belongs to $\mathcal{IR}$ unless they share the same upper-level decision, i.e., $x_1 = x_2$. When handling constraints, metaheuristics often used some kind of distance to the feasible search space to penalize non-feasible solutions. For bi-level optimization problems, it is very difficult to obtain such distance measure since bi-level feasibility implies lower-level optimality. For comparing two bi-level solutions, we should be able to measure how well they approximate lower-level optimality. Since each upper-level decision x generates a new lower-level instance, the lower-level optimal value should be taken relatively to the upper-level solution. Therefore, it makes no sense to compare two lower-level fitness values coming from different algorithms. We should not compare the absolute lower-level fitness values but the distance to their respective optimal value. This is the reason why we have chosen the %-gap that is defined as follows:

$$\%\text{-}gap = 100 * \frac{\overline{v}_h - \underline{v}^{lp}}{\overline{v}^{lp}} \tag{6.1}$$

In total, 30 runs are performed for each BCPOP instance and all algorithms. All the experiments have been conducted on the High Performance Computing (HPC) platform of the University of Luxembourg [357]. The python library DEAP [108] has been considered for the GP implementation.

6.6 Experimental results

Tables 6.5, 6.6 and 6.7 represent the %-gap obtained for each class of instances on each of the 5 test folds. For example, Fold 1 in Table 6.5 with instances (N=100 ; m=5) have an average %-gap of 2.25. This means that the best heuristic obtained through the learning process on the training instances from the union of fold 2 to fold 5 provided an %-average gap on the test instances of Fold 1 equals to 2.25. Therefore for each class, 5 heuristics are trained, i.e.one per fold. For Table 6.5, a total of 45 heuristics have been trained and generated through the GP. In order to fairly evaluate the performance of the novel heuristics, we also apply the H1 and H2 heuristics on the same test folds for each class of instances. Table 6.6 depicts the performance of the H1 heuristics while Table 6.7 described the performance of the H2 heuristic. We can easily observe that the average %-gap for all classes and folds is better, i.e., 1.69 than the one obtained for the H1 heuristic, i.e., 7.68 and H2 heuristic, i.e., 9.03. Additionally for each class of instances, the average performance on the 5 folds is always better for the trained heuristics. Figure 6.4 described the average performance of the three type of heuristics for each fold. The non parametric kruskal-Wallis test [222] has also been performed to statistically determine if the results obtained on the 5 folds differ for each heuristic type. The 5 tests, i.e., one per fold provided very low p-values. Assuming an α-value=0.05, we can conclude that the performance of the three heuristics on the folds are statistically different. Fig. 6.4 clearly shows that the trained heuristics outperforms H1 and H2 in terms of gap on all classes of instances. These results confirm that we can automatically create efficient heuristics using some prior knowledge from the lower-level problem. These heuristics can also be more efficient than the human-based ones as shown in Figure 6.4.

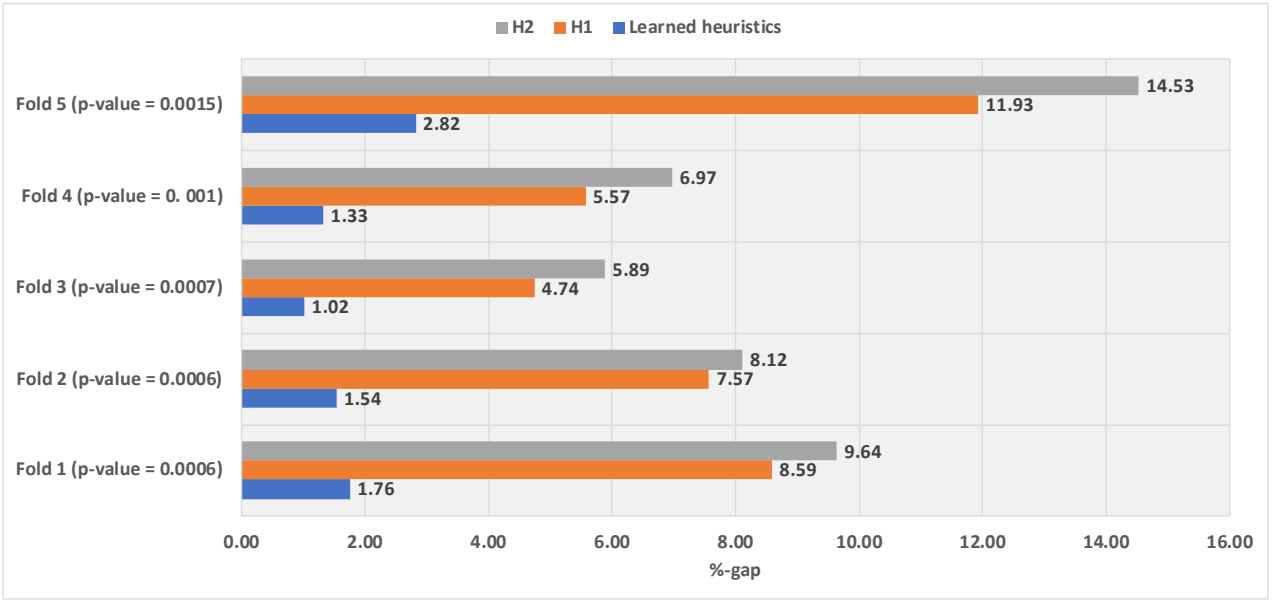


FIGURE 6.4: Performance of the heuristics in %-gap for each fold

Table 6.8 reports the average lower-level %-gap of the best bi-level solution obtained over the 30 runs for each random instances and for each algorithm. This average lower-level %-gap measures the abilities of a bi-level solution (x, y) to solve the lower-level problem parametrized by x. For example, the average %-gap achieved for

Table 6.5: Automatic heuristic generation performance on each fold

		Average %-gap					
# Variables	# Constraints	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
100	5	2.42	1.82	1.31	1.33	4.36	2.25
100	10	0.85	0.87	0.36	0.67	1.27	0.80
100	30	0.37	0.35	0.18	0.23	0.61	0.35
250	5	3.14	2.22	1.57	2.51	5.08	2.90
250	30	1.24	1.26	0.74	0.89	1.67	1.16
250	10	0.46	0.60	0.40	0.48	0.94	0.58
500	5	4.30	3.87	2.57	3.63	6.73	4.22
500	10	2.00	1.88	1.34	1.39	2.94	1.91
500	30	1.02	0.99	0.68	0.81	1.78	1.05
average		1.76	1.54	1.02	1.33	2.82	1.69

Table 6.6: H1 performance on each test fold

		Average %-gap					
# Variable	# Constraint	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
100	5	9.37	9.02	6.39	5.13	15.54	9.09
100	10	6.78	4.88	3.41	4.61	9.34	5.80
100	30	4.86	3.87	2.44	3.16	5.81	4.03
250	5	12.31	12.42	6.37	9.15	19.28	11.90
250	30	9.78	7.22	4.30	4.22	10.91	7.29
250	10	5.04	5.11	3.01	3.78	8.25	5.04
500	5	13.82	11.73	6.74	9.08	16.70	11.61
500	10	9.59	7.72	6.34	6.84	11.75	8.45
500	30	5.75	6.17	3.69	4.16	9.80	5.91
average		8.59	7.57	4.74	5.57	11.93	7.68

Table 6.7: H2 performance on each test fold

		Average %-gap					
# Variable	# Constraint	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
100	5	8.03	6.10	4.19	4.20	11.70	6.85
100	10	3.62	3.04	1.85	2.24	3.42	2.83
100	30	1.79	1.51	0.99	1.17	2.31	1.55
250	5	13.63	10.74	8.68	9.52	21.53	12.82
250	30	4.77	5.45	3.76	3.94	9.56	5.49
250	10	3.00	3.00	1.94	2.01	4.05	2.80
500	5	29.14	22.81	17.15	22.17	45.66	27.39
500	10	14.77	13.01	9.51	10.99	20.44	13.74
500	30	8.02	7.38	4.91	6.50	12.10	7.78
average		9.64	8.12	5.89	6.97	14.53	9.03

the BCPOP instance (100;5) is 3.60 for GA+AGH, 14.32 for GA+HA1, 23.84 for GA+H2, 18.39 for GA+H1H2 and 9.71 for COBRA. According to the p-values obtained after performing Kruskall-Wallis tests (or one-way ANOVA on ranks)[222], the results are statistically different for all approaches on each instance. We can observe that the solutions obtained by GA+AGH offer better %-gaps at lower-level for all instances. In addition, one can also notice that the standard deviations ($\pm std$) are significantly lower for the GA+AGH algorithm. This implies that during bi-level optimization, lower-level instances are solved efficiently using the trained heuristics. Thus, the results confirm and validate our expectations in terms of heuristic efficiency. If we observe COBRA, we can see that the distance to optimality at lower-level is very large.

COBRA separates both levels, performs some independent improvements for each level and finally employs a co-evolutionary operator to share information between levels. This strategy presents some risks because the links between the upper-level and lower-level decision variables are strongly epistatic. In addition, both levels have negatively correlated objective functions. The exchange of information performed by the co-evolutionary operator does not guarantee any common improvement for both levels. COBRA could basically alternate good upper-level solutions then good lower-level solutions without finding a compromise between both. Figure 6.5 depicts such a situation that occurred during the experiments described in this work. COBRA also assumes that the upper-level solution and lower-level solution are fully compatible which is not necessarily the case. As illustrated by Figure 6.5, small gap variations at lower-level can generate high variation at upper-level.

Concerning the human-based heuristics used in combination with the GA, we can observe that they provide better results than COBRA except for the instance (100,10) and (100,3). In general, we observe that none of the human-based heuristics obtained better results on all instances. Compared to GA+AGH that has been trained, the variability of the results is much higher for the GA+H1, GA+H2 and GA+H1H2. These results give us confidence about the abilities of the trained heuristics to solve efficiently various lower-level instances which is a key of major importance for bi-level optimization.

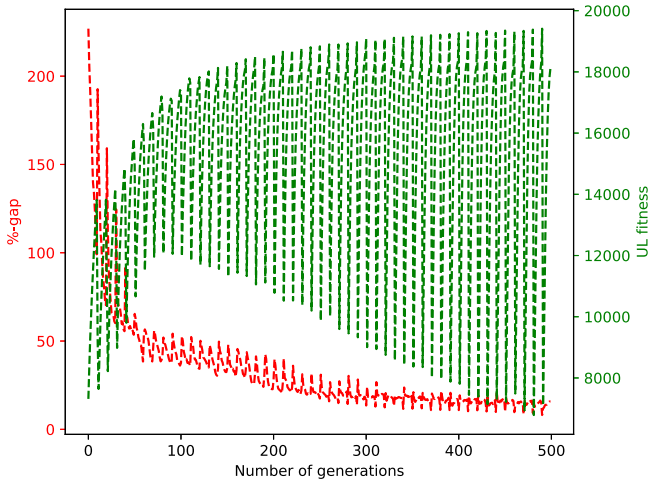


FIGURE 6.5: Example of convergence issues impacting COBRA

Due to the $\mathcal{NP}$-hard lower-level embedded in the BCPOP problem, finding a bi-level feasible solution is $\mathcal{NP}$-hard as well. As consequence, all solutions presented in this work are by definition not bi-level feasible since this feasibility is only ensured when lower-level instances are solved to optimality. The average %-gap observed in Table 6.8 measures the average distance to lower-level optimality of the best solutions obtained by each algorithm. As a consequence, we can conclude that the solutions obtained by GA+AGH are closer to $\mathcal{IR}$ bi-level feasible search space) that the other alternatives shown in Table 6.8. From a practical point of view, the lower-level decision maker buys bundles while minimizing its costs. It is highly important for the upper-level

decision maker to be able to predict the lower-level rational reaction and propose a realistic pricing. Therefore the upper-level decision maker should be able to solve the parametric lower-level problem and get a solution as close as possible to the lower-level rational reaction. Indeed approximating the lower-level rational reaction leads to a solution which over-estimates the expenses of the lower-level.

TABLE 6.8: %-gap

		%-GAP to lower-level optimality					
# Var	# Cstr	GA + AGH	GA + H1	GA + H2	GA + H1H2	COBRA	**P-value**
100	5	3.60 ±0.69	14.32 ±1.84	23.84 ±1.20	8.39 ±4.88	9.71 ±8.96	$\leq$ 1e−6
100	10	4.30 ±0.98	22.26 ±3.45	41.54 ±4.09	37.92 ±4.41	12.33 ±11.01	$\leq$ 1e−6
100	30	8.15 ±1.39	52.62 ±4.36	28.68 ±0.96	37.90 ±13.4	23.31 ±8.18	$\leq$ 1e−6
250	5	1.44 ±0.49	6.48 ±0.96	14.81 ±3.88	6.35 ±1.32	25.19 ±7.11	$\leq$ 1e−6
250	10	1.97 ±0.48	0.49 ±1.11	17.09 ±2.91	12.02 ±2.22	26.08 ±5.50	$\leq$ 1e−6
250	30	3.22 ±0.45	23.23 ±1.82	21.67 ±3.87	24.03 ±2.41	27.75 ±5.30	$\leq$ 1e−6
500	5	0.45 ±0.15	3.55 ±0.57	18.22 ±1.62	13.64 ±2.20	30.07 ±6.41	$\leq$ 1e−6
500	10	0.94 ±0.28	5.85 ±0.52	19.75 ±2.63	10.18 ±3.59	34.68 ±4.84	$\leq$ 1e−6
500	30	1.46 ±0.34	12.18 ±1.42	16.73 ±1.94	13.35 ±1.49	35.19 ±4.47	$\leq$ 1e−6
Average		2.84 ±0.58	16.77 ±1.78	22.48 ±2.57	19.03 ±3.99	24.92 ±6.86	

With regard to the previous observations, we now focus on Table 6.9 that reports the average upper-level objective values achieved by each algorithm for each class of instances. Again, we can observe that the results are significantly different according to Kruskal-Wallis tests (see P-values). We recall that the upper-level objective function is maximized. The highest results are obtained by COBRA over the 9 instances. In these circumstances, we could wrongly assume that COBRA obtained the best bi-level results. Nonetheless, let us consider now both the gap and the upper-level function value for all algorithms. We can clearly observe that the smallest upper-level objective values are obtained by GA+AGH while having the best lower-level %-gaps. On the contrary, COBRA has the worst %-gap results but highest upper-level objective values. Therefore, we could suppose that COBRA over-estimates the upper-level revenues if the lower-level approximations are too wide. We observe the same trend with GA+(H1,H2,H1H2) except that their values lie between the results of GA+AGH and COBRA.

In order to prove theoretically and experimentally that COBRA over-estimates the upper-level objective value, we consider an alternative but equivalent formulation of the general bi-level optimization problem described in Chapter 2. The max-min nested structure of the BCPOP has some interesting properties that are worth mentioning. First, let us consider an alternative but equivalent reformulation:

$$max\{F(x,y)| \quad f(x,y) \leq w(x)\} \tag{6.2}$$

$$\text{with} \quad w(x) = min\{f(x,y)|g(x,y) \leq 0\} \tag{6.3}$$

with $x \geq 0$ and $y \geq 0$. $w(x)$ is the optimal lower-level solution value for the upper-level decision x.

Now suppose that we have an approximation of the lower-level optimal solution with regard to x: $\bar{w}(x)$ and a relaxation bound with regard to x: $\underline{w}(x)$. Since the lower-level is a minimization problem, we have $\underline{w}(x) \leq w(x) \leq \bar{w}(x)$. At this point, consider the following sets:

$$\begin{aligned} S_{approx} &= \{f(x,y) \leq \bar{w}(x)\} \\ S_{opt} &= \{f(x,y) \leq w(x)\} \\ S_{relax} &= \{f(x,y) \leq \underline{w}(x)\} \end{aligned} \tag{6.4}$$

We can then affirm that $S_{relax} \subset S_{opt} \subset S_{approx}$ and therefore:

$$max\{F(x,y)|S_{relax}\} \leq max\{F(x,y)|S_{opt}\} \leq max\{F(x,y)|S_{approx}\} \tag{6.5}$$

Any resolution approaches that tend to approximate the lower-level problem leads to an upper-level problem that is less constrained. If the upper-level is then less constrained, it can be considered as relaxed. Therefore, $max\{F(x,y)|S_{opt}\} \leq max\{F(x,y)|S_{approx}\}$. On the contrary, any approach that yield a relaxation bound of the lower-level problem leads to an upper-level that is more constrained. As a consequence, $max\{F(x,y)|S_{relax}\} \leq max\{F(x,y)|S_{opt}\}$.

In this work, all algorithms presented here solve approximately lower-level instances. Consequently, we are in this situation: $max\{F(x,y)|S_{opt}\} \leq max\{F(x,y)|S_{approx}\}$. A bad approximation can lead to a high upper-level objective value and is not representative of what could really happen. Since all the presented algorithms are approximations, we claim that the better lower-level gaps obtained by GA+AGH provide more confidence regarding the results obtained at upper-level. To show the implications of approximating lower-level instances on the upper-level objective function, we need to bound the optimal lower-level solution. We can achieve it using $max\{F(x',y)|S_{relax}\}$ and $max\{F(x',y)|S_{approx}\}$ for a given upper-level solution x' and measure how far they are from each other.

Table 6.9 already provides us the average $max\{F(x',y)|S_{approx}\}$ for each instances and algorithms with x' the best upper-level solution found during bi-level optimization. Therefore, we only need to compute $max\{F(x',y)|S_{relax}\}$ using the same upper-level solution x' . For this purpose, S_{relax} is obtained after continuous relaxation of the corresponding lower-level instance that is parametrized by x'. This allows us to measure how far are the results for all algorithms to a respective lower-bound. We define this measure as the upper-level gap (UL-gap) for a given upper-level solution and is formally expressed as follows:

$$UL\_gap(x') = \frac{\max\{F(x',y)|S_{approx}\} - \max\{F(x',y)|S_{relax}\}}{\max\{F(x',y)|S_{relax}\}} \tag{6.6}$$

with x' the considered upper-level solution. A large UL-gap would mean that the algorithm is not accurate and over-estimates the true objective value at x' since it is far from the lower-bound obtained by solving $\max\{F(x,y)|S_{relax}\}$. Table 6.10 summarizes all UL-gaps obtained for the different algorithms. Notice here that we selected the best upper-level solution from the results obtained in Table 6.9.

TABLE 6.9: UL objective values

		UL objective value					
#Var	#Cstr	GA + AGH	GA + H1	GA + H2	GA + H1H2	COBRA	**P-value**
100	5	12336.98 ±260.13	12952.95 ±236.62	13243.43 ±173.61	13024.36 ±210.72	14710.78 ±928.98	≤ 1e−6
100	10	10195.56 ±290.37	11166.93 ±461.89	12836.03 ±214.86	12788.70 ±172.13	15226.79 ±713.71	≤ 1e−6
100	30	11464.33 ±484.66	13998.58 ±411.50	13382.70 ±779.61	13535.58 ±253.20	14762.83 ±863.36	≤ 1e−6
250	5	28604.54 ±966.96	29046.60 ±848.13	27773.38 ±1323.17	29387.63 ±1246.92	35479.64 ±2147.84	≤ 1e−6
250	10	29391.77 ±661.76	30889.73 ±554.45	30442.85 ±1052.13	29963.24 ±829.11	38283.71 ±1740.42	≤ 1e−6
250	30	28094.64 ±454.11	31388.80 ±578.49	29186.03 ±576.81	30059.45 ±570.91	39368.26 ±1891.78	≤ 1e−6
500	5	52487.52 ±1092.99	53665.68 ±1110.01	56653.52 ±1902.37	55059.31 ±902.05	73529.34 ±3887.39	≤ 1e−6
500	10	53800.62 ±1467.47	55651.07 ±1135.35	54750.17 ±1006.22	54683.81 ±1462.71	75041.02 ±3277.08	≤ 1e−6
500	30	53735.45 ±903.92	57604.96 ±1047.51	58430.30 ±1117.32	56323.13 ±781.44	75386.02 ±3815.59	≤ 1e−6
Average		31123.49 ±731.37	32929.48 ±709.32	32966.49 ±905.12	32758.36 ±714.35	42420.93 ±1922.34	

TABLE 6.10: Upper-level gap based on the best solution obtained at upper-level

		UL gap * 100				
#Var	#Cstr	GA + AGH	GA + H1	GA + H2	GA + H1H2	COBRA
100	5	9.47	16.79	43.90	26.85	915.40
100	10	12.77	31.53	154.93	110.00	1116.58
100	30	20.36	66.95	83.26	69.96	341.21
250	5	2.32	7.43	23.45	6.45	347.08
250	10	3.63	12.43	33.13	14.57	362.88
250	30	7.29	26.41	51.12	23.59	401.98
500	5	1.62	4.50	36.64	13.36	394.26
500	10	1.52	6.51	38.18	10.72	362.94
500	30	3.35	12.94	33.52	14.63	335.04
Average		4.14	13.15	40.58	18.17	386.29

As expected, COBRA obtained a very high gap contrary to the other implementations. GA+AGH has tighter gap except for the instance (100,30). In fact, all algorithms have difficulties for this very specific instance. Not surprisingly, GA+H1H2 has better results than GA+H2 but worse than GA+H1. The difference between the results proposed by COBRA and the bounds computed directly through the UL-gap shows how a bad approximation can have strong implications on the upper-level objective function. One should not forget that the algorithm maximize the profits. If the profits are too optimistic, the solver will continue to misinterpret results and will increase the deviations. This justify and validate our first proposition to train heuristics for unseen lower-level instances with the aim of making them less sensitive and more general (robust) to solve different lower-level instances.

6.7 Conclusion and Perspectives

In this work, we use the concept of GP hyper-heuristic to train efficient heuristics with the aim of tackling a large-scale and combinatorial bi-level problem, i.e., the Bi-level Cloud Pricing Optimization Problem (BCPOP). Few approaches are able to deal with such a large-scale and combinatorial bi-level problem, especially when the lower-level problem is $\mathcal{NP}$-hard. Generally, nested optimization combining heuristics and meta-heuristics at both levels are employed to obtain approximations. However, an approximation of lower-level instances leads to solutions that are generally not bi-level feasible, i.e., they do not belong to the Inducible Region ($\mathcal{IR}$). Therefore, it is highly important to ensure that the generated solutions are located as close as possible to $\mathcal{IR}$ to avoid "illusory" results at upper-level. Due to the various lower-level instances that could be encountered, it is difficult to obtain the same efficiency on all these instances. For this purpose, we relied on machine learning concepts to train greedy heuristics in order to be as "close" as possible to $\mathcal{IR}$.

Using a greedy heuristic template, we only trained scoring functions which permits to rank decision variables, i.e., bundles. Training only the key elements of an heuristic shorten time processing by reducing the search space. A genetic programming algorithm then evolved the scoring functions which are placed into the template for evaluation. We classified the instances by size and learn them using cross-validation, i.e., a technique widely employed in machine learning. Training heuristics for the lower-level problem allows us to obtain more robust heuristics. Numerical experiments have been performed on BCPOP instances to evaluate the potential of the new trained heuristics to tackle lower-level instances. We also compared the results with two human-based heuristics, a selection-based hyper-heuristic and a bi-level metaheuristic, i.e., COBRA.

Results have shown that the trained heuristics can lead to superior results in terms of distance to lower-level optimality which was the primary goal. Furthermore, we show the impact on the upper-level objective value and demonstrated that an inaccurate approximation can lead cloud service providers to over-estimate their final profits. In some cases, this over-estimation can lead to a worst case, i.e., no sold bundles. Future work will attempt to extend the GP hyper-heuristic approach to other multi-level optimization problems.

As it has been shown in this chapter, bi-level problems can be tackled in two phases. The first one tackles the parametric lower-level problem by training heuristics for it. Then, the second phase relies on classical metaheuristics (e.g. evolutionary algorithms) to tackle the original bi-level problem. The next chapter will attempt to gather these two phases in a single one in order to train heuristics and solve the bi-level problem at the same time. For this purpose, we will use a modified co-evolutionary approach.

Chapter 7

CARBON, a hybrid co-evolutionary Bi-level Optimization algorithm

Contents

7.1 Introduction

In Chapter 5, we have raised the question of generating automatically dedicated heuristics for combinatorial optimization problems. We contributed to the field of Hyper-heuristic by proposing to tackle the MKP instances with a new set of terminals that allow to learn any instance size. As mentioned in the previous chapters, many approaches have been proposed in the literature to solve continuous bi-level optimization problems. On the contrary, very few algorithms have been designed to cope with combinatorial versions even if they are the most encountered problems in real applications. Metaheuristics have been proposed to cope with bi-level optimization complexity. Nevertheless most of them are based on a nested optimization scheme which repeatedly solves one level after the other. This scheme is very time consuming. Chapter 5 laid the foundations of the "Learn to optimize" paradigm using GP Hyper-heuristics that we implemented for bi-level optimization in chapter 6 to approximate lower-level instances. Nested optimization for bi-level problem requires sequential optimization of both levels. Every time, the upper-level decision maker sets its decision, a new lower-level instance is generated and has to be solved in order to get the optimal lower-level solution. Instead of generating these instances iteratively, we generated a training set of instances and trained efficient heuristics to solve them. Then, these

new heuristics are used as lower-level solver into the evaluation operator of a classical genetic algorithm. Notice that any other metaheuristics (e.g. simulated annealing, tabu search)could have been considered as well. The key of this contribution relied on the fact that we extracted a set of potential lower-level instances and then learned a set of fast and dedicated set of heuristics before solving the bi-level problem.

The approach developed in chapter 6 has demonstrated that the upper-level decision maker should first design a strategy to estimate accurately the response of the lower-level decision maker. Nonetheless, the proposed approach has two static phases. Once the heuristics are trained, they do not evolve anymore which can be problematic if they were not able to capture all the necessary knowledge to solve the various lower-level instances encountered during bi-level optimization. It could be more interesting to adopt a dynamic strategy gathering both phases,i.e, the heuristics could be trained during bi-level optimization.

For this purpose, we rely in this chapter on a modified *co-evolution* approach. Indeed, the main issue with classical co-evolutionary scheme in bi-level optimization is the strong epistatic links between the upper-level and lower-level decision variables. The nested structure is a drawback since each upper-level decision leads to a different lower-level instance and thus search space. Consequently, some upper-level and lower-level solutions may not be compatible when they are paired. In order to obtain independence between the two populations, we can consider another approach which consists in having one population representing the upper-level decisions and another one representing a set of heuristics which are evolved to solve any kind of lower-level instances. We not only evolve the upper-level decision but also the abilities of heuristics to solve lower-level instances. We chose to evolve greedy heuristics since they are fast, easily modifiable and have been proved efficient in chapter 6. This strategy permits therefore to hybridize the co-evolutionary and the "learning to optimize" paradigm. We experiment our new approach on the same Bi-level Cloud Pricing instances considered in chapter 6. We compare our numerical results against a classical bi-level co-evolutionary algorithm, i.e., COBRA.

The remainder of this chapter is organized as follows. We first explain the principle of "co-evolution". Then, we discussed "co-evolution" in bi-level optimization in section 3. Section 4 introduces CARBON, i.e., the proposed competitive and hybrid co-evolutionary algorithm. Experiment setups and results are discussed in section 5. Finally, we close this chapter by providing our conclusions.

7.2 Co-evolution

Co-evolution occurs when intimate species influence each other's evolution. In his book "Origin of Species" [91], Charles Darwin already observed the evolutionary interactions between some species such as plants and insects. Nonetheless, Ehrlich and Raven [117] were the first to introduce formally the term "co-evolution". They also demonstrated its essential role in evolutionary transitions which show that the evolution of a specie can be dramatically influenced when an external selection pressure is applied by other species. Several co-evolutionary models have inspired the evolutionary computing field. The predator/prey model (see Figure. 7.1) is a famous example. We generally distinguish two classes of co-evolutionary algorithms: competitive and cooperative. Competitive co-evolution has been first proposed by Hillis in [168] for Sorting Networks. Then many applications stemmed from his results and notably in Game Theory where players often compete. Competition focuses on the abilities of species to evolve and develop new skills to outperform another specie during a so called "armed race". On the contrary, cooperative co-evolution relies on the skills emerging from species tending to collectively work when facing a common problem as a team work (see Figure 7.2). The seminal work of Potter and De Jong [303] showed how cooperative co-evolution can be employed to optimize multi-dimensional functions. In [202], the authors used cooperative co-evolution as a constraint handling method by decomposing the constraint set through multiple populations. Finally, loosely coupled genetic algorithms

designed by Seredynski [325, 326] can be assimilated as a non-cooperative co-evolution and it is a kind of compromise between the two original models: *competitive* and *cooperative* co-evolution.

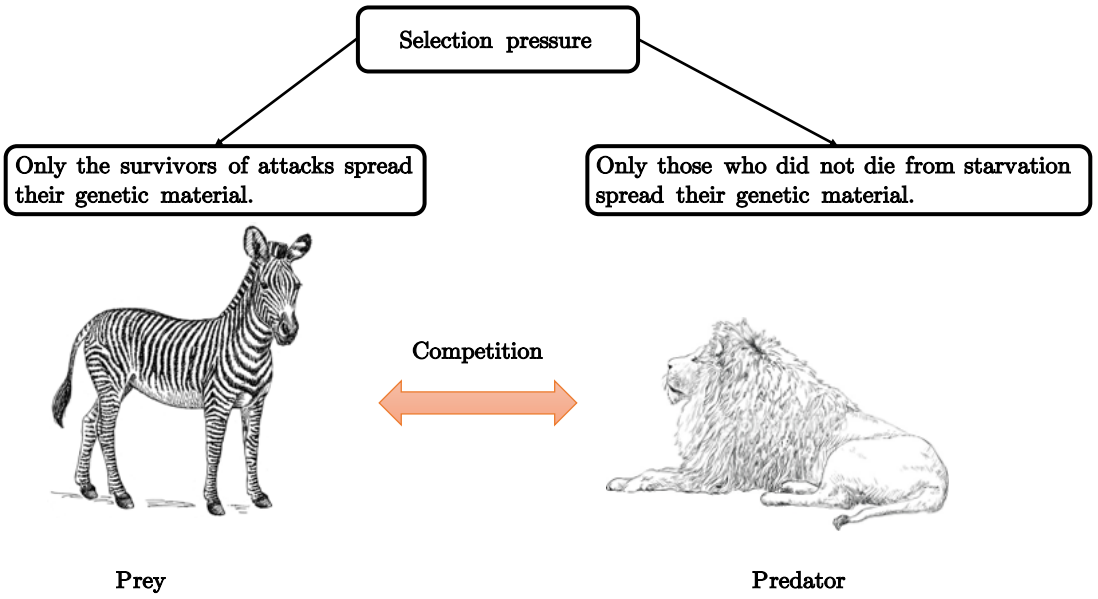


FIGURE 7.1: The prey/predator model of competitive co-evolution

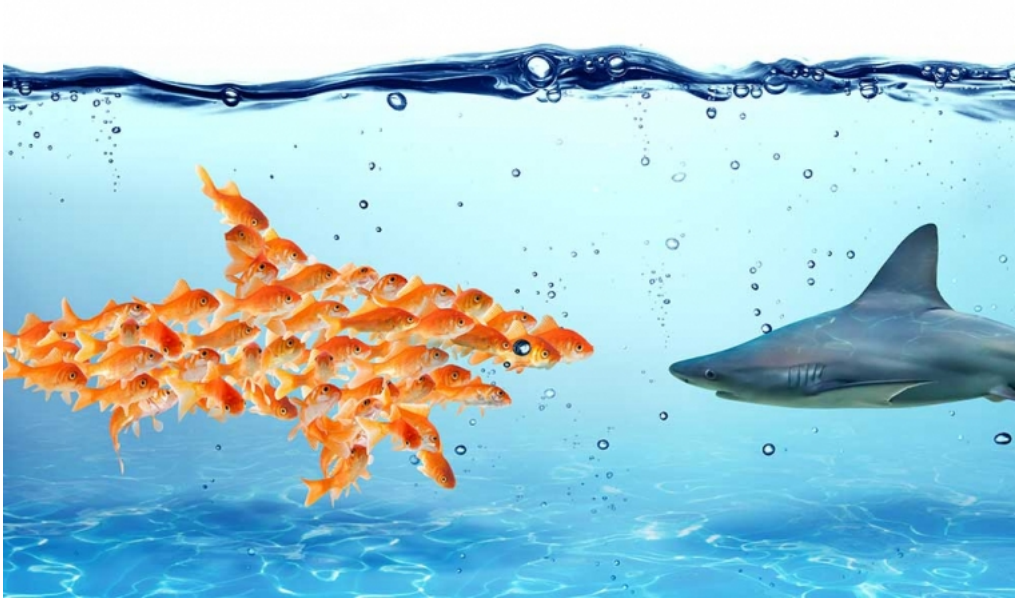


FIGURE 7.2: Cooperation co-evolution as team work source: https://thissolution.com/how-to-make-teamwork-work/

7.3 Discussion on classical co-evolution for bi-level optimization

Co-evolutionary algorithms in bi-level optimization have been discussed in section 2.5.3 of chapter 2. Two mains algorithms have been identified: BIGA [288] and COBRA[235]. Although BIGA employs an exchange of information between both levels, the strategy employed by Oduguwa and Roy is closer to a classical nested optimization scheme. In order to fix this issue, Legillon et al. developed COBRA by adding independent improvement phases for both levels. The exchange of information in COBRA only occurs when both populations have evolved separately during a specific number of iterations. The co-evolutionary mechanism is achieved by population recombination (see Figure 7.3).

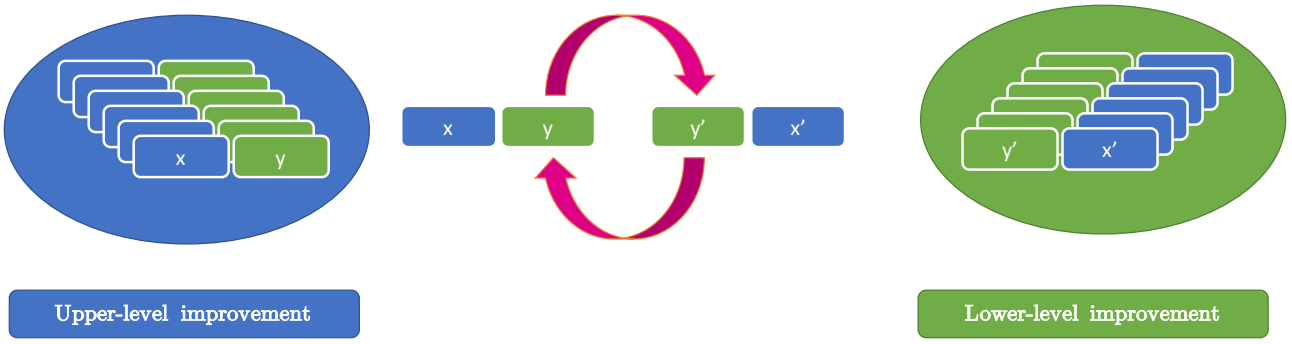


FIGURE 7.3: Population recombination as co-evolutionary mechanism

7.3.1 Convergence issues due to the co-evolutionary operator

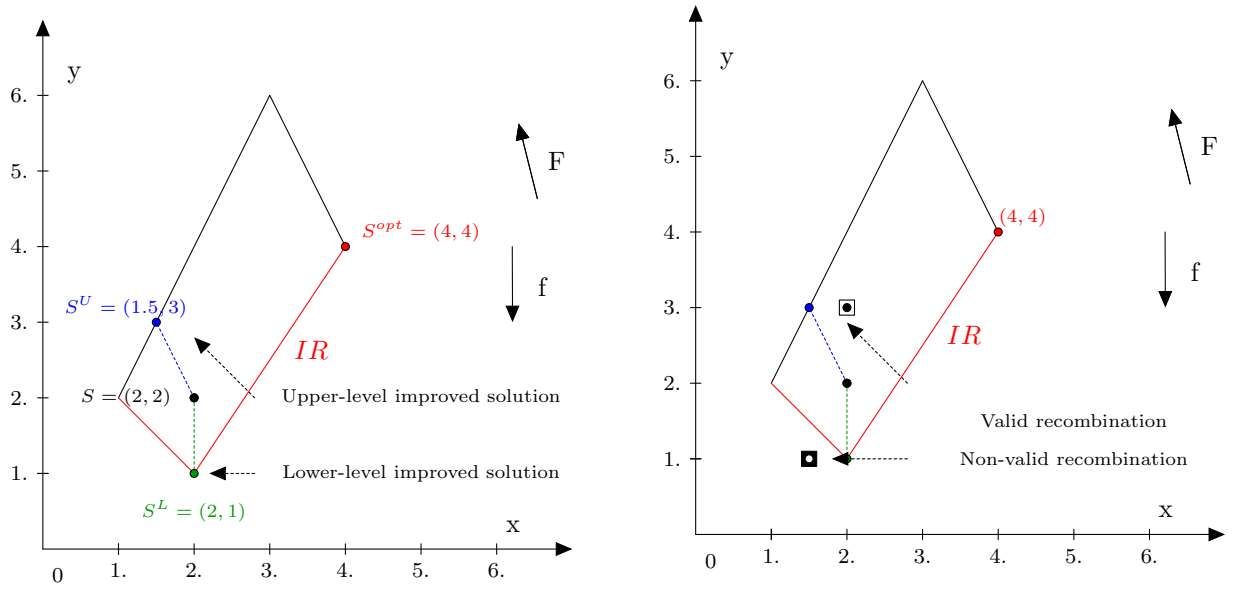
Co-evolution permits generally to tackle problems in a decentralized manner by decomposing them into sub-problems while keeping an exchange of information between them. This methodology can be negatively impacted when both sub-problems have strong epistatic links. The nested structure of bi-level optimization problems implies strongly epistatic links between the upper-level and lower-level variables. As we have observed in the previous chapter, the upper-level objective can only be evaluated using the decision obtained at lower-level. In addition, the lower-level problem remains a parametric problem until the upper-level decision maker sets its decision first.

To face this issue, the two populations of COBRA work on the entire bi-level solution, i.e., upper-level and lower-level solutions. While the upper-level population only evolves the upper-level part of the bi-level solution, the lower-level population evolves only on the lower-level part. In both populations, some parts remains fixed during optimization and are only modified when the co-evolutionary operator comes into play as described in Figure 7.4. Since both levels are competitive by nature, this exchange cannot guaranty a real improvement towards the optimal bi-level solution. It can even destroy any positive progresses at both levels leading to convergence issues.

For instance, let us consider Figure 7.4a that depicts the independent improvements of a solution $S = (2, 2)$ where the upper-level objective function is F and the lower-level objective function is f. These improvements give birth to two new solutions $S^U = (1.5, 3)$ and $S^L = (2, 1)$. When applying the co-evolutionary operator implemented in COBRA, the recombination can yield non-valid solutions as depicted in Figure 7.4b. Even valid solutions can be bad in term of lower-level quality. The solution S was definitely closer to $\mathcal{IR}$ than the valid recombination. In fact, there is no move that can improve both levels at the same time. As a consequence, this co-evolutionary operator may generate oscillating values that improve one level but degrade the other one.

7.3.2 Lower-level issues

Another interesting issue can be highlighted. As aforementioned, each upper-level decision leads to a specific lower-level instances with its own optimal solution. According to the bi-level definition (see chapter 2), two bi-level solutions $S_1 = (x_1, y_1)$ and $S_2 = (x_2, y_2)$ can be compared if and only if both solutions are bi-level feasible (1) unless they share the same upper-level decision, i.e., $x_1 = x_2$ (2) . Since bi-level feasibility implies optimality at lower-level, (1) is unlikely to happen in the case of a $\mathcal{NP}$-hard lower-level when it is solved by mean of heuristics or metaheuristics. As a consequence, two solutions such as $x_1 \neq x_2$ cannot be directly compared in terms of upper-level fitness. Let us turn our attention to Figure 7.5, the upper-level decisions $x_1 = 2$ and $x_2 = 3$ provide two different lower-level optimal solutions, namely $\hat{y}_1 = 3$ and $\hat{y}_2 = 6$. In fact, there is an infinite



(A) Independent improvement phase at each level

(B) Application of the co-evolutionary operator after improvements

FIGURE 7.4: Example of application of the co-evolutionary operator implemented in COBRA

number of lower-level instances that can be generated in the example of Figure 7.5. Therefore, the lower-level population in COBRA has to deal not only with a single lower-level instance but with many possible instances with different optimal solutions. To be more precise, two solutions in the lower-level population with different upper-level decision does not belong to the same lower-level problem instance. Consequently, the lower-level population can have difficulties to convergence since it optimizes many different lower-level instances at the same time. In fact, we should have as many lower-level populations as lower-level instances which is definitely impracticable.

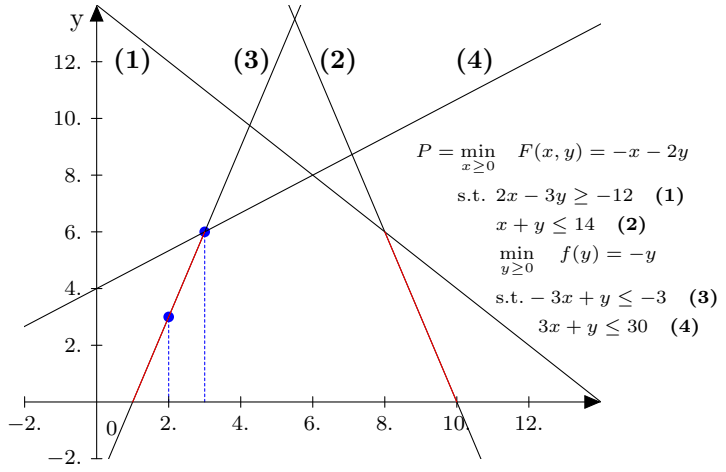
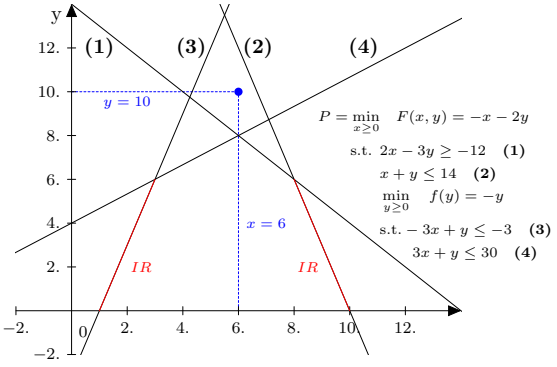


FIGURE 7.5: Bi-level problems can have an infinite number of lower-level instances with different optimal solution and value

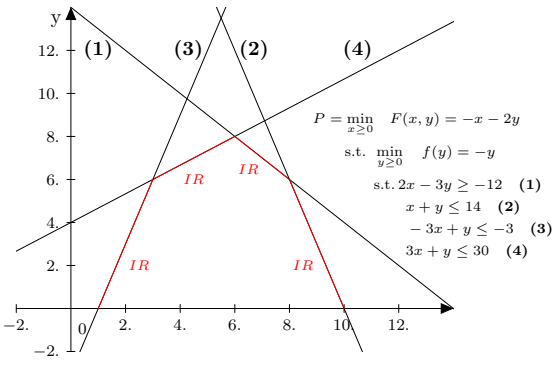
7.3.3 Classical feasibility issues

Finally, breaking the nested structure by optimizing independently both levels can raise feasibility issues. Indeed, it may be possible that pairing the two level decisions x and y provides a non-feasible solution. This can be easily seen on the example presented in Figure 7.6a and has also been discussed in chapter 2. Let us

suppose that a co-evolutionary algorithm optimize both levels in two separated populations as it has been done in the literature. Suppose that both levels provide $x = 6$ and $y = 10$ to be evaluated. Since the lower-level problem is indifferent to the upper-level constraints, the lower-level variables $y = 10$ is legal. Nonetheless such a lower-level decision make the global bi-level solution not feasible. As a consequence, two feasible solutions x and y for each separate level does not necessary provide a valid bi-level solution (x, y). To cope with such issue, some people proposed to push the upper-level constraints into the lower-level constraint set. Using such a strategy definitely changes the meaning of the original problem and would change the location of the bi-level optimal solution. Figure 7.6a depicts the original problem with upper-level constraints. The inducible region $\mathcal{IR}$ is a discontinuous piecewise function with an optimal bi-level solution located at $(x = 8; y = 6)$. On the contrary, Figure 7.6b represents the problem with no upper-level constraints constraints. They have been pushed to the lower-level. Notice now that the inducible region $\mathcal{IR}$ is no longer discontinuous and the bi-level optimal solution moved to $(x = 6; y = 8)$. As a consequence, reassigning constraints is a bad strategy that cannot be applied to control bi-level feasibility.



(A) Example of bi-level problem with upper-level constraints



(B) Upper-level constraints moved the lower-level constraint set

FIGURE 7.6: Example of problems where classical co-evolution can raise issues

In the next section, we propose a new methodology based on chapters 5 and 6 that will fixed such an issue and reveal the interest of co-evolution for bi-level optimization.

7.4 CARBON: a hybrid bi-level co-evolutionary algorithm

In the previous section, we introduced co-evolution and its application to bi-level problems. As we discussed it, co-evolution relies on the notion of independence while bi-level optimization models entangled choices between two decision makers. In this context, classical co-evolution as done in [235] could experience some troubles. In this chapter, we propose an alternative approach that permits to obtain two independent populations for each levels.

In chapters 5 and 6, we focused on the automatic generation of heuristics. Instead of working in the space of solutions, we went one layer of abstraction higher and consider heuristics as solutions. Such a strategy has been successfully employed to cope with the nested structure of large-scale and combinatorial bi-level problems such as the Bi-level Cloud Pricing Optimization Problem (BCPOP). We first train heuristics to tackle efficiently any lower-level instances and then embedded them into a genetic algorithm that tackle the upper-level. Heuristics are evolved only one single time before solving the upper-level. Therefore this approach can be considered as static and may reach some limitations when the numbers of upper-level decision variables in the parametric lower-level problem is large. Indeed this implies that the number of lower-level instances would be very large,

i.e., exponential. Learning only a static subset of instances would not be sufficient to train efficient heuristics for any lower-level instances. A solution for this issue would be to evolve the upper-level decision variables and the lower-level heuristics at the same time. For this purpose, co-evolution would be the most appropriate solution since upper-level variables and lower-level heuristics are not link together.

Hereafter, we introduce CARBON: a competitive hybrid co-evolutionary algorithm. In this co-evolutionary algorithm, two populations obey to the predator-prey model except that the second population will not directly evolve lower-level solutions but the mean to get to them. With this approach, we obtain a second population independent from any upper-level decision. The prey will be the upper-level decision variables and the predators will be the lower-level heuristics evolved to provide lower-level rational reactions with low gap. According to Figure 7.7, upper-level decision variables are evolved using evolutionary operators that can be found typically in Genetic Algorithm (GA) while heuristics are evolved using Genetic Programming (GP) operators. The goal of this second population is basically to train dynamically a set of accurate heuristics as it has been done in chapter 6.

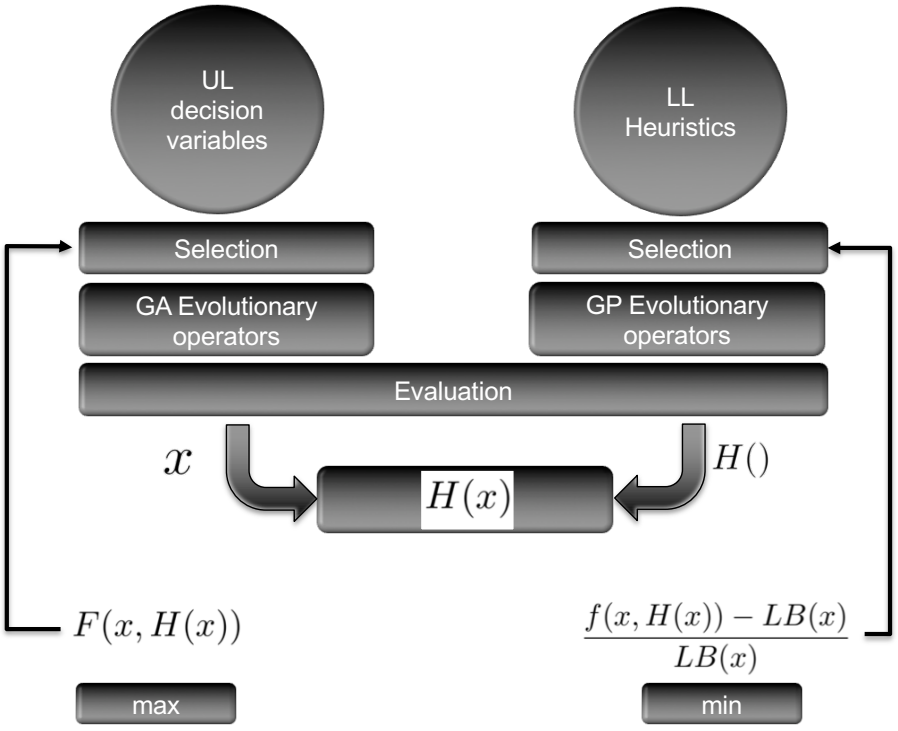


FIGURE 7.7: CARBON workflow

CARBON attempts to find the optimal upper-level solutions while finding the best strategies (heuristics) to determine the lower-level rational reactions. We need to adopt the upper-level decision maker point of view. When we face a competitive problem where the decision of the opponent has direct consequences on our payoff, we always wish to determine the impact of our decision first. Therefore, we look for strategies that can be adopted by this opponent. This concept is embedded in CARBON.

As for chapter 6, the original data from the lower-level problem formed the "Terminal set". The goal of the GP is to encode a population of scoring functions as a population of syntax trees where each terminal node is picked in the "Terminal set" and each intermediate node is picked in the "Operator set". According to Fig. 7.7, the upper-level population will evolve upper-level solutions and the lower-level population will evolve scoring functions. These functions, once embedded in the same template of greedy heuristic utilized in chapter 6, will constitute valid heuristics to solve the lower-level instances parametrized with the upper-level decisions. Table 7.1 describes the terminal and operator sets considered to solve the lower-level covering problem embedded in the

BCPOP constraint set. Notice that we consider the dual values and relaxed optimal solution after continuous relaxation. Indeed, continuous relaxation will be in any case computed since we require it to compute the lower-level gap which is the objective value of the second population (see Fig. 7.7).

Contrary to chapter 6 where heuristics have been trained in order to minimize the lower-level objective function, we decided to employ the gap as new objective function. We recall that the gap is computed as follows:

$$\%\text{-}gap(x) = 100 * \frac{A(x) - LB(x)}{LB(x)} \tag{7.1}$$

where $A(x)$ is the lower-level solution value obtained after applying an algorithm A on the upper-level decision x. $LB(x)$ is a lower-bound according to the upper-level decision x.

This choice is justified since each upper-level decision x generates a new lower-level instance, the lower-level optimal value should be taken relatively to x in order to obtain a good measure of bi-level feasibility.

Using the gap we can compare two solutions even if they did not stem from the same upper-level decision. The solution with the lowest gap indicates that the upper-level decision maker has a better estimation of the lower-level rational reaction.

TABLE 7.1: Functions and terminal sets implemented in this work

Name	**Description**
Operators	
+	Add two inputs
-	Substract two inputs
*	Multiply two inputs
%	Divide two inputs with protection
mod	Modulo b.t.w. two inputs with protection
Terminal sets/ Arguments	
c_j	Cost of the current item j
q_j^k	Distribution of service k in bundle j
b^k	Required number of service k
d_k	Dual value for service k after LP relaxation
$\bar{x}_j$	Solution value for bundle j after LP relaxation

In the next section, a comparison is provided between CARBON and COBRA (described in chapter 6). First, we introduce all parameters and the considered instances to perform the experiments. Then, we discuss the results and explain the main differences between these two bi-level co-evolutionary algorithms.

7.5 Experimental results

7.5.1 Setups and parameters

The set of BCPOP instances considered for these experiments are exactly the same that the ones defined in chapter 6. We recall that they are divided into 9 different types of instances with $N \in \{100, 250, 500\}$ decision variables and $M \in \{5, 10, 30\}$ constraints. As in chapter 6, 25% of the bundles belongs to the upper-level decision maker, i.e, the CSP. The remaining bundles belong to competitive providers on the market.

Concerning the algorithms, Table 7.2 describes all parameters used during the experiments. In order to fairly compare them, we adopted the same number of upper-level and lower-level fitness evaluations. COBRA implements archives at both levels to keep track of the best results. We also adopted this strategy in CARBON. The

only difference between these two algorithms occurs at lower-level since we employ GP operators and not GA operators for CARBON. Therefore, you can observe a reproduction operator which is very typical in GP.

TABLE 7.2: Parameters used for both Bi-level evolutionary approaches

	CARBON	COBRA
Independent runs	30	30
UL encoding	continuous values	continuous values
UL Population size	100	100
UL Archive size	100	100
UL Fitness evaluations	50000	50000
UL Selection	Binary Tournament	Binary Tournament
UL Crossover operator	Simulated binary	Simulated binary
UL Crossover probability	0.85	0.85
UL Mutation operator	Polynomial	Polynomial
UL Mutation probability	0.01	0.01
LL encoding	syntax trees	binary values
LL fitness evaluations	50000	50000
LL Archive size	100	100
LL Selection	Tournament	Binary Tournament
LL Crossover operator	(GP) One-point	(GA) Two-points
LL Crossover probability	0.85	0.85
LL Mutation operator	(GP) uniform	(GA) swap
LL Mutation probability	0.1	$\frac{1}{\#variables}$
LL Reproduction probability	0.05	–

Finally, experiments for both algorithms were carried out using the HPC facility of the University of Luxembourg [357]. The python library DEAP [108] has been considered for the implementation CARBON and COBRA.

7.5.2 Numerical results

After 30 runs for each algorithms on the 9 different instances, we recorded the best results in terms of %-gap (see Table 7.3) and upper-level fitness values (see Table 7.4). In both cases, we performed Kruskal-Wallis tests to determine if the difference between results are statistically significant. P-values can be observed in the last column of each table. Our first discussion will be devoted to results illustrated in Table 7.3. It can be easily observed that CARBON provides better lower-level solutions than COBRA. The %-gap which measures the distance from lower-level solutions to their continuous lower bound is much smaller for CARBON. These results indicates that upper-level decision maker can forecast more accurately the lower-level rational reaction of the lower-level decision maker. We recall that CARBON directly minimize the gap while COBRA minimize the lower-level objective value. In Bi-level Optimization, each upper-level decision x leads to a different lower-level instances with distinct lower-level optimal solution. The disadvantages of COBRA is a work-flow which does not take into account that many different lower-level instances stem from distinct upper-level decisions. On the contrary, CARBON does not rely on any lower-level solution value but on a relative measure of optimality,i.e., the gap. Therefore, whatever upper-level decisions you are considering, CARBON can evaluate how accurate are the resulting lower-level solutions and compare them in terms of distance to their respective lower-level optimality.

We also reported the results obtained by GA+AGH, i.e, the hybrid genetic algorithm designed in chapter 6. Interestingly, CARBON obtained better lower-level gap than AG+AGH. We recall that GA+AGH learn first a static set of lower-level heuristics and then apply a GA to the bi-level problem while CARBON dynamically generates lower-level heuristics through co-evolutionary competition. The advantage of CARBON over GA+AGH is that learning and optimization are performed at the same time.

TABLE 7.3: %-gap

		%-GAP to LL optimality			
# Variables	# Constraints	CARBON	COBRA	GA+AGH	**p-value**
100	5	1.02 ±0.26	9.71 ±8.96	3.60 ±0.69	$\leq$ 1e−6
100	10	1.91 ±0.33	12.33 ±11.01	4.30 ±0.98	$\leq$ 1e−6
100	30	3.32 ±0.60	23.31 ±8.18	8.15 ±1.39	$\leq$ 1e−6
250	5	0.34 ±0.08	25.19 ±7.11	1.44 ±0.49	$\leq$ 1e−6
250	10	0.76 ±0.08	26.08 ±5.50	1.97 ±0.48	$\leq$ 1e−6
250	30	1.44 ±0.15	27.75 ±5.30	3.22 ±0.45	$\leq$ 1e−6
500	5	0.14 ±0.02	30.07 ±6.41	0.45 ±0.15	$\leq$ 1e−6
500	10	0.34 ±0.04	34.68 ±4.84	0.94 ±0.28	$\leq$ 1e−6
500	30	0.97 ±1.17	35.19 ±4.47	1.46 ±0.34	$\leq$ 1e−6
Average		1.14 ±0.30	24.92 ±6.86	2.84 ±0.58	

Let us focus now on the upper-level fitness values reported in Table 7.4. A first observation let us believe that the upper-level objective value is much better for COBRA than CARBON. Indeed, the upper-level fitness value is higher for each instance. In the context of the BCPOP, it means that the pricing obtained with COBRA lead to a better payoff than for CARBON. In fact, this is not true. We cannot observe the upper-level fitness value without the gap values. COBRA led to larger gaps than CARBON. This means that COBRA has less accurate lower-level solutions than CARBON.

As we proved it in chapter 6 when we compared COBRA against GA+AGH, an inaccurate approximation of the lower-level conducts to a relaxation of the upper-level problem. In the context of the BCPOP, this means that a cloud service provider overestimates the number of bundles sold to the cloud service customers. As a consequence, this same provider over-estimates its final payoff. Only lower-level solutions that are close to the Inducible Region IR minimize the risk of overestimation error that can be disastrous for a provider.

We reported the results of GA+AGH in Table 7.4 and one can observed that CARBON has lower upper-level fitness value that GA+AGH which confirm our thought. Indeed since CARBON has better lower-level gap, its estimation of the number of bundles sold to customers is more accurate than the one of GA+AGH. In order to prove it, we rely on the UL-gap described in chapter 6 and defined as follows:

$$UL\_gap(x') = \frac{\max\{F(x',y)|S_{approx}\} - \max\{F(x',y)|S_{relax}\}}{\max\{F(x',y)|S_{relax}\}} \tag{7.2}$$

The UL-gap measures the effect of having an lower-level solution on the upper-level fitness. It permits to determine if the results obtained by each algorithm are illusory or close to what should be expected. UL-gaps are reported in Table 7.5 and illustrate the accuracy of CARBON to produce prices that are not illusory. We can notice that UL-gaps are the lowest for CARBON results.

TABLE 7.4: UL objective values

		UL objective value			
# Variables	# Constraints	CARBON	COBRA	GA+AGH	**p-value**
100	5	10898.81 ±567.39	14710.78 ±928.98	12336.98 ±260.13	$\leq$ 1e−6
100	10	8679.70 ±624.10	15226.79 ±713.71	10195.56 ±290.37	$\leq$ 1e−6
100	30	8104.59 ±1678.16	14762.83 ±863.36	11464.33 ±484.66	$\leq$ 1e−6
250	5	25988.73 ±1399.70	35479.64 ±2147.84	28604.54 ±966.96	$\leq$ 1e−6
250	10	26916.55 ±998.66	38283.71 ±1740.42	29391.77 ±661.76	$\leq$ 1e−6
250	30	24254.18 ±1929.44	39368.26 ±1891.78	28094.64 ±454.11	$\leq$ 1e−6
500	5	49810.29 ±1186.12	73529.34 ±3887.39	52487.52 ±1092.99	$\leq$ 1e−6
500	10	50152.13 ±1843.40	75041.02 ±3277.08	53800.62 ±1467.47	$\leq$ 1e−6
500	30	46670.80 ±5808.65	75386.02 ±3815.59	53735.45 ±903.92	$\leq$ 1e−6
Average		27941.76 ±1781.74	42420.93 ±1922.34	31123.49 ±731.37	

In conclusion, the approximation done at lower-level lead to an upper-level relaxation since the upper-level is less constrained. Here, the results obtained in Table 7.4 implies that CARBON provides tighter upper-bounds than COBRA and GA+AGH since the upper-level problem is a maximization problem.

TABLE 7.5: Upper-level gap based on the best solution obtained at upper-level

		UL gap * 100		
# Variables	# Constraints	CARBON	GA + AGH	COBRA
100	5	2.32	9.47	915.40
100	10	5.19	12.77	1116.58
100	30	8.90	20.36	341.21
250	5	0.79	2.32	347.08
250	10	1.05	3.63	362.88
250	30	3.87	7.29	401.98
500	5	0.51	1.62	394.26
500	10	0.73	1.52	362.94
500	30	3.28	3.35	335.04
Average		2.96	4.14	386.29

Finally to conclude this section, we choose to discuss two average convergence curves for the instance with $n = 500$ variables and $m = 30$ constraints. Additionnal convergence curves can be found in Annexe E. The

first one depicted in Figure 7.8 represents the average convergence over the 30 runs obtained with CARBON. Figure 7.9 illustrates the ones obtained with COBRA. We can easily observe that the convergence curves are smoother for CARBON with a steady increase for the upper-level fitness and a steady decrease for the gap. On the contrary, we can note that both convergence curves have a see-saw shape which indicates us that each improvements phase deteriorates the other level. This observation confirms our belief discussed in section 7.3.1.

In conclusion, COBRA still seems very close to a nested metaheuristics despite the use of a co-evolutionary operator. The fact that it relies on independent improvement phases that do not facilitate its parameterization. Indeed, how should be set the number of improvement generation for each level ? Should it be unbalanced ? Unlike COBRA, we see that CARBON is able to break the nested structure. The convergence curves clearly show that both populations have steady improvements contrary to COBRA.

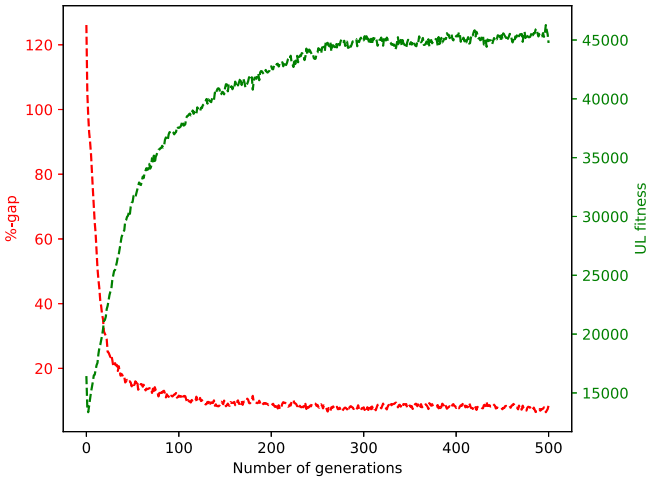


FIGURE 7.8: Example of convergence curve obtained for both CARBON populations

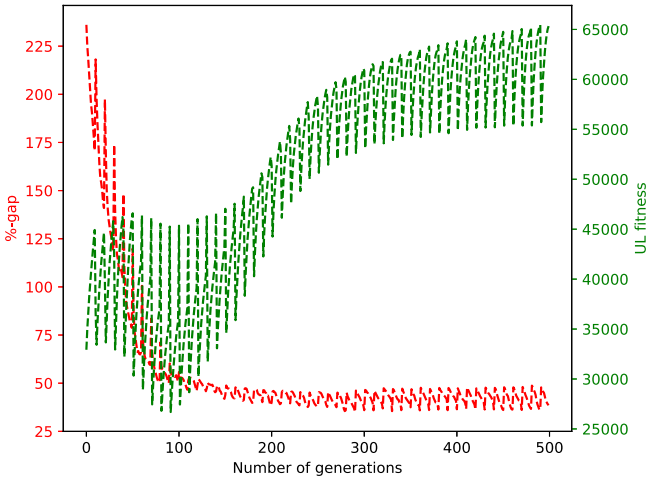


FIGURE 7.9: Example of convergence curve for both COBRA populations

7.6 Conclusions and perspectives

In this chapter, we introduced CARBON, a novel competitive co-evolutionary algorithm to solve bi-level optimization problems. We discussed the main difficulty to create a co-evolutionary algorithm when dealing which two nested optimization problems. Instead of considering, two strongly dependent populations, we made use of

GP hyper-heuristic to generate and learn automatically efficient heuristics to solve the parametric lower-level problem. Indeed, each upper-level decision induces a new lower-level instance that needs to be solved. Heuristics are fast but less accurate than standard metaheuristics. Therefore, using the evolutionary paradigm, it is possible to evolve those heuristics to make them more accurate. The proposed hybrid bi-level co-evolutionary algorithm, i.e., CARBON, pairs the bests upper-level decision with the best heuristics to provide a very good approximation of the lower-level rational reactions and thus a realistic payoff for the upper-level decision maker. Numerical experiments on the Bi-level Cloud Pricing Optimization Problem have shown that CARBON can better handle the nested structure than COBRA, a reference in terms of bi-level co-evolutionary algorithm. Future works could be devoted to multiple-level problems with deeper nested structure in order to analyze the limitations of CARBON in terms of co-evolution.

Chapter 8

Conclusion

Contents

8.1 Thesis summary

Before proposing any new resolution approaches or bi-level models, bi-level theoretical aspects and complexity have been investigated in chapter 2 in order to grasp all the implications of having two nested optimization levels. During this very important phase (objective **O2**), relationships with other classes of problems have been demonstrated before surveying classical resolution and metaheuristics approaches.

Before diving directly into novel resolution approaches, chapter 3 introduced the "Bi-level Clustering Optimization problem", i.e., a novel bi-level mathematical clustering formulation. Despite the fact that it is $\mathcal{NP}$-hard, its particular structure allows us to employ efficiently the classical nested optimization strategy through a hybrid and parallel genetic algorithm. Indeed, the lower-level problem features a totally unimodular coefficient matrix allowing us to solve it exactly and efficiently with a linear solver. In order to examine the added value brought by the proposed bi-level clustering (objective **O3**), numerical experiments have been performed in cooperation with the Luxembourg Centre for Systems Biomedicine (LCSB) on real datasets such as disease maps (e.g. Parkinson, Alzheimer).

Unfortunately, many bi-level optimization problems does not have such interesting properties. Most often, the lower-level problem is a $\mathcal{NP}$-hard problem and the nested optimization strategy becomes unsuitable even using metaheuristics approaches. In order to deal with this pitfall, chapter 4 studied an alternative approach relying on hybridizations with machine learning approaches in order to avoid time consuming nested resolutions (objective **O3**). In this chapter, upper-level solution values in the Inducible Region are approximated using a surrogate-based optimization approach, i.e., Bayesian Optimization. This hybrid approach is compared against the bi-level evolutionary algorithm based on quadratic approximations (BLEAQ)which is a reference in the bi-level evolutionary field. Contrary to BLEAQ which approximates the lower-level decision variables, our approach approximate directly the fitness solution value. Numerical results have shown that we can drastically reduce the cost of lower-level optimization while obtaining very accurate results.

Despite their very interesting potential, approximation algorithms can only handle small and continuous bi-level optimization problems whereas many bi-level optimization problems are large-scale and have combinatorial properties. In short, surrogate functions cannot be employed in the discrete case to approximate the objective function value but it does not mean that a machine learning strategy cannot be applied. The lower-level problem is technically a parametric optimization problem which strongly depends on the upper-level decision variables. Consequently, the different lower-level instances can be considered as family of instances with a common and a parametrized part. Instead of approximating the objective function, we could automatically train dedicated heuristics to tackle this family of instances. For this purpose, GP Hyper-heuristics have been investigated in chapter 5 on a case study, i.e., the multidimensional knapsack problem. Although this problem is a single-level one, it allowed us to validate the concept of "learning to optimize".

Chapter 6 is the direct implementation of the knowledge gained through chapter 5. Instead of solving directly a bi-level problem, a set of heuristics have been trained to tackle solely the parametric lower-level problem by generating a family of instances as learning set (objective **O3**). To prove the validity of the newly designed approach, chapter 6 solve the "Bi-level Cloud Pricing Problem", i.e., a large scale and mixed-integer bi-level optimization problem (objective **O2**). This problem is a second modeling contribution relying on bi-level pricing models introduced in chapter 2.

Apart from the approximation of the lower-level by mean of machine learning techniques , we investigated a decentralized approach, i.e., co-evolution, to break the nested structure. Some seminal works have attempted to deal with it but with more or less success because of the strong epistatic links between the upper-level and lower-level decision variables. In chapter 7, we designed a competitive co-evolutionary algorithm (objective **O4**) that permits to solve the upper-level while training heuristics to cope with lower-level instances. Such an approach allows us to work with two independent populations: one evolving the upper-level solutions while the second one evolves strategies (heuristics). Figure 8.1 recalls the different paths followed in this thesis work.

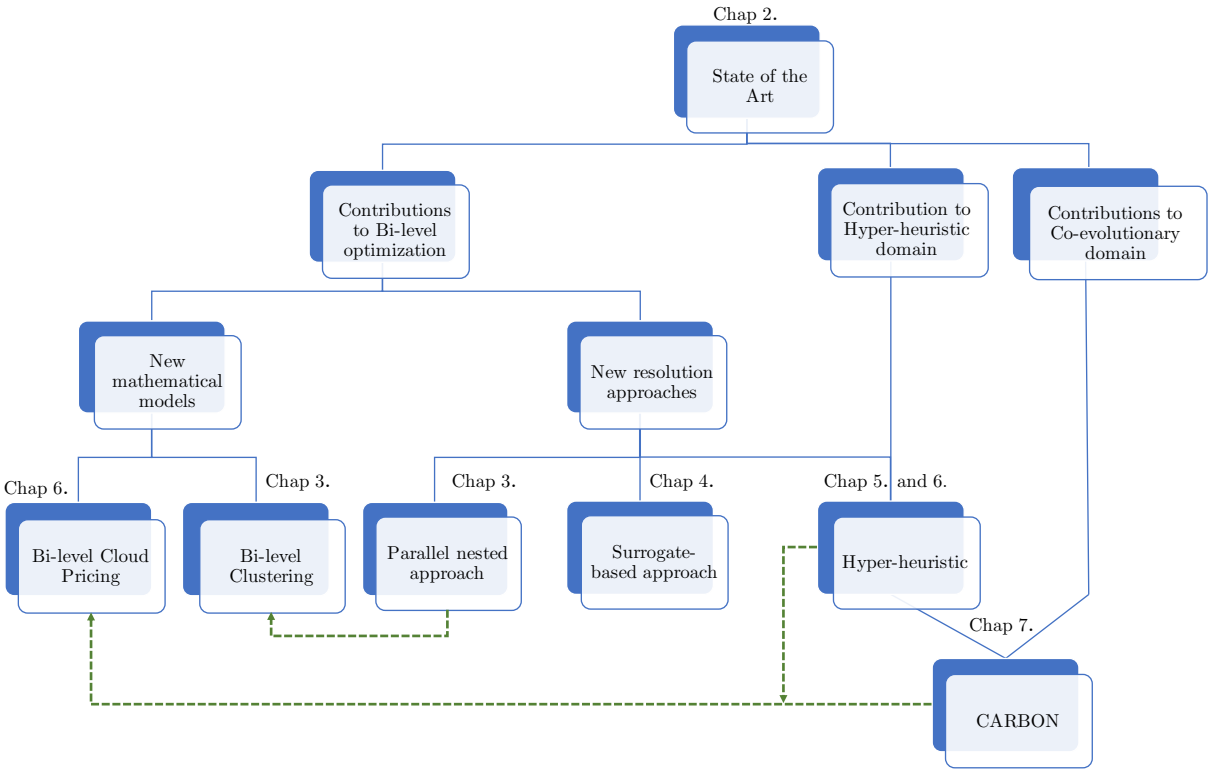


FIGURE 8.1: Thesis workflow

8.2 Contributions

This thesis work has shown the pertinence of considering bi-level models to tackle complex problems with inherent hierarchical structure. Through chapters 3 and 6, we precisely tackled objective O2 by developing new mathematical models for clustering and pricing in the Cloud.

Clustering is a sensitive task that generally depends on several distance metrics. In the case of disease maps, they have been developed by researchers in Bioinformatics in order to categorize proteins and genes responsible for a specific disease. Despite their strong importance, they only provide partial information and many works attempt to combine them in order to utilize them with a classical clustering approach (e.g. Hierarchical clustering). Needless to say, it is not trivial at all to find the right combinations. The bi-level clustering model leveraged the utilization of different distance metrics which are key of major importance for distance-based clusterings (e.g. K-means, K-medoids). The proposed bi-level model enables the choice of centroids with a specific distance metric that can be different from the one used to assign data to clusters. Such an approach clearly permits to prioritize these metrics and have been studied successfully in this work. Furthermore, we proposed a parallel and hybrid evolutionary approach to solve this novel two-level model.

The second bi-level model introduced in this work is a particular and very interesting model for pricing set of services in the Cloud. Many growing businesses now face complex pricing problems due to high competition on the market. Therefore, a pricing model has been proposed to take into account competitors as well as the rational reaction of Cloud Service Customers (CSC) to the proposed prices. With such a model, a CSP can determine what are the optimal prices that guaranty him a maximum revenue knowing the current state of the market. Closely inspired by the famous toll-setting problem [53], the novel bi-level problem is a large scale and combinatorial problem that challenge the few but existing resolution approaches for discrete bi-level problems. Chapters 6 and 7 defined two new resolution approaches that can efficiently deal with it.

O3 has been addressed by chapters 4, 5 and 6 where the approximation of the lower-level problem has been investigated in the continuous and then in the discrete/combinatorial case. In the continuous case, Bayesian Optimization has been employed to approximate the upper-level objective value in the inducible region ($\mathcal{IR}$). Such strategies based on approximation have already been considered in the scientific literature but none were able to reduce dramatically the number of lower-level optimization calls. Furthermore, they attempted to approximate the lower-level decision variable instead of the objective function which is not suitable for large scale problems. Even though Bayesian Optimization has shown impressive properties to deal with continuous problems, it is unfortunately powerless in the discrete case. The main innovation of this thesis is certainly the automatic design of fast but efficient dedicated heuristics. Instead of optimizing a surrogate model of the lower-level objective function, heuristics have been trained to solve lower-level instances while providing good and resilient lower-level solutions. Investigations on GP Hyper-Heuristics have shown the potential of the"learning to optimize" paradigm.

Studies on co-evolutionary aspects have been also performed. The advantage of co-evolution lies in the decomposition that can be obtained to solve a problem more efficiently. Very few co-evolutionary approaches have been designed to handle bi-level optimization problems due to the strong epistatic links existing between both levels. Indeed, each upper-level decision conducts to a different lower-level instance with specific optimal solution. It is very hard to separate both problems. To cope with this issue, a hybrid co-evolutionary bi-level algorithm has been implemented to tackle large scale combinatorial problems such as the Bi-level Cloud Pricing Problem. This algorithm gathered all the knowledge and results obtained during this PhD thesis. Based on the competitive co-evolutionary paradigm, this new algorithm allows to break the nested structure and therefore the strong links between the two levels. This algorithm is presented in chapter 7.

All the publications related to this thesis work are listed below:

- Emmanuel Kieffer, Matthias Rudolf Brust, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Tackling large-scale and combinatorial bi-level problems with genetic programming hyper-heuristic. *IEEE Transactions on Evolutionary Computation*, (in review), 2018
- Marek Ostaszewski, Emmanuel Kieffer, Grégoire Danoy, Reinhard Schneider, and Pascal Bouvry. Clustering approaches for visual knowledge exploration in molecular interaction networks. *BMC Bioinformatics*, 19, aug 2018
- Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A competitive approach for bi-level co-evolution. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018
- Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Bayesian optimization approach of general bi-level problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1614–1621. ACM, 2017
- Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A new modeling approach for the biobjective exact optimization of satellite payload configuration. *International Transactions in Operational Research*, 2017
- Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A new co-evolutionary algorithm based on constraint decomposition. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 492–500, May 2017
- Emmanuel Kieffer, Grégoire Danoy, and Pascal Bouvry. On bi-level approach for scheduling problems. In *New Challenges in Scheduling Theory*, 2016
- Emmanuel Kieffer, Mateusz Guzek, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A novel co-evolutionary approach for constrained genetic algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, pages 47–48. ACM, 2016
- Emmanuel Kieffer, Grégoire Danoy, and Pascal Bouvry. On bi-level approach for scheduling problems. In *New Challenges in Scheduling Theory*, 2016

A PhD thesis is a long-lasting work that enable collaborations and exchanges with other researchers. Some of ours contributions are indirectly related to the main topic and followed fruitful discussions. In some cases, they had indirect impacts on our thesis methodology or on future investigations. Despite the fact that we are not developing them in detail in this manuscript, it is still worthwhile to highlight them briefly. They have been conducted in the context of Unmanned Aerial Vehicle Mobility Models and co-evolutionary optimization which are areas of expertise of the Parallel Computing & Optimisation Group (http://pcog.uni.lu/) headed by Prof. Dr. Bouvry. Below are listed additional publications having an indirect link to the main topic. More details are provided in Annexe A.

- Emmanuel Kieffer, Martin Rosalie, Grégoire Danoy, and Pascal Bouvry. Bayesian optimization to enhance coverage performance of a swarm of uav with chaotic dynamics. May 2018
- Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Hybrid mobility model with pheromones for uav detection task. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016

- Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A new co-evolutionary algorithm based on constraint decomposition. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 492–500, May 2017

8.3 Perspectives

Investigations on GP Hyper-Heuristics have shown the potential of generate ad-hoc heuristics with can surpass human-based ones. We plan to propose a project that is an extension of the successful results undertaken in this PhD work and notably on GP Hyper-Heuristics. Its goal is to go beyond the traditional gradient-based learning techniques and heuristics that can be found in the literature. We claim that the maturity reached by machine learning could allow to learn more complex structures such as metaheuristics. Indeed, since the beginning of the 21st century, developments in the Artificial Intelligence field have steadily grown. Machine learning has certainly been the most prolific area and has made a large technological leap forward. Methods like Deep Learning (DL) pushed the frontiers of the possible with applications like "Alpha GO" which are now overtaking human decisions. This was made feasible by the development of new paradigms such as "learning to learn". Machines are not learning directly anymore, but develop themselves their own learning strategies. This project will propose to investigate the potential of a new paradigm: "Learning to optimize". Alpha GO was able to excel but only on one specific game where the combinatorial aspects are strongly represented. We could propose to automatically generate algorithms to efficiently tackle some very difficult optimization problems. Contrary to most of the contributions found in the literature, this project would not just seek for yet other hybrid algorithms but for a new paradigm where "machine learning" would serve "optimization".

Appendix A

Contributions to other research topics

Bayesian optimization to enhance coverage performance of a swarm of UAV with chaotic dynamics [208]

Abstract:

A mobility model of swarm of UAVs has been recently proposed which purpose is to cover an unknown area: CACOC (Chaotic Ant Colony Optimization to Coverage). This algorithm is based on Ant Colony Optimization where chaotic dynamics are used to enhance the exploration part of the algorithm. CACOC mobility model uses repulsive pheromones to guide the UAVs over the area they have to cover. The UAVs share a map of virtual pheromones that indicates recently visited areas when high pheromone concentrations are present. The UAVs then have a higher probability to move to the least recently visited areas. When there is no pheromone to guide the UAVs, the introduction of chaotic dynamics permits to obtain an efficient exploration of the unknown area. Since the chaotic dynamics are obtained from a three differential equations system with parameters, we can tune one parameter to obtain another chaotic dynamic. The aim of this work consists in discovering a chaotic dynamic leading to the best coverage which can be only computed after a CACOC simulation. For this purpose, Bayesian Optimization has been considered which has been originally designed for time-consuming black-box optimization. Indeed in order to evaluate the chaotic dynamic for a specific parameter, a full simulation should be realized. Global optimization techniques (e.g. population-based heuristics) could be very time-consuming contrary to a surrogate-based approach that minimize the number of simulations to perform and determine the best parameters of the chaotic system.

Link to the main topic:

Although this application is far from Bi-level Optimization, they still have one common characteristic: the evaluation of a solution is time-consuming. Indeed, for each instanced parameter, a simulation has to be run in order to determine its impact. Likewise, Bi-level Optimization requires to determine the optimal solution of the lower-level in order to compute the upper-level fitness value for a given upper-level solution.

Hybrid mobility model with pheromones for UAV detection task [199]

Abstract:

Over the last years, the activities related to unmanned aerial vehicle have seen an exponential growth in several application domains. In that context, a great interest has been devoted to search and tracking scenarios, which require the development of novel UAV mobility management solutions. Recent works on mobility models have shown that bio-inspired algorithms such as ant colonies, have a real potential to tackle complex scenarios. Nevertheless, most of these algorithms are either modified path planning algorithms or dynamical algorithms with no a priori knowledge. We developed a hybrid model based on Markov chains and pheromones to take advantage of both static and dynamic methods. Markov chains are evolved to generate a global behavior guiding UAVs to promising areas while pheromones allow local and dynamical mobility management thanks to information sharing between UAVs via stigmergy. Experimental results have demonstrated the ability of the proposed approach to rapidly detect and keep watch on targets compared to random and classical pheromone based models.

Link to the main topic:

This work was a first attempt to model static cooperation between two levels of UAVs. The future step will be devoted to propose a bi-level (stackelberg game) version of this hybrid Markov mobility model where clusters (zones) may change based on a dynamic cooperation between quad-copters and high-altitude UAVs. In fact, the problem could be model as a Principal-agent model where high-altitude uavs delegates detection tasks to low-altitudes ones. Both uavs do not have the same properties and their priorities may be conflicting. In this case, a bi-level modeling would be very appropriated.

A new Co-evolutionary Algorithm Based on Constraint Decomposition [203]

Abstract:

Handling constraints is not a trivial task in evolutionary computing. Even if different techniques have been proposed in the literature, very few have considered co-evolution which tends to decompose problems into easier sub-problems. Existing co-evolutionary approaches have been mainly used to separate the decision vector. In this article we propose a different co-evolutionary approach, referred to as co-evolutionary constraint decomposition algorithm (CCDA), that relies on a decomposition of the constraints. Indeed, it is generally the conjunction of some specific constraints that hardens problems. The proposed CCDA generates one sub-population for each constraint and optimizes its own local fitness. A sub-population will first try to satisfy its assigned constraint, then the remaining constraints from other sub-populations using a cooperative mechanism, and finally the original objective function. Thanks to this approach, sub-populations will have different behaviors and solutions will approach the feasible domain from different sides. An exchange of information is performed using crossover between individuals from different sub-populations while mutation is applied locally. Promising mutated features are then transmitted through mating. The proposed CCDA has been validated on 8 well-known benchmarks from the literature. Experimental results show the relevance of constraint decomposition in the context of co-evolution compared to state-of-the-art algorithms.

Link to the main topic:

This work has been performed during our first investigations on co-evolutionary algorithms for bi-level optimization. Since bi-level problems can be considered as strongly constrained problems, we focused our attention to the literature on constraint handling techniques and surveyed it. We then proposed a novel decomposition scheme that relies on constraint separation. The co-evolutionary model applied in this work belongs to the class of cooperative co-evolution.

Appendix B

Bi-level clustering quality and terms enrichment (chapter 3)

Map	AlzPathway		AlzPathway Reorg		PD map	
Enrichment with	DO clusters/terms	GO clusters/terms	DO clusters/terms	GO clusters/terms	DO clusters/terms	GO clusters/terms
Eu >GO BP	3/5	3/19	3/1	3/1	52/363	52/1860
Eu >GO CC	5/37	5/38	3/1	3/9	4/76	4/686
Eu >GO MF	2/1	2/1	None	None	2/1	2/156
Eu >Net	68/101	68/238	98/17	98/18	88/372	88/1929
Eu*GO BP	61/86	97/179	13/6	90/14	89/334	97/1669
Eu*GO CC	47/73	89/225	74/22	93/9	88/345	98/1696
Eu*GO MF	51/82	95/194	27/28	92/23	88/334	99/1682
Eu*Net	58/75	90/201	99/8	99/1	93/339	98/1641
GO BP >Eu	51/76	51/196	32/76	32/720	61/353	61/2203
GO BP >Net	48/72	48/196	35/13	35/409	59/368	59/2211
GO CC >Eu	53/44	53/205	37/1	37/100	33/316	33/1590
GO CC >Net	57/37	57/182	45/16	45/13	50/305	50/1550
GO MF >Eu	47/2	47/321	50/8	50/265	50/246	50/1235
GO MF >Net	59/1	59/220	73/5	73/264	74/218	74/1282
Net >Eu	71/85	71/343	65/1	65/23	67/158	67/1463
Net >GO BP	6/1	6/36	4/1	4/1	40/171	40/1373
Net >GO CC	8/20	8/1	2/1	2/1	9/48	9/479
Net >GO MF	4/1	4/1	None	None	10/52	10/558
Net*GO BP	49/47	55/182	2/1	97/136	81/289	86/1563
Net*GO CC	95/48	95/186	99/1	42/27	56/250	79/1398
Net*GO MF	49/53	99/169	34/3	99/1	83/162	91/1306
Expert-based	20/1	20/43	20/1	20/70	36/275	36/1449

TABLE B.1: Maximum numbers of enriched terms with their associated number of clusters for all tested clustering solutions

Map	AlzPathway		AlzPathway Reorg		PD map	
Clustering method	Bilevel clusters/Fmeasure	HCW clusters/Fmeasure	Bilevel clusters/Fmeasure	HCW clusters/Fmeasure	Bilevel clusters/Fmeasure	HCW clusters/Fmeasure
Eu	None	7/0.6389	None	7/0.7157	None	3/0.7045
Eu >GO BP	3/0.5748	None	3/0.5110	None	4/0.7136	None
Eu >GO CC	4/0.6118	None	3/0.5830	None	2/0.6278	None
Eu >GO MF	2/0.4523	None	None	None	1/0.5350	None
Eu >Net	6/0.6349	None	8/0.7134	None	3/0.6757	None
Eu*GO BP	None	12/0.6562	None	12/0.7844	None	4/0.7259
Eu*GO CC	None	12/0.6695	None	12/0.6981	None	8/0.7650
Eu*GO MF	None	13/0.6496	None	5/0.6355	None	3/0.6985
Eu*Net	None	47/0.5164	None	17/0.6034	None	3/0.6796
GO BP	None	2/0.3288	None	2/0.3400	None	2/0.5284
GO BP >Eu	2/0.4123	None	1/0.4111	None	1/0.5350	None
GO BP >Net	1/0.4121	None	None	None	1/0.5350	None
GO CC	None	2/0.3289	None	2/0.3392	None	2/0.4890
GO CC >Eu	2/0.4123	None	None	None	2/0.5373	None
GO CC >Net	6/0.4132	None	None	None	1/0.5350	None
GO MF	None	2/0.3318	None	2/0.3372	None	2/0.4842
GO MF >Eu	1/0.4121	None	None	None	1/0.5350	None
GO MF >Net	3/0.4130	None	None	None	1/0.5350	None
Net	None	39/0.4466	None	14/0.4842	None	4/0.5076
Net >Eu	11/0.4370	None	33/0.4309	None	3/0.5766	None
Net >GO BP	6/0.4165	None	None	None	3/0.5762	None
Net >GO CC	7/0.4150	None	None	None	6/0.5837	None
Net >GO MF	1/0.4121	None	None	None	4/0.5731	None
Net*GO BP	None	44/0.4396	None	10/0.4772	None	3/0.5151
Net*GO CC	None	45/0.4392	None	11/0.4844	None	3/0.5151
Net*GO MF	None	46/0.4359	None	11/0.4875	None	3/0.5151

TABLE B.2: Maximum F-measure values with their associated number of clusters for all tested clustering solutions

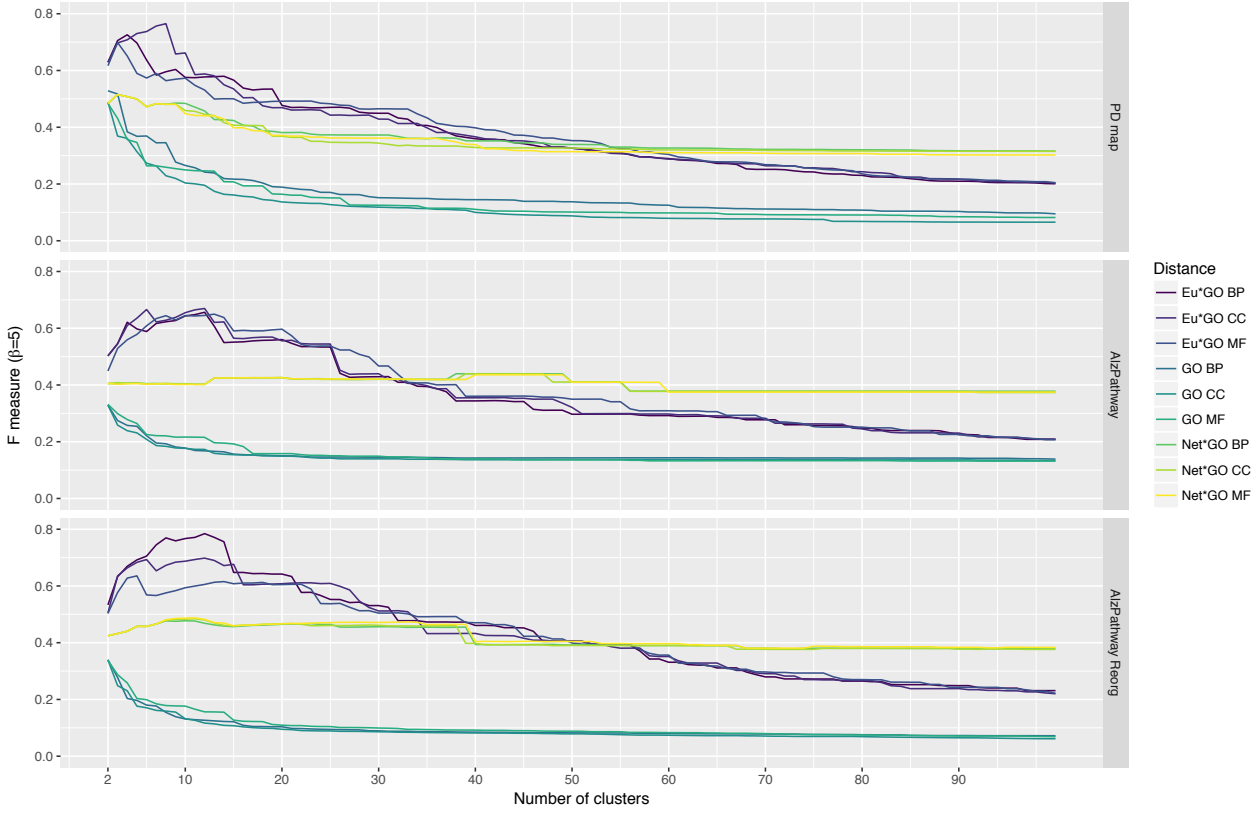


FIGURE B.1: Hierarchical clustering (HCW) quality for different Gene Ontologies (GO) distance metrics

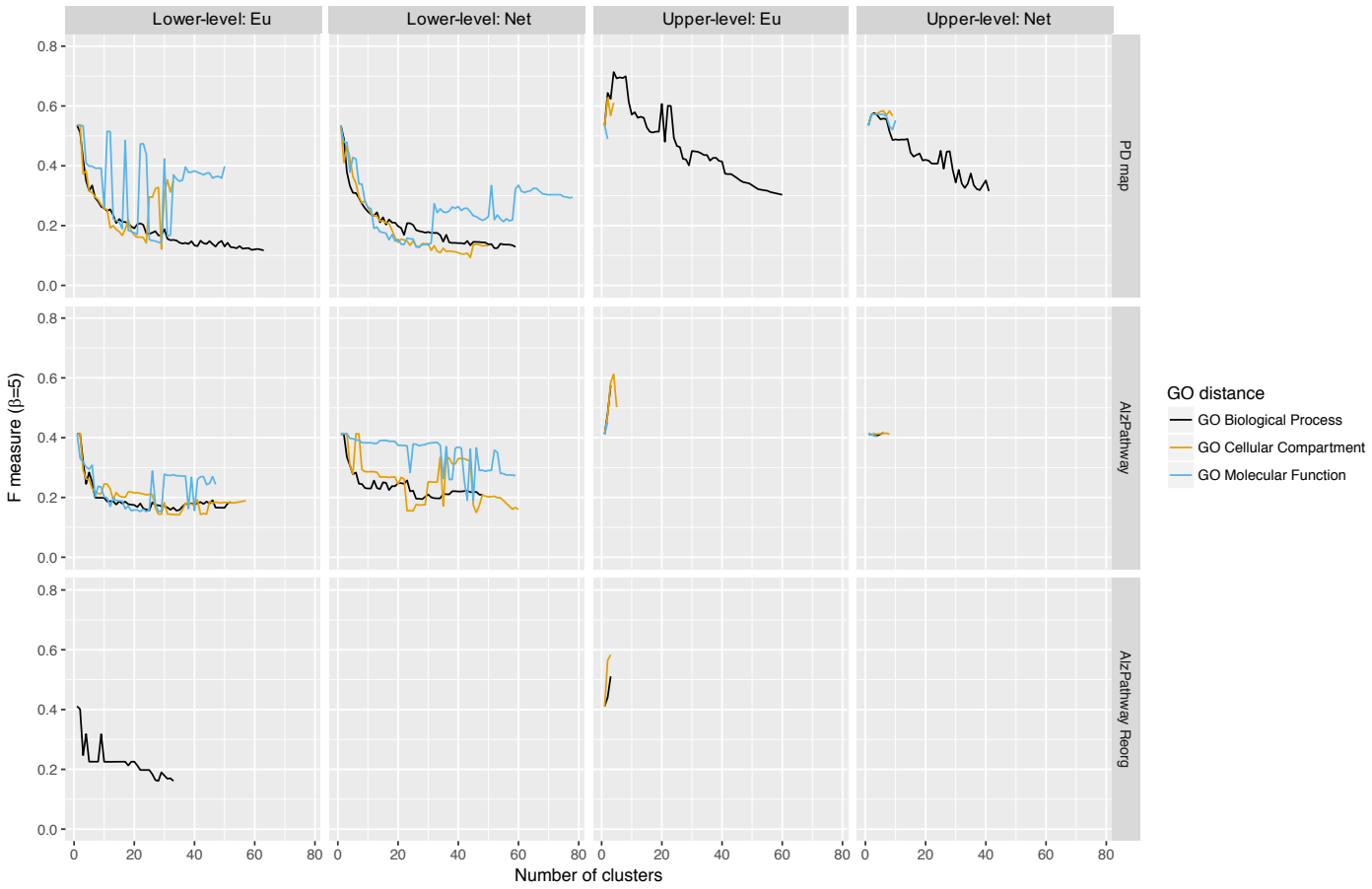


FIGURE B.2: Bi-level clustering quality for different Gene Ontologies (GO) distance metrics

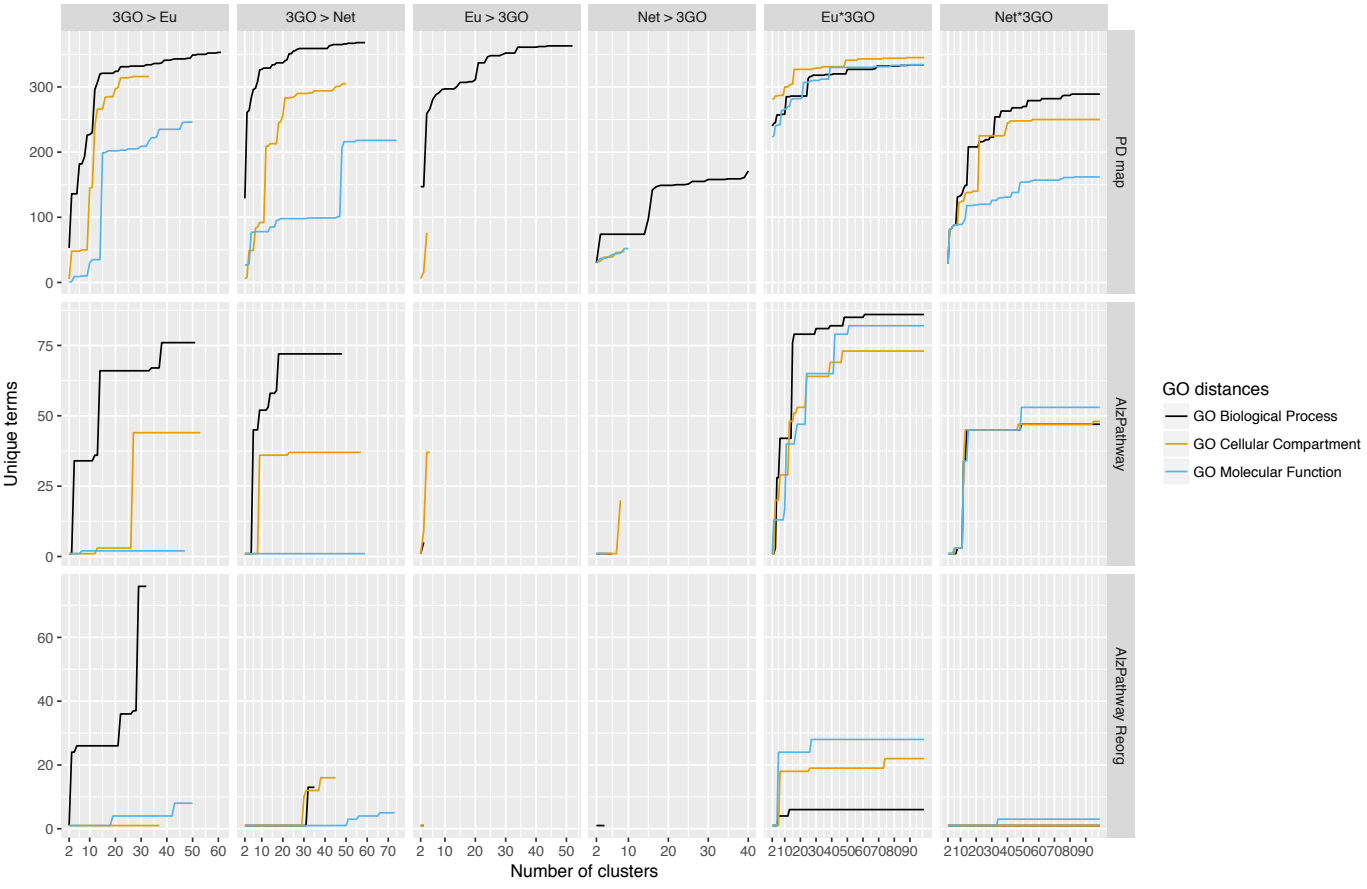


FIGURE B.3: Disease Ontology terms enriched for Gene Ontologies (GO) distance metrics

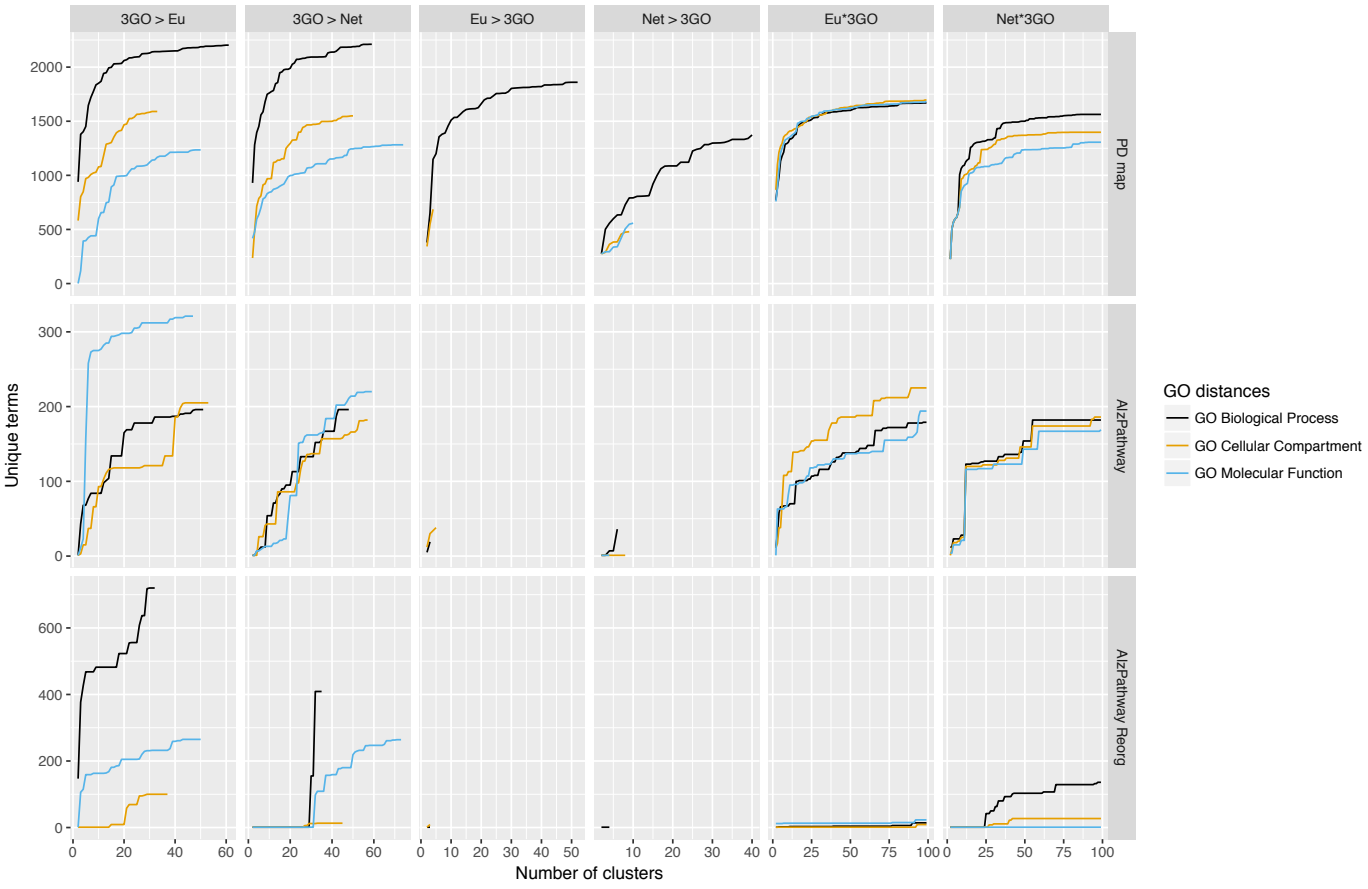
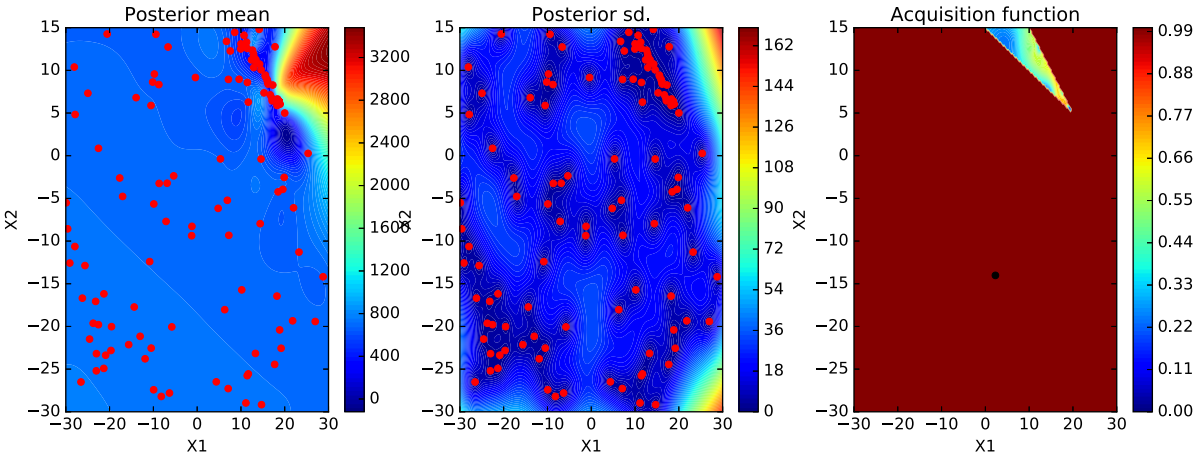
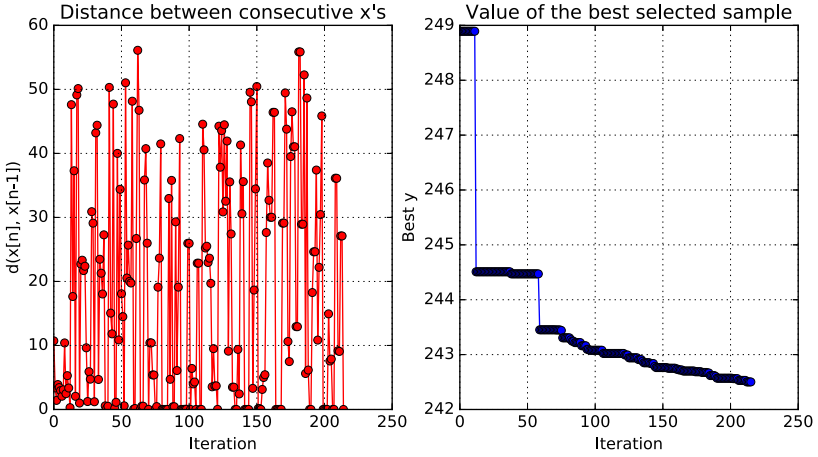


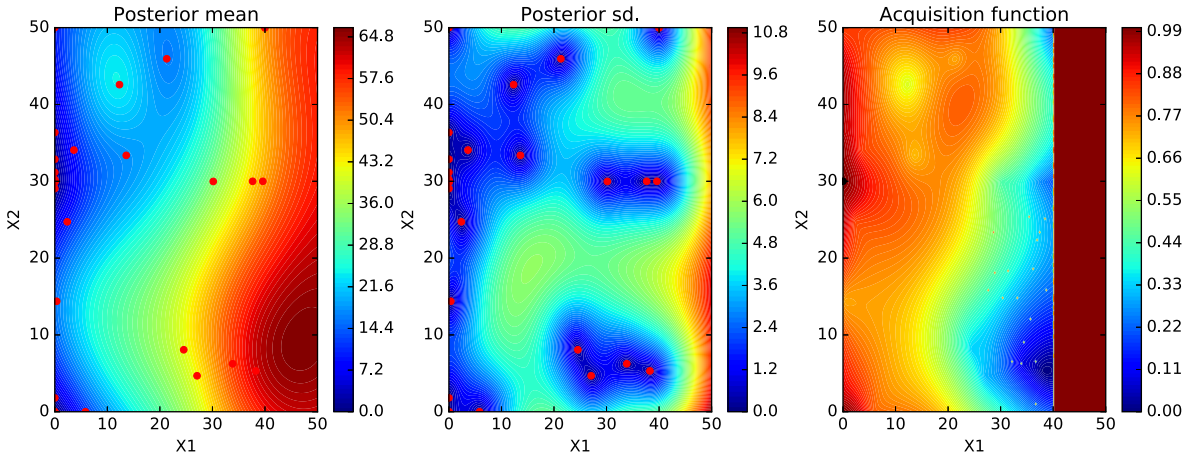
FIGURE B.4: Gene Ontology terms enriched for Gene Ontologies (GO) distance metrics

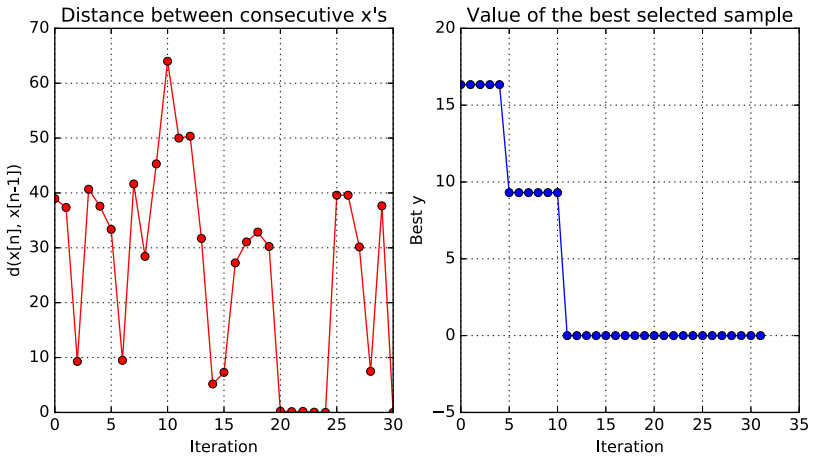
Appendix C

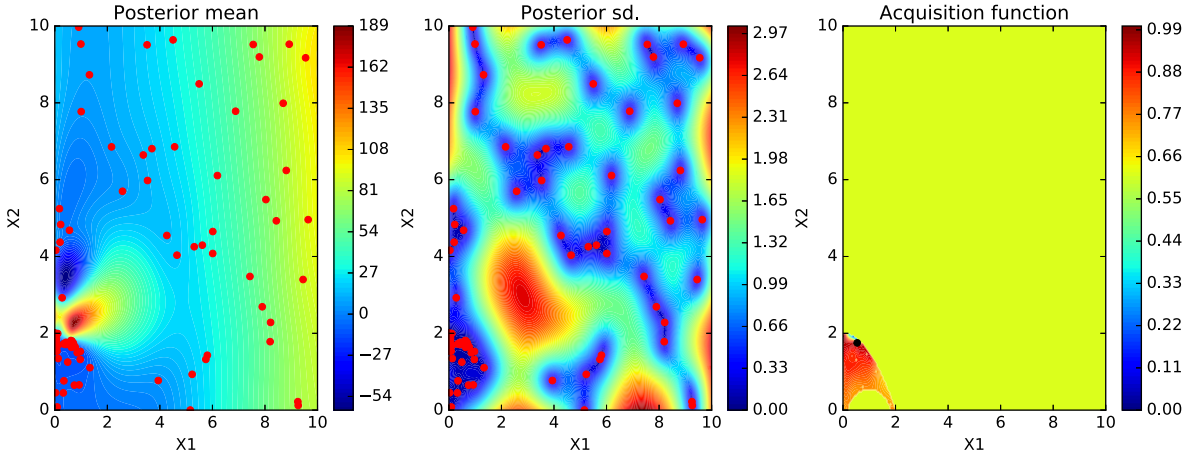
Heat maps, acquisition function and convergence TP benchmarks (chapter 4)

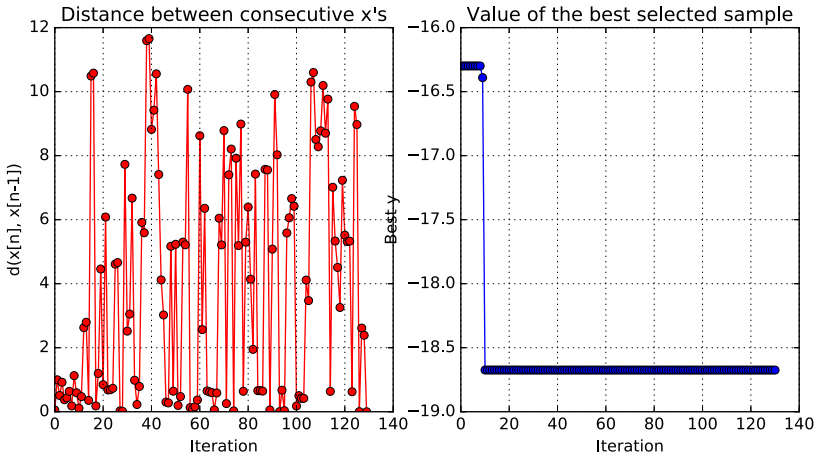


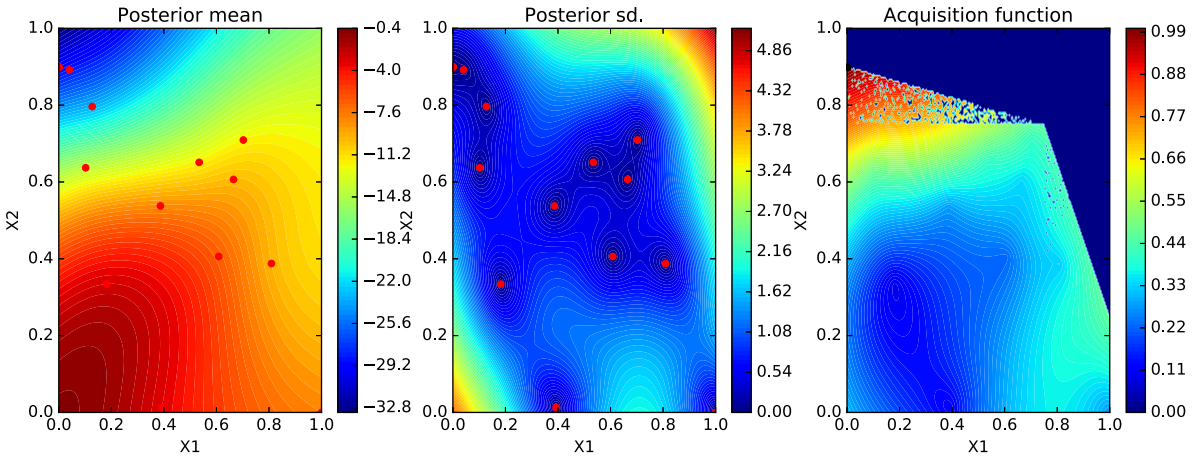


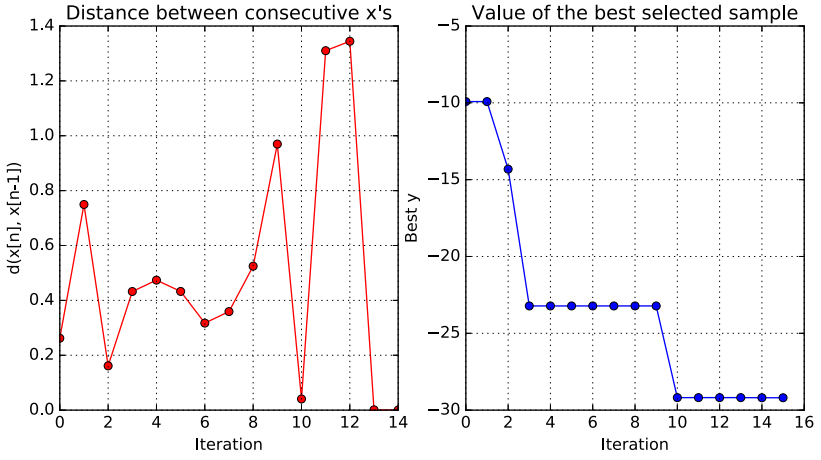

Posterior mean
Posterior sd.
Acquisition function
X1
X2

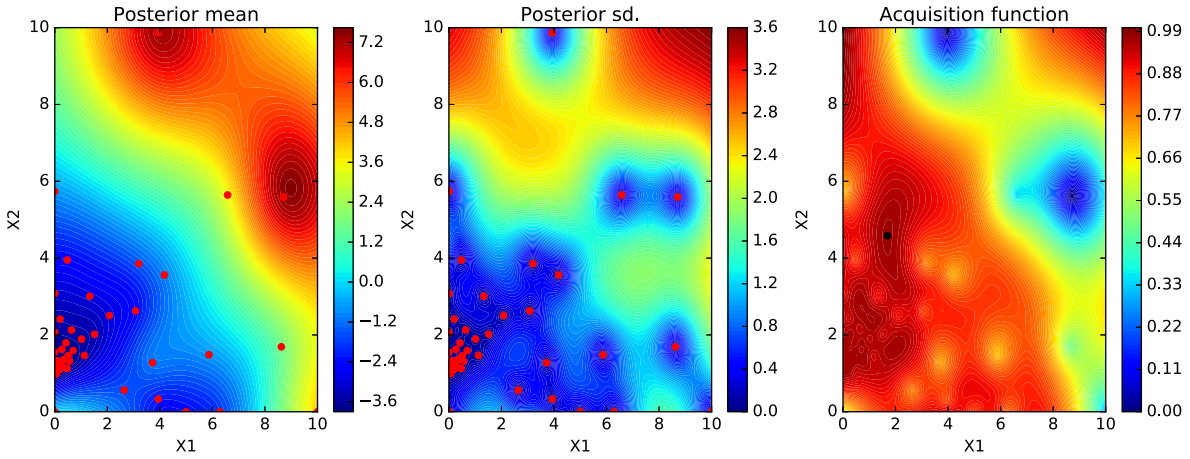

Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration

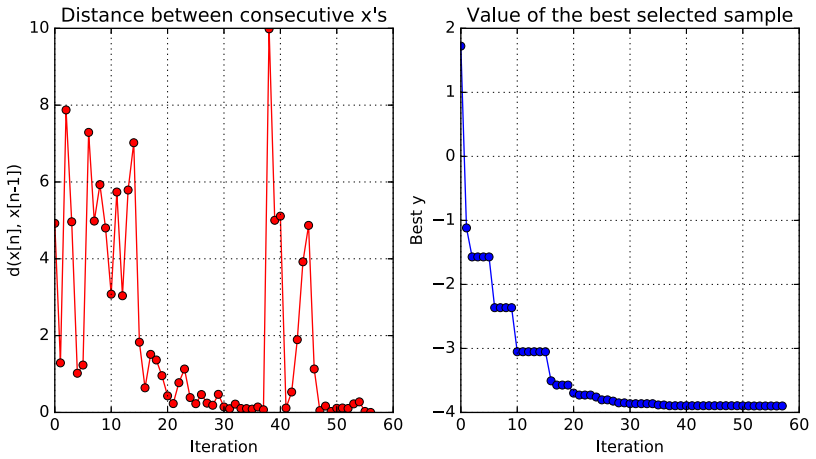

Posterior mean
Posterior sd.
Acquisition function
X1
X2

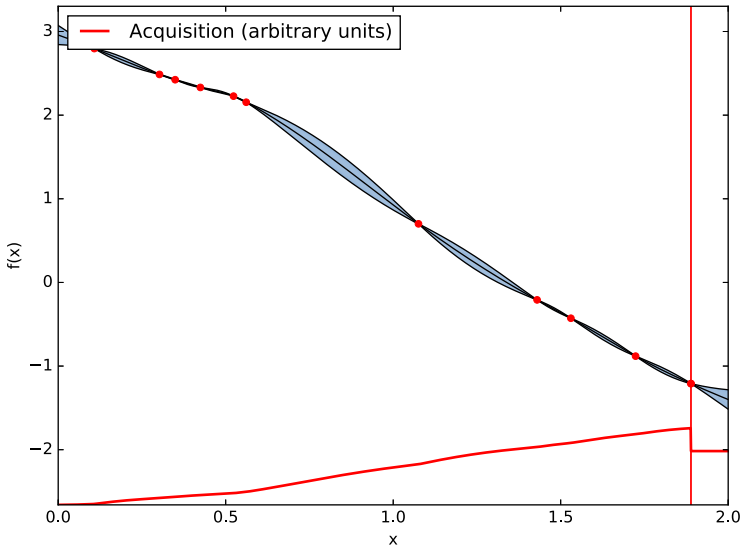

Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration

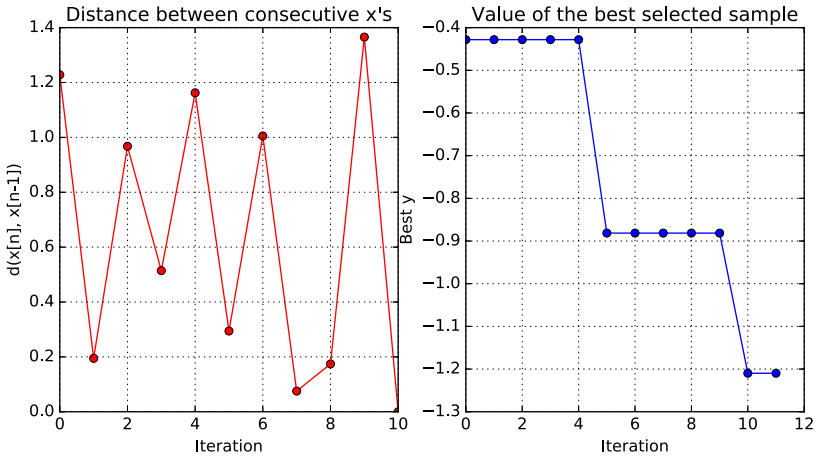

Posterior mean
X2
X1
Posterior sd.
X2
X1
Acquisition function
X2
X1

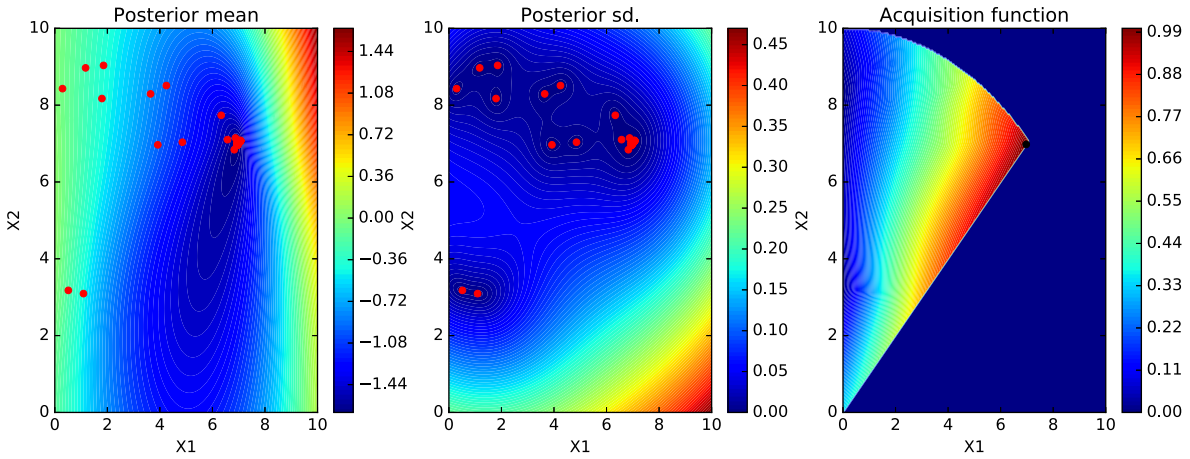

Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration

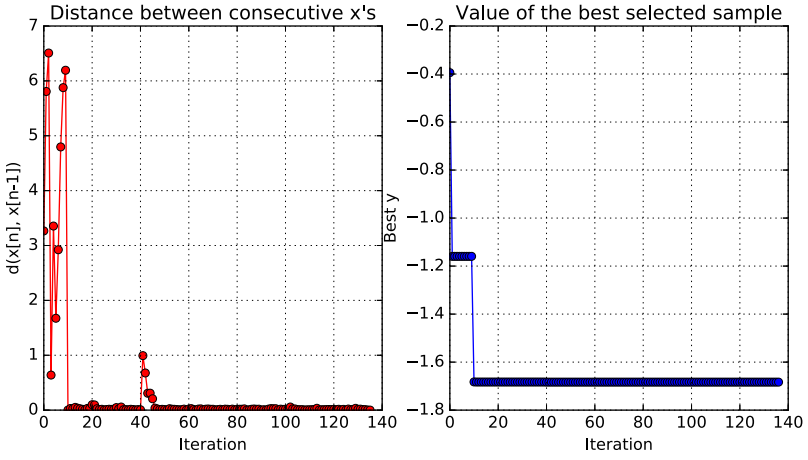

Posterior mean
Posterior sd.
Acquisition function
X1
X2

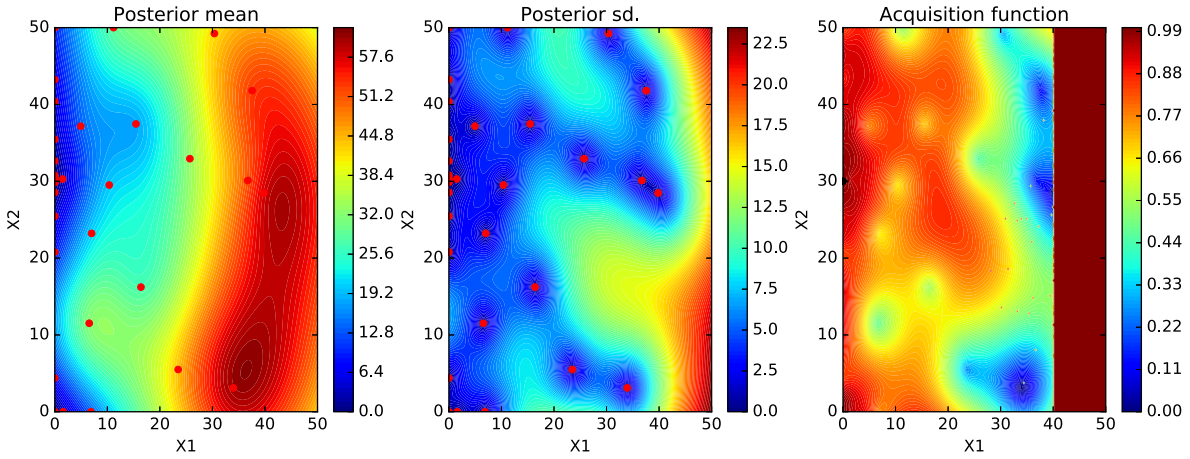

Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration

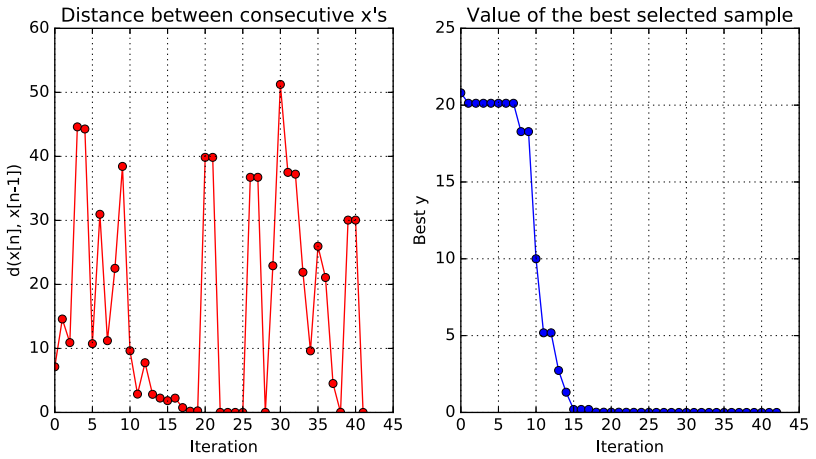

Acquisition (arbitrary units)
f(x)
x

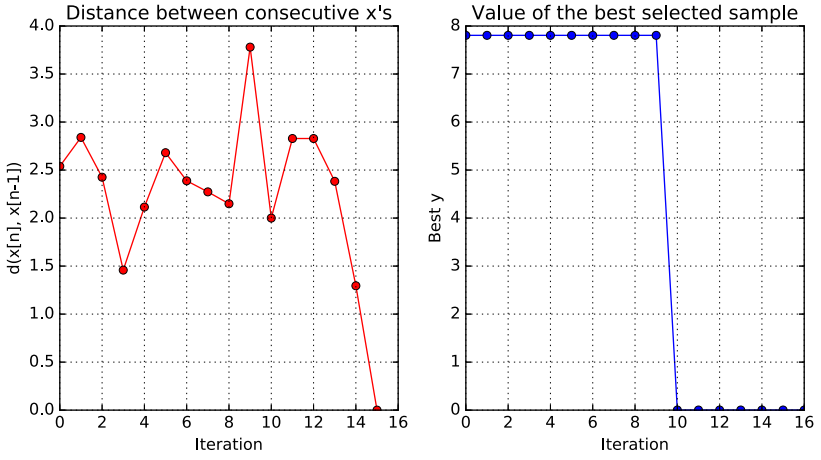

Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration

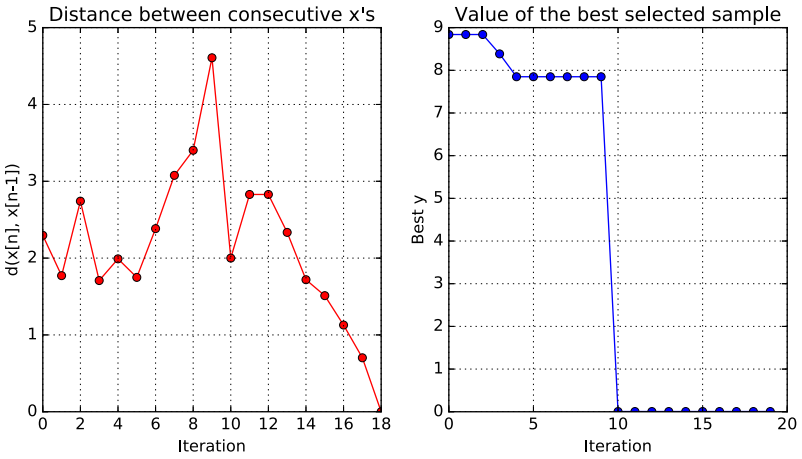

Posterior mean
X2
X1
Posterior sd.
X2
X1
Acquisition function
X2
X1


Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration


Posterior mean
Posterior sd.
Acquisition function
X1
X2


Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration


Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration


Distance between consecutive x's
d(x[n], x[n-1])
Iteration
0
1
2
3
4
5
0
2
4
6
8
10
12
14
16
18
Value of the best selected sample
Best y
Iteration
0
1
2
3
4
5
6
7
8
9
0
5
10
15
20

Appendix D

Automatic heuristic gap for all MKP instances (chapter 5)

TABLE D.1: Application of the learned heuristics on instances with n=100

Heuristics \ Instances			$n=100$								
			$m=5$			$m=10$			$m=30$		
			$r=0.25$	$r=0.5$	$r=0.75$	$r=0.25$	$r=0.5$	$r=0.75$	$r=0.25$	$r=0.5$	$r=0.75$
$n=100$	$m=5$	$r=0.25$	1.40	1.39	1.90	2.26	1.76	2.72	2.92	1.79	5.52
		$r=0.5$	1.55	0.99	2.04	2.25	1.60	2.76	3.40	2.08	5.26
		$r=0.75$	1.53	1.42	1.90	2.26	1.63	2.72	3.06	1.71	5.36
	$m=10$	$r=0.25$	1.63	1.26	2.19	2.13	1.76	2.91	2.98	1.86	5.32
		$r=0.5$	1.62	1.17	2.32	2.25	1.44	3.07	3.30	1.90	4.93
		$r=0.75$	1.51	1.17	2.08	2.23	1.45	2.62	4.22	2.42	4.89
	$m=30$	$r=0.25$	1.52	1.17	2.01	2.39	1.57	2.89	2.75	1.73	5.14
		$r=0.5$	2.21	1.91	2.40	2.33	2.03	2.83	3.10	1.73	5.25
		$r=0.75$	1.81	1.29	2.89	2.43	1.89	3.31	3.42	2.07	4.73
$n=250$	$m=5$	$r=0.25$	1.47	1.16	2.19	2.27	1.63	2.63	2.98	1.81	5.21
		$r=0.5$	1.61	1.12	2.04	2.35	1.56	3.03	3.04	1.82	5.50
		$r=0.75$	1.45	1.23	1.83	2.27	1.70	2.60	3.02	2.01	5.07
	$m=10$	$r=0.25$	1.80	1.13	2.63	2.29	1.76	3.08	3.12	1.77	5.31
		$r=0.5$	1.42	0.98	2.00	2.41	1.72	3.13	2.84	1.71	5.25
		$r=0.75$	1.88	1.55	3.12	2.25	1.57	2.85	4.44	2.38	5.50
	$m=30$	$r=0.25$	1.49	1.08	2.04	2.35	1.56	2.82	3.12	1.82	5.07
		$r=0.5$	2.01	1.70	2.72	2.36	1.55	2.89	2.98	1.90	5.27
		$r=0.75$	1.50	1.26	2.42	2.59	1.83	3.07	3.05	2.02	4.85
$n=500$	$m=5$	$r=0.25$	1.65	1.05	2.04	2.35	1.56	3.15	3.03	1.82	5.47
		$r=0.5$	1.64	1.14	2.00	2.25	1.56	2.85	3.12	1.71	5.27
		$r=0.75$	1.66	1.09	2.70	3.05	1.99	4.35	5.35	3.49	8.04
	$m=10$	$r=0.25$	1.99	1.44	3.49	2.38	1.58	2.89	2.75	1.73	5.14
		$r=0.5$	1.65	1.14	2.15	2.25	1.56	3.02	3.12	1.71	5.23
		$r=0.75$	1.48	1.20	1.93	2.28	1.61	2.72	2.98	1.81	5.02
	$m=30$	$r=0.25$	1.52	1.11	1.95	2.39	1.57	2.88	2.96	1.67	5.10
		$r=0.5$	4.42	3.80	6.42	2.70	2.19	3.79	3.38	2.03	4.99
		$r=0.75$	1.72	1.39	2.54	2.35	1.49	3.07	3.05	1.88	5.58
		average	1.75	1.35	2.44	2.36	1.67	2.99	3.24	1.94	5.31
		best	1.40	0.98	1.83	2.13	1.44	2.60	2.75	1.67	4.73
		std	0.57	0.53	0.89	0.18	0.18	0.37	0.57	0.36	0.59

TABLE D.2: Application of the learned heuristics on instances with n=250

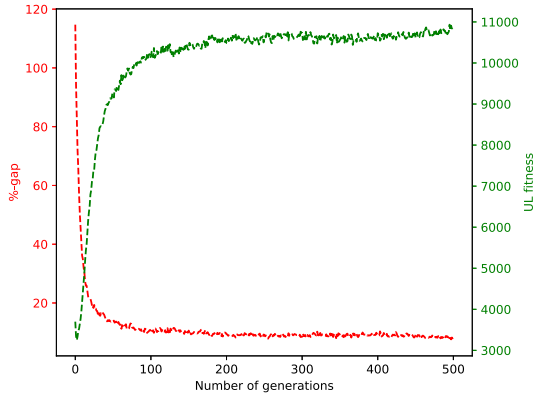
Heuristics \ Instances			$n = 250$								
			$m = 5$			$m = 10$			$m = 30$		
			$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 0.25$	$r = 0.5$	$r = 0.75$
$n = 100$	$m = 5$	$r = 0.25$	0.60	0.36	0.75	0.89	0.57	1.13	1.37	1.00	2.24
		$r = 0.5$	0.64	0.65	0.71	0.95	1.18	1.20	2.32	1.75	2.39
		$r = 0.75$	0.78	0.42	0.76	0.94	0.65	1.09	1.33	1.04	2.23
	$m = 10$	0.25	0.66	0.50	0.84	1.02	0.69	1.14	1.35	0.98	2.20
		$r = 0.5$	0.63	0.42	0.75	0.79	0.58	1.15	1.47	0.87	1.99
		$r = 0.75$	0.64	0.40	0.73	0.92	0.69	1.08	2.22	1.23	2.62
	$m = 30$	0.25	0.59	0.38	0.69	0.93	0.54	1.08	1.38	0.94	1.90
		$r = 0.5$	0.97	0.63	1.44	0.95	0.85	1.38	1.43	0.99	2.16
		$r = 0.75$	0.63	0.46	0.86	1.00	0.68	1.04	1.38	1.09	1.97
$n = 250$	$= 5$	$r = 0.25$	0.54	0.37	0.65	0.87	0.57	1.14	1.43	1.02	2.01
		$r = 0.5$	0.62	0.35	0.72	0.88	0.60	1.04	1.25	0.98	2.06
		$r = 0.75$	0.56	0.39	0.61	0.88	0.57	1.16	1.42	1.02	2.05
	$m = 10$	$r = 0.25$	0.82	0.54	1.08	0.85	0.74	1.56	1.43	0.89	2.02
		$r = 0.5$	0.70	0.35	0.75	0.86	0.52	1.15	1.30	0.94	2.18
		$r = 0.75$	1.04	0.75	1.59	0.83	0.56	0.91	2.55	1.44	2.85
	$m = 30$	$r = 0.25$	0.67	0.35	0.72	0.88	0.57	1.04	1.33	0.93	2.06
		$r = 0.5$	1.23	0.63	1.31	1.06	0.57	1.16	1.36	0.87	2.14
		$r = 0.75$	0.60	0.43	0.83	0.91	0.52	1.13	1.42	1.11	1.81
$n = 500$	$m = 5$	$r = 0.25$	0.66	0.36	0.70	0.87	0.57	1.02	1.38	0.92	2.06
		$r = 0.5$	0.73	0.35	0.67	0.89	0.61	1.04	1.36	0.89	1.87
		$r = 0.75$	0.53	0.41	0.73	1.13	0.66	1.34	2.58	1.58	3.82
	$m = 10$	$r = 0.25$	0.96	0.59	1.93	0.89	0.52	1.11	1.42	0.93	1.90
		$r = 0.5$	0.57	0.41	0.74	0.84	0.59	1.00	1.37	0.88	1.87
		$r = 0.75$	0.56	0.40	0.71	0.91	0.54	1.13	1.43	1.02	2.01
	$m = 30$	$r = 0.25$	0.61	0.38	0.69	0.88	0.56	1.08	1.44	0.90	1.95
		$r = 0.5$	4.53	3.54	6.58	1.40	0.91	1.76	1.40	0.91	1.82
		$r = 0.75$	0.84	0.48	0.97	0.94	0.61	1.10	1.40	0.97	2.06
		average	0.85	0.57	1.09	0.93	0.64	1.15	1.54	1.04	2.16
		best	0.53	0.35	0.61	0.79	0.52	0.91	1.25	0.87	1.81
		std	0.76	0.60	1.14	0.12	0.14	0.18	0.38	0.22	0.41

TABLE D.3: Application of the learned heuristics on instances with n=500

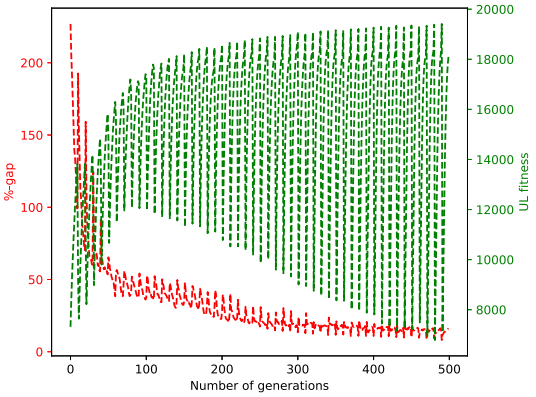
Heuristics \ Instances			$n = 500$								
			5			10			30		
			$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 0.25$	$r = 0.5$	$r = 0.75$
$n = 100$	$m = 5$	$r = 0.25$	0.25	0.23	0.38	0.42	0.32	0.56	0.80	0.51	1.03
		$r = 0.5$	0.80	1.74	0.74	1.42	1.49	0.91	2.37	1.91	1.93
		$r = 0.75$	0.28	0.23	0.39	0.39	0.33	0.59	0.82	0.49	0.95
	$m = 10$	$r = 0.25$	0.36	0.25	0.47	0.41	0.43	0.64	1.04	0.58	1.19
		$r = 0.5$	0.32	0.15	0.40	0.48	0.25	0.62	0.72	0.45	0.97
		$r = 0.75$	0.33	0.18	0.39	0.42	0.31	0.54	1.07	0.63	1.46
	$m = 30$	$r = 0.25$	0.26	0.18	0.39	0.38	0.29	0.56	0.80	0.48	0.93
		$r = 0.5$	0.59	0.53	0.92	0.60	0.49	0.80	0.76	0.51	0.92
		$r = 0.75$	0.31	0.28	0.38	0.47	0.42	0.61	0.92	0.51	1.15
$n = 250$	$m = 5$	$r = 0.25$	0.23	0.19	0.36	0.43	0.30	0.49	0.81	0.49	1.00
		$r = 0.5$	0.28	0.17	0.43	0.40	0.29	0.59	0.76	0.49	0.93
		$r = 0.75$	0.23	0.21	0.37	0.43	0.29	0.49	0.81	0.49	1.01
	$m = 10$	$r = 0.25$	0.37	0.21	0.43	0.49	0.30	0.64	0.76	0.46	1.18
		$r = 0.5$	0.27	0.18	0.38	0.51	0.32	0.58	0.78	0.50	0.96
		$r = 0.75$	0.91	0.59	1.61	0.35	0.23	0.54	1.28	0.97	1.95
	$m = 30$	$r = 0.25$	0.32	0.18	0.39	0.40	0.29	0.56	0.76	0.49	0.95
		$r = 0.5$	0.56	0.65	0.96	0.36	0.31	0.62	0.75	0.53	0.97
		$r = 0.75$	0.26	0.29	0.34	0.46	0.35	0.54	0.90	0.48	1.13
$n = 500$	$m = 5$	$r = 0.25$	0.24	0.17	0.37	0.35	0.28	0.55	0.75	0.48	0.90
		$r = 0.5$	0.27	0.15	0.39	0.35	0.25	0.55	0.75	0.47	0.99
		$r = 0.75$	0.26	0.16	0.27	0.50	0.36	0.64	1.35	0.95	1.86
	$m = 10$	$r = 0.25$	0.68	0.27	1.03	0.35	0.30	0.58	0.80	0.48	0.93
		$r = 0.5$	0.33	0.18	0.37	0.38	0.22	0.56	0.74	0.47	1.00
		$r = 0.75$	0.25	0.19	0.38	0.44	0.30	0.50	0.83	0.51	0.99
	$m = 30$	$r = 0.25$	0.26	0.20	0.39	0.37	0.34	0.57	0.66	0.54	0.93
		$r = 0.5$	4.19	3.30	6.09	1.05	0.51	0.93	0.71	0.42	1.04
		$r = 0.75$	0.38	0.24	0.61	0.45	0.27	0.59	0.75	0.51	0.89
		average	0.51	0.42	0.73	0.48	0.37	0.61	0.90	0.59	1.12
		best	0.23	0.15	0.27	0.35	0.22	0.49	0.66	0.42	0.89
		std	0.76	0.66	1.11	0.23	0.24	0.11	0.34	0.29	0.31

Appendix E

Convergence curves for CARBON and COBRA (chapter 7)



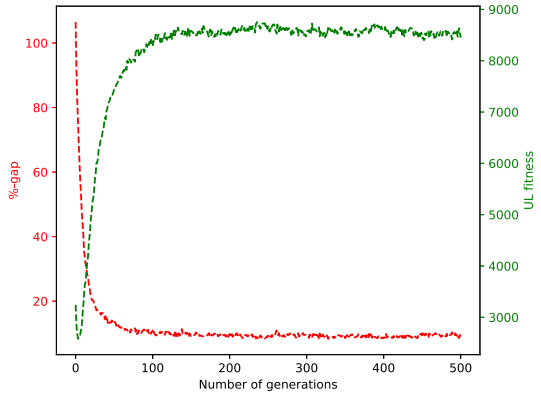
(A) CARBON



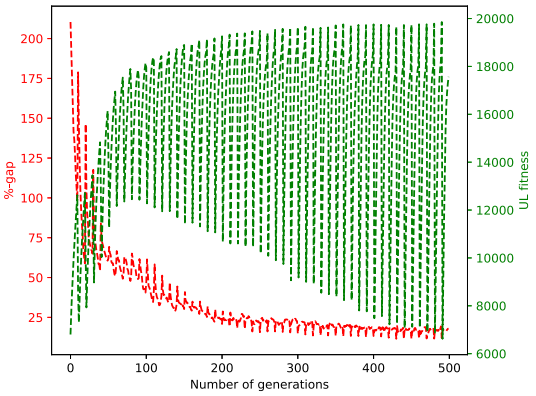
(B) COBRA

s

lack Convergence curve for both sub-populations on the instance class (100;5)



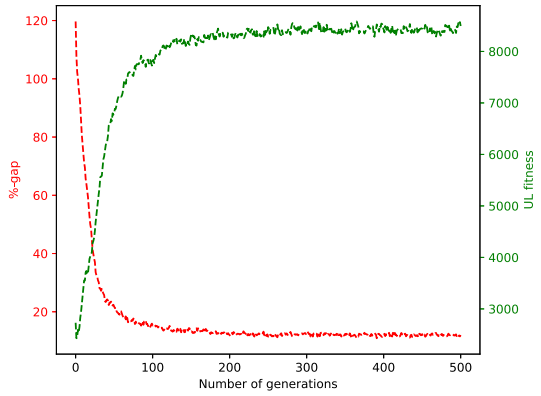
(A) CARBON



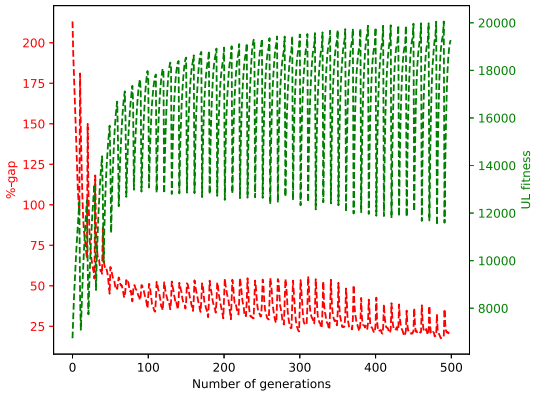
(B) COBRA

Convergence curve for both sub-populations on the instance class (100;10)

..

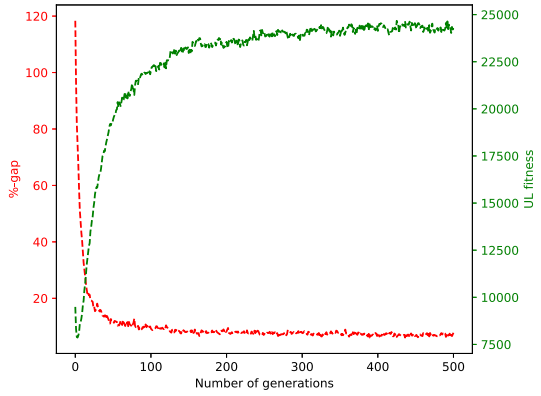


(A) CARBON

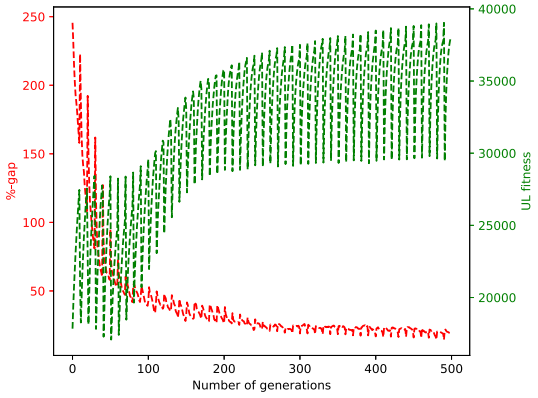


(B) COBRA

Convergence curve for both sub-populations on the instance class (100;30)

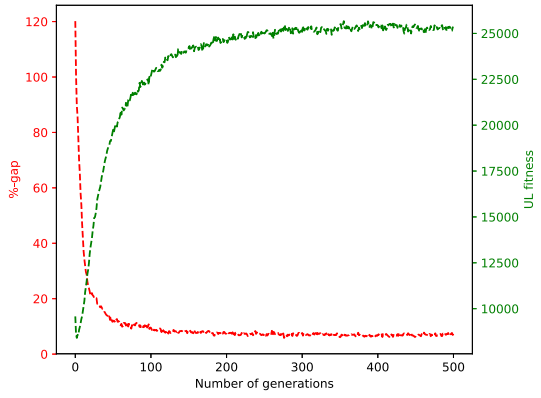


(A) CARBON

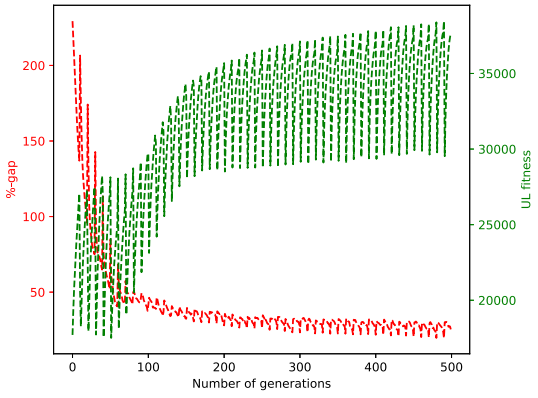


(B) COBRA

Convergence curve for both sub-populations on the instance class (250;5)

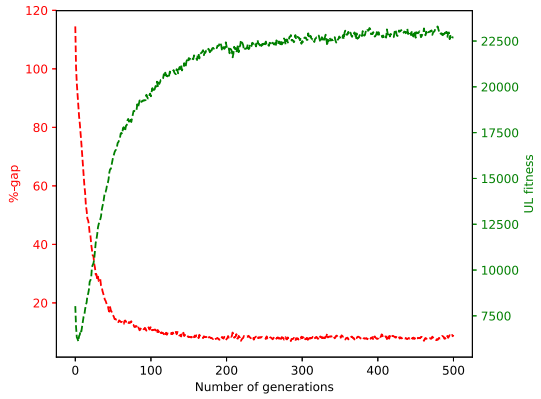


(A) CARBON

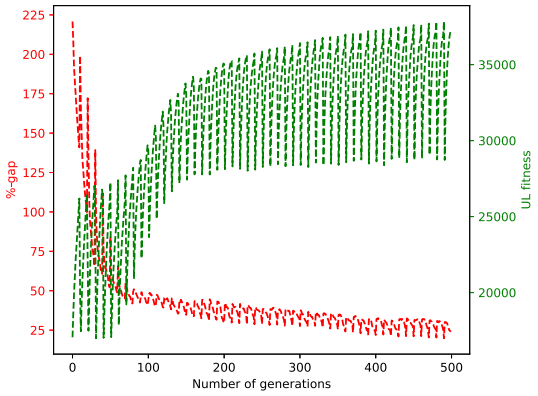


(B) COBRA

Convergence curve for both sub-populations on the instance class (250;10)

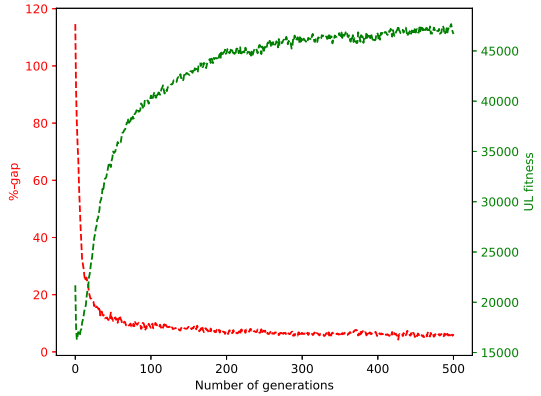


(A) CARBON

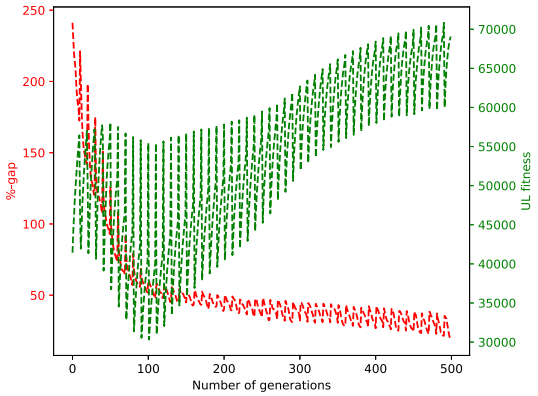


(B) COBRA

Convergence curve for both sub-populations on the instance class (250;30)

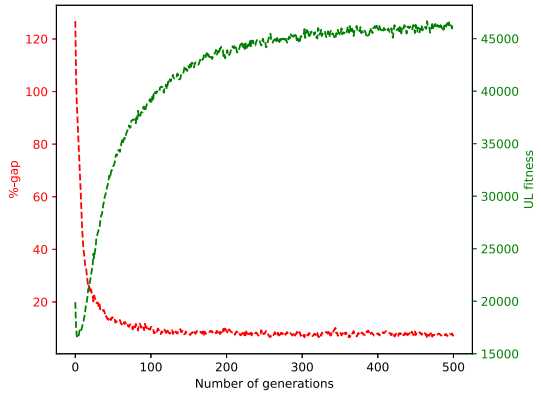


(A) CARBON

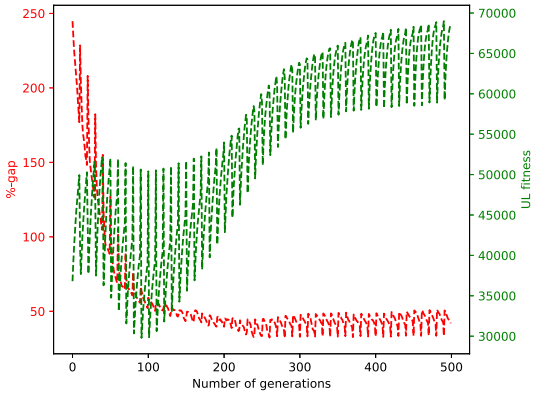


(B) COBRA

Convergence curve for both sub-populations on the instance class (500;5)

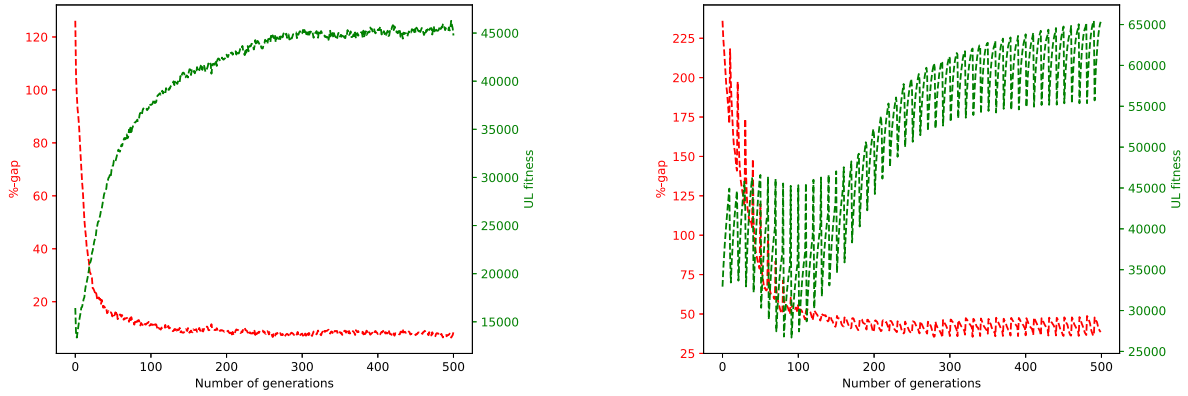


(A) CARBON



(B) COBRA

Convergence curve for both sub-populations on the instance class (500;10)



(A) CARBON (B) COBRA

Convergence curve for both sub-populations on the instance class (500;30)

Bibliography

[1] K Aisaki, S Aizawa, H Fujii, J Kanno, and H Kanno. Glycolytic inhibition by mutation of pyruvate kinase gene increases oxidative stress and causes apoptosis of a pyruvate kinase deficient cell line. *Experimental Hematology*, 35(8):1190–1200, aug 2007.
One citation in page 76.

[2] E. Aiyoshi and K. Shimizu. Hierarchical decentralized systems and its new solution by a barrier method. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):444–449, 1981.
2 citations in pages 32 and 38.

[3] E. Aiyoshi and K. Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29(12):1111–1114, dec 1984.
2 citations in pages 32 and 38.

[4] Yalçın Akçay, Haijun Li, and Susan H. Xu. Greedy algorithm for the general multidimensional knapsack problem. *Annals of Operations Research*, 150(1):17–29, dec 2006.
One citation in page 103.

[5] Deniz Aksen and Necati Aras. A bilevel fixed charge location model for facilities under imminent attack. *Computers & Operations Research*, 39(7):1364–1381, jul 2012.
2 citations in pages 13 and 17.

[6] Deniz Aksen and Necati Aras. A matheuristic for leader-follower games involving facility location-protection-interdiction decisions. In *Metaheuristics for Bi-level Optimization*, pages 115–151. Springer Berlin Heidelberg, 2013.
One citation in page 18.

[7] Faiz A. Al-Khayyal, Reiner Horst, and Panos M. Pardalos. Global optimization of concave functions subject to quadratic constraints: An application in nonlinear bilevel programming. *Annals of Operations Research*, 34(1):125–147, dec 1992.
2 citations in pages 32 and 38.

[8] Abdulrahman Alguwaizani, Pierre Hansen, Nenad Mladenović, and Eric Ngai. Variable neighborhood search for harmonic means clustering. *Applied Mathematical Modelling*, 35(6):2688–2694, jun 2011.
One citation in page 51.

[9] Sam Allen, Edmund K. Burke, Matthew Hyde, and Graham Kendall. Evolving reusable 3d packing heuristics with genetic programming. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*. ACM Press, 2009.
One citation in page 95.

[10] Mahyar A. Amouzegar and Khosrow Moshirvaziri. Determining optimal pollution control policies: An application of bilevel programming. *European Journal of Operational Research*, 119(1):100–120, nov 1999.
2 citations in pages 16 and 17.

[11] G. Anandalingam and D.J. White. A solution method for the linear static stackelberg problem using penalty functions. *IEEE Transactions on Automatic Control*, 35(10):1170–1173, 1990.
2 citations in pages 33 and 38.

[12] Martin Andersson, Sunith Bandaru, Amos Ng, and Anna Syberfeldt. Parameter tuning of MOEAs using a bilevel optimization approach. In *Lecture Notes in Computer Science*, pages 233–247. Springer International Publishing, 2015.
2 citations in pages 15 and 17.

[13] Peter J. Angeline. Genetic programming: On the programming of computers by means of natural selection,. *Biosystems*, 33(1):69–73, jan 1994.
3 citations in pages 96, 97, and 111.

[14] Jaqueline S. Angelo and Helio J. C. Barbosa. A study on the use of heuristics to solve a bilevel programming problem. *International Transactions in Operational Research*, 22(5):861–882, jan 2015.
2 citations in pages 40 and 46.

[15] Jaqueline S. Angelo, Eduardo Krempser, and Helio J.C. Barbosa. Differential evolution assisted by a surrogate model for bilevel programming problems. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, jul 2014.
2 citations in pages 45 and 46.

[16] Ritu Arora, David Schmitt, Balasubramanyam Karanam, Ming Tan, Clayton Yates, and Windy Dean-Colomb. Inhibition of the Warburg effect with a natural compound reveals a novel measurement for determining the metastatic potential of breast cancers. *Oncotarget*, 6(2), jan 2015.
One citation in page 76.

[17] J.M. Arroyo and F.J. Fernandez. A genetic algorithm approach for the analysis of electric grid interdiction with line switching. In *2009 15th International Conference on Intelligent System Applications to Power Systems*. IEEE, nov 2009.
One citation in page 18.

[18] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard. On the linear maxmin and related programming problems. In *Multilevel Optimization: Algorithms and Applications*, pages 181–208. Springer US, 1998.
One citation in page 27.

[19] Md. Abul Kalam Azad, Ana Maria A. C. Rocha, and Edite M. G. P. Fernandes. Solving large 0–1 multidimensional knapsack problems by a new simplified binary artificial fish swarm algorithm. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 14(3):313–330, feb 2015.
One citation in page 97.

[20] Gary D Bader and Christopher WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1):2, 2003.
One citation in page 52.

[21] Mohamed Bader-El-Den and Riccardo Poli. Generating SAT local-search heuristics using a GP hyper-heuristic framework. In *Lecture Notes in Computer Science*, pages 37–49. Springer Berlin Heidelberg.
2 citations in pages 94 and 95.

[22] R Bai, E K Burke, and G Kendall. Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *Journal of the Operational Research Society*,

59(10):1387–1397, oct 2008.
One citation in page 96.

[23] Ruibin Bai, Jacek Blazewicz, Edmund K. Burke, Graham Kendall, and Barry McCollum. A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR*, 10(1):43–66, nov 2011.
One citation in page 96.

[24] Sanghamitra Bandyopadhyay and Ujjwal Maulik. An evolutionary technique based on k-means algorithm for optimal clustering in RN. *Information Sciences*, 146(1-4):221–237, oct 2002.
2 citations in pages 51 and 52.

[25] Sanghamitra Bandyopadhyay, Ujjwal Maulik, and Anirban Mukhopadhyay. Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1506–1511, may 2007.
2 citations in pages 51 and 52.

[26] J. F. Bard. Some properties of the bilevel programming problem. *Journal of Optimization Theory and Applications*, 68(2):371–378, feb 1991.
3 citations in pages 2, 23, and 30.

[27] Jonathan F Bard. Coordination of a multidivisional organization through two levels of management. *Omega*, 11(5):457–468, jan 1983.
One citation in page 11.

[28] Jonathan F. Bard. An investigation of the linear three level programming problem. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14(5):711–717, sep 1984.
2 citations in pages 43 and 46.

[29] Jonathan F. Bard. Optimality conditions for the bilevel programming problem. *Naval Research Logistics Quarterly*, 31(1):13–26, mar 1984.
One citation in page 21.

[30] Jonathan F. Bard. *Practical Bilevel Optimization*. Springer US, 1998.
6 citations in pages 24, 31, 32, 34, 36, and 38.

[31] Jonathan F. Bard and James E. Falk. An explicit solution to the multi-level programming problem. *Computers & Operations Research*, 9(1):77–100, jan 1982.
4 citations in pages 28, 29, 31, and 38.

[32] Jonathan F. Bard and James T. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292, mar 1990.
One citation in page 37.

[33] Jonathan F. Bard and James T. Moore. An algorithm for the discrete bilevel programming problem. *Naval Research Logistics*, 39(3):419–435, apr 1992.
3 citations in pages 31, 36, and 38.

[34] J. E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, nov 1990.
One citation in page 113.

[35] Omar Ben-Ayed and Charles E. Blair. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3):556–560, jun 1990.
2 citations in pages 2 and 23.

[36] Omar Ben-Ayed, David E. Boyce, and Charles E. Blair. A general bilevel linear programming formulation of the network design problem. *Transportation Research Part B: Methodological*, 22(4):311–318, aug 1988.
2 citations in pages 32 and 38.

[37] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science*, 2(1):3–19, jan 2005.
3 citations in pages 29, 37, and 55.

[38] K.P. Bennett, Jing Hu, Xiaoyun Ji, G. Kunapuli, and Jong-Shi Pang. Model selection via bilevel optimization. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE.
2 citations in pages 15 and 17.

[39] Kristin P. Bennett, Gautam Kunapuli, Jing Hu, and Jong-Shi Pang. Bilevel optimization and machine learning. In *Lecture Notes in Computer Science*, pages 25–47. Springer Berlin Heidelberg.
2 citations in pages 15 and 17.

[40] B. Bhattacharjee, P. Lemonidis, W.H. Green Jr., and P.I. Barton. Global solution of semi-infinite programs. *Mathematical Programming*, 103(2):283–307, apr 2005.
2 citations in pages 27 and 29.

[41] Binita Bhattacharjee, William H. Green, and Paul I. Barton. Interval methods for semi-infinite programs. *Computational Optimization and Applications*, 30(1):63–93, jan 2005.
2 citations in pages 27 and 29.

[42] Hua BI, Hong-Li LIANG, and Jue WANG. Resampling methods and machine learning. *Chinese Journal of Computers*, 32(5):862–877, aug 2009.
One citation in page 81.

[43] W. Bialas and M. Karwan. On two-level optimization. *IEEE Transactions on Automatic Control*, 27(1):211–214, feb 1982.
2 citations in pages 30 and 38.

[44] Wayne F. Bialas and Mark H. Karwan. Two-level linear programming. *Management Science*, 30(8):1004–1020, aug 1984.
2 citations in pages 32 and 38.

[45] Lucio Bianco, Massimiliano Caramia, and Stefano Giordani. A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: Emerging Technologies*, 17(2):175–196, apr 2009.
One citation in page 18.

[46] C. G. E. Boender and Jonas Mockus. Bayesian approach to global optimization–theory and applications. *Mathematics of Computation*, 56(194):878, apr 1991.
One citation in page 81.

[47] Eric Bonnet, Eric Viara, Inna Kuperstein, Laurence Calzone, David P. A. Cohen, Emmanuel Barillot, and Andrei Zinovyev. NaviCell web service for network-based data visualization. *Nucleic Acids Research*, 43(W1):W560–W565, may 2015.
One citation in page 49.

[48] Moriah Bostian, Gerald Whittaker, Brad Barnhart, Rolf Färe, and Shawna Grosskopf. Valuing water quality tradeoffs at different spatial scales: An integrated approach using bilevel optimization. *Water Resources and Economics*, 11:1–12, jul 2015.
3 citations in pages 11, 16, and 17.

[49] Moriah Bostian, Gerald Whittaker, Ankur Sinha, and Bradley Barnhart. Incorporating data envelopment analysis solution methods into bilevel multi-objective optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, may 2015.
2 citations in pages 16 and 17.

[50] Anthony Brabazon, Michael O'Neill, and Seán McGarraghy. Grammar-based and developmental genetic programming. In *Natural Computing Algorithms*, pages 345–356. Springer Berlin Heidelberg, 2015.
2 citations in pages 96 and 98.

[51] Jerome Bracken and James T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, feb 1973.
2 citations in pages 11 and 18.

[52] Jerome Bracken and James T. McGill. Defense applications of mathematical programs with optimization problems in the constraints. *Operations Research*, 22(5):1086–1096, oct 1974.
3 citations in pages 11, 12, and 17.

[53] Luce Brotcorne, Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4):345–358, nov 2001.
6 citations in pages 11, 12, 17, 18, 108, and 139.

[54] Gerald Brown, Matthew Carlyle, Douglas Diehl, Jeffrey Kline, and Kevin Wood. A two-sided optimization for theater ballistic missile defense. *Operations Research*, 53(5):745–763, oct 2005.
2 citations in pages 13 and 17.

[55] Gerald Brown, Matthew Carlyle, Robert Harney, Eric Skroch, and Kevin Wood. Interdicting a nuclear-weapons project. Technical report, dec 2007.
3 citations in pages 13, 17, and 18.

[56] Gerald Brown, Matthew Carlyle, Javier Salmerón, and Kevin Wood. Defending critical infrastructure. *Interfaces*, 36(6):530–544, dec 2006.
2 citations in pages 13 and 17.

[57] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, dec 2013.
One citation in page 95.

[58] Edmund K. Burke, Mathew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R. Woodward. Exploring hyper-heuristic methodologies with genetic programming. In *Intelligent Systems Reference Library*, pages 177–201. Springer Berlin Heidelberg, 2009.
One citation in page 96.

[59] Edmund K. Burke, Matthew R. Hyde, and Graham Kendall. Grammatical evolution of local search heuristics. *IEEE Transactions on Evolutionary Computation*, 16(3):406–417, jun 2012.
One citation in page 96.

[60] Y. Cakir and C. Guzelis. Dynamic solvers for linear optimization problems. In *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference, 2004.* IEEE.
One citation in page 68.

[61] Herminia I. Calvete, Carmen Galé, and José A. Iranzo. An efficient evolutionary algorithm for the ring star problem. *European Journal of Operational Research*, 231(1):22–33, nov 2013.
2 citations in pages 40 and 46.

[62] Herminia I. Calvete, Carmen Galé, and Pedro M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1):14–28, jul 2008.
2 citations in pages 39 and 46.

[63] Herminia I. Calvete, Carmen Galé, and María-José Oliveros. Bilevel model for production–distribution planning solved by using ant colony optimization. *Computers & Operations Research*, 38(1):320–327, jan 2011.
2 citations in pages 40 and 46.

[64] José-Fernando Camacho-Vallejo, Álvaro Eduardo Cordero-Franco, and Rosa G. González-Ramírez. Solving the bilevel facility location problem under preferences by a stackelberg-evolutionary algorithm. *Mathematical Problems in Engineering*, 2014:1–14, 2014.
2 citations in pages 13 and 17.

[65] José-Fernando Camacho-Vallejo, Julio Mar-Ortiz, Francisco López-Ramos, and Ricardo Pedraza Rodríguez. A genetic algorithm for the bi-level topological design of local area networks. *PLOS ONE*, 10(6):e0128067, jun 2015.
2 citations in pages 14 and 17.

[66] Wilfred Candler, Jose Fortuny-Amat, and Bruce McCarl. The potential role of multilevel programming in agricultural economics. *American Journal of Agricultural Economics*, 63(3):521, aug 1981.
One citation in page 11.

[67] Wilfred Candler and Robert Townsley. A linear two-level programming problem. *Computers & Operations Research*, 9(1):59–76, jan 1982.
2 citations in pages 30 and 38.

[68] Massimiliano Caramia and Renato Mari. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters*, 10(5):997–1019, jul 2015.
2 citations in pages 13 and 17.

[69] Massimiliano Caramia and Renato Mari. Enhanced exact algorithms for discrete bilevel linear problems. *Optimization Letters*, 9(7):1447–1468, apr 2015.
2 citations in pages 37 and 38.

[70] Etienne Caron, Samik Ghosh, Yukiko Matsuoka, Dariel Ashton-Beaucage, Marc Therrien, Sébastien Lemieux, Claude Perreault, Philippe P Roux, and Hiroaki Kitano. A comprehensive map of the mTOR signaling network. *Molecular Systems Biology*, 6, dec 2010.
One citation in page 49.

[71] Case, Lori Michelle. *An lÂŠ1 penalty function approach to the nonlinear bilevel programming problem*. PhD thesis, 1997.
2 citations in pages 33 and 38.

[72] Mark Cecchini, Joseph Ecker, Michael Kupferschmid, and Robert Leitch. Solving nonlinear principal-agent problems using bilevel programming. *European Journal of Operational Research*, 230(2):364–373, oct 2013.
One citation in page 17.

[73] Abir Chaabani, Slim Bechikh, and Lamjed Ben Said. A co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, may 2015.
One citation in page 42.

[74] Irène Charon and Olivier Hudry. The noising method: a new method for combinatorial optimization. *Operations Research Letters*, 14(3):133–137, oct 1993.
One citation in page 97.

[75] Anthony Chen, Juyoung Kim, Seungjae Lee, and Youngchan Kim. Stochastic multi-objective models for network design problem. *Expert Systems with Applications*, 37(2):1608–1619, mar 2010.
2 citations in pages 14 and 17.

[76] Guang Ya Chen and Johannes Jahn. Optimality conditions for set-valued optimization problems. *Mathematical Methods of Operations Research*, 48(2):187–200, nov 1998.
2 citations in pages 28 and 29.

[77] Jiumei CHEN and Ying GONG. Particle swarm optimization for two-echelon location-routing problem. *Journal of Computer Applications*, 33(8):2261–2264, nov 2013.
One citation in page 97.

[78] Alexey Chernyshev. Bayesian optimization of spiking neural network parameters to solving the time series classification task. In *Advances in Intelligent Systems and Computing*, pages 39–45. Springer International Publishing, 2016.
One citation in page 81.

[79] S. Christiansen, M. Patriksson, and L. Wynter. Stochastic bilevel programming in structural optimization. *Structural and Multidisciplinary Optimization*, 21(5):361–371, jul 2001.
2 citations in pages 14 and 17.

[80] P.C. Chu and J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.
3 citations in pages 97, 100, and 103.

[81] P.A. Clark and A.W. Westerberg. Bilevel programming for steady-state chemical process design—i. fundamentals and algorithms. *Computers & Chemical Engineering*, 14(1):87–97, jan 1990.
2 citations in pages 16 and 17.

[82] Benoît Colson, Patrice Marcotte, and Gilles Savard. A trust-region method for nonlinear bilevel programming: Algorithm and computational experience. *Computational Optimization and Applications*, 30(3):211–227, mar 2005.
2 citations in pages 33 and 38.

[83] I Constantin. Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. *International Transactions in Operational Research*, 2(2):149–164, apr 1995.
2 citations in pages 12 and 17.

[84] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–24, jan 1943.
One citation in page 18.

[85] M.C. Cowgill, R.J. Harvey, and L.T. Watson. A genetic algorithm approach to cluster analysis. *Computers & Mathematics with Applications*, 37(7):99–108, apr 1999.
2 citations in pages 51 and 52.

[86] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg, 2001.
One citation in page 95.

[87] Noel Cressie. The origins of kriging. *Mathematical Geology*, 22(3):239–252, Apr 1990.
One citation in page 81.

[88] Tunchan Cura. A particle swarm optimization approach to clustering. *Expert Systems with Applications*, 39(1):1582–1588, jan 2012.
One citation in page 51.

[89] Paul Inuwa Dalatu, Anwar Fitrianto, and Aida Mustapha. Hybrid distance functions for k-means clustering algorithms. *Statistical Journal of the IAOS*, 33(4):989–996, nov 2017.
One citation in page 50.

[90] Frank Dammeyer and Stefan Voß. Dynamic tabu list management using the reverse elimination method. *Annals of Operations Research*, 41(2):29–46, jun 1993.
One citation in page 97.

[91] Charles Darwin. *On the origin of species*. D. Appleton and Co.,, 1871.
One citation in page 125.

[92] Bianca de Almeida Dantas and Edson Norberto Caceres. A parallel implementation to the multidimensional knapsack problem using augmented neural networks. In *2014 XL Latin American Computing Conference (CLEI)*. IEEE, sep 2014.
One citation in page 97.

[93] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, apr 2002.
2 citations in pages 62 and 68.

[94] S. Dempe. A simple algorithm for the-linear bilevel programming problem. *Optimization*, 18(3):373–385, jan 1987.
2 citations in pages 30 and 38.

[95] S. Dempe. Set-valued optimization – an introduction with applications. *Optimization*, 65(4):905–906, jul 2015.
2 citations in pages 28 and 29.

[96] S. Dempe, J. Dutta, and B. S. Mordukhovich. New necessary optimality conditions in optimistic bilevel programming. *Optimization*, 56(5-6):577–604, oct 2007.
One citation in page 21.

[97] S. Dempe, B.S. Mordukhovich, and A.B. Zemkoho. Necessary optimality conditions in pessimistic bilevel programming. *Optimization*, 63(4):505–533, jun 2012.
One citation in page 22.

[98] Stephan Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, 2002.
3 citations in pages 20, 21, and 22.

[99] Stephan Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 52(3):333–359, jun 2003.
2 citations in pages 28 and 29.

[100] Stephan Dempe and Maria Pilecka. Necessary optimality conditions for optimistic bilevel programming problems using set-valued programming. *Journal of Global Optimization*, 61(4):769–788, may 2014.
One citation in page 21.

[101] S. T. DeNegre and T. K. Ralphs. A branch-and-cut algorithm for integer bilevel linear programs. In *Operations Research and Cyber-Infrastructure*, pages 65–78. Springer US, 2009.
3 citations in pages 34, 37, and 38.

[102] Xiaotie Deng. Complexity issues in bilevel linear programming. In *Multilevel Optimization: Algorithms and Applications*, pages 149–164. Springer US, 1998.
One citation in page 23.

[103] Loukas Dimitriou, Theodore Tsekeris, and Antony Stathopoulos. Genetic computation of road network design and pricing stackelberg games with multi-class users. In *Lecture Notes in Computer Science*, pages 669–678. Springer Berlin Heidelberg, 2008.
2 citations in pages 40 and 46.

[104] Kasper Dinkla, Mohammed El-Kebir, Cristina-Iulia Bucur, Marco Siderius, Martine J Smit, Michel A Westenberg, and Gunnar W Klau. eXamine: Exploring annotated modules in networks. *BMC Bioinformatics*, 15(1):201, 2014.
One citation in page 52.

[105] Trivikram Dokka, Alain B. Zemkoho, Sonali Sen Gupta, and Fabrice T. Nobibon. Robust toll pricing: A novel approach. *arxiv:1712.01570v1*, 2017.
2 citations in pages 12 and 17.

[106] A M Dolga, A de Andrade, L Meissner, H-G Knaus, M Höllerhage, P Christophersen, H Zischka, N Plesnila, G U Höglinger, and C Culmsee. Subcellular expression and neuroprotective effects of SK channels in human dopaminergic neurons. *Cell Death & Disease*, 5(1):e999–e999, jan 2014.
One citation in page 76.

[107] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, nov 2005.
One citation in page 38.

[108] Bernabé Dorronsoro, Patricia Ruiz, Grégoire Danoy, Yoann Pigné, and Pascal Bouvry. *Evolutionary Algorithms for Mobile Ad Hoc Networks*. John Wiley & Sons, Inc., apr 2014.
3 citations in pages 101, 117, and 132.

[109] John H Drake, Matthew Hyde, Khaled Ibrahim, and Ender Ozcan. A genetic programming hyper-heuristic for the multidimensional knapsack problem. *Kybernetes*, 43(9/10):1500–1511, nov 2014.
13 citations in pages xii, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, and 110.

[110] John H. Drake, Ender Özcan, and Edmund K. Burke. A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional knapsack problem. *Evolutionary Computation*, 24(1):113–141, mar 2016.
One citation in page 94.

[111] John H. Drake, Ender Özcan, and Edmund K. Burke. A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional knapsack problem. *Evolutionary Computation*, 24(1):113–141, mar 2016.
One citation in page 103.

[112] Ding-Zhu Du and Panos M. Pardalos, editors. *Minimax and Applications*. Springer US, 1995.
2 citations in pages 27 and 29.

[113] Gunter Dueck and Tobias Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175, sep 1990.
One citation in page 97.

[114] Joydeep Dutta, Kalyanmoy Deb, Rupesh Tulshyan, and Ramnik Arora. Approximate KKT points and a proximity measure for termination. *Journal of Global Optimization*, 56(4):1463–1499, may 2012.
2 citations in pages 41 and 46.

[115] T.A. Edmunds and J.F. Bard. Algorithms for nonlinear bilevel mathematical programs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(1):83–89, 1991.
2 citations in pages 28 and 29.

[116] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.
One citation in page 58.

[117] Paul R. Ehrlich and Peter H. Raven. Butterflies and plants: A study in coevolution. *Evolution*, 18(4):586, dec 1964.
One citation in page 125.

[118] Achiya Elyasaf, Ami Hauptman, and Moshe Sipper. Evolutionary design of FreeCell solvers. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(4):270–281, dec 2012.
One citation in page 96.

[119] Antonio Fabregat, Konstantinos Sidiropoulos, Phani Garapati, Marc Gillespie, Kerstin Hausmann, Robin Haw, Bijay Jassal, Steven Jupe, Florian Korninger, Sheldon McKay, Lisa Matthews, Bruce May, Marija Milacic, Karen Rothfels, Veronica Shamovsky, Marissa Webber, Joel Weiser, Mark Williams, Guanming Wu, Lincoln Stein, Henning Hermjakob, and Peter D'Eustachio. The reactome pathway knowledgebase. *Nucleic Acids Research*, 44(D1):D481–D487, dec 2015.
One citation in page 49.

[120] James E. Falk. A linear max—min problem. *Mathematical Programming*, 5(1):169–188, dec 1973.
2 citations in pages 27 and 29.

[121] James E. Falk and Karla Hoffman. A nonconvex max-min problem. *Naval Research Logistics Quarterly*, 24(3):441–450, sep 1977.
2 citations in pages 27 and 29.

[122] James E. Falk and Jiming Liu. On bilevel programming, part i: General nonlinear cases. *Mathematical Programming*, 70(1-3):47–72, oct 1995.
2 citations in pages 32 and 38.

[123] Wei Fan. Optimal congestion pricing toll design for revenue maximization: comprehensive numerical results and implications. *Canadian Journal of Civil Engineering*, 42(8):544–551, aug 2015.
2 citations in pages 12 and 17.

[124] Wei Fan and Randy Machemehl. Bi-level optimization model for public transportation network redesign problem. *Transportation Research Record: Journal of the Transportation Research Board*, 2263:151–162, dec 2011.
2 citations in pages 14 and 17.

[125] Henrique Fingler, Edson N. Cáceres, Henrique Mongelli, and Siang W. Song. A CUDA based solution to the multidimensional knapsack problem using the ant colony optimization. *Procedia Computer Science*, 29:84–94, 2014.
One citation in page 97.

[126] Bruce Finlayson. Chemical reaction equilibrium. In *Introduction to Chemical Engineering Computing*, pages 41–54. John Wiley & Sons, Inc., feb 2006.
One citation in page 18.

[127] Matteo Fischetti, Ivana Ljubić, Michele Monaci, and Markus Sinnl. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65(6):1615–1637, dec 2017.
2 citations in pages 37 and 38.

[128] J. Fliege and L. N. Vicente. Multicriteria approach to bilevel optimization. *Journal of Optimization Theory and Applications*, 131(2):209–225, nov 2006.
2 citations in pages 43 and 46.

[129] Lope A. Flórez, Christoph R. Lammers, Raphael Michna, and Jörg Stülke. CellPublisher: a web platform for the intuitive visualization and sharing of metabolic, signalling and regulatory pathways: Fig. 1. *Bioinformatics*, 26(23):2997–2999, oct 2010.
One citation in page 49.

[130] Christodoulos A. Floudas and Oliver Stein. The adaptive convexification algorithm: A feasible point method for semi-infinite programming. *SIAM Journal on Optimization*, 18(4):1187–1208, jan 2008.
2 citations in pages 27 and 29.

[131] Pirmin Fontaine and Stefan Minner. Benders decomposition for discrete–continuous linear bilevel problems with application to traffic network design. *Transportation Research Part B: Methodological*, 70:163–172, dec 2014.
2 citations in pages 37 and 38.

[132] José Fortuny-Amat and Bruce McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, 32(9):783–792, sep 1981.
2 citations in pages 31 and 38.

[133] C. C. Foster. Information retrieval. In *Proceedings of the 1965 20th national conference on* -. ACM Press, 1965.
One citation in page 66.

[134] Anton Frantsev, Ankur Sinha, and Pekka Malo. Finding optimal strategies in multi-period stackelberg games using an evolutionary framework. *IFAC Proceedings Volumes*, 45(25):33–38, 2012.
2 citations in pages 26 and 29.

[135] Arnaud Fréville. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21, may 2004.
One citation in page 97.

[136] Arnaud Freville and Gérard Plateau. An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. *Discrete Applied Mathematics*, 49(1-3):189–212, mar 1994.
2 citations in pages 97 and 103.

[137] Terry L. Friesz, G. Anandalingam, Nihal J. Mehta, Keesung Nam, Samir J. Shah, and Roger L. Tobin. The multiobjective equilibrium network design problem revisited: A simulated annealing approach. *European Journal of Operational Research*, 65(1):44–57, feb 1993.
2 citations in pages 41 and 46.

[138] Muhammad Marwan Muhammad Fuad. Differential evolution-based weighted combination of distance metrics for k-means clustering. In *Theory and Practice of Natural Computing*, pages 193–204. Springer International Publishing, 2014.
One citation in page 58.

[139] Kazuhiro A. Fujita, Marek Ostaszewski, Yukiko Matsuoka, Samik Ghosh, Enrico Glaab, Christophe Trefois, Isaac Crespo, Thanneer M. Perumal, Wiktor Jurkowski, Paul M. A. Antony, Nico Diederich, Manuel Buttini, Akihiko Kodama, Venkata P. Satagopam, Serge Eifes, Antonio del Sol, Reinhard Schneider, Hiroaki Kitano, and Rudi Balling. Integrating pathways of parkinson's disease in a molecular interaction map. *Molecular Neurobiology*, 49(1):88–102, jul 2013.
5 citations in pages 49, 50, 52, 63, and 76.

[140] Shiho Fujiwara, Naoko Wada, Yawara Kawano, Yutaka Okuno, Yoshitaka Kikukawa, Shinya Endo, Nao Nishimura, Nina Ueno, Hiroaki Mitsuya, and Hiroyuki Hata. Lactate, a putative survival factor for myeloma cells, is incorporated by myeloma cells through monocarboxylate transporters 1. *Experimental Hematology & Oncology*, 4(1):12, dec 2015.
One citation in page 76.

[141] Christian Gagn. DEAP : Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.
One citation in page 69.

[142] Mariano Gallo, Luca D'Acierno, and Bruno Montella. A meta-heuristic approach for solving the urban network design problem. *European Journal of Operational Research*, 201(1):144–157, feb 2010.
2 citations in pages 14 and 17.

[143] Ya Gao, Guangquan Zhang, Jie Lu, and Hui-Ming Wee. Particle swarm optimization for bi-level pricing problems in supply chains. *Journal of Global Optimization*, 51(2):245–254, sep 2010.
2 citations in pages 40 and 46.

[144] Alberto García-Villoria, Said Salhi, Albert Corominas, and Rafael Pastor. Hyper-heuristic approaches for the response time variability problem. *European Journal of Operational Research*, 211(1):160–169, may 2011.
One citation in page 96.

[145] John E. Garen. Executive compensation and principal-agent theory. *Journal of Political Economy*, 102(6):1175–1199, dec 1994.
One citation in page 17.

[146] Pablo Garrido and María Cristina Riff. DVRP: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics*, 16(6):795–834, feb 2010.
One citation in page 96.

[147] Piotr Gawron, Marek Ostaszewski, Venkata Satagopam, Stephan Gebel, Alexander Mazein, Michal Kuzma, Simone Zorzan, Fintan McGee, Benoît Otjacques, Rudi Balling, and Reinhard Schneider. MINERVA—a platform for visualization and curation of molecular interaction networks. *npj Systems Biology and Applications*, 2(1), sep 2016.
One citation in page 49.

[148] Raffaele Giancarlo, Giosuè Lo Bosco, and Luca Pinello. Distance functions, clustering algorithms and microarray data analysis. In *Lecture Notes in Computer Science*, pages 125–138. Springer Berlin Heidelberg, 2010.
One citation in page 50.

[149] Enrico Glaab and Reinhard Schneider. Comparative pathway and network analysis of brain transcriptome changes during adult aging and in Parkinson's disease. *Neurobiology of Disease*, 74:1–13, feb 2015.
One citation in page 74.

[150] Linda Glover and Peter Butler. High-performance work systems, partnership and the working lives of HR professionals. *Human Resource Management Journal*, 22(2):199–215, jan 2011.
One citation in page 95.

[151] R. Gnanadesikan. *Methods for Statistical Data Analysis of Multivariate Observations*. John Wiley & Sons, Inc., jan 1997.
One citation in page 50.

[152] Philipp Gobrecht, Marco Leibinger, Anastasia Andreadaki, and Dietmar Fischer. Sustained GSK3 activity markedly facilitates nerve regeneration. *Nature Communications*, 5, jul 2014.
One citation in page 76.

[153] Atul Gohad, Nanjangud C. Narendra, and Parathasarthy Ramachandran. Cloud pricing models: A survey and position paper. In *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE, oct 2013.
One citation in page 107.

[154] I.E. Grossmann and C.A. Floudas. Active constraint strategy for flexibility analysis in chemical processes. *Computers & Chemical Engineering*, 11(6):675–693, jan 1987.
One citation in page 37.

[155] Kai gui WU. Formal analysis of IEEE 802.11i. *Journal of Computer Applications*, 28(5):1125–1127, oct 2008.
One citation in page 50.

[156] Richard F. Gunst. Response surface methodology: Process and product optimization using designed experiments. *Technometrics*, 38(3):284–286, aug 1996.
2 citations in pages 45 and 46.

[157] Aaron M. Gusdon, Jianhui Zhu, Bennett Van Houten, and Charleen T. Chu. ATP13A2 regulates mitochondrial bioenergetics through macroautophagy. *Neurobiology of Disease*, 45(3):962–972, mar 2012.
One citation in page 76.

[158] W. Halter and S. Mostaghim. Bilevel optimization of multi-component chemical systems using particle swarm optimization. In *2006 IEEE International Conference on Evolutionary Computation*. IEEE.
3 citations in pages 11, 16, and 17.

[159] Said Hanafi and Christophe Wilbaut. Scatter search for the 0–1 multidimensional knapsack problem. *Journal of Mathematical Modelling and Algorithms*, 7(2):143–159, feb 2008.
One citation in page 103.

[160] Julia Handl and Joshua Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76, feb 2007.
2 citations in pages 51 and 52.

[161] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, sep 1992.
6 citations in pages 23, 28, 29, 30, 32, and 38.

[162] Patrick T Harker and Jong-Shi Pang. Existence of optimal solutions to mathematical programs with equilibrium constraints. *Operations Research Letters*, 7(2):61–64, apr 1988.
2 citations in pages 18 and 21.

[163] Juris Hartmanis. Computers and intractability: A guide to the theory of NP-completeness (michael r. garey and david s. johnson). *SIAM Review*, 24(1):90–91, jan 1982.
2 citations in pages 38 and 96.

[164] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4-6):175–181, dec 2000.
One citation in page 52.

[165] S.R. Hejazi, A. Memariani, G. Jahanshahloo, and M.M. Sepehri. Linear bilevel programming solution by genetic algorithm. *Computers & Operations Research*, 29(13):1913–1925, nov 2002.
2 citations in pages 41 and 46.

[166] Mehdi Hemmati and J. Cole Smith. A mixed-integer bilevel programming approach for a competitive prioritized set covering problem. *Discrete Optimization*, 20:105–134, may 2016.
2 citations in pages 37 and 38.

[167] J. Herskovits, A. Leontiev, G. Dias, and G. Santos. Contact shape optimization: a bilevel programming approach. *Structural and Multidisciplinary Optimization*, 20(3):214–221, nov 2000.
3 citations in pages 11, 14, and 17.

[168] W.Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, jun 1990.
2 citations in pages 42 and 125.

[169] John H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314, jul 1962.
One citation in page 38.

[170] Eduardo R. Hruschka, Ricardo J. G. B. Campello, and Leandro N. de Castro. Improving the efficiency of a clustering genetic algorithm. In *Advances in Artificial Intelligence – IBERAMIA 2004*, pages 861–870. Springer Berlin Heidelberg, 2004.
2 citations in pages 51 and 52.

[171] Eduardo R. Hruschka, Ricardo J.G.B. Campello, and Leandro N. de Castro. Evolving clusters in gene-expression data. *Information Sciences*, 176(13):1898–1927, jul 2006.
2 citations in pages 51 and 52.

[172] E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, and A.C.P.L.F. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(2):133–155, mar 2009.
One citation in page 51.

[173] Jasmine Irani, Nitin Pise, and Madhura Phatak. Clustering techniques and the similarity measures used in clustering: A survey. *International Journal of Computer Applications*, 134(7):9–14, jan 2016.
One citation in page 50.

[174] Yo Ishizuka and Eitaro Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1):73–88, dec 1992.
2 citations in pages 32 and 38.

[175] Md Monjurul Islam, Hemant Kumar Singh, and Tapabrata Ray. A nested differential evolution based algorithm for solving multi-objective bilevel optimization problems. In *Lecture Notes in Computer Science*, pages 101–112. Springer International Publishing, 2016.
2 citations in pages 40 and 46.

[176] Md Monjurul Islam, Hemant Kumar Singh, and Tapabrata Ray. A surrogate assisted approach for single-objective bilevel optimization. *IEEE Transactions on Evolutionary Computation*, 21(5):681–696, oct 2017.
2 citations in pages 45 and 46.

[177] Eitan Israeli and R. Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, aug 2002.
One citation in page 18.

[178] Johannes Jahn. Set-valued optimization. In *Encyclopedia of Optimization*, pages 2352–2355. Springer US.
2 citations in pages 28 and 29.

[179] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, sep 1999.
One citation in page 51.

[180] Yeongjun Jang, Namhee Yu, Jihae Seo, Sun Kim, and Sanghyuk Lee. MONGKIE: an integrated tool for network analysis and visualization for multi-omics data. *Biology Direct*, 11(1), mar 2016.
One citation in page 52.

[181] Robert G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32(2):146–164, jun 1985.
One citation in page 23.

[182] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, nov 2004.
One citation in page 51.

[183] Yan Jiang, Xuyong Li, Chongchao Huang, and Xianing Wu. Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem. *Applied Mathematics and Computation*, 219(9):4332–4339, jan 2013.
2 citations in pages 41 and 46.

[184] José Jiménez and Josep Ginebra. pyGPGO: Bayesian optimization for python. *The Journal of Open Source Software*, 2(19):431, nov 2017.
One citation in page 88.

[185] Miles Johnson, Navid Aghasadeghi, and Timothy Bretl. Inverse optimal control for deterministic continuous-time nonlinear systems. In *52nd IEEE Conference on Decision and Control*. IEEE, dec 2013.
One citation in page 11.

[186] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, sep 1967.
One citation in page 50.

[187] J. J. Júdice and A. M. Faustino. A sequential LCP method for bilevel linear programming. *Annals of Operations Research*, 34(1):89–106, dec 1992.
2 citations in pages 32 and 38.

[188] Kazunari Kaizu, Samik Ghosh, Yukiko Matsuoka, Hisao Moriya, Yuki Shimizu-Yoshida, and Hiroaki Kitano. A comprehensive molecular interaction map of the budding yeast cell cycle. *Molecular Systems Biology*, 6, sep 2010.
One citation in page 49.

[189] Vyacheslav Kalashnikov, Fernando Camacho, Ronald Askin, and Nataliya Kalashnykova. Comparison of algorithms for solving a bi-level toll setting problem. *International Journal of Innovative Computing, Information and Control*, 6(8):3529–3549, 2010.
2 citations in pages 12 and 17.

[190] Vyacheslav V. Kalashnikov, Roberto Carlos Herrera Maldonado, José-Fernando Camacho-Vallejo, and Nataliya I. Kalashnykova. A heuristic algorithm solving bilevel toll optimization problems. *The International Journal of Logistics Management*, 27(1):31–51, may 2016.
3 citations in pages 11, 12, and 17.

[191] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, 45(D1):D353–D361, nov 2016.
One citation in page 49.

[192] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, dec 1979.
One citation in page 53.

[193] G Kendall. Scheduling english football fixtures over holiday periods. *Journal of the Operational Research Society*, 59(6):743–755, jun 2008.
One citation in page 96.

[194] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*. IEEE.
One citation in page 38.

[195] Akhtar A. Khan, Christiane Tammer, and Constantin Zălinescu. *Set-valued Optimization*. Springer Berlin Heidelberg, 2015.
3 citations in pages 27, 28, and 29.

[196] Amirsaman Kheirkhah, HamidReza Navidi, and Masume Messi Bidgoli. A bi-level network interdiction model for solving the hazmat routing problem. *International Journal of Production Research*, 54(2):459–471, sep 2015.
2 citations in pages 13 and 17.

[197] Emmanuel Kieffer, Matthias Rudolf Brust, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Tackling large-scale and combinatorial bi-level problems with genetic programming hyper-heuristic. *IEEE Transactions on Evolutionary Computation*, (in review), 2018.

[198] Emmanuel Kieffer, Grégoire Danoy, and Pascal Bouvry. On bi-level approach for scheduling problems. In *New Challenges in Scheduling Theory*, 2016.

[199] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Hybrid mobility model with pheromones for uav detection task. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.
One citation in page 143.

[200] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Bayesian optimization approach of general bi-level problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*. ACM Press, 2017.
3 citations in pages 18, 45, and 46.

[201] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. Bayesian optimization approach of general bi-level problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1614–1621. ACM, 2017.

[202] Emmanuel Kieffer, Gregoire Danoy, Pascal Bouvry, and Anass Nagih. A new co-evolutionary algorithm based on constraint decomposition. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, may 2017.
One citation in page 125.

[203] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A new co-evolutionary algorithm based on constraint decomposition. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 492–500, May 2017.
One citation in page 143.

[204] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A new modeling approach for the biobjective exact optimization of satellite payload configuration. *International Transactions in Operational Research*, 2017.

[205] Emmanuel Kieffer, Gregoire Danoy, Pascal Bouvry, and Anass Nagih. A competitive approach for bi-level co-evolution. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, page 609—618. IEEE, may 2018.
One citation in page 18.

[206] Emmanuel Kieffer, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A competitive approach for bi-level co-evolution. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018.
One citation in page 108.

[207] Emmanuel Kieffer, Mateusz Guzek, Grégoire Danoy, Pascal Bouvry, and Anass Nagih. A novel co-evolutionary approach for constrained genetic algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, pages 47–48. ACM, 2016.

[208] Emmanuel Kieffer, Martin Rosalie, Grégoire Danoy, and Pascal Bouvry. Bayesian optimization to enhance coverage performance of a swarm of uav with chaotic dynamics. May 2018.
One citation in page 142.

[209] A. D. King, N. Przulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, jun 2004.
One citation in page 52.

[210] Polyxeni-M. Kleniati and Claire S. Adjiman. Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. part II: Convergence analysis and numerical results. *Journal of Global Optimization*, 60(3):459–481, jan 2014.
2 citations in pages 33 and 38.

[211] Polyxeni-M. Kleniati and Claire S. Adjiman. A generalization of the branch-and-sandwich algorithm: From continuous to mixed-integer nonlinear bilevel problems. *Computers & Chemical Engineering*, 72:373–386, jan 2015.
2 citations in pages 33 and 38.

[212] Polyxeni-Margarita Kleniati and Claire S. Adjiman. Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. part i: Theoretical development. *Journal of Global Optimization*, 60(3):425–458, jan 2014.
2 citations in pages 33 and 38.

[213] Gary A. Kochenberger, Bruce A. McCarl, and F. Paul Wyman. A HEURISTIC FOR GENERAL INTEGER PROGRAMMING. *Decision Sciences*, 5(1):36–44, jan 1974.
One citation in page 97.

[214] M. Kočvara. Topology optimization with displacement constraints: a bilevel programming approach. *Structural Optimization*, 14(4):256–263, dec 1997.
2 citations in pages 14 and 17.

[215] Michal Kočvara and Jiří V. Outrata. On the solution of optimum design problems with variational inequalities. In *Recent Advances in Nonsmooth Optimization*, pages 172–192. WORLD SCIENTIFIC, sep 1995.
2 citations in pages 14 and 17.

[216] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
One citation in page 114.

[217] Bhawna Kohli. Optimality conditions for optimistic bilevel programming problem using convexifactors. *Journal of Optimization Theory and Applications*, 152(3):632–651, oct 2011.
One citation in page 21.

[218] C. D. Kolstad and L. S. Lasdon. Derivative evaluation and computational experience with large bilevel mathematical programs. *Journal of Optimization Theory and Applications*, 65(3):485–499, jun 1990.
2 citations in pages 32 and 38.

[219] Xiangyu Kong, GuoLin Yu, and Wei Liu. Optimality for set-valued optimization in the sense of vector and set criteria. *Journal of Inequalities and Applications*, 2017(1), feb 2017.
2 citations in pages 28 and 29.

[220] Mina Moradi Kordmahalleh, Mohammad Gorji Sefidmazgi, and Abdollah Homaifar. A bilevel parameter tuning strategy of partially connected ANNs. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, dec 2015.
2 citations in pages 15 and 17.

[221] JohnR. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2), jun 1994.
One citation in page 94.

[222] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.
2 citations in pages 117 and 119.

[223] G. Kunapuli, K.P. Bennett, Jing Hu, and Jong-Shi Pang. Classification model selection via bilevel programming. *Optimization Methods and Software*, 23(4):475–489, aug 2008.
2 citations in pages 15 and 17.

[224] Ludmila I Kuncheva and James C Bezdek. Selection of cluster prototypes from data by a genetic algorithm. In *Proc. 5th European congress on Intelligent techniques and soft computing*, pages 1683–1688, 1997.
2 citations in pages 51 and 52.

[225] R.J. Kuo and C.C. Huang. Application of particle swarm optimization algorithm for solving bi-level linear programming problem. *Computers & Mathematics with Applications*, 58(4):678–685, aug 2009.
2 citations in pages 40 and 46.

[226] I Kuperstein, E Bonnet, H-A Nguyen, D Cohen, E Viara, L Grieco, S Fourquet, L Calzone, C Russo, M Kondratova, M Dutreix, E Barillot, and A Zinovyev. Atlas of cancer signalling network: a systems biology resource for integrative analysis of cancer data with google maps. *Oncogenesis*, 4(7):e160–e160, jul 2015.
2 citations in pages 49 and 50.

[227] Martina Kutmon, Anders Riutta, Nuno Nunes, Kristina Hanspers, Egon L. Willighagen, Anwesha Bohler, Jonathan Mélius, Andra Waagmeester, Sravanthi R. Sinha, Ryan Miller, Susan L. Coort, Elisa Cirillo, Bart Smeets, Chris T. Evelo, and Alexander R. Pico. WikiPathways: capturing the full diversity of pathway knowledge. *Nucleic Acids Research*, 44(D1):D488–D494, oct 2015.
One citation in page 49.

[228] Hande Küçükaydın, Necati Aras, and İ. Kuban Altınel. A hybrid tabu search heuristic for a bilevel competitive facility location model. In *Hybrid Metaheuristics*, pages 31–45. Springer Berlin Heidelberg, 2010.
2 citations in pages 40 and 46.

[229] Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12-part-1):1608–1622, dec 1998.
2 citations in pages 12 and 17.

[230] Philippe Laborie and Jérôme Rogerie. Temporal linear relaxation in IBM ILOG CP optimizer. *Journal of Scheduling*, 19(4):391–400, nov 2014.
One citation in page 68.

[231] Guoming Lai, Dehui Yuan, and Shenyun Yang. A new hybrid combinatorial genetic algorithm for multi-dimensional knapsack problems. *The Journal of Supercomputing*, 70(2):930–945, jul 2014.
One citation in page 97.

[232] Xiangjing Lai and Jin-Kao Hao. Iterated variable neighborhood search for the capacitated clustering problem. *Engineering Applications of Artificial Intelligence*, 56:102–120, nov 2016.
One citation in page 51.

[233] T. Van Le. Evolutionary fuzzy clustering. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. IEEE.
2 citations in pages 51 and 52.

[234] Larry J. LeBlanc and David E. Boyce. A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows. *Transportation Research Part B: Methodological*, 20(3):259–265, jun 1986.
2 citations in pages 14 and 17.

[235] F Legillon, A Liefooghe, and E Talbi. CoBRA: A cooperative coevolutionary algorithm for bi-level optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, 2012.
6 citations in pages 42, 46, 115, 116, 126, and 129.

[236] Chuanjia Li and Lei Guo. A single-level reformulation of mixed integer bilevel programming problems. *Operations Research Letters*, 45(1):1–5, jan 2017.
2 citations in pages 37 and 38.

[237] Hecheng Li. A genetic algorithm using a finite search space for solving nonlinear/linear fractional bilevel programming problems. *Annals of Operations Research*, 235(1):543–558, may 2015.
2 citations in pages 41 and 46.

[238] Hecheng LI and Yuping WANG. An evolutionary algorithm based on a new decomposition scheme for nonlinear bilevel programming problems. *International Journal of Communications, Network and System Sciences*, 03(01):87–93, 2010.
2 citations in pages 40 and 46.

[239] Xiangyong Li, Peng Tian, and Xiaoping Min. A hierarchical particle swarm optimization for solving bilevel programming problems. In *Artificial Intelligence and Soft Computing – ICAISC 2006*, pages 1169–1178. Springer Berlin Heidelberg, 2006.
2 citations in pages 40 and 46.

[240] Jason Zhi Liang and Risto Miikkulainen. Evolutionary bilevel optimization for complex control tasks. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*. ACM Press, 2015.
3 citations in pages 11, 15, and 17.

[241] M. B. Lignola and J. Morgan. Topological existence and stability for stackelberg problems. *Journal of Optimization Theory and Applications*, 84(1):145–169, jan 1995.
2 citations in pages 26 and 29.

[242] Maria Beatrice Lignola and Jacqueline Morgan. Existence of solutions to bilevel variational problems in banach spaces. In *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Models*, pages 161–174. Kluwer Academic Publishers.
One citation in page 21.

[243] Bei Liu, Le Niu, Ming-Zhi Shen, Lei Gao, Chao Wang, Jie Li, Li-Jia Song, Ye Tao, Qiang Meng, Qian-Li Yang, Guo-Dong Gao, and Hua Zhang. Decreased Astroglial Monocarboxylate Transporter 4 Expression in Temporal Lobe Epilepsy. *Molecular Neurobiology*, 50(2):327–338, oct 2014.
One citation in page 76.

[244] Bo Liu, Qingfu Zhang, and Georges G. E. Gielen. A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(2):180–192, apr 2014.
One citation in page 84.

[245] Guoshan Liu, Jiye Han, and Shouyang Wang. A trust region algorithm for bilevel programing problems. *Chinese Science Bulletin*, 43(10):820–824, may 1998.
2 citations in pages 33 and 38.

[246] Richa Loohach and Kanwal Garg. Effect of distance functions on simple k-means clustering algorithm. *International Journal of Computer Applications*, 49(6):7–9, jul 2012.
One citation in page 50.

[247] Eunice López-Camacho, Hugo Terashima-Marín, Peter Ross, and Manuel Valenzuela-Rendón. Problem-state representations in a hyper-heuristic approach for the 2d irregular BPP. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*. ACM Press, 2010.
One citation in page 96.

[248] Pierre Loridan and Jacqueline Morgan. Weak via strong stackelberg problem: New results. *Journal of Global Optimization*, 8(3):263–287, apr 1996.
3 citations in pages 22, 26, and 29.

[249] James H. Lorie and Leonard J. Savage. Three problems in rationing capital. *The Journal of Business*, 28(4):229, jan 1955.
One citation in page 96.

[250] Richard Loulou and Eleftherios Michaelides. New greedy-like heuristics for the multidimensional 0-1 knapsack problem. *Operations Research*, 27(6):1101–1114, dec 1979.
One citation in page 97.

[251] Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, and Susan J Brown. *BMC Bioinformatics*, 5(1):172, 2004.
2 citations in pages 51 and 52.

[252] R. Lucchetti, F. Mignanego, and G. Pieri. Existence theorems of equilibrium points in stackelberg. *Optimization*, 18(6):857–866, jan 1987.
3 citations in pages 22, 26, and 29.

[253] Yibing Lv, Tiesong Hu, Guangmin Wang, and Zhongping Wan. A penalty function method based on kuhn–tucker condition for solving linear bilevel programming. *Applied Mathematics and Computation*, 188(1):808–813, may 2007.
2 citations in pages 28 and 29.

[254] P.C.H. Ma, K.C.C. Chan, Xin Yao, and D.K Chiu. An evolutionary clustering algorithm for gene expression microarray data analysis. *IEEE Transactions on Evolutionary Computation*, 10(3):296–314, jun 2006.
2 citations in pages 51 and 52.

[255] M.J. Magazine and Osman Oguz. A heuristic algorithm for the multidimensional zero-one knapsack problem. *European Journal of Operational Research*, 16(3):319–326, jun 1984.
2 citations in pages 97 and 103.

[256] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, jul 2012.
One citation in page 50.

[257] Sayuri Maldonado-Pinto, Martha-Selene Casas-Ramírez, and José-Fernando Camacho-Vallejo. Analyzing the performance of a hybrid heuristic for solving a bilevel location problem under different approaches to tackle the lower level. *Mathematical Problems in Engineering*, 2016:1–10, 2016.
2 citations in pages 13 and 17.

[258] O.L Mangasarian and S Fromovitz. The fritz john necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, 17(1):37–47, jan 1967.
One citation in page 21.

[259] P. Marcotte, G. Savard, and D.L. Zhu. A trust region algorithm for nonlinear bilevel programming. *Operations Research Letters*, 29(4):171–179, nov 2001.
2 citations in pages 33 and 38.

[260] Patrice Marcotte and Gérald Marquis. Efficient implementation of heuristics for the continuous network design problem. *Annals of Operations Research*, 34(1):163–176, dec 1992.
2 citations in pages 14 and 17.

[261] Patrice Marcotte and Gilles Savard. Bilevel programming. In *Encyclopedia of Optimization*, pages 155–156. Springer US.
3 citations in pages 24, 33, and 38.

[262] Patrice Marcotte and Gilles Savard. A note on the pareto optimality of solutions to the linear bilevel programming problem. *Computers & Operations Research*, 18(4):355–359, jan 1991.
2 citations in pages 43 and 46.

[263] Patrice Marcotte, Gilles Savard, and Frédéric Semet. A bilevel programming approach to the travelling salesman problem. *Operations Research Letters*, 32(3):240–248, may 2004.
2 citations in pages 12 and 17.

[264] Miroslav Maric, Zorica Stanimirovic, Nikola Milenkovic, and Aleksandar Djenic. Metaheuristic approaches to solving large-scale bilevel uncapacitated facility location problem with clients' preferences. *Yugoslav Journal of Operations Research*, 25(3):361–378, 2015.
2 citations in pages 13 and 17.

[265] Yannis Marinakis, Athanasios Migdalas, and Panos M. Pardalos. A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm. *Journal of Global Optimization*, 38(4):555–580, oct 2006.
One citation in page 18.

[266] Harry M. Markowitz and Alan S. Manne. On the solution of discrete programming problems. *Econometrica*, 25(1):84, jan 1957.
One citation in page 96.

[267] Yukiko Matsuoka, Hiromi Matsumae, Manami Katoh, Amie J Eisfeld, Gabriele Neumann, Takeshi Hase, Samik Ghosh, Jason E Shoemaker, Tiago JS Lopes, Tokiko Watanabe, Shinji Watanabe, Satoshi Fukuyama, Hiroaki Kitano, and Yoshihiro Kawaoka. A comprehensive map of the influenza a virus replication cycle. *BMC Systems Biology*, 7(1):97, 2013.
One citation in page 49.

[268] P M Mell and T Grance. The NIST definition of cloud computing. Technical report, 2011.
One citation in page 107.

[269] Ayalew Getachew Mersha and Stephan Dempe. Linear bilevel programming with upper level constraints depending on the lower level solution. *Applied Mathematics and Computation*, 180(1):247–254, sep 2006.
2 citations in pages 18 and 23.

[270] Peter Merz and Andreas Zell. Clustering gene expression profiles with memetic algorithms. In *Parallel Problem Solving from Nature — PPSN VII*, pages 811–820. Springer Berlin Heidelberg, 2002.
2 citations in pages 51 and 52.

[271] Mahmoud Mesbah, Majid Sarvi, and Graham Currie. Optimization of transit priority in the transportation network using a genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):908–919, sep 2011.
2 citations in pages 14 and 17.

[272] Athanasios Migdalas. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization*, 7(4):381–405, dec 1995.
3 citations in pages 11, 12, and 17.

[273] Guang min Wang, Xian jia Wang, Zhong ping Wan, and Shi hui Jia. An adaptive genetic algorithm for solving bilevel linear programming problem. *Applied Mathematics and Mechanics*, 28(12):1605–1612, dec 2007.
2 citations in pages 39 and 46.

[274] Alexander Mitsos, Panayiotis Lemonidis, and Paul I. Barton. Global solution of bilevel programs with a nonconvex inner program. *Journal of Global Optimization*, 42(4):475–513, dec 2007.
2 citations in pages 33 and 38.

[275] Alexander Mitsos, Panayiotis Lemonidis, Cha Kun Lee, and Paul I. Barton. Relaxation-based bounds for semi-infinite programs. *SIAM Journal on Optimization*, 19(1):77–113, jan 2008.
2 citations in pages 27 and 29.

[276] Satoshi Mizuno, Risa Iijima, Soichi Ogishima, Masataka Kikuchi, Yukiko Matsuoka, Samik Ghosh, Tadashi Miyamoto, Akinori Miyashita, Ryozo Kuwano, and Hiroshi Tanaka. AlzPathway: a comprehensive map of signaling pathways of alzheimer's disease. *BMC Systems Biology*, 6(1):52, 2012.
2 citations in pages 49 and 63.

[277] Gregory Moore, Charles Bergeron, and Kristin P. Bennett. Model selection for primal SVM. *Machine Learning*, 85(1-2):175–208, apr 2011.
2 citations in pages 15 and 17.

[278] James T. Moore and Jonathan F. Bard. The mixed integer linear bilevel programming problem. *Operations Research*, 38(5):911–921, oct 1990.
2 citations in pages 36 and 38.

[279] José Luis Morales and Jorge Nocedal. Remark on "algorithm 778: L-BFGS-b: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software*, 38(1):1–4, nov 2011.
One citation in page 80.

[280] C.A. Murthy and Nirmalya Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832, jul 1996.
2 citations in pages 51 and 52.

[281] Su Nguyen, Mengjie Zhang, and Mark Johnston. A genetic programming based hyper-heuristic approach for combinatorial optimisation. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*. ACM Press, 2011.
One citation in page 96.

[282] Anna Niarakis, Yacine Bounab, Luca Grieco, Romain Roncagalli, Anne-Marie Hesse, Jérôme Garin, Bernard Malissen, Marc Daëron, and Denis Thieffry. Computational modeling of the main signaling pathways involved in mast cell activation. In *Fc Receptors*, pages 69–93. Springer International Publishing, 2014.
One citation in page 49.

[283] Ichiro Nishizaki, Masatoshi Sakawa, Keiichi Niwa, and Yasuhiro Kitaguchi. A computational method using genetic algorithms for obtaining stackelberg solutions to two-level linear programming problems. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 85(6):55–62, feb 2002.
4 citations in pages 26, 29, 41, and 46.

[284] Alberto Noronha, Anna Dröfn Daníelsdóttir, Piotr Gawron, Freyr Jóhannsson, Soffía Jónsdóttir, Sindri Jarlsson, Jón Pétur Gunnarsson, Sigurur Brynjólfsson, Reinhard Schneider, Ines Thiele, and Ronan M. T. Fleming. ReconMap: an interactive visualization of human metabolism. *Bioinformatics*, page btw667, dec 2016.
One citation in page 50.

[285] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, Frank T Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villéger, Sarah E Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn, and Hiroaki Kitano. The systems biology graphical notation. *Nature Biotechnology*, 27(8):735–741, aug 2009.
One citation in page 65.

[286] Kanae Oda and Hiroaki Kitano. A comprehensive map of the toll-like receptor signaling network. *Molecular Systems Biology*, 2, apr 2006.
One citation in page 49.

[287] Kanae Oda, Yukiko Matsuoka, Akira Funahashi, and Hiroaki Kitano. A comprehensive pathway map of epidermal growth factor receptor signaling. *Molecular Systems Biology*, 1(1):E1–E17, may 2005.
One citation in page 49.

[288] V. Oduguwa and R. Roy. Bi-level optimisation using genetic algorithm. In *Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002)*. IEEE Comput. Soc, 2002.
3 citations in pages 42, 46, and 126.

[289] Mihai Oltean. Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation*, 13(3):387–410, sep 2005.
One citation in page 96.

[290] Jose Carlos Ortiz-Bayliss, Ender Ozcan, Andrew J. Parkes, and Hugo Terashima-Marin. Mapping the performance of heuristics for constraint satisfaction. In *IEEE Congress on Evolutionary Computation*. IEEE, jul 2010.
One citation in page 96.

[291] Marek Ostaszewski, Emmanuel Kieffer, Grégoire Danoy, Reinhard Schneider, and Pascal Bouvry. Clustering approaches for visual knowledge exploration in molecular interaction networks. *BMC Bioinformatics*, 19, aug 2018.

[292] J. V. Outrata. Necessary optimality conditions for stackelberg problems. *Journal of Optimization Theory and Applications*, 76(2):305–320, feb 1993.
One citation in page 21.

[293] Malay K. Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification. *Fuzzy Sets and Systems*, 155(2):191–214, oct 2005.
2 citations in pages 51 and 52.

[294] András Pál. fitsh- a software package for image processing. *Monthly Notices of the Royal Astronomical Society*, 421(3):1825–1837, mar 2012.
One citation in page 83.

[295] A. A. Panin, M. G. Pashchenko, and A. V. Plyasunov. Bilevel competitive facility location and pricing problems. *Automation and Remote Control*, 75(4):715–727, apr 2014.
3 citations in pages 12, 17, and 18.

[296] George Papavassilopoulos. Algorithms for static stackelberg games with linear costs and polyhedra constraints. In *1982 21st IEEE Conference on Decision and Control*. IEEE, dec 1982.
2 citations in pages 30 and 38.

[297] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, mar 2009.
2 citations in pages 53 and 54.

[298] Han-Saem Park, Si-Ho Yoo, and Sung-Bae Cho. Evolutionary fuzzy clustering algorithm with knowledge-based evaluation and applications for gene expression profiling. *Journal of Computational and Theoretical Nanoscience*, 2(4):524–533, dec 2005.
2 citations in pages 51 and 52.

[299] Ren Peng, Xu Rui-hua, and Qin Jin. Bi-level simulated annealing algorithm for facility location problem. In *2008 International Conference on Information Management, Innovation Management and Industrial Engineering*. IEEE, dec 2008.
2 citations in pages 13 and 17.

[300] Livia Perfetto, Leonardo Briganti, Alberto Calderone, Andrea Cerquone Perpetuini, Marta Iannuccelli, Francesca Langone, Luana Licata, Milica Marinkovic, Anna Mattioni, Theodora Pavlidou, Daniele Peluso, Lucia Lisa Petrilli, Stefano Pirrò, Daniela Posca, Elena Santonico, Alessandra Silvestri, Filomena Spada, Luisa Castagnoli, and Gianni Cesareni. SIGNOR: a database of causal relationships between biological entities. *Nucleic Acids Research*, 44(D1):D548–D554, oct 2015.
One citation in page 49.

[301] C. O. Pieume, L. P. Fotso, and P. Siarry. Solving bilevel programming problems with multicriteria optimization techniques. *OPSEARCH*, 46(2):169–183, jun 2009.
2 citations in pages 43 and 46.

[302] Hasan Pirkul. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 34(2):161–172, apr 1987.
2 citations in pages 97 and 103.

[303] Mitchell A. Potter and Kenneth A. Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature — PPSN III*, pages 249–257. Springer Berlin Heidelberg, 1994.
2 citations in pages 42 and 125.

[304] Dexter Pratt, Jing Chen, David Welker, Ricardo Rivas, Rudolf Pillich, Vladimir Rynkov, Keiichiro Ono, Carol Miello, Lyndon Hicks, Sandor Szalma, Aleksandar Stojmirovic, Radu Dobrin, Michael Braxenthaler, Jan Kuentzer, Barry Demchak, and Trey Ideker. NDEx, the network data exchange. *Cell Systems*, 1(4):302–305, oct 2015.
One citation in page 49.

[305] Nestor V Queipo, Javier V Goicochea, and Salvador Pintos. Surrogate modeling-based optimization of SAGD processes. *Journal of Petroleum Science and Engineering*, 35(1-2):83–93, jul 2002.
2 citations in pages 45 and 46.

[306] Arvind U Raghunathan and Lorenz T Biegler. Mathematical programs with equilibrium constraints (MPECs) in process engineering. *Computers & Chemical Engineering*, 27(10):1381–1392, oct 2003.
3 citations in pages 11, 16, and 17.

[307] T.K. Ralphs. Multistage Discrete Optimization. In S Dempe, V Kalashnikov, and B Mordukhovich, editors, *Bilevel Programming: Theory and Applications*. Bantham Science Publishers, 2016.
One citation in page 29.

[308] C E Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
One citation in page 81.

[309] W. D. Ray. User's guide to linear programming. *Journal of the Operational Research Society*, 22(1):93–93, mar 1971.
One citation in page 97.

[310] Aihong Ren and Yuping Wang. An approach for solving a fuzzy bilevel programming problem through nearest interval approximation approach and KKT optimality conditions. *Soft Computing*, 21(18):5515–5526, apr 2016.
2 citations in pages 42 and 46.

[311] Gang Ren, Zhengfeng Huang, Yang Cheng, Xing Zhao, and Yong Zhang. An integrated model for evacuation routing and traffic signal optimization with background demand uncertainty. *Journal of Advanced Transportation*, 47(1):4–27, oct 2012.
2 citations in pages 14 and 17.

[312] K.S. Nawaz Ripon, Chi-Ho Tsang, S. Kwong, and Man-Ki Ip. Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In *18th International Conference on Pattern Recognition (ICPR'06)*. IEEE, 2006.
2 citations in pages 51 and 52.

[313] Berc Rustem and Melendres Howe. *Algorithms for worst-case design and applications to risk management*. Princeton University Press, 2009.
2 citations in pages 27 and 29.

[314] Sauli Ruuska and Kaisa Miettinen. Constructing evolutionary algorithms for bilevel multiobjective optimization. In *2012 IEEE Congress on Evolutionary Computation*. IEEE, jun 2012.
One citation in page 44.

[315] Sauli Ruuska, Kaisa Miettinen, and Margaret M. Wiecek. Connections between single-level and bilevel multiobjective optimization. *Journal of Optimization Theory and Applications*, 153(1):60–74, oct 2011.
2 citations in pages 43 and 46.

[316] G. K. Saharidis and M. G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44(1):29–51, mar 2008.
2 citations in pages 37 and 38.

[317] Kemal H. Sahin and Amy R. Ciric. A dual temperature simulated annealing approach for solving bilevel programming problems. *Computers & Chemical Engineering*, 23(1):11–25, dec 1998.
2 citations in pages 41 and 46.

[318] Javier Salmeron, Kevin Wood, and Ross Baldick. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on Power Systems*, 24(1):96–104, feb 2009.
One citation in page 18.

[319] Venkata Satagopam, Wei Gu, Serge Eifes, Piotr Gawron, Marek Ostaszewski, Stephan Gebel, Adriano Barbosa-Silva, Rudi Balling, and Reinhard Schneider. Integration and visualization of translational medicine data for better understanding of human diseases. *Big Data*, 4(2):97–108, jun 2016.
One citation in page 49.

[320] Gilles Savard and Jacques Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15(5):265–272, jun 1994.
2 citations in pages 32 and 38.

[321] Maria P. Scaparra and Richard L. Church. A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, 35(6):1905–1923, jun 2008.
2 citations in pages 13 and 17.

[322] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, aug 2007.
One citation in page 52.

[323] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.
One citation in page 56.

[324] Shizuo Senju and Yoshiaki Toyoda. An approach to linear programming with 0–1 variables. *Management Science*, 15(4):B–196–B–207, dec 1968.
One citation in page 97.

[325] Franciszek Seredynski. Loosely coupled distributed genetic algorithms. In *Parallel Problem Solving from Nature — PPSN III*, pages 514–523. Springer Berlin Heidelberg, 1994.
One citation in page 126.

[326] Franciszek Seredynski. Competitive coevolutionary multi-agent systems: The application to mapping and scheduling problems. *Journal of Parallel and Distributed Computing*, 47(1):39–57, nov 1997.
One citation in page 126.

[327] P.S Shelokar, V.K Jayaraman, and B.D Kulkarni. An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195, may 2004.
One citation in page 51.

[328] Weiguo Sheng and Xiaohui Liu. A hybrid algorithm for k-medoid clustering of large data sets. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. IEEE.
2 citations in pages 51 and 52.

[329] Simon Shepherd and Agachai Sumalee. A genetic algorithm based approach to optimal toll level and location problems. *Networks and Spatial Economics*, 4(2):161–179, jun 2004.
2 citations in pages 39 and 46.

[330] Chenggen Shi, Jie Lu, and Guangquan Zhang. An extended kuhn–tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, 162(1):51–63, mar 2005.
3 citations in pages 23, 32, and 38.

[331] Chenggen Shi, Guangquan Zhang, and Jie Lu. On the definition of linear bilevel programming solution. *Applied Mathematics and Computation*, 160(1):169–176, jan 2005.
One citation in page 23.

[332] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. An improved bilevel evolutionary algorithm based on quadratic approximations. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, jul 2014.
8 citations in pages xi, 18, 45, 46, 80, 83, 85, and 88.

[333] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Test problem construction for single-objective bilevel optimization. *Evolutionary Computation*, 22(3):439–477, sep 2014.
One citation in page 18.

[334] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Transportation policy formulation as a multi-objective bilevel optimization problem. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, may 2015.
2 citations in pages 12 and 17.

[335] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Solving optimistic bilevel programs by iteratively approximating lower level optimal value function. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, jul 2016.
2 citations in pages 45 and 46.

[336] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 257(2):395–411, mar 2017.
2 citations in pages 45 and 46.

[337] Ankur Sinha, Pekka Malo, Anton Frantsev, and Kalyanmoy Deb. Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, jun 2013.
2 citations in pages 16 and 17.

[338] Ankur Sinha, Pekka Malo, Peng Xu, and Kalyanmoy Deb. A bilevel optimization approach to automated parameter tuning. In *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14*. ACM Press, 2014.
3 citations in pages 11, 15, and 17.

[339] Ankur Sinha, Tharo Soun, and Kalyanmoy Deb. Evolutionary bilevel optimization using KKT proximity measure. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, jun 2017.
2 citations in pages 41 and 46.

[340] Alejandro Sosa-Ascencio, Gabriela Ochoa, Hugo Terashima-Marin, and Santiago Enrique Conant-Pablos. Grammar-based generation of variable-selection heuristics for constraint satisfaction problems. *Genetic Programming and Evolvable Machines*, 17(2):119–144, 2016.
One citation in page 96.

[341] Alejandro Sosa-Ascencio, Hugo Terashima-Marin, and Manuel Valenzuela-Rendon. Grammar-based genetic programming for evolving variable ordering heuristics. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013.
One citation in page 96.

[342] Sweta Srivastava and Sudip Kumar Sahana. Nested hybrid evolutionary model for traffic signal optimization. *Applied Intelligence*, 46(1):113–123, jul 2016.
2 citations in pages 40 and 46.

[343] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. Springer Berlin Heidelberg, 2009.
One citation in page 50.

[344] Huijun Sun, Ziyou Gao, and Jianjun Wu. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied Mathematical Modelling*, 32(4):610–616, apr 2008.
2 citations in pages 13 and 17.

[345] Shyam Sundar, Alok Singh, and André Rossi. An artificial bee colony algorithm for the 0–1 multidimensional knapsack problem. In *Communications in Computer and Information Science*, pages 141–151. Springer Berlin Heidelberg, 2010.
One citation in page 97.

[346] Jung Mo Sung. Idolatry: A reading key for the market economy? *Dialog*, 55(1):25–30, mar 2016.
3 citations in pages 5, 26, and 29.

[347] Varun Suryan, Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Handling inverse optimal control problems using evolutionary bilevel optimization. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, jul 2016.
One citation in page 11.

[348] Damian Szklarczyk, Andrea Franceschini, Stefan Wyder, Kristoffer Forslund, Davide Heller, Jaime Huerta-Cepas, Milan Simonovic, Alexander Roth, Alberto Santos, Kalliopi P. Tsafou, Michael Kuhn, Peer Bork, Lars J. Jensen, and Christian von Mering. STRING v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic Acids Research*, 43(D1):D447–D452, oct 2014.
One citation in page 49.

[349] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Inc., jun 2009.
One citation in page 40.

[350] El-Ghazali Talbi. A taxonomy of metaheuristics for bi-level optimization. In *Metaheuristics for Bi-level Optimization*, pages 1–39. Springer Berlin Heidelberg, 2013.
2 citations in pages 38 and 39.

[351] Yoshiaki Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, 21(12):1417–1427, aug 1975.
One citation in page 97.

[352] Angelos Tsoukalas, Panos Parpas, and Berç Rustem. A smoothing algorithm for finite min–max–min problems. *Optimization Letters*, 3(1):49–62, jun 2008.
2 citations in pages 27 and 29.

[353] Angelos Tsoukalas, Berç Rustem, and Efstratios N. Pistikopoulos. A global optimization algorithm for generalized semi-infinite, continuous minimax with coupled constraints and bi-level problems. *Journal of Global Optimization*, 44(2):235–250, jul 2008.
4 citations in pages 27, 29, 33, and 38.

[354] Ann van Ackere. The principal/agent paradigm: Its relevance to various functional fields. *European Journal of Operational Research*, 70(1):83–103, oct 1993.
One citation in page 17.

[355] D.W. van der Merwe and A.P. Engelbrecht. Data clustering using particle swarm optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.* IEEE.
One citation in page 51.

[356] Rinde R.S. van Lon, Tom Holvoet, Greet Vanden Berghe, Tom Wenseleers, and Juergen Branke. Evolutionary synthesis of multi-agent systems for dynamic dial-a-ride problems. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*. ACM Press, 2012.
One citation in page 96.

[357] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos. Management of an academic hpc cluster: The ul experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, pages 959–967, Bologna, Italy, July 2014.
5 citations in pages 69, 88, 101, 117, and 132.

[358] F. Guerra Vázquez, J.-J. Rückmann, O. Stein, and G. Still. Generalized semi-infinite programming: A tutorial. *Journal of Computational and Applied Mathematics*, 217(2):394–419, aug 2008.
2 citations in pages 27 and 29.

[359] José Luis González Velarde, José Fernando Camacho Vallejo, and Gabriel Pinto Serrano. A scatter search algorithm for solving a bilevel optimization model for determining highway tolls. *Computación y Sistemas*, 19(1), mar 2015.
3 citations in pages 11, 12, and 17.

[360] Pascale Vicat-Blanc, Sébastien Soudan, Romaric Guillier, and Brice Goglin. *Computing Networks*. John Wiley & Sons, Inc, jan 2013.
One citation in page 97.

[361] L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2):379–399, may 1994.
3 citations in pages 24, 32, and 38.

[362] Luis N. Vicente and Paul H. Calamai. Geometry and local optimality conditions for bilevel programs with quadratic strictly convex lower levels. In *Nonconvex Optimization and Its Applications*, pages 141–151. Springer US, 1995.
2 citations in pages 27 and 29.

[363] A. Volgenant and J. A. Zoon. An improved heuristic for multidimensional 0-1 knapsack problems. *Journal of the Operational Research Society*, 41(10):963–970, oct 1990.
2 citations in pages 97 and 103.

[364] G. Gary Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. In *Volume 1: 32nd Design Automation Conference, Parts A and B*. ASME, 2006.
2 citations in pages 44 and 46.

[365] Guangmin Wang, Zhongping Wan, Xianjia Wang, and Yibing Lv. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications*, 56(10):2550–2555, nov 2008.
One citation in page 39.

[366] Judith Y.T. Wang, Matthias Ehrgott, Kim N. Dirks, and Abhishek Gupta. A bilevel multi-objective road pricing model for economic, environmental and health sustainability. *Transportation Research Procedia*, 3:393–402, 2014.
3 citations in pages 12, 17, and 18.

[367] Y. Wang, Y.-C. Jiao, and H. Li. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 35(2):221–232, may 2005.
3 citations in pages 40, 41, and 46.

[368] Yong WANG, Zi-Xing CAI, Yu-Ren ZHOU, and Chi-Xin XIAO. Constrained optimization evolutionary algorithms. *Journal of Software*, 20(1):11–29, apr 2009.
One citation in page 84.

[369] Yuping Wang, Hong Li, and Chuangyin Dang. A new evolutionary algorithm for a class of nonlinear bilevel programming problems and its global convergence. *INFORMS Journal on Computing*, 23(4):618–629, nov 2011.
One citation in page 41.

[370] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, mar 1963.
One citation in page 68.

[371] Ue-Pyng Wen and Shuh-Tzy Hsu. A note on a linear bilevel programming algorithm based on bicriteria programming. *Computers & Operations Research*, 16(1):79–83, jan 1989.
2 citations in pages 43 and 46.

[372] U.P. Wen and Y.H. Yang. Algorithms for solving the mixed integer two-level linear programming problem. *Computers & Operations Research*, 17(2):133–142, jan 1990.
2 citations in pages 36 and 38.

[373] D. J. White and G. Anandalingam. A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, 3(4):397–419, 1993.
2 citations in pages 33 and 38.

[374] Gerald Whittaker, Rolf Färe, Shawna Grosskopf, Bradley Barnhart, Moriah Bostian, George Mueller-Warrant, and Stephen Griffith. Spatial targeting of agri-environmental policy using bilevel evolutionary optimization. *Omega*, 66:15–27, jan 2017.
3 citations in pages 11, 16, and 17.

[375] Wolfram Wiesemann, Angelos Tsoukalas, Polyxeni-Margarita Kleniati, and Berç Rustem. Pessimistic bilevel optimization. *SIAM Journal on Optimization*, 23(1):353–380, jan 2013.
One citation in page 22.

[376] Frank Wilcoxon. Individual comparisons by ranking methods. In *Springer Series in Statistics*, pages 196–202. Springer New York, 1992.
One citation in page 88.

[377] Christian Wiwie, Jan Baumbach, and Richard Röttger. Comparing the performance of biomedical clustering methods. *Nature Methods*, 12(11):1033–1038, sep 2015.
One citation in page 50.

[378] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, apr 1997.
One citation in page 94.

[379] R. Kevin Wood. *Bilevel Network Interdiction Models: Formulations and Solutions*. American Cancer Society, 2011.
2 citations in pages 13 and 17.

[380] Huang XIE, Ping wei ZHANG, and Sheng LUO. Spectral clustering based on global k-means. *Journal of Computer Applications*, 30(7):1936–1937, aug 2010.
One citation in page 50.

[381] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, jun 2015.
One citation in page 50.

[382] Jiuping Xu, Yan Tu, and Ziqiang Zeng. A nonlinear multiobjective bilevel model for minimum cost network flow problem in a large-scale construction project. *Mathematical Problems in Engineering*, 2012:1–40, 2012.
2 citations in pages 14 and 17.

[383] Rui Xu and Donald WunschII. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, may 2005.
One citation in page 51.

[384] Xiaowei Xu, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings 14th International Conference on Data Engineering*. IEEE Comput. Soc.
One citation in page 50.

[385] Tadashi Yamada, Bona Frazila Russ, Jun Castro, and Eiichi Taniguchi. Designing multimodal freight transport networks: A heuristic approach and applications. *Transportation Science*, 43(2):129–143, may 2009.
2 citations in pages 14 and 17.

[386] Yasuhiro Yamamoto, Hiromi Koma, and Tatsurou Yagami. Hydrogen peroxide mediated the neurotoxicity of an antibody against plasmalemmal neuronspecific enolase in primary cortical neurons. *NeuroToxicology*, 49:86–93, jul 2015.
One citation in page 76.

[387] Bo YANG, Da-You LIU, Jiming LIU, Di JIN, and Hai-Bin MA. Complex network clustering algorithms. *Journal of Software*, 20(1):54–66, apr 2009.
One citation in page 50.

[388] Hai Yang and Michael G. H. Bell. Models and algorithms for road network design: a review and some new developments. *Transport Reviews*, 18(3):257–278, jul 1998.
2 citations in pages 14 and 17.

[389] Yafeng Yin. Genetic-algorithms-based approach for bilevel programming models. *Journal of Transportation Engineering*, 126(2):115–120, mar 2000.
2 citations in pages 14 and 17.

[390] Yafeng Yin. Multiobjective bilevel optimization for transportation planning and management problems. *Journal of Advanced Transportation*, 36(1):93–105, sep 2002.
2 citations in pages 12 and 17.

[391] G. Yu, F. Li, Y. Qin, X. Bo, Y. Wu, and S. Wang. GOSemSim: an r package for measuring semantic similarity among GO terms and gene products. *Bioinformatics*, 26(7):976–978, feb 2010.
One citation in page 66.

[392] Guangchuang Yu, Li-Gen Wang, Yanyan Han, and Qing-Yu He. clusterProfiler: an r package for comparing biological themes among gene clusters. *OMICS: A Journal of Integrative Biology*, 16(5):284–287, may 2012.
One citation in page 67.

[393] Guangchuang Yu, Li-Gen Wang, Guang-Rong Yan, and Qing-Yu He. DOSE: an r/bioconductor package for disease ontology semantic and enrichment analysis. *Bioinformatics*, 31(4):608–609, oct 2014.
One citation in page 67.

[394] Bo Yuan, G.J. Klir, and J.F. Swan-Stone. Evolutionary fuzzy c-means clustering algorithm. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems. The International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium*. IEEE.
2 citations in pages 51 and 52.

[395] Peiyong Zhai, Shumin Gao, Eric Holle, Xianzhong Yu, Atsuko Yatani, Thomas Wagner, and Junichi Sadoshima. Glycogen Synthase Kinase-3α Reduces Cardiac Growth and Pressure Overload-induced Cardiac Hypertrophy by Inhibition of Extracellular Signal-regulated Kinases. *Journal of Biological Chemistry*, 282(45):33181–33191, nov 2007.
One citation in page 76.

[396] Guangquan Zhang, Jie Lu, and Tharam Dillon. Decentralized multi-objective bilevel decision making with fuzzy demands. *Knowledge-Based Systems*, 20(5):495–507, jun 2007.
One citation in page 17.

[397] Jiawei Zhang and Lining Xing. A survey of multiobjective evolutionary algorithms. In *22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE, jul 2017.
One citation in page 44.

[398] Yuqing Zhang, William H.K. Lam, and Agachai Sumalee. Transit schedule design in dynamic transit network with demand and supply uncertainties. *Proceedings of the Eastern Asia Society for Transportation Studies*, 2009:250–250, 2009.
2 citations in pages 39 and 46.

[399] Bao-Jiang Zhao. An ant colony clustering algorithm. In *2007 International Conference on Machine Learning and Cybernetics*. IEEE, 2007.
One citation in page 51.

[400] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, mar 2011.
One citation in page 44.

[401] Wenjie Zuo. Bi-level optimization for the cross-sectional shape of a thin-walled car body frame with static stiffness and dynamic frequency stiffness constraints. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 229(8):1046–1059, oct 2014.
2 citations in pages 14 and 17.

[402] Zhi ZUO, Ryo KANAMORI, Tomio MIWA, and Takayuki MORIKAWA. A study of both optimal locations and toll levels road pricing using genetic algorithm. *Journal of the Eastern Asia Society for Transportation Studies*, 8:145–156, 2010.
One citation in page 18.

[403] Özalp Özer and Gal Raz. Supply chain sourcing under asymmetric information. *Production and Operations Management*, 20(1):92–115, jan 2011.
One citation in page 17.

[404] Ender Özcan and Can Başaran. A case study of memetic algorithms for constraint optimization. *Soft Computing*, 13(8-9):871–882, jul 2008.
One citation in page 103.