Combinatorial optimization methods for the (α, β)-k Feature Set Problem

Amir Salehipour

Faculty of Engineering and Built Environment
School of Electrical Engineering and Computing
University of Newcastle, NSW, Australia

Combinatorial optimization methods for the (α, β)-k Feature Set Problem

Amir Salehipour

March 2019

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

Statement of Originality

The thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I hereby certify that the work embodied in the thesis is my own work, conducted under normal supervision. I give consent to the final version of my thesis being made available worldwide when deposited in the University's Digital Repository, subject to the provisions of the Copyright Act 1968 and any approved embargo.

Amir Salehipour 28 March 2019

Dedication

To my parents, my wife and my son, and my sisters.

Acknowledgment

My first acknowledgment must go to my lovely wife, Dr Leila Moslemi Naeni. I want to thank her for all the support that she gave me during these long four years of my research.
Secondly, I would like to thank my supervisors, Prof. Pablo Moscato and Prof. Regina Berretta for providing me with this opportunity to pursue my PhD studies at the University of Newcastle, Australia. I would like to thank all the past and present members of the CIBM team (Center for Bioinformatics, Biomarker Discovery and Information-Based Medicine, the University of Newcastle) for their friendly support, chats, and helpful ideas. Starting with ex-member Dr Carlos Riveros, an amazing supportive person, and a true friend, who helped me many times with programming, and modeling. Then, Dr Renato Vimieiro, Dr Mateus Rocha de Paula and Dr Ahmed Shamsul Arefin for their friendship. Finally, thanks to all my colleagues and friends that supported me thorough this journey: Ademir, Amer, Claudio, Francia, Heloisa, Inna, Lukas, Luke, Marta, Mohammad, Nader, Nisha and Shannon.
I am very thankful to the University of Newcastle for my scholarships and for the opportunity to be part of an incredible academic environment. My Doctorate was fully funded by the University of Newcastle's Postgraduate Research Scholarships; those financial supports are very much appreciated.
I would like to thank my mother and my father, for the love and support that they have given me from the other side of the world. You gave me the strength and motivation to continue and finish my PhD.

Amir Salehipour
March 2019

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

This PhD research thesis proposes novel and efficient combinatorial optimization-based solution methods for the (α, β)-k Feature Set Problem. The (α, β)-k Feature Set Problem is a combinatorial optimization-based feature selection approach proposed in 2004, and has several applications in computational biology and Bioinformatics. The (α, β)-k Feature Set Problem aims to select a minimum cost set of features such that similarities between entities of the same class and differences between entities of different classes are maximized.

The developed solution methods of this research include heuristic and exact methods. While this research focuses on utilizing exact methods, we also developed mathematical properties, and heuristics and problem-driven local searches and applied them in certain stages of the exact methods in order to guide exact solvers and deliver high quality solutions. The motivation behind this stems from computational difficulty of exact solvers in providing good quality solutions for the (α, β)-k Feature Set Problem. Our proposed heuristics deliver very good quality solutions including optimal, and that in a reasonable amount of time.

The major contributions of the presented research include: 1) investigating and exploring mathematical properties and characteristics of the (α, β)-k Feature Set Problem for the first time, and utilizing those in order to design and develop algorithms and methods for solving large instances of the (α, β)-k Feature Set Problem; 2) extending the basic modeling, algorithms and solution methods to the weighted variant of the (α, β)-k Feature Set Problem (where features have a cost); and, 3) developing algorithms and solution methods that are capable of solving large instances of the (α, β)-k Feature Set Problem in a reasonable amount of time (prior to this research, many of those instances pose a computational challenge for the exact solvers).

To this end, we showed the usefulness of the developed algorithms and methods by applying them on three sets of 346 instances, including real-world, weighted, and randomly generated instances, and obtaining high quality solutions in a short time. To the best of our knowledge, the developed algorithms of this research have obtained the best results for the (α, β)-k Feature Set Problem. In particular, they outperform state-of-the-art algorithms and exact solvers, and have a very competitive performance over large instances because they always deliver feasible solutions, and obtain new best solutions for a majority of large instances in a reasonable amount of time.

Awards, Publications, and Outcomes

Part of the material presented in this research thesis has been already presented, and published in peer-reviewed conferences. The list of publications and presentations is provided below. It is worth mentioning that during my PhD studies I won one major award of the University of Newcastle for conducting an outstanding original research on the (α, β)-k Feature Set Problem.

Awards

1. Winner of the "2015 Research Poster Prize Competition", awarded by Faculty of Engineering and Built Environment, University of Newcastle, 2015.
2. Two PGRSS travel grants (approximately, $4,500), awarded by the Faculty of Engineering and Built Environment, University of Newcastle, 2016.

Conference proceedings

1. "Tight lower bounds and a hybrid heuristic for a problem of selecting features", orally presented at EURO 2016 International Conference (peer-reviewed). Poznan, 3 – 6 July 2016.
2. "An optimization approach towards selecting features in biological datasets", orally presented at 24th National Conference of the Australian Society for Operations Research (peer-reviewed). Canberra, 16 – 18 November 2016.

Working papers

1. "Efficient solution methods for the Min k (α, β)-k Feature Set Problem" (75% complete; will be submitted to an international journal in next few months).

2. "A heuristic algorithm for the (α, β)-k Feature Set Problem" (80% complete; will be submitted to a top tier journal in the next few weeks).

Chapter 1

Introduction

Abstract

This chapter brings a short introduction into the presented research thesis, as well as a brief discussion regarding the role of optimization and operations research into computational biology and Bioinformatics. The research question answered by this research thesis, and four research goals, which we achieved and accomplished in this thesis, are also discussed. Finally, the chapter explains the thesis structure, and reviews the contents of every chapter of the thesis.

1.1 Introduction

Bioinformatics has been defined in many different ways. Although, usually it refers to any use of computers to analyze and characterize the molecular components of living organisms, generally, it is the use of computers for processing biologically-derived data and information. This definition is according to the biological science view towards Bioinformatics (Waterman, 1995). In fact, Bioinformatics is associated with information science and information technology of biology. Note that both definitions share a common view: *information contained within the biological data*. Furthermore, both definitions imply that large amounts of data should be managed and analyzed.

According to Luscombe et al. (2001), "Bioinformatics is conceptualizing biology in terms of macromolecules (in the sense of physical-chemistry) and then applying informatics techniques (derived from disciplines such as applied maths, computer science, and statistics) to understand and organize the information associated with these molecules, on a large-scale".

We can state that Bioinformatics is an interdisciplinary field with the goal of developing methods and tools for storing, organizing, exploring, and analyzing large biological data as well as discussing and interpreting the outcomes. For this purpose, Bioinformatics benefits from many areas of computer science, mathematics and engineering. Here, efficient methods and algorithms that can store, process and analyze biological data are a must.

During this Ph.D project, outcomes of which have been presented in this research thesis, I designed, developed, and implemented models and solution techniques for an important and applicable optimization problem arising in computational biology and Bioinformatics, that is, the (α, β)-k Feature Set Problem. Later in this chapter, I shall state the research question and research goals of this research. I will conclude this chapter by explaining the structure of the thesis, and pointing out the subject and contents of every chapter.

1.2 Optimization in Bioinformatics

Optimization has found its way in healthcare and Bioinformatics. It has helped healthcare professionals to improve their decisions and processes; in fact, to better utilize the scarce resources available. A few examples of operations research applications in healthcare include locating healthcare facilities (Farahani et al., 2012), distributing blood products among hospitals (Salehipour and Sepehri, 2012), and locating ambulances to minimize the delay and maximize the coverage (Brotcorne et al., 2003). For more applications of operations research techniques in healthcare we refer the interested reader to Batun and Begen (2013); Brandeau et al. (2005); Brandeau et al. (2004). Rais and Viana (2011) provided current research trends of optimization and operations research in healthcare.

The role of optimization in biology and Bioinformatics is well reflected by certain aspects of problems, which may be stated in the form of making the best decision, out of huge number of acceptable decisions, with respect to available resources, as well as other practical considerations and limitations. Practical considerations or problem requirements (known as constraints in modeling) heavily impact decision making. Although, a decision is preferred to be verifiable in terms of quality, and that in a reasonable amount of time, even obtaining such a decision might computationally be very expensive, in particular, in the case of complex problems including large instances and datasets (many optimization problems in the domain of computational biology and Bioinformatics fall in this category). For this reason, an alternative approach is to obtain very good quality decisions instead of obtaining the best decision (whenever obtaining the best decision is very expensive). The methods that look for high quality decisions are known as heuristic or approximation methods, and the associated decision is referred to *local optimum decision* (versus exact methods, and *global optimum decision* or the best decision). Although, heuristic methods may even obtain the global optimum decision, in general and in most cases there is still no performance guarantee[1].

Many optimization problems in the areas of computational biology and Bioinformatics are very difficult to solve to optimality (i.e. to obtain the global optimal solution). Despite

[1] Notice that several well-known optimization problems are solved to optimality by heuristic methods, and that in polynomial time. These methods also provide proof of optimality. Several examples include heuristics developed for the Shortest Path and Minimum Spanning Tree problems (Dasgupta et al., 2008; Cormen et al., 2009; Aini and Salehipour, 2012).

this, many authors analyzed and developed statistical tools and combinatorial optimization problems and methods, as well as efficient algorithms, which can obtain high quality solutions in a reasonable amount of time. For example, Ravetti et al. (2010) analyzed a microarray[2] dataset of 31 samples associated with the Alzheimer's disease (AD), including 22 AD samples, and 9 healthy samples. They utilized certain statistics as well as mathematical programming models, and uncovered biomarkers including 1372 probe gene expression signatures. These agree with the already established markers of progression in AD. Wang et al. (2008) proposed an evolutionary method for the problem of probe selection. The problem includes finding a minimal non-unique probe set, which can be used as identifiers. Brinza and Zelikovsky (2006) studied the problem of searching for the most disease-associated and the most disease-resistant multi-gene interactions for a given sample of diseased and healthy individuals. More precisely, they studied disease susceptibility prediction problem. For this, they developed several search methods to address the problem of Multi-SNP (Multi Single Nucleotide Polymorphisms).

Sequence alignment is to arrange the sequences of DNA, RNA, or protein to identify regions of similarity (Mount, 2004). The sequence alignment methods include both exact (e.g. Dynamic Programming) and heuristics. Due to the size of sequence local alignments may be preferred to global alignments. Global alignments globally optimize the sequence alignment (thus, through entire sequence), while local alignments focus on parts of the sequence. Another example is protein threading or fold recognition, which includes developing models for proteins. Wagner et al. (2004) presented large-scale optimization techniques for the problem of proteins folding. The objective of their model is correct prediction of the structure of known proteins. Related to the problem of proteins folding, Xu et al. (2000) developed a network flow formulation for the problem of protein domain decomposition. Structural domains are the basic and semi-independent units of protein folding. For a classification of applications of mathematical optimization in computational biology and Bioinformatics we refer the interested reader to Ramsden (2009); Banga (2008); Polanski and Kimmel (2007). Lancia (2008) provided a survey on mathematical programming methods in computational biology and Bioinformatics.

1.3 Feature selection in Bioinformatics

In this section, we briefly explain an interesting problem in the area of feature selection, which has several applications in Bioinformatics. Chapter 2 will extensively discuss this.

Generally speaking, feature selection is to choose a subset of features, out of a set of candidate features, such that the selected set best represents the whole in a particular aspect. As discussed by Paula (2012), removing irrelevant or redundant features, and reducing the dimensionality of the dataset are two reasons to perform feature selection. These criteria are both interesting and important because given the size of the datasets and the amount of data we

[2] Microarray is a two dimensional array on a chip that can keep huge amount of biological data. One type of such data is gene expression. Gene expression is the process of using gene's information to synthesis gene products, and typically includes amount and timing of appearance of a functional product of a gene.

encounter in many practical applications, they ease the analysis, utilization, and interpretation of high-dimensional datasets. For example, Inostroza-Ponta et al. (2008); Inostroza-Ponta et al. (2011) modeled a visualization problem as a Quadratic Assignment Problem (QAP).

Feature selection is an important technique in refining data. In practice, different criteria and measures might be of interest for selecting such a subset, for example a *less expensive* set, *stronger classifier*, and *new/independent* features in the set, among others. These criteria are additional motivations for selecting only a subset of features rather than selecting the whole set. For example, in Bioinformatics studying the whole set of probes or genes in a microarray dataset is highly resource demanding. On top of this, more often obtaining a set of probes to act as a biomarker is of one of the primary goals of the analysis. For example, it is important to find out which genes, probes, or SNPs are useful in distinguishing a certain group of people or diagnosis of a given disease.

Feature selection has a broad applications in computational biology and Bioinformatics. Recently, Wang et al. (2016a) provided a thorough survey reviewing some of the feature selection applications in Bioinformatics where focus is on combinatorial optimization methods and big data analysis. This is one of the first studies that categorizes feature selection methods into exhaustive search, heuristic search, and hybrid search methods (instead of traditional filter, wrapper, and embedded approaches). Wang (2012) discussed several examples of feature selection applications in Bioinformatics. As one example, the author introduced new statistical methods for feature selection, and showed that how a set of only two genes can successfully classify Lymphoma dataset, which were previously classified by a set of 48 genes (Alizadeh et al., 2000), while the accuracy is kept at the same level of 100%.

Another interesting example of feature selection in Bioinformatics includes distinguishing between two classes of healthy and disease samples. Given a set of disease and healthy samples and a set of genes or SNPs with their values of expression level for every sample, we are interested in selecting the minimum number of genes (a subset of all genes) such that a classification between healthy and disease samples with a good accuracy when predicting classes can be concluded. In other words, the selected subset of genes will act as a biomarker. For example, Ravetti et al. (2009); Ravetti and Moscato (2008) employed the combinatorial optimization approach of (α, β)-k Feature Set Problem (see Chapter 2 for more details) to select features, and proposed novel biomarkers for the prediction of Alzheimer's disease and Prostate Cancer. The studies showed that these biomarkers are superior to others found in the literature from the classification point of view, since they lead to a better accuracy when predicting classes using classifiers.

Notice that the feature selection is far more than obtaining a smaller feature set. Its primary purpose is to obtain a classifier with a better accuracy when predicting classes by using the classifier. This is particularly important for many biological datasets, as the associated analyses are often very resource-demanding. Given that the biological datasets include hundreds of samples and thousands of features (e.g. genes, probes, and SNPs), exploring and selecting a subset of features with the desired behavior and characteristic is very important. One of the goals of this research is to develop efficient optimization-based algorithms and methods to select this subset of features. It should be noted that other methods such as classification and clustering may have the ability to select the right features as a part of their learning (Guyon and Elisseeff, 2003; Chormunge and Jena, 2018; Huang et al., 2018; Şeref et al., 2018). To do so, we study the problem of (α, β)-k Feature Set, which was proposed by Cotta et al. (2004); Berretta et al. (2005), and has several applications in the areas of computational biology and Bioinformatics, and we develop mathematical properties and efficient optimization algorithms and methods in order to obtain a subset of features with the desired characteristics.

1.4 Research question and research goals

The major research question, which we investigate in this thesis, and provide answer for it includes:

- Research question. Can we develop efficient combinatorial optimization-based algorithms and methods for the (α, β)-k Feature Set Problem (FSP), in order to select a subset of features, out of a larger set, and that in a reasonable amount of time?

As we will discuss in Chapter 2, the research question attempts to overcome limitations of the available combinatorial optimization-based algorithms and methods for the (α, β)-k FSP.

The major goal of this research is to "design, develop and implement modeling techniques, and efficient and advanced optimization-based algorithms and methods for the (α, β)-k FSP". This goal is inspired by the computational complexity of obtaining optimal solutions for large instances of the (α, β)-k FSP in a reasonable amount of time (we will discuss those algorithms and solution methods in details in Chapter 4 and Chapter 5), and lack of existence of algorithms and solution methods for the (α, β)-k FSP, in particular, for large instances (see Chapter 2). The major goal of this research thesis can be further narrowed down into the following two goals.

- Research goal 1. Investigating and exploring mathematical properties and characteristics of the (α, β)-k FSP, and utilizing those in designing and developing algorithms and methods to solve the problem. This goal is accomplished in Chapter 3. Furthermore, this research extends the developed modeling, algorithms and solution methods to the *weighted* variant of the (α, β)-k FSP. In the weighted variant, features are given costs associated with their importance, weight, preference, etc. One application of this is when certain features are always preferred to be selected. This goal is fulfilled by incorporating the features' costs in all modeling, algorithms, solution methods and computational experiments.

- Research goal 2. Contributing to solving the (α, β)-k FSP by developing efficient algorithms and methods, in particular, for instances that exact solvers including the well-known solver CPLEX cannot obtain good quality solutions in a reasonable amount of time. While this research thesis focuses on utilizing exact solvers, we developed certain heuristics and problem-driven local searches to facilitate and speed-up the exact solvers. This goal is achieved and accomplished in Chapters 4 and 5. In addition to this, we showed the usefulness of the developed algorithms and methods by applying them to real-world biological datasets ranging from medium to large. To the best of our knowledge, and at the time of writing this research thesis, the algorithms and methods of this research can efficiently solve large instances of the (α, β)-k FSP in a reasonable amount of time. Such an achievement is not available prior to this research.

1.5 Thesis structure

After discussing the research question and research goals in this chapter, we define the research problem of this thesis in Chapter 2 (i.e. the (α, β)-k Feature Set Problem (FSP)), and discuss the research motivation. The chapter establishes the notations and mathematical foundations for the (α, β)-k FSP. The remaining of Chapter 2 reviews the most important methods and techniques of feature selection, as well as relevant works studying the applications of feature selection in computational biology and Bioinformatics.

Chapter 3 investigates the mathematical models and properties of the (α, β)-k FSP. In this research thesis, we solve the (α, β)-k FSP by following a four-stage approach. Indeed, this approach decomposes the (α, β)-k FSP into a set of related optimization problems, which we call *sub-problems*, and sequentially solves each sub-problem. Those sub-problems are the Min k (α, β)-k Feature Set Problem, the Max β (α, β)-k Feature Set Problem, and the Max Cover (α, β)-k Feature Set Problem. The chapter reviews mathematical programming formulations for the sub-problems. Notations, and a graph representation are thoroughly discussed in this chapter. After establishing those foundations, we develop several important mathematical properties and bounds for the sub-problems. The bounds and properties are utilized in Chapters 4 and 5 in order to develop highly efficient heuristic algorithms for the (α, β)-k FSP.

Chapter 4 develops several heuristic algorithms for both weighted and unweighted Min k (α, β)-k Feature Set Problem. While in the weighted variant there is a cost associated with a feature, in the unweighted variant the cost is unique and equal across all features. The proposed heuristic algorithms include greedy construction and improvements, and one very efficient exact+heuristic (EH) algorithm, which combines both exact and heuristic algorithms, and obtains very high quality solutions for the Min k (α, β)-k Feature Set Problem. We tested those algorithms over three sets of real-world, weighted and randomly generated instances, and in total 346 instances. Computational results show that the proposed EH algorithm provides very good quality solutions, including new best solutions, and competes well against the state-of-the-art algorithms.

Chapter 5 develops exact and heuristic solution algorithms for the Max β (α, β)-k Feature

Set Problem. The major proposed solution method includes an exact+heuristic (EH) algorithm for the Max β (α, β)-k Feature Set Problem. To the best of our knowledge, the EH algorithm obtains the best results for the Max β (α, β)-k Feature Set Problem to date, in particular, for large instances. This is verified through solving 136 instances including real-world and randomly generated instances. This chapter also studies the Max Cover (α, β)-k Feature Set Problem, and proposes a simple but effective solution method for the Max Cover (α, β)-k Feature Set Problem.

In Chapter 6 we wrap up this research thesis by pointing out the major outcomes of the research, in terms of both algorithms and solutions methods, and the computational achievements. Furthermore, we discuss the limitations of the research, and a few directions for the future research.

1.6 Conclusion

This chapter mainly provided an introduction into this research thesis, and explained the research question and goals. We also discussed the structure of thesis, including a short summary on every chapter.

Chapter 2

Research Problem and Literature Review

Abstract

This chapter defines the research problem of this thesis, and states the research motivation. The main research problem of this thesis is to develop optimization methods for the (α, β)-k Feature Set Problem (FSP). The (α, β)-k FSP is a new combinatorial optimization-based approach proposed in 2004 for the feature selection. The (α, β)-k FSP selects features such that the selected set of features maximizes the similarities between entities of the same class and differences between entities of different classes. This chapter discusses the (α, β)-k FSP, and explains how an instance of the (α, β)-k FSP may be built. Additionally, we review some of the most important techniques of feature selection, and note several applications of feature selection, particularly, in the context of computational biology and Bioinformatics. Finally, the chapter discusses research motivation.

2.1 Introduction

The main research problem of this thesis is to develop optimization methods for the (α, β)-k Feature Set Problem (FSP). The (α, β)-k FSP aims to select features in order to distinguish two classes of data such that the selected set of features maximizes the similarities between entities of the same class and differences between entities of different classes. The (α, β)-k FSP was proposed in 2004 by Cotta et al. (2004), and is a combinatorial optimization-based feature selection approach. As authors discussed, the (α, β)-k FSP is a generalization of the k-Feature Set Problem, which is proven $\mathcal{NP}$-Hard (Cotta et al., 2004). Hence, the (α, β)-k FSP is also $\mathcal{NP}$-Hard.

This research is motivated by a broad range of applications that the (α, β)-k FSP has in practice, and in a variety of domains including classification, data mining, computational biology and Bioinformatics, and lack of efficient solution methods, particularly, for large instances and datasets. One such application is to obtain a *biomarker*, which can be a set of SNPs, genes, probes, etc. in order to distinguish healthy and disease samples (see for example Ravetti and Moscato (2008); Ravetti et al. (2009); Ravetti et al. (2010); Paula et al. (2011)). Some of these application are reviewed in Section 2.3.

The remaining of this chapter is organized as follows. The problem of this PhD research is discussed in Section 2.2. This includes problem's concept and framework, notations, and a graph representation. Section 2.3 provides a general literature review on the feature selection methods, in particular, with respect to applications in computational biology and Bioinformatics. We discuss the research motivation in Section 2.4. Finally, Section 2.5 concludes the chapter.

2.2 Problem statement

Presume we are given a dataset, in which two *classes* of data exist, for example, Class 1 and Class 2, and a set $J = \{1, \ldots, n\}, |J| = n$ of *features*, each with a *profile* $P_j, \forall j \in J$ ($P = \{P_j\}, \forall j \in J$). A feature profile P_j includes a set of discrete values in the ranges of 0 and 1. Expectedly, features may not have a unique cost (here, cost is a general term and may model feature's weight, importance, preference, relation to other features, etc.). Thus, $c_j \in \mathbb{R}^+, \forall j \in J$ is the cost associated with selecting feature j (notice that this is an extension to the original (α, β)-k Feature Set Problem (FSP), in which all features have a unique cost of $\mathcal{C}$, where $\mathcal{C} \in \mathbb{R}^+$ is a constant). Furthermore, let S_1 and S_2 denote the set of all entities in Class 1 and Class 2, where $S_1 = \{s_{11}, \ldots, s_{1,n_1}\}, |S_1| = n_1$, and $S_2 = \{s_{21}, \ldots, s_{2,n_2}\}, |S_2| = n_2$. Let I_1 and I_2 represent sets of pairs of entities of different classes, and of the same class. Then I_1 includes all pairs of entities (every combination of size two of entities) belonging to different classes, and I_2 includes all pairs of entities (every combination of size two) belonging to the same class. Sets I_1 and I_2 can be formed by using Equation (2.1) and Equation (2.2). The cardinality of these sets may be derived by using Equation (2.3) and Equation (2.4). We shall call I_1 and I_2 sets of *elements*.

$$I_1 = \{(s_{11}, s_{21}), \ldots, (s_{11}, s_{2,n_2}), \ldots, (s_{1,n_1}, s_{2,n_2})\} \tag{2.1}$$

I_1 is the set of all pairs of entities $(s_{1,t}, s_{2,t'})$, where $s_{1,t} \in S_1, \forall t = 1, \ldots, n_1$, and $s_{2,t'} \in S_2, \forall t' = 1, \ldots, n_2$.

$$I_2 = \{(s_{11}, s_{12}), \ldots, (s_{11}, s_{1,n_1}), \ldots, (s_{21}, s_{22}), \ldots, (s_{21}, s_{2,n_2})\} \tag{2.2}$$

Similarly, I_2 includes all pairs of entities $(s_{1,t}, s_{1,t'})$, where $(s_{1,t}, s_{1,t'}) \in S_1, \forall t, t' = 1, \ldots, n_1, t \neq t'$, and $(s_{2,t}, s_{2,t'}) \in S_2$, where $(s_{2,t}, s_{2,t'}) \in S_2, \forall t, t' = 1, \ldots, n_2, t \neq t'$.
The cardinality of I_1 and I_2 is then

$$|I_1| = \binom{|S_1|}{1} \times \binom{|S_2|}{1} = n_1 \times n_2 \tag{2.3}$$

$$|I_2| = \binom{|S_1|}{2} + \binom{|S_2|}{2} = \frac{n_1 \times (n_1 - 1)}{2} + \frac{n_2 \times (n_2 - 1)}{2} \tag{2.4}$$

Indeed, the (α, β)-k FSP looks for a minimum cost set of features, where the set maximizes the differences between the entities of different classes (set I_1) and similarities between the entities of the same class (set I_2). We denote this set of features by $J^* \subseteq J$. In computational biology and Bioinformatics features may represent proteins, genes, probes, SNPs, etc., while Class 1 may represent a set of healthy samples, and Class 2 a set of disease samples.

Given these notations and definitions, we can proceed to mathematically explain the (α, β)-k FSP. Notice that the model of (α, β)-k FSP, which was originally proposed by Cotta et al. (2004); Berretta et al. (2005), and was also discussed in Paula (2012), does not consider costs for selecting features. We extend that by considering costs of selecting features in all models and solution methods of this research; so one may model the cost of features in future analysis. Therefore, instead of a selecting a set of k features, we may select a set of minimum cost features. The original (α, β)-k FSP is defined with three positive integer parameters α, β, and k. The value of α represents the minimum number of features that must explain the differences between any pair of entities of different classes. The value of β represents the minimum number of features that must explain the similarities between any pair of entities of the same class. Finally, k represents the number of features to be selected. More precisely, the (α, β)-k FSP has the following characteristics.

- Every element in I_1 (pair of entities of different classes) must be "explained" by at least α features. We re-state this as element $i, \forall i \in I_1$ must be *covered* by at least α features, where α is a parameter, and $1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+$ (Requirement 1).

- A set $J^* \subseteq J$ of features with the minimum *cost*, among all alternative sets, must be selected (Objective 1). In other words, $\Sigma_{j \in J^*} c_j$ has the smallest cost among every alternative set of features.

- Every element in I_2 (pair of entities of the same class) must be "explained" by at least β features, where $1 \leq \beta \leq \beta^*, \beta \in \mathbb{Z}^+$ (Objective 2).

Now let us explain how we can build an instance of the (α, β)-k FSP from a dataset with two classes (groups) of data. For the sake of illustration, we shall explain this by bringing a small example. Assume we are given a dataset that includes two different classes (groups) of data (see Table 2.1). Class 1 (e.g. Control) consists of three healthy samples (entities), and Class 2 (e.g. Case) consists of three disease samples (the number of elements in the classes do not need to be equal). Furthermore, the dataset includes five features, which may be protein, genes, probes, SNPs, etc.

Table 2.1: A dataset with two classes (groups) of data. The dataset includes five features (e.g. proteins, probes, genes, SNPs, etc.), and three samples (entities) in each class. Every feature has an equal cost of one. Moreover, a feature appears either up-regulated (associated with a value of 1) or down-regulated (associated with a value of 0) in a sample.

Feature	Cost	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6
A	1	0	0	0	0	1	0
B	1	1	1	1	1	0	1
C	1	1	1	1	0	1	0
D	1	1	1	1	1	0	0
E	1	0	0	1	0	0	0
Class		Healthy	Healthy	Healthy	Disease	Disease	Disease

The entities of Table 2.1 may refer to discretized gene expression levels (Berretta et al., 2005). The first column in Table 2.1 states the name of features, and the second column holds the cost of selecting a feature. For this example, we assumed that all features have a unique cost of 1 (this may be referred to *unicost* or *unweighted* case). The last row in Table 2.1 states the label of the classes. Hence, one may distinguish Class 1 and Class 2. For example, Samples 1, 2 and 3 belong to the healthy class (Class 1) while Samples 4, 5 and 6 belong to the disease class (Class 2). Row j corresponds to the values of expression level of feature j for samples. Indeed, a feature may be up-regulated (associated with a value of 1) or down-regulated (associated with a value of 0) in a sample. For the sake of simplicity of this example, we assumed that the values for expression level only take 0, for a down-regulated level, or 1, for an up-regulated level. Because the required data for the (α, β)-k FSP must be discrete and the values found on many datasets are often real numbers, the method of Fayyad and Irani (1993) can be applied to the values of data in order to discretize them; see also Cotta et al. (2004); Paula (2012).

Given the dataset presented in Table 2.1, the first step in building an instance of the (α, β)-k FSP is to derive sets I_1 and I_2. The sets I_1 and I_2 can be derived by using Equation (2.1) and Equation (2.2):

$$I_1 = \{(1,4), (1,5), (1,6), (2,4), (2,5), (2,6), (3,4), (3,5), (3,6)\}$$

and

$$I_2 = \{(1,2), (1,3), (2,3), (4,5), (4,6), (5,6)\}$$

The second step is to derive feature profiles. The profile of feature j can be modeled by a set of binary values. More precisely, $P_j = \{a_{ij} \in \{0,1\}, \forall i \in I_1 \cup I_2, \forall j \in J\}$. For element $i \in I_1$ (pairs of entities of different classes), if feature j has different values of expression level for the pair (for example, one entity has a value of 1 and the other 0), then $a_{ij} = 1$. Otherwise,

Table 2.2: Building an instance of the (α, β)-k Feature Set Problem (FSP) from the dataset presented in Table 2.1. To do so, feature profiles are extracted by determining the values of parameter a_{ij}. Notice that if feature j has different values of expression level for a pair i of entities (samples), then $a_{ij} = 1$, otherwise $a_{ij} = 0$.

Pair (members of set I_1)	P_A	P_B	P_C	P_D	P_E
(1,4)	0	0	1	0	0
(1,5)	1	1	0	1	0
(1,6)	0	0	1	1	0
(2,4)	0	0	1	0	0
(2,5)	1	1	0	1	0
(2,6)	0	0	1	1	0
(3,4)	0	0	1	0	1
(3,5)	1	1	0	1	1
(3,6)	0	0	1	1	1

$a_{ij} = 0$. Equation (2.5) illustrates the calculation of values of parameter a_{ij}. Notice that $a_{ij}, \forall i \in I_2, j \in J$ will differently be determined than $a_{ij}, \forall i \in I_1, j \in J$.

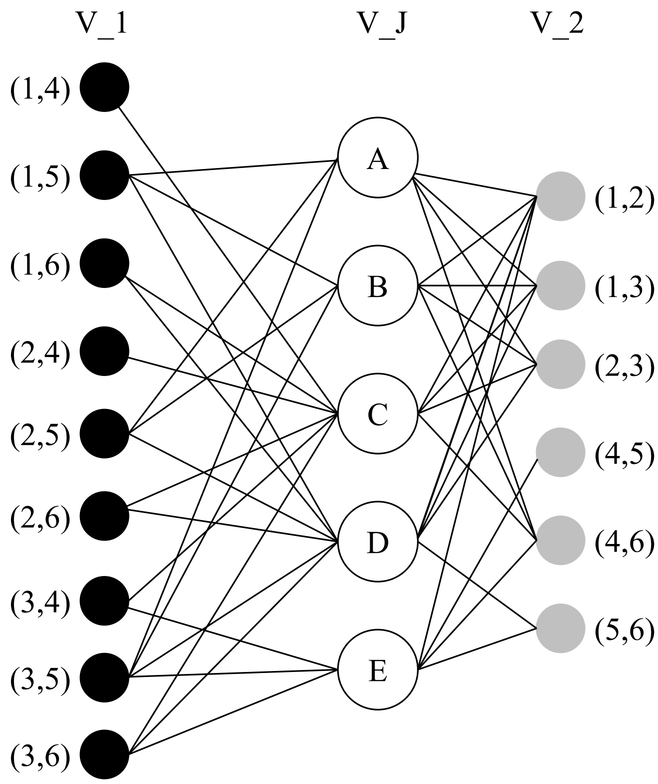
$$a_{ij} = \begin{cases} 1, & \text{if } \varepsilon_{jt} \neq \varepsilon_{jt'}, \text{ where } i \in I_1, t = 1, \ldots, n_1, t' = 1, \ldots, n_2 \\ 1, & \text{if } \varepsilon_{jt} = \varepsilon_{jt'}, \text{ where } i \in I_2, t, t' = 1, \ldots, n_1, t \neq t' \vee t, t' = 1, \ldots, n_2, t \neq t' \\ 0, & \text{Otherwise} \end{cases} \tag{2.5}$$

where $\varepsilon_{jt}, \forall j \in J, t = 1, \ldots, n_1 \vee t = 1, \ldots, n_2$ is the value of expression level of feature j for entity t.

Table 2.2 shows the values for $a_{ij}, \forall i \in I_1, j \in J$ (here, we only show $a_{ij}, \forall i \in I_1, j \in J$). The first column of Table 2.2 shows all members of set I_1. Entries of table (values of 0 and 1) are the values of parameter a_{ij}. For example, the first entry, which is associated with feature "A" and pair "(1,4)", has a value of 0 because feature "A" does not have different values of expression level for each entity in the pair. In fact, feature "A" has expression level values of 0 for both Samples 1 and 4; hence, $a_{11} = 0$. On the other hand, $a_{21} = 1$ (associated with feature "A" and pair "(1,5)") because feature "A" has different values of expression level for Samples 1 and 5.

One may realize that Table 2.2 represents feature profiles $P = \{P_j, \forall j \in J\}$, where P_j is profile of feature j. A feature profile explains whether a feature distinguishes (*covers*) a set of elements, where an element is a pair of entities. For example, feature "A" distinguishes pairs of Samples (1,5), (2,5), and (3,5), while feature "E" distinguishes pairs of Samples (3,4), (3,5), and (3,6). According to the features profile presented in Table 2.2, we can see that features A and C are able to distinguish (cover) all elements of I_1 (pairs of entities belonging to different classes).

Figure 2.1: A bipartite graph $G = (V_J \cup V_1 \cup V_2, E_1 \cup E_2)$ associated with the (α, β)-k Feature Set Problem (FSP). The graph is built upon the example of Table 2.1, where V_1 and V_2 are sets of vertices associated with I_1 and I_2, and V_J is the set of vertices representing features. E_1 and E_2 are two sets of disjoint edges: $E_1 = \{e_{ij} | a_{ij} = 1, \forall i \in V_1, j \in V_J\}$, and $E_2 = \{e_{ij} | a_{ij} = 1, \forall i \in V_2, j \in V_J\}$. The sets of vertices of V_1, V_2, and V_J are shown by using different colors.



We can represent features and elements as a bipartite graph by three sets of disjoints vertices (nodes). Let V_J denotes the set of vertices representing features, where $|V_J| = |J|$, V_1 the set of vertices representing elements of I_1, where $|V_1| = |I_1| = n_1$, and V_2 the set of vertices representing elements of I_2, where $|V_2| = |I_2| = n_2$. An edge $e_{ij}, \forall i \in V_1 \cup V_2, j \in V_J$ connects vertex i to vertex j if and only if $a_{ij} = 1$. We may see that this results in a bipartite graph[1] $G = (V_J \cup V_1 \cup V_2, E_1 \cup E_2)$, where $E_1 = \{e_{ij} | a_{ij} = 1, \forall i \in V_1, j \in V_J\}$ and $E_2 = \{e_{ij} | a_{ij} = 1, \forall i \in V_2, j \in V_J\}$. The bipartite graph associated with the example of Table 2.1 is illustrated in Figure 2.1.

Figure 2.1 reveals several important concepts regarding the combinatorial optimization problem of (α, β)-k FSP. Firstly, because every element of I_1 must be explained (covered) by at least α features, the number of edges adjacent to every vertex $i \in V_1$ must at least be α. This implies that $|J^*| \geq \alpha$. Secondly, a set of features with the minimum cost is indeed a set of vertices $J^* \subseteq V_J$ with the minimum total cost. Thirdly, in order to maximize the similarities

[1] Given graph $G = (V, E)$ with vertex set V and edge set E, a bipartite graph of G divides V into two disjoint subsets V' and V'' such that their joint is V, and an edge between V' and V'' has one end point in either.

between any pair of entities of the same class the selected subset of features must have the maximum degree[2] over E_2. Finally, if more than one set of features with these properties exist, we may wish to choose the set with the largest possible covering, that is, the set of vertices $J^* \subseteq V_J$ has the maximum degree over E_1 and E_2 (later in Chapter 3 we shall see this refers to the combinatorial optimization problem of Maximum Cover (α, β)-k Feature Set).

2.3 A review of feature selection methods

In this section, we shall provide a brief review of the relevant research in the context of feature selection, and related to the problem of this research thesis. The "feature selection", also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features for the purpose of classification and clustering, and have a broad range of applications including machine learning and prediction. For example in urban transport network systems (Ferchichi et al., 2009), market investment and stock price prediction (Tsai and Hsiao, 2010; Meiri and Zahavi, 2006), and computational biology and Bioinformatics (Ravetti and Moscato, 2008; Ravetti et al., 2009; Ravetti et al., 2010; Paula et al., 2011; Haque et al., 2016).

Feature selection is an important tool in data mining. The main idea and motivation behind feature selection is that data contain many redundant, irrelevant or unimportant features, which may not be of interest when generating a model, or analyzing data. Research on this problem is very rich, see for example Liu and Motoda (1998); Guyon and Elisseeff (2003); Zhao et al. (2010). We refer the interested reader to Liu and Motoda (2007) for more details.

The methods of selecting feature operate by selecting a set of features, out of a larger set, and evaluating the set by some criteria. One such criterion is the amount of error, and hence, we are interested in a set of features with the minimum error. The main difference of the feature selection methods lies in the evaluation criteria. The major feature selection methods include (Chandrashekar and Sahin, 2014):

- Wrapper methods. Wrapper methods use the classifier data to guide the search; thus, they are dependent on the classifiers. Upon finding a new set of features, the set is applied to a training data, and its outcome is evaluated against the benchmark outcome. The gap in the outcomes may be evaluated as the error of the set of features. Because every time a new set is obtained, it is applied on a training data, Wrapper methods are computationally very expensive, however, they usually provide the best outcome, i.e. the best set of features. Sequential search is one of the most used algorithms in Wrapper methods. For example Inza et al. (2004) compared the outcomes of their sequential search method with a filter method for Colon and Leukemia cancer data. Another example is the sequential search algorithm of Xiong et al. (2001). The complexity of Wrapper methods has lead to development of heuristic algorithms in order to speed up Wrapper methods,

[2] The degree of a vertex is the number of edges adjacent to the vertex.

particularly, the evolutionary algorithms. Examples include the studies of Duval and Hao (2009); Blanco et al. (2004); Jirapech-Umpai and Aitken (2005).

- Filter methods. These methods leave aside those irrelevant features by filtering out the most relevant and most important features. Because performance evaluation criterion in Filter methods is directly calculated from data, Filter methods are independent of any classifier/predictor. Due to this, typically Filter methods are computationally less expensive than Wrapper and Embedded methods. A comparison of several Filter methods was reported in Sánchez-Maroño et al. (2007). According to Saeys et al. (2007), Filter methods are among the most applicable methods, and hence, a very rich literature on the methods and applications exists. For several examples, we refer the interested reader to Breitling et al. (2004); Fox and Dimmic (2006); Wang et al. (2005); Jafari and Azuaje (2006).

- Embedded methods. These methods combine different criteria, algorithms, and approaches, some of which belong to the methods of Wrapper and Filter. Embedded methods interact with the classifier, hence, they are dependent on the classifier. However, they do not have the computational complexity of Wrapper methods. The studies of Díaz-Uriarte and Andrés (2006); Jiang et al. (2004); Ma and Huang (2005) investigated and developed several Embedded methods, and applied them to biological data.

- Combinatorial Optimization methods. Although this is not mentioned as an independent category in previous studies (for example in Saeys et al. (2007); Paula (2012)), following the focus of this research on combinatorial optimization methods for the (α, β)-k Feature Set Problem, we shall discuss these methods as a new category (similarly, Wang et al. (2016a) categorizes feature selection methods into three categories of exhaustive search, heuristic search, and hybrid search methods, instead of traditional Wrapper, Filter, and Embedded methods). However, one may realize that Combinatorial Optimization methods can be categorized under Filter methods because both filter out irrelevant features, and both are independent of classifiers. Several major works in this area, which are relevant to the presented research, are due to Berretta et al. (2005); Berretta et al. (2008). The authors' major idea to discard irrelevant features is to select the smallest subset of features that maximizes the similarities between entities of the same class and differences between entities of different classes. Depending on datasets, the computational complexity of Combinatorial Optimization methods may greatly vary, and hence, heuristic algorithms have been considered as well. The studies of Berretta et al. (2007); Berretta et al. (2008); Ravetti and Moscato (2008); Ravetti et al. (2009); Paula et al. (2011) focused on utilizing the standard exact solvers, and that only for instances that the exact solvers are capable of solving in a reasonable amount of time. In contrast to those, Paula (2012) developed the first heuristic algorithms. His algorithms include Variable Neighborhood Search+Tabu Search (VNS+TS) algorithms with certain randomized local searches. The author validated his algorithms on a set of randomly generated instances, as well as a set of six biological datasets (we used the same sets to conduct our computational experiments).

There are many studies applying feature selection to a variety of problems in the areas of computational biology and Bioinformatics. Saeys et al. (2007) provides an excellent review of those problems and applications. In this regard, two well-studied algorithms are evolutionary and population based algorithms. For instance, Yongming et al. (2009) developed a Genetic Algorithm (GA) for selecting features. Chuang et al. (2009) developed a hybrid algorithm where a TS guides the Particle Swarm Optimization (PSO) algorithm as a local search. In another study, Unler and Murat (2010) implemented a PSO algorithm where relevancy and dependency of features are dynamically checked. The PSO algorithm of Chuang et al. (2011) uses the k-Nearest Neighbor (kNN) as an objective function criterion. Yang and Olafsson (2009) used nested partitions method as a local search. Kabir et al. (2012) proposed a hybrid Ant Colony Optimization (ACO) to be used within Wrapper and Filter methods. Jain et al. (2018) proposed a hybrid model for cancer classification, where the main component of the model is a PSO-based algorithm. The model was shown to obtain better sets of features (in terms of classification accuracy and the number of selected genes) than available methods. Ghosh et al. (2019) proposed a Memetic Algorithm (MA) for gene selection. More precisely, they utilized a recursive MA and tested it on seven microarray datasets. The developed algorithm was reported to obtain promising results.

For details of other non-optimization-based methods, we refer the interested reader to Urbanowicz et al. (2018), who investigates a set of filter-style feature selection algorithms aiming at developing efficient algorithms and tackling large scale and various datasets, Kuncheva and Rodriguez (2018), who performs a comparative study investigating performance of certain feature selection methods on sets of high-dimensional datasets, and to Shukla et al. (2019), who investigates several Filter methods.

Although most of the studies on feature selection have focused on benchmark and standard datasets and applications other than computational biology and Bioinformatics, there are several studies related to Bioinformatics, more particularly, on gene expression datasets. For example, Albrecht (2006) applied an algorithm on a gene expression data of Leukemia. The algorithm revealed three genes with zero errors. Fan and Chaovalitwongse (2010) applied an optimization algorithm to select weighted and unweighed features with the objective of maximizing a "correct classification of data". Their findings related to several biological datasets including epilepsy, breast cancer, heart disease, diabetes and liver disorders revealed that the algorithm uses fewer features for classification, compared to the previous studies. Bar et al. (2018) discusses an interesting application of feature selection for pathology detection in chest X-rays, in which the set of most informative features are selected. The proposed method was tested on a dataset of about 600 samples. Kong and Yu (2018) developed an Embedded-based method, and tested it on two datasets in the contexts of breast and kidney cancers. The feature selection method of Emura et al. (2019) was applied on the lung cancer datasets and was shown to deliver an optimal subset of genes for prediction. The approach of Pati et al. (2019) includes first grouping genes with similar gene expressions into clusters, and then selecting informative genes from each cluster. They applied the method on several publicly accessible datasets.

The usage of more than one criterion as an evaluation measure for feature selection has also been studied. Wang and Huang (2009) formulated feature selection as a multi-objective optimization problem. Vieira et al. (2010); Vieira et al. (2012) used fuzzy models in classification. This allows flexibility in defining objectives and in weighting different objectives. For this purpose, they developed an ACO algorithm with objectives of both minimizing the number of features and classification error. Dashtban et al. (2018) introduced a multi-objective bat algorithm for selecting genes from cancer datasets. The studies of Dos Santos et al. (2018); Lai (2018) utilized multi-objective genetic and swarm optimization algorithms for feature and gene selection applications. Recently, González et al. (2019) developed a multi-objective evolutionary algorithm and proposed solutions for a Wrapper-based method.

To the best of our knowledge, the only studies on developing combinatorial optimization models, and mathematical programming formulations for feature selection, in particular, in Bioinformatics are due to Cotta et al. (2004); Berretta et al. (2005); Berretta et al. (2008). They call their model (α,β)-k Feature Set Problem (FSP). The model differs from previous studies in that it aims to obtain the minimum number of features such that similarities between entities of the same class and differences between entities of different classes are maximized. The major drawbacks of their formulation are that it relies on standard solvers, and therefore, it can only solve small and medium sized datasets (usually many biological datasets are large), and that their methods are unweighted (in contrast to this research thesis, they have not modeled the cost of features). Later, Ravetti and Moscato (2008) and Ravetti et al. (2009) applied the (α, β)-k FSP to select features from the Alzheimer's Disease and Prostate Cancer datasets. As discussed by the authors, the selected features led to novel biomarkers with better accuracy for prediction of those diseases.

2.4 Research motivation

As discussed in Section 2.3, there exists only a few studies on the combinatorial optimization methods for the (α, β)-k Feature Set Problem (FSP). Although those studies utilized both exact and heuristic methods, they have several drawbacks and limitations. The major limitation of the exact methods is that they rely on the standard solver CPLEX, and hence, are computationally very expensive (see for example Cotta et al. (2004); Berretta et al. (2008)). Having said that, they are unable to be applied to large datasets, and not to mention that many real world applications of (α, β)-k FSP include large datasets, and hence, they demand for effective and efficient methods. As heuristic algorithms, the major drawback of the algorithms of

Paula (2012) is that they rely on general heuristics and randomized elements developed for the traditional combinatorial optimization problems, whereas it is well accepted in the literature that problem-driven local searches usually lead to superior outcomes.

In addition to those limitations, the cost associated with selecting features was not studied by Cotta et al. (2004); Berretta et al. (2008); Paula (2012). This is indeed important because in practice features may have different distinguishing factors including cost, importance, impact, dependency on other features, etc. These limitations and drawbacks are the main motivations behind this research. More importantly, this research aims to overcome the limitations of the existing methods by developing efficient algorithms that are able to deliver high quality solutions for large instances of the (α, β)-k FSP, and that in a reasonable amount of time. In particular, this research may well be distinguished from the previous studies by

- developing greedy and problem-driven local searches, as well as hybrid algorithms (exact+heuristic) in order to efficiently solve the (α, β)-k FSP;
- developing and implementing algorithms for the (α, β)-k FSP that are applicable to large datasets, and are quite capable of providing effective and efficient solutions for those datasets; and,
- including the cost associated with selecting features. The cost may model distinguishing factors of features, for example, their importance, their correlation with other features, their dependency on other features, etc.

2.5 Conclusion

In this chapter we stated the main research problem of this thesis. Feature selection is a fundamental concept in the areas of machine learning, classification, and prediction with a huge number of applications. A review of the state-of-the-art methods and techniques for feature selection, as well as for the (α, β)-k FSP, revealed that there are gaps in the previous studies in terms of efficient solution methods for the (α, β)-k FSP, particularly, for large datasets and instances. The latter is very important because many applications of the (α, β)-k FSP include dealing with large datasets.

Chapter 3

Mathematical Models and Properties

Abstract

This chapter investigates several mathematical properties of the (α, β)-k Feature Set Problem (FSP). Furthermore, the chapter discusses a four-stage approach to solve the (α, β)-k FSP, which leads to solving four optimization problems. We show mathematical connections among these problems, and develop several bounds and propositions for them. These bounds and propositions will be utilized in Chapters 4 and 5 to develop highly efficient algorithms and solution methods for the (α, β)-k FSP.

3.1 Introduction

This chapter provides mathematical foundation of this thesis by exploring and developing mathematical properties and bounds for the (α, β)-k Feature Set Problem (FSP). As discussed in Chapter 2, this problem selects a minimum cost/cardinality set of features such that similarities between entities of the same class and differences between entities of different classes are maximized (Paula, 2012). Moreover, the chapter discusses a four-step decomposition-based approach for solving the (α, β)-k FSP. This four-step approach decomposes the problem into four combinatorial optimization problems (sub-problems). For this reason, this chapter also studies models, bounds, and mathematical propositions for these sub-problems. In a previous study by Paula (2012) a similar four-step approach was proposed in order to solve the (α, β)-k FSP, however, that study did not investigate mathematical properties of the (α, β)-k FSP.

Let us start by explaining the four-step decomposition-based approach for solving the (α, β)-k FSP. Recall from our earlier discussion in Chapter 2 that the (α, β)-k FSP has three positive integer parameters: α, β, and k. The value of α represents the minimum number of features that must explain the differences between any pair of entities of different classes. The value of β represents the minimum number of features that must explain the similarities between any pair of entities of the same class, and k represents the number of features to be selected. The four-step decomposition-based approach in solving the (α, β)-k FSP, which involves four optimization problems (sub-problems), includes the followings.

- Step 1. Obtaining the maximum value of α (i.e. $\alpha^* \in \mathbb{Z}^+$) such that there exists a feasible solution for an instance of the (α, β)-k FSP. Clearly, the value of α^* depends on the instance under investigation. However, as we will see later α^* can be derived in polynomial time (Sub-problem 1).

- Step 2. Obtaining the minimum number of features (i.e. k^*) necessary to explain the dichotomy between the classes (in the weighted variant, k^* refers to the minimum cost set of features), considering that at least α^* features do so for each pair of entities of different classes (Sub-problem 2). This can be modeled as an integer program (IP), where α^* (obtained in Step 1) is a parameter. This problem is known as the Min k (α, β)-k Feature Set Problem (FSP). Obviously, any positive integer value less than α^* is still possible and will lead to a different value for k^*. If the cost of selecting features is unique, then the problem is *unweighted* or *unicost*. Because there is no distinguishing factors to model the importance of features. Otherwise, the problem is *weighted*. Interestingly, the unweighted variant is more difficult to solve than the weighted variant (Vasko and Wilson, 1986).

- Step 3. Obtaining the maximum value of β (i.e. $\beta^* \in \mathbb{Z}^+$) such that a set of minimum cost features are selected to explain the dichotomy between the classes, and at least α^* features do so for each pair of entities of different classes. This can be modeled as an IP, where α^* and k^* (obtained in Steps 1 and 2) are parameters. This problem is known the Max β (α, β)-k Feature Set Problem (FSP). Notice that this step maximizes the internal consistency of the entities in the same class; hence, it provides a more robust feature set. In fact, in the Max β (α, β)-k FSP, the values of α^* and k^* are known, and the set of features is sought (Sub-problem 3).

- Step 4. Among alternative minimum cost set of features (each with the cost of k^*), obtaining one that provides more explanations (*coverage*) in total, either to the differences between the classes or similarity within entities in the same class. This may be modeled as an IP, where α^*, β^*, and k^* (obtained in the previous steps) are parameters. This optimization problem is called the Maximum (Max) Cover (α, β)-k Feature Set Problem (FSP) (Paula, 2012) (Sub-problem 4). The solution to the Max Cover (α, β)-k FSP is a minimum cost set of features that maximizes the similarities between entities of the same class and the differences between entities of different classes, and has more explanations (coverage) in total.

Notice that in Step 1 the value of α^* is determined such that there exists a feasible solution for an instance of the (α, β)-k FSP. Therefore, we already know that at least one such set of features exists. This along with the fact that we implement a four-step approach to solve the (α, β)-k FSP imply that Steps 2, 3, and 4 are guaranteed to deliver a feasible solution. Finally, we need to efficiently solve each step in order to obtain high quality solutions for the (α, β)-k FSP. To do so, we will design and develop efficient algorithms and methods in Chapter 4 and Chapter 5. These algorithms and solution methods utilize the bounds and properties that we develop in this chapter. Such a study into mathematics of the (α, β)-k FSP has not previously been performed. The major contribution of this chapter lies in developing bounds and mathematical properties and propositions for the (α, β)-k FSP.

The remainder of this chapter is organized as follows. Section 3.2 defines concepts and mathematical notations. Section 3.3 provides a graph representation for the (α, β)-k FSP similar to the one discussed in Chapter 2, however, focuses on the problem's concepts rather than definition. The mathematical models will be discussed in Section 3.5. Lower and upper bounds, and properties and propositions for the (α, β)-k FSP as well as their proofs will be discussed in Section 3.6 and Section 3.7. The chapter ends with a few conclusions in Section 3.8.

3.2 Definitions and notations

Before going into the details of mathematical models, bounds, and propositions we first define all concepts and mathematical notations. As stated earlier in Chapter 2, the problem of this research is to develop efficient algorithms and solution methods for the (α, β)-k Feature Set Problem (FSP).

Let $J = \{1, \dots, n\}$, where $|J| = n$, be the set of all features, out of which a set $J^* \subseteq J$, which has the minimum cost/cardinality, must be chosen, and $P = \{P_j\}, \forall j \in J$, be the set of profiles of features. Also, we have two sets of elements (sets of *universes*): $I_1 = \{I_{1i}\}, i = 1, \dots, m_1, |I_1| = m_1$ (pairs of entities of different classes), and $I_2 = \{I_{2i}\}, i = 1, \dots, m_2, |I_2| = m_2$ (pairs of entities of the same class). The set of all elements is then $I = I_1 \cup I_2$. For the reasons that we will discuss in Chapter 4, we may use words *feature* and *column* interchangeably, and words *element* and *row*.

A profile P_j of feature j can be defined by a set of binary values, and by observing that in which elements feature j has a value of 1 and in which it has a value of 0. We may characterize this by parameter $a_{ij} \in \{0, 1\}, \forall i \in I_1 \cup I_2, j \in J$. Equivalently, if feature j has a value of 1 in element i we may say feature j *explains or covers* element i. Thus, P_j is a list of elements that feature j covers. Typically, feature j may not cover all elements.

The *cost* of feature j represents the cost, weight, priority, importance, etc. of the feature. For this reason, parameter $c_j \in \mathbb{R}^+, \forall j \in J$ denotes the cost of feature j. Thus, if feature j is chosen, it incurs a cost of c_j. One special case is where $c_j = \mathcal{C}, \forall j \in J, \mathcal{C} \in \mathbb{R}^+$. This may also be referred to the *unweighted* or *unicost* variant because all features have a unique cost, and furthermore, the cost of features does not impact the solution.

The *value* of feature j (*coverage degree*), which is denoted by $v_j \in \mathbb{Z}^+, \forall j \in J$, is derived by counting the number of elements that feature j covers. Equation (3.1) calculates v_j. In Section 3.3 we will explain that when we represent the (α, β)-k FSP on a bipartite graph, vertices represent features. Thus, the value of a feature is equivalent to the degree of the associated vertex.

$$v_j = \Sigma_{i \in I_1 \cup I_2} a_{ij} \tag{3.1}$$

Interestingly, the *coverage level* of an element, i.e., the total number of times that an element $i, \forall i \in I_1$ can be covered (by all features) may be denoted by $\alpha_i \in \mathbb{Z}^+, \forall i \in I_1$, and can be calculated by Equation (3.2). In addition to this, $\alpha^* \in \mathbb{Z}^+$ (which is also the solution to Sub-problem 1) may be derived by Equation (3.3).

$$\alpha_i = \Sigma_{j \in J} a_{ij}, \forall i \in I_1 \tag{3.2}$$

$$\alpha^* = \min_{i \in I_1}(\alpha_i) \tag{3.3}$$

Firstly, notice that any value greater than α^* will lead to infeasiblity. Secondly, recall from our previous discussion that α^* is the minimum number of features that explain the differences between any pair of entities of different classes. Table 3.1 summarizes all sets, indices, parameters and decision variables associated with the (α, β)-k FSP.

Given all sets, notations, parameters and decision variables, we may establish those by one illustrative example. Assume we have five features, the set of which is $J = \{1, 2, 3, 4, 5\}$, where the set of their profile is $P = \{P_1, P_2, P_3, P_4, P_5\}$, and two sets of elements $I_1 = \{I_{11}, I_{12}, I_{13}, I_{14}, I_{15}, I_{16}\}, |I_1| = 6$, and $I_2 = \{I_{21}, I_{22}, I_{23}, I_{24}, I_{25}\}, |I_2| = 5$. Table 3.2 shows the sets as well as values for parameter a_{ij}. According to the table, feature 1 covers every element of set I_1 because it has a value of 1 for every element. More precisely, $a_{ij} = 1, \forall i \in I_1, j = 1$. On the other hand, feature 5 does not cover any element of I_1 because $a_{ij} = 0, \forall i \in I_1, j = 5$. The remaining features (features 2, 3, and 4) each covers certain elements of I_1, and this can be recognized by looking into the values of parameter $a_{ij}, \forall i \in I_1, j = 2, 3, 4$. Now let us discuss how we may obtain $\alpha_i, \forall i \in I_1$ (the values of which are shown in the last column of the table). Remember that α_i denotes the coverage level of element $i, \forall i \in I_1$ (by all features); that is, the total number of features covering element i. Given α_i, we may derive $\alpha^* = \min_{i \in I_1}(\alpha_i) = \min(2, 3, 3, 2, 3, 3) = 2$. This implies that not every element of I_1 can be covered by more than $\alpha^* = 2$ features. For example, even if we choose all features, not all elements of I_1 can be covered by more than 2 features. Hence, based on the value of α^* the feasibility of a given coverage level may easily be evaluated. For instance, here $\alpha = 3$ leads to infeasibility. Further in the example, in order to obtain the value/degree of features, we follow Equation (3.1), and obtain $v_1 = 11, v_2 = 5, v_3 = 6, v_4 = 6$, and $v_5 = 4$.

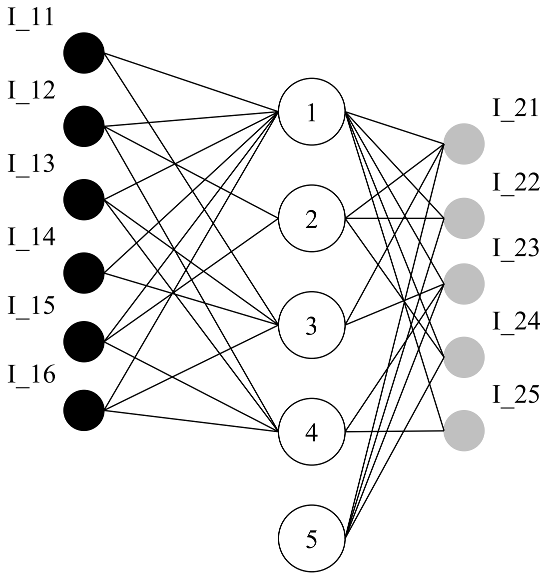
Table 3.1: A summary of all sets, indices, parameters and decision variables used in the (α, β)-k Feature Set Problem.

Type	Notation	Explanation	Value
Parameter	J	Set of all features	$J = \{1, \ldots, n\}$
	P_j	Set of profiles of features	$P = \{P_j\}$
	I_1	The first set of elements (pairs of entities of different classes)	$I_1 = \{I_{11}, \ldots, I_{1,m_1}\}, \lvert I_1\rvert = m_1$
	I_2	The second set of elements (pairs of entities of the same class)	$I_2 = \{I_{21}, \ldots, I_{2,m_2}\}, \lvert I_2\rvert = m_2$
	I	The set of all elements	$I = I_1 \cup I_2$
	a_{ij}	States whether feature j covers element i	$a_{ij} \in \{0, 1\}$
	c_j	Cost (weight, priority, importance, etc.) of feature j	$c_j \in \mathbb{R}^+$
	v_j	Value of feature j (coverage degree)	$v_j = \Sigma_{i \in I_1 \cup I_2} a_{ij}, v_j \in \mathbb{Z}^+$
	α_i	Coverage level of element $i \in I_1$ by all features	$\alpha_i = \Sigma_{j \in J} a_{ij}, \alpha_i \in \mathbb{Z}^+$
Decision variables	x_j	Takes 1 if feature j is chosen to be in a solution, and 0 otherwise	$x_j \in \{0, 1\}$
	α^*	Minimum number of features covering elements of I_1	$\alpha^* = \min_{i \in I_1}(\alpha_i)$
	β^*	Minimum number of features covering elements of I_2	$\beta^* \in \mathbb{Z}^+$

Table 3.2: An illustrative example to explain the notations and concepts of (α, β)-k Feature Set Problem (FSP). The example includes five features, and a total of 11 elements in two sets.

Element	Profile					α_i
	P_1	P_2	P_3	P_4	P_5	
I_{11}	1	0	1	0	0	2
I_{12}	1	1	0	1	0	3
I_{13}	1	0	1	1	0	3
I_{14}	1	0	1	0	0	2
I_{15}	1	1	0	1	0	3
I_{16}	1	0	1	1	0	3
I_{21}	1	1	1	0	1	-
I_{22}	1	1	0	0	1	-
I_{23}	1	0	1	1	1	-
I_{24}	1	0	0	0	1	-
I_{25}	1	1	0	1	0	-
Value of feature (v_j)	11	5	6	6	4	

Figure 3.1: An undirected bipartite graph associated with the data of Table 3.2. In this graph, the black vertices (on the left) represent set I_1, the gray vertices (on the right) represent set I_2, and the white vertices (in the middle) represent features. An edge between feature j and element i is formed if and only if $a_{ij} = 1$. In this graph vertex "5" is not connected to any black vertex because it does not cover any element of I_1. Note that α_i refers to the degree of black vertices, and v_j refers to the degree of white vertices. Finally, there are no edges connecting black vertices (similarly, gray vertices, and white vertices). Additionally, there are no edges connecting black and gray vertices.



3.3 A bipartite graph representation

We may present the (α, β)-k Feature Set Problem (FSP) on an undirected bipartite graph $G = (V, E, C)$, where $V = J \cup I_1 \cup I_2$ is the set of all vertices (nodes), $|V| = |J| + |I_1| + |I_2|$, and $E = \{\{e_{ij}|a_{ij} = 1, i \in I_1, j \in J\} \cup \{e_{ij}|a_{ij} = 1, i \in I_2, j \in J\}\}$ is the set of all edges, where $|E| = \Sigma_{j \in J} v_j$. Set $C = \{c_j|\forall j \in J\}$ holds the features' weights. Although a similar representation was discussed in Chapter 2, the focus of this section is on illustrating the notations and concepts of the (α, β)-k FSP rather than the problem's definition. Figure 3.1 illustrates this bipartite graph, which has three groups of vertices. In fact, this graph is a combination of two bipartite graphs (the left one includes vertices associated with sets J and I_1, and the right one includes vertices associated with sets J and I_2). The vertices on the left (colored black) represent elements of set I_1, and the vertices on the right (colored gray) represent elements of set I_2. The vertices in the middle (colored white) represent features.

Graph G includes two sets of edges. The first set denotes features covering elements of I_1. This set is $E_1 = \{e_{ij}|a_{ij} = 1, i \in I_1, j \in J\}$. The second set denotes features covering elements of I_2, and is $E_2 = \{e_{ij}|a_{ij} = 1, i \in I_2, j \in J\}$. The set of all edges is $E = E_1 \cup E_2$. Notice that edges are formed based on the values of parameter $a_{ij}, \forall i \in I_1 \cup I_2, j \in J$. Here, the concept of $1 \leq \alpha \leq \alpha^*$ features covering ("explaining") the differences between any pair of entities of different classes (elements of I_1) implies that at least α edges must be adjacent to every black vertex. Equivalently, the degree of every black vertex must be at least α. More precisely, the (α, β)-k FSP can be re-stated as obtaining a set of minimum cost white vertices such that

- every black vertex has a minimum degree of $1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+$; and
- every gray vertex has a minimum degree of $1 \leq \beta \leq \beta^*, \beta \in \mathbb{Z}^+$.

3.4 Illustrative examples

In this section, we discuss two illustrative examples to further elaborate on how the solution of the (α, β)-k FSP may look like. For those examples, we use biological datasets for Down Syndrome (DS) and Alzheimer's Disease (AD). We will later provide technical details of the solution procedure.

Let start by the DS dataset, which was proposed by Lockstone et al. (2007) and contains 73 genes (column entitled "Feature"), and 15 samples with seven cases of DS and eight controls. The size of this dataset makes it easy to understand the operation and outcome of the four-step approach. Next, we explain the outcome of each step.

Step 1 aims to obtain α^* (the largest number of features) such that the dichotomy between each pair of entities of different classes can be explained by at least α^* features. We used Equation (3.3) and obtained $\alpha^* = 50$.

Given α^*, Step 2 delivers k^* (the minimum number of features, and a list of features as a by-product of the solution) such that at least α^* features explain the dichotomy between each pair of entities of different classes. We obtained $k^* = 65$. The column "Min k" in Table 3.3 shows the selected list of features.

Given α^* and k^*, Step 3 delivers a set of features such that the internal consistency (denoted by β) of the entities in the same class is maximized (i.e. it provides a more robust feature set). The column "Max β" in Table 3.3 shows the list of features we obtained. The maximum internal consistency is 51, i.e. $\beta^* = 51$.

Finally, in Step 4 we would like to obtain a list of features that not only satisfies the conditions of α^*, k^*, and β^*, but also provides more explanations in total (denoted by the "coverage" score), either to the differences between the classes or similarity within entities in the same class. The list of such features is shown in column "Max Cover" in Table 3.3, with a maximum coverage score of 5,341.
It is not difficult to see that the optimal solution includes 65 genes, i.e. $k^* = 65$.

Now, let discuss the dataset of Alzheimer's Disease (AD). The dataset was proposed by Paula et al. (2011) and has 683 features, which are (pairs of) proteins, and 83 samples with 43 cases of AD and 40 controls. The dataset is denoted as ADMF.

Following Step 1, we obtain $\alpha^* = 86$. Given $\alpha^* = 86$, Step 2 delivered $k^* = 292$ features, which are listed in column "Min k" in Section 3.4. To obtain β^*, we proceed to Step 3. The

Table 3.3: The selected list of features for dataset DS.

No.	Feature	Min k	Max β	Max Cover	No.	Feature	Min k	Max β	Max Cover
1	DDR1	X	X	X	38	CCND2	X	X	X
2	CYP2E1	X	X	X	39	PRDX2	X	X	X
3	CYP2A6	X	X	X	40	DUSP1	X	X	X
4	RPL28	X	X	X	41	HLA-DPB1	X	X	X
5	SRP14	X	X	X	42	DDX3X	X	X	X
6	RPL11	X	X	X	43	CTBP2	X	X	X
7	DAD1	X	X	X	44	HNRNPAB	X	X	X
8	SPAG7	X	X	X	45	OGDH	X	X	X
9	NONO	X	X	X	46	CUL4A	X	X	X
10	RPS6	X	X	X	47	DDX23	X	X	X
11	TCEB2	X	X	X	48	TIA1	X	X	X
12	RPL4	X	X	X	49	DCTD	X	X	X
13	DSP	X	X	X	50	ICMT	X	X	X
14	WDR1	X	X	X	51	DARS	X	X	X
15	KIAA0152	X	X	X	52	SCARB2	X	X	X
16	SF3B2	X	X	X	53	CCND3	X	X	X
17	MARCKSL1	X	X	X	54	LUM	X	X	X
18	GLUL	X	X	X	55	ALDH3A2	X	X	X
19	GNB2L1	X	X	X	56	VPS72	X	X	X
20	CD63	X	X	X	57	PLSCR1			
21	BG537255	X	X	X	58	PPL	X	X	X
22	RPL32	X	X	X	59	U47924	X	X	X
23	GRN	X	X	X	60	MAP3K11			
24	UBE2L3	X	X	X	61	THBD	X	X	X
25	KDELR2	X	X	X	62	PEX3	X	X	X
26	LITAF	X	X	X	63	EML2	X	X	X
27	RPL13A	X	X	X	64	EIF1AY			
28	ACTR2	X	X	X	65	NEFH	X	X	X
29	LRP1	X	X	X	66	MN1	X	X	X
30	DAZAP2	X	X	X	67	STRN	X	X	X
31	PAFAH1B1	X	X	X	68	CPA4			
32	NCOR1	X	X	X	69	HERC3			
33	S100A10	X	X	X	70	PCTK2			
34	PPM1J	X	X	X	71	USP9Y	X	X	X
35	PHC2	X	X	X	72	CHI3L1			
36	RERE	X	X	X	73	MAST2			
37	HMGN1	X	X	X					

list of features obtained during the step is shown in column "Max β" in Section 3.4, where $\beta^* = 118$. Finally, Step 4 leads to a coverage score of 581,608, and a list of features that is shown in column "Max Cover" in Section 3.4. We should note that because it might be possible that more than one optimal set of features exist with cardinality 292 is obtained during Step 2, Steps 3 and 4 aim to select a set, from such optimal sets, such that β and "coverage" are maximized. Therefore, Steps 2, 3 and 4 produce different sets of features, however, each with $\alpha^* = 86$, $k^* = 292$, and $\beta^* = 118$.

3.5 Mathematical models

The mathematical models for the (α, β)-k Feature Set Problem (FSP) were originally developed in previous studies; see for example, Berretta et al. (2005) and Paula (2012). Following the four-step approach explained in the beginning of this chapter, we discuss mathematical models for every step, except for the first step because as we showed in Equation (3.3), α^* can easily be obtained, and without solving an optimization problem. The mathematical models for Steps 2, 3, and 4 are integer programs (IPs). We discuss these in Sections 3.5.1 to 3.5.3.

3.5.1 An integer program for the Min k (α, β)-k Feature Set Problem

Recall that Step 2 of the four-step approach determines k^*, which is the minimum cost for a set of features. More precisely, Step 2 obtains a minimum cost set of features (among alternative minimum cost sets of features) that explains the dichotomy between the classes, considering that at least α^* features do so for each pair of entities of different classes (elements of I_1). Paula (2012) calls the associated optimization problem with Step 2 the Min k (α, β)-k Feature Set Problem (FSP). This problem can mathematically be modeled as an integer program (IP), where α^* (obtained in Step 1) is a parameter. Model $\mathcal{IP}_{\mathcal{MCFSP}}$ discusses this. The model has one set of binary decision variables: x_j, which takes 1 if feature j is selected, and 0 otherwise.

Model $\mathcal{IP}_{\mathcal{MCFSP}}$

$$z = \min \sum_{j \in J} c_j x_j \tag{3.4}$$

$$\sum_{j \in J} a_{ij} x_j \geq \alpha, \forall i \in I_1, 1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+ \tag{3.5}$$

$$x_j \in \{0, 1\}, \forall j \in J \tag{3.6}$$

The objective function (Equation (3.4)) minimizes the cost of selecting features. Equation (3.5) ensures every element of set I_1 is covered by at least α features. This implies that

Table 3.4: The selected list of features for dataset ADMF.

No.	Feature	Min k	Max β	Max Cover	No.	Feature	Min k	Max β	Max Cover
1	BMP-6_1		X	X	61	CNTF_1'-'PDGF-BB_1			
2	EGF_1				62	CNTF_1'-'TNF-a_1			
3	IL-1a_1				63	CNTF_1'-'ENA-78_1			
4	IL-3_1	X	X	X	64	CNTF_1'-'IL-8_1			
5	IL-6_1				65	EGF_1'-'GCP-2_1			
6	MCP-3_1	X	X	X	66	EGF_1'-'GM-CSF_1	X	X	X
7	MIP-1d_1	X	X	X	67	EGF_1'-'IGFBP-2_1	X	X	X
8	PDGF-BB_1				68	EGF_1'-'IL-15_1	X	X	X
9	RANTES_1				69	EGF_1'-'NT-3_1			
10	TNF-a_1				70	EGF_1'-'TNF-b_1	X		
11	GCSF_1				71	EGF_1'-'AgRP(ART)_1			
12	IL-11_1	X	X	X	72	EGF_1'-'ANG-2_1			
13	ANG_1'-'EGF_1	X	X	X	73	EGF_1'-'AR_1			
14	ANG_1'-'IL-1a_1	X	X	X	74	EGF_1'-'AXL_1	X	X	X
15	ANG_1'-'RANTES_1	X	X	X	75	EGF_1'-'bFGF	X	X	X
16	BDNF_1'-'Eotaxin-3_1				76	EGF_1'-'BTC_1	X	X	X
17	BDNF_1'-'IL-1a_1				77	EGF_1'-'DTK_1	X	X	X
18	BDNF_1'-'IL-3_1	X	X	X	78	EGF_1'-'EGF-R_1	X	X	X
19	BDNF_1'-'PDGF-BB_1	X	X	X	79	EGF_1'-'FAS_1	X	X	X
20	BDNF_1'-'SCF_1				80	EGF_1'-'FGF-9_1	X	X	X
21	BDNF_1'-'GCSF_1	X	X	X	81	EGF_1'-'GITR_1	X	X	X
22	BLC_1'-'EGF_1				82	EGF_1'-'GRO_1	X	X	X
23	BLC_1'-'Eotaxin_1				83	EGF_1'-'GRO-a_1	X	X	X
24	BLC_1'-'GDNF_1				84	EGF_1'-'ICAM-1_1	X	X	X
25	BLC_1'-'IL-1a_1	X	X	X	85	EGF_1'-'IGF-1_SR	X	X	X
26	BLC_1'-'IL-3_1	X	X	X	86	EGF_1'-'IGFBP3_1	X	X	X
27	BLC_1'-'IL-4_1	X	X	X	87	EGF_1'-'IGFBP-6_1	X	X	X
28	BLC_1'-'MCP-3_1				88	EGF_1'-'IL-1_RI_1			
29	BLC_1'-'M-CSF_1				89	EGF_1'-'IL-11_1	X	X	X
30	BLC_1'-'MDC_1	X	X	X	90	EGF_1'-'IL-12_p40_1	X	X	X
31	BLC_1'-'PDGF-BB_1				91	EGF_1'-'IL-12_p70_1	X	X	X
32	BLC_1'-'RANTES_1	X	X	X	92	EGF_1'-'IL-2_Ra_1	X	X	X
33	BLC_1'-'TNF-a_1				93	EGF_1'-'IL-6_R_1	X	X	X
34	BLC_1'-'BTC_1				94	EGF_1'-'IL-8_1	X	X	X
35	BMP-4_1'-'EGF_1		X	X	95	EGF_1'-'Lymphotactin_1	X	X	X
36	BMP-4_1'-'GDNF_1				96	EGF_1'-'MIF_1	X	X	X
37	BMP-4_1'-'IGF-1_1				97	EGF_1'-'MIP-1a_1	X	X	X
38	BMP-4_1'-'IL-1a_1	X	X	X	98	EGF_1'-'MIP-1b_1			
39	BMP-4_1'-'LEPTIN(OB)_1	X	X	X	99	EGF_1'-'MIP-3b_1	X	X	X
40	BMP-4_1'-'PDGF-BB_1	X	X	X	100	EGF_1'-'NT-4_1	X	X	X
41	BMP-4_1'-'TNF-a_1				101	EGF_1'-'OSM_1	X	X	X
42	BMP-6_1'-'EGF_1	X	X	X	102	EGF_1'-'OST_1	X	X	X
43	BMP-6_1'-'IL-1a_1	X	X	X	103	EGF_1'-'PIGF_1			
44	BMP-6_1'-'NT-3_1		X	X	104	EGF_1'-'sTNF_RI_1	X	X	X
45	BMP-6_1'-'PDGF-BB_1	X	X	X	105	EGF_1'-'TPO_1	X	X	X
46	BMP-6_1'-'BTC_1				106	EGF_1'-'TRAIL_R3_1	X	X	X
47	BMP-6_1'-'IL-11_1	X	X	X	107	EGF_1'-'TRAIL_R4_1	X	X	X
48	CK_b8-1_1'-'EGF_1				108	EGF_1'-'uPAR_1			
49	CK_b8-1_1'-'Eotaxin-3_1				109	EGF_1'-'VEGF-B_1	X	X	X
50	CK_b8-1_1'-'Fractalkine_1	X	X	X	110	EGF_1'-'VEGF-D_1	X	X	X
51	CK_b8-1_1'-'GDNF_1	X			111	Eotaxin_1'-'NT-3_1	X	X	X
52	CK_b8-1_1'-'IL-10_1				112	Eotaxin-2_1'-'IL-1a_1	X	X	X
53	CK_b8-1_1'-'IL-1a_1		X	X	113	Eotaxin-2_1'-'NT-3_1			
54	CK_b8-1_1'-'MCP-3_1		X	X	114	Eotaxin-2_1'-'BTC_1	X	X	X
55	CK_b8-1_1'-'M-CSF_1	X	X	X	115	Eotaxin-2_1'-'IL-11_1	X		
56	CK_b8-1_1'-'PDGF-BB_1	X	X	X	116	Eotaxin-2_1'-'IL-6_R_1	X	X	X
57	CK_b8-1_1'-'TNF-a_1				117	Eotaxin-2_1'-'MIF_1			
58	CK_b8-1_1'-'ICAM-3_1	X	X	X	118	Eotaxin-3_1'-'IGFBP-2_1			
59	CNTF_1'-'IL-1a_1	X	X	X	119	Eotaxin-3_1'-'IL-10_1			
60	CNTF_1'-'NT-3_1	X	X	X	120	Eotaxin-3_1'-'IL-1a_1	X	X	X

No.	Feature	Min k	Max β	Max Cover
121	Eotaxin-3_1'-'M-CSF_1	X		
122	Eotaxin-3_1'-'NT-3_1	X	X	X
123	Eotaxin-3_1'-'TNF-a_1			
124	Eotaxin-3_1'-'ANG-2_1			
125	Eotaxin-3_1'-'IL-11_1	X	X	X
126	Eotaxin-3_1'-'TRAIL_R4_1			
127	FGF-6_1'-'NT-3_1	X	X	X
128	FGF-6_1'-'RANTES_1		X	X
129	FGF-6_1'-'TNF-a_1			
130	FGF-6_1'-'AgRP(ART)_1	X	X	X
131	FGF-6_1'-'ANG-2_1	X	X	X
132	FGF-6_1'-'AXL_1			
133	FGF-6_1'-'FAS_1			
134	FGF-6_1'-'IL-1_RI_1	X	X	X
135	FGF-6_1'-'IL-11_1	X	X	X
136	FGF-6_1'-'IL-8_1		X	X
137	FGF-6_1'-'TPO_1			
138	FGF-6_1'-'TRAIL_R4_1			
139	FGF-6_1'-'uPAR_1	X	X	X
140	FGF-7_1'-'IGFBP-2_1			
141	FGF-7_1'-'IL-1a_1	X	X	X
142	FGF-7_1'-'IL-3_1	X	X	X
143	FGF-7_1'-'IL-6_1	X	X	X
144	FGF-7_1'-'M-CSF_1			
145	FGF-7_1'-'MIG_1			
146	FGF-7_1'-'NT-3_1	X	X	X
147	FGF-7_1'-'PDGF-BB_1			
148	FGF-7_1'-'RANTES_1			
149	FGF-7_1'-'TNF-a_1			
150	Fit-3_Ligand_1'-'GDNF_1			
151	Fit-3_Ligand_1'-'IL-1a_1			
152	Fit-3_Ligand_1'-'NT-3_1	X	X	X
153	Fit-3_Ligand_1'-'TNF-a_1			
154	Fit-3_Ligand_1'-'ANG-2_1	X	X	X
155	Fit-3_Ligand_1'-'FAS_1	X	X	X
156	Fractalkine_1'-'IL-10_1			
157	Fractalkine_1'-'IL-1a_1			
158	Fractalkine_1'-'M-CSF_1			
159	Fractalkine_1'-'TNF-a_1	X	X	X
160	GCP-2_1'-'IGFBP-2_1			
161	GCP-2_1'-'IL-10_1			
162	GCP-2_1'-'IL-1a_1		X	X
163	GCP-2_1'-'NT-3_1			
164	GCP-2_1'-'PDGF-BB_1			
165	GCP-2_1'-'TNF-a_1	X	X	X
166	GCP-2_1'-'FAS_1			
167	GDNF_1'-'IGFBP-2_1	X	X	X
168	GDNF_1'-'IL-1b_1			
169	GDNF_1'-'IL-1ra_1			
170	GDNF_1'-'IL-3_1	X	X	X
171	GDNF_1'-'MIG_1			
172	GDNF_1'-'NT-3_1	X	X	X
173	GDNF_1'-'PDGF-BB_1			
174	GDNF_1'-'SCF_1			
175	GDNF_1'-'TNF-b_1			
176	GDNF_1'-'ANG-2_1	X	X	X
177	GDNF_1'-'AR_1			
178	GDNF_1'-'AXL_1			
179	GDNF_1'-'BTC_1	X	X	X
180	GDNF_1'-'DTK_1	X	X	X
181	GDNF_1'-'ENA-78_1	X	X	X
182	GDNF_1'-'FAS_1		X	
183	GDNF_1'-'ICAM-1_1	X	X	X
184	GDNF_1'-'IL-1_RI_1	X	X	X
185	GDNF_1'-'IL-11_1	X	X	X
186	GDNF_1'-'IL-2_Ra_1	X	X	X
187	GDNF_1'-'IL-6_R_1	X	X	X
188	GDNF_1'-'MIP-1a_1			
189	GDNF_1'-'MIP-3b_1			
190	GDNF_1'-'OSM_1	X	X	X
191	GDNF_1'-'OST_1			
192	GDNF_1'-'PIGF_1			
193	GDNF_1'-'TRAIL_R4_1			
194	GDNF_1'-'VEGF-B_1			
195	GDNF_1'-'VEGF-D_1			
196	GM-CSF_1'-'I-309_1			
197	GM-CSF_1'-'IL-16_1			
198	GM-CSF_1'-'IL-1a_1			
199	GM-CSF_1'-'IL-3_1	X	X	X
200	GM-CSF_1'-'IL-6_1			
201	GM-CSF_1'-'M-CSF_1			
202	GM-CSF_1'-'PDGF-BB_1			
203	GM-CSF_1'-'TGF-b_1	X	X	X
204	GM-CSF_1'-'TNF-a_1	X	X	X
205	I-309_1'-'IL-1a_1			
206	I-309_1'-'BTC_1			
207	I-309_1'-'IL-11_1	X	X	X
208	I-309_1'-'TRAIL_R4_1			
209	IFN-g_1'-'IL-1a_1			
210	IFN-g_1'-'M-CSF_1		X	X
211	IFN-g_1'-'MIP-1d_1	X	X	X
212	IFN-g_1'-'PDGF-BB_1	X	X	X
213	IFN-g_1'-'TNF-a_1	X	X	X
214	IFN-g_1'-'ANG-2_1			
215	IGF-1_1'-'IL-10_1			
216	IGF-1_1'-'IL-1a_1	X	X	X
217	IGF-1_1'-'PDGF-BB_1	X		
218	IGF-1_1'-'ANG-2_1			
219	IGFBP-1_1'-'ICAM-1_1	X	X	X
220	IGFBP-2_1'-'IL-10_1	X	X	X
221	IGFBP-2_1'-'IL-13_1			
222	IGFBP-2_1'-'IL-16_1			
223	IGFBP-2_1'-'IL-1a_1	X	X	X
224	IGFBP-2_1'-'IL-3_1			
225	IGFBP-2_1'-'IL-6_1			
226	IGFBP-2_1'-'M-CSF_1			
227	IGFBP-2_1'-'NAP-2_1			
228	IGFBP-2_1'-'PDGF-BB_1	X	X	X
229	IGFBP-2_1'-'DTK_1			
230	IGFBP-2_1'-'ICAM-3_1			
231	IGFBP-4_1'-'IL-1a_1	X	X	X
232	IGFBP-4_1'-'PDGF-BB_1	X	X	X
233	IGFBP-4_1'-'ANG-2_1	X	X	X
234	IL-10_1'-'LEPTIN(OB)_1			
235	IL-10_1'-'MIG_1		X	X
236	IL-10_1'-'NT-3_1	X	X	X
237	IL-10_1'-'TGF-b3_1			
238	IL-10_1'-'IL-11_1	X	X	X
239	IL-10_1'-'TPO_1	X	X	X
240	IL-10_1'-'TRAIL_R4_1			

No.	Feature	Min k	Max β	Max Cover	No.	Feature	Min k	Max β	Max Cover
241	IL-10_1'-'uPAR_1				301	IL-1a_1'-'IGF-1_SR			
242	IL-13_1'-'IL-15_1				302	IL-1a_1'-'IGFBP3_1			
243	IL-13_1'-'IL-1a_1	X	X	X	303	IL-1a_1'-'IGFBP-6_1	X	X	X
244	IL-13_1'-'MCP-4_1				304	IL-1a_1'-'IL-1_RI_1			
245	IL-13_1'-'NT-3_1				305	IL-1a_1'-'IL-11_1			
246	IL-13_1'-'PDGF-BB_1				306	IL-1a_1'-'IL-12_p40_1	X	X	X
247	IL-13_1'-'TNF-a_1				307	IL-1a_1'-'IL-12_p70_1			
248	IL-13_1'-'IL-11_1	X	X	X	308	IL-1a_1'-'IL-17_1			
249	IL-15_1'-'IL-1a_1				309	IL-1a_1'-'IL-1R4_/ST2_1			X
250	IL-15_1'-'IL-3_1				310	IL-1a_1'-'IL-2_Ra_1	X		
251	IL-15_1'-'IL-6_1				311	IL-1a_1'-'IL-8_1			
252	IL-15_1'-'M-CSF_1				312	IL-1a_1'-'MIF_1			
253	IL-15_1'-'TNF-a_1				313	IL-1a_1'-'MIP-1a_1			
254	IL-15_1'-'ANG-2_1	X	X	X	314	IL-1a_1'-'MIP-1b_1	X	X	X
255	IL-15_1'-'FAS_1				315	IL-1a_1'-'MIP-3b_1			
256	IL-16_1'-'IL-1a_1				316	IL-1a_1'-'MSP-a_1	X	X	X
257	IL-16_1'-'IL-3_1	X	X	X	317	IL-1a_1'-'NT-4_1	X	X	X
258	IL-16_1'-'NT-3_1	X	X	X	318	IL-1a_1'-'OSM_1			
259	IL-16_1'-'PDGF-BB_1				319	IL-1a_1'-'OST_1			
260	IL-16_1'-'TNF-a_1				320	IL-1a_1'-'PIGF_1			
261	IL-16_1'-'IL-11_1				321	IL-1a_1'-'sTNF_RI_1	X	X	X
262	IL-1a_1'-'IL-1b_1				322	IL-1a_1'-'TPO_1			
263	IL-1a_1'-'IL-1ra_1				323	IL-1a_1'-'TRAIL_R4_1			
264	IL-1a_1'-'IL-2_1				324	IL-1a_1'-'uPAR_1			
265	IL-1a_1'-'IL-4_1		X	X	325	IL-1a_1'-'VEGF-B_1			
266	IL-1a_1'-'IL-5_1	X	X	X	326	IL-1a_1'-'VEGF-D_1			
267	IL-1a_1'-'IL-6_1				327	IL-1b_1'-'IL-3_1	X	X	X
268	IL-1a_1'-'IL-7_1	X	X	X	328	IL-1b_1'-'M-CSF_1	X	X	X
269	IL-1a_1'-'LIGHT_1				329	IL-1b_1'-'MDC_1			
270	IL-1a_1'-'MCP-1_1				330	IL-1b_1'-'PDGF-BB_1			
271	IL-1a_1'-'MCP-2_1				331	IL-1b_1'-'TNF-a_1			
272	IL-1a_1'-'MCP-3_1	X	X	X	332	IL-1b_1'-'BTC_1			
273	IL-1a_1'-'MCP-4_1				333	IL-1ra_1'-'M-CSF_1			
274	IL-1a_1'-'MDC_1	X	X	X	334	IL-1ra_1'-'TNF-a_1			
275	IL-1a_1'-'MIG_1				335	IL-2_1'-'IL-3_1	X	X	X
276	IL-1a_1'-'MIP-3a_1	X	X	X	336	IL-2_1'-'M-CSF_1	X		
277	IL-1a_1'-'NT-3_1	X	X	X	337	IL-2_1'-'TNF-a_1			
278	IL-1a_1'-'PARC_1				338	IL-2_1'-'FAS_1			
279	IL-1a_1'-'SDF-1_1				339	IL-2_1'-'IL-11_1	X	X	X
280	IL-1a_1'-'TARC_1				340	IL-3_1'-'NT-3_1			
281	IL-1a_1'-'TGF-b3_1	X	X	X	341	IL-3_1'-'ANG-2_1	X	X	X
282	IL-1a_1'-'TNF-b_1				342	IL-3_1'-'AXL_1			
283	IL-1a_1'-'AgRP(ART)_1	X	X	X	343	IL-3_1'-'ENA-78_1			
284	IL-1a_1'-'ANG-2_1	X	X	X	344	IL-3_1'-'FAS_1			
285	IL-1a_1'-'AR_1				345	IL-3_1'-'FGF-9_1			
286	IL-1a_1'-'AXL_1	X	X	X	346	IL-3_1'-'GITR-Light_1			
287	IL-1a_1'-'BTC_1				347	IL-3_1'-'HGF_1			
288	IL-1a_1'-'CCL-28_1				348	IL-3_1'-'ICAM-1_1			
289	IL-1a_1'-'CTACK_1				349	IL-3_1'-'IGF-1_SR			
290	IL-1a_1'-'DTK_1	X	X	X	350	IL-3_1'-'IL-1_RI_1			
291	IL-1a_1'-'EGF-R_1				351	IL-3_1'-'IL-11_1	X		
292	IL-1a_1'-'ENA-78_1	X	X	X	352	IL-3_1'-'IL-17_1			
293	IL-1a_1'-'FAS_1				353	IL-3_1'-'IL-1R4_/ST2_1			
294	IL-1a_1'-'FGF-9_1				354	IL-3_1'-'IL-8_1			
295	IL-1a_1'-'GITR_1	X	X	X	355	IL-3_1'-'I-TAC_1			
296	IL-1a_1'-'GITR-Light_1	X	X	X	356	IL-3_1'-'MIF_1	X	X	X
297	IL-1a_1'-'GRO_1				357	IL-3_1'-'MIP-1a_1			
298	IL-1a_1'-'GRO-a_1				358	IL-3_1'-'MIP-3b_1	X	X	X
299	IL-1a_1'-'HGF_1				359	IL-3_1'-'NT-4_1			
300	IL-1a_1'-'ICAM-1_1				360	IL-3_1'-'TPO_1			

No.	Feature	Min k	Max β	Max Cover	No.	Feature	Min k	Max β	Max Cover
361	IL-3_1'-'TRAIL_R4_1				421	M-CSF_1'-'ANG-2_1			
362	IL-3_1'-'VEGF-D_1	X	X	X	422	M-CSF_1'-'AXL_1			
363	IL-4_1'-'PDGF-BB_1				423	M-CSF_1'-'bFGF	X	X	X
364	IL-4_1'-'ANG-2_1				424	M-CSF_1'-'FAS_1			
365	IL-4_1'-'TRAIL_R4_1				425	M-CSF_1'-'FGF-9_1	X	X	X
366	IL-5_1'-'NT-3_1	X	X	X	426	M-CSF_1'-'GRO_1	X	X	X
367	IL-5_1'-'FAS_1				427	M-CSF_1'-'ICAM-1_1	X	X	X
368	IL-5_1'-'IL-11_1				428	M-CSF_1'-'IGFBP-6_1	X	X	X
369	IL-5_1'-'TRAIL_R4_1				429	M-CSF_1'-'IL-1_RI_1			
370	IL-6_1'-'M-CSF_1				430	M-CSF_1'-'IL-11_1			
371	IL-6_1'-'NT-3_1	X	X	X	431	M-CSF_1'-'IL-8_1	X	X	X
372	IL-6_1'-'TNF-a_1				432	M-CSF_1'-'MIF_1			
373	IL-6_1'-'TNF-b_1	X	X		433	M-CSF_1'-'sTNF_RI_1			
374	IL-6_1'-'DTK_1				434	M-CSF_1'-'TPO_1			
375	IL-6_1'-'ENA-78_1				435	M-CSF_1'-'TRAIL_R4_1			
376	IL-6_1'-'FAS_1				436	M-CSF_1'-'uPAR_1	X	X	X
377	IL-6_1'-'FGF-9_1				437	M-CSF_1'-'VEGF-B_1			
378	IL-6_1'-'GITR-Light_1				438	M-CSF_1'-'VEGF-D_1			
379	IL-6_1'-'GRO-a_1				439	MDC_1'-'NT-3_1	X	X	X
380	IL-6_1'-'IL-1_RI_1				440	MDC_1'-'TNF-a_1			
381	IL-6_1'-'IL-11_1	X	X	X	441	MDC_1'-'ANG-2_1			
382	IL-7_1'-'TNF-a_1				442	MDC_1'-'bFGF			
383	IL-7_1'-'FAS_1				443	MDC_1'-'GRO-a_1			
384	IL-7_1'-'GCSF_1				444	MDC_1'-'IGF-1_SR			
385	IL-7_1'-'IL-12_p40_1				445	MDC_1'-'IL-1_RI_1			
386	IL-7_1'-'TRAIL_R4_1				446	MDC_1'-'IL-11_1			
387	LEPTIN(OB)_1'-'M-CSF_1				447	MDC_1'-'IL-1R4_/ST2_1			
388	LEPTIN(OB)_1'-'ANG-2_1	X	X	X	448	MIG_1'-'PDGF-BB_1			
389	LIGHT_1'-'TNF-a_1				449	MIG_1'-'TNF-a_1	X	X	X
390	LIGHT_1'-'ENA-78_1				450	MIG_1'-'ANG-2_1			
391	MCP-1_1'-'M-CSF_1	X	X	X	451	MIG_1'-'GCSF_1	X	X	X
392	MCP-1_1'-'NT-3_1	X	X	X	452	MIP-1d_1'-'NT-3_1	X	X	X
393	MCP-1_1'-'PDGF-BB_1				453	MIP-1d_1'-'TNF-a_1			
394	MCP-1_1'-'TNF-a_1				454	MIP-1d_1'-'ANG-2_1	X	X	X
395	MCP-2_1'-'NT-3_1	X	X	X	455	MIP-1d_1'-'BTC_1	X	X	X
396	MCP-2_1'-'RANTES_1	X	X	X	456	MIP-1d_1'-'ICAM-1_1	X	X	X
397	MCP-2_1'-'FAS_1	X	X	X	457	MIP-1d_1'-'IL-11_1	X	X	X
398	MCP-3_1'-'NT-3_1	X	X	X	458	MIP-1d_1'-'IL-12_p40_1	X	X	X
399	MCP-3_1'-'TARC_1				459	MIP-1d_1'-'MIF_1	X	X	X
400	MCP-3_1'-'TNF-a_1				460	MIP-1d_1'-'TRAIL_R4_1			
401	MCP-3_1'-'ANG-2_1	X	X	X	461	MIP-1d_1'-'uPAR_1	X	X	X
402	MCP-3_1'-'BTC_1	X	X	X	462	MIP-3a_1'-'PDGF-BB_1			
403	MCP-3_1'-'EGF-R_1				463	MIP-3a_1'-'TNF-a_1			
404	MCP-3_1'-'ENA-78_1				464	MIP-3a_1'-'ANG-2_1			
405	MCP-3_1'-'FAS_1				465	MIP-3a_1'-'AXL_1			
406	MCP-3_1'-'IL-11_1	X	X	X	466	MIP-3a_1'-'bFGF			
407	MCP-3_1'-'TRAIL_R4_1				467	MIP-3a_1'-'TRAIL_R4_1	X	X	X
408	MCP-3_1'-'VEGF-D_1	X	X	X	468	NAP-2_1'-'PDGF-BB_1	X	X	X
409	MCP-4_1'-'NT-3_1	X	X	X	469	NT-3_1'-'PDGF-BB_1			
410	MCP-4_1'-'TNF-a_1				470	NT-3_1'-'RANTES_1			
411	MCP-4_1'-'BTC_1				471	NT-3_1'-'SDF-1_1	X	X	X
412	MCP-4_1'-'ENA-78_1				472	NT-3_1'-'TGF-b_1	X	X	X
413	MCP-4_1'-'FAS_1				473	NT-3_1'-'TNF-a_1			
414	MCP-4_1'-'IL-11_1				474	NT-3_1'-'ICAM-3_1	X	X	X
415	MCP-4_1'-'TRAIL_R4_1	X	X	X	475	PARC_1'-'PDGF-BB_1	X	X	X
416	M-CSF_1'-'MIG_1	X	X	X	476	PARC_1'-'RANTES_1			
417	M-CSF_1'-'MIP-3a_1				477	PARC_1'-'TNF-a_1	X	X	X
418	M-CSF_1'-'NT-3_1				478	PARC_1'-'GCSF_1	X	X	X
419	M-CSF_1'-'TARC_1				479	PDGF-BB_1'-'SCF_1			
420	M-CSF_1'-'AgRP(ART)_1	X	X	X	480	PDGF-BB_1'-'SDF-1_1			

No.	Feature	Min k	Max β	Max Cover
481	PDGF-BB_1'-'TARC_1			
482	PDGF-BB_1'-'TGF-b_1			
483	PDGF-BB_1'-'TNF-b_1	X	X	X
484	PDGF-BB_1'-'AgRP(ART)_1	X	X	X
485	PDGF-BB_1'-'ANG-2_1	X	X	X
486	PDGF-BB_1'-'AR_1			
487	PDGF-BB_1'-'AXL_1			
488	PDGF-BB_1'-'bFGF			
489	PDGF-BB_1'-'BTC_1	X	X	X
490	PDGF-BB_1'-'CCL-28_1	X		
491	PDGF-BB_1'-'CTACK_1			
492	PDGF-BB_1'-'DTK_1			X
493	PDGF-BB_1'-'EGF-R_1			
494	PDGF-BB_1'-'ENA-78_1			
495	PDGF-BB_1'-'FAS_1	X	X	X
496	PDGF-BB_1'-'FGF-4_1			
497	PDGF-BB_1'-'FGF-9_1			
498	PDGF-BB_1'-'GITR_1			
499	PDGF-BB_1'-'GITR-Light_1			
500	PDGF-BB_1'-'GRO_1	X	X	X
501	PDGF-BB_1'-'GRO-a_1			
502	PDGF-BB_1'-'HCC-4_1			
503	PDGF-BB_1'-'HGF_1			
504	PDGF-BB_1'-'ICAM-1_1	X		
505	PDGF-BB_1'-'IGF-1_SR			
506	PDGF-BB_1'-'IGFBP3_1			
507	PDGF-BB_1'-'IGFBP-6_1			
508	PDGF-BB_1'-'IL-1_RI_1			
509	PDGF-BB_1'-'IL-11_1			
510	PDGF-BB_1'-'IL-12_p40_1			
511	PDGF-BB_1'-'IL-17_1	X	X	X
512	PDGF-BB_1'-'IL-2_Ra_1			
513	PDGF-BB_1'-'IL-6_R_1	X	X	X
514	PDGF-BB_1'-'IL-8_1			
515	PDGF-BB_1'-'Lymphotactin_1			
516	PDGF-BB_1'-'MIF_1			
517	PDGF-BB_1'-'MIP-1a_1			
518	PDGF-BB_1'-'MIP-1b_1			
519	PDGF-BB_1'-'MIP-3b_1	X	X	X
520	PDGF-BB_1'-'MSP-a_1			
521	PDGF-BB_1'-'OSM_1	X	X	X
522	PDGF-BB_1'-'OST_1			
523	PDGF-BB_1'-'PIGF_1			
524	PDGF-BB_1'-'spg130_1			
525	PDGF-BB_1'-'sTNF_RI_1			
526	PDGF-BB_1'-'TIMP-1_1			X
527	PDGF-BB_1'-'TPO_1	X	X	X
528	PDGF-BB_1'-'TRAIL_R3_1			
529	PDGF-BB_1'-'TRAIL_R4_1	X		
530	PDGF-BB_1'-'uPAR_1			
531	PDGF-BB_1'-'VEGF-B_1	X	X	X
532	PDGF-BB_1'-'VEGF-D_1			
533	RANTES_1'-'SDF-1_1			
534	RANTES_1'-'TARC_1			
535	RANTES_1'-'TNF-b_1	X	X	X
536	RANTES_1'-'ANG-2_1	X		
537	RANTES_1'-'AR_1	X	X	X
538	RANTES_1'-'BTC_1	X	X	X
539	RANTES_1'-'EGF-R_1	X	X	X
540	RANTES_1'-'ENA-78_1	X	X	X
541	RANTES_1'-'FAS_1		X	
542	RANTES_1'-'FGF-4_1	X	X	X
543	RANTES_1'-'FGF-9_1		X	X
544	RANTES_1'-'GRO_1	X	X	X
545	RANTES_1'-'GRO-a_1		X	X
546	RANTES_1'-'HCC-4_1	X	X	X
547	RANTES_1'-'HGF_1	X	X	X
548	RANTES_1'-'ICAM-1_1			
549	RANTES_1'-'IGF-1_SR	X	X	X
550	RANTES_1'-'IGFBP3_1	X	X	X
551	RANTES_1'-'IGFBP-6_1	X	X	X
552	RANTES_1'-'IL-1_RI_1			
553	RANTES_1'-'IL-11_1		X	X
554	RANTES_1'-'IL-12_p40_1	X	X	X
555	RANTES_1'-'IL-17_1	X		
556	RANTES_1'-'IL-1R4_/ST2_1	X	X	X
557	RANTES_1'-'IL-2_Ra_1	X	X	X
558	RANTES_1'-'IL-6_R_1			
559	RANTES_1'-'IL-8_1	X	X	X
560	RANTES_1'-'MIF_1	X	X	X
561	RANTES_1'-'MIP-3b_1	X	X	X
562	RANTES_1'-'OSM_1	X		
563	RANTES_1'-'OST_1			
564	RANTES_1'-'PIGF_1	X		X
565	RANTES_1'-'TPO_1			
566	RANTES_1'-'TRAIL_R3_1	X	X	X
567	RANTES_1'-'TRAIL_R4_1	X	X	X
568	RANTES_1'-'uPAR_1	X	X	X
569	RANTES_1'-'VEGF-D_1	X	X	X
570	SCF_1'-'TNF-a_1			
571	SCF_1'-'ENA-78_1			
572	SCF_1'-'GITR-Light_1			
573	SCF_1'-'IL-11_1			
574	SCF_1'-'VEGF-D_1			
575	SDF-1_1'-'TNF-a_1			
576	SDF-1_1'-'ANG-2_1	X	X	X
577	SDF-1_1'-'BTC_1			
578	SDF-1_1'-'FAS_1	X	X	X
579	SDF-1_1'-'IL-11_1	X	X	X
580	SDF-1_1'-'MIP-1a_1			
581	SDF-1_1'-'MIP-3b_1			
582	SDF-1_1'-'VEGF-D_1	X	X	X
583	TARC_1'-'TNF-a_1			
584	TARC_1'-'FAS_1			
585	TARC_1'-'GCSF_1			
586	TARC_1'-'IL-11_1	X	X	X
587	TGF-b_1'-'TNF-a_1			
588	TGF-b_1'-'ANG-2_1	X	X	X
589	TGF-b_1'-'AXL_1	X	X	X
590	TGF-b_1'-'FAS_1	X	X	X
591	TGF-b_1'-'IL-11_1	X	X	X
592	TGF-b_1'-'IL-12_p40_1	X	X	X
593	TGF-b_1'-'TPO_1	X	X	X
594	TGF-b_1'-'TRAIL_R4_1			
595	TGF-b_1'-'VEGF-B_1	X	X	X
596	TGF-b3_1'-'TNF-a_1			
597	TGF-b3_1'-'AXL_1			
598	TNF-a_1'-'TNF-b_1			
599	TNF-a_1'-'ANG-2_1	X	X	X
600	TNF-a_1'-'AR_1			

No.	Feature	Min k	Max β	Max Cover
601	TNF-a_1'-'AXL_1			
602	TNF-a_1'-'bFGF	X	X	X
603	TNF-a_1'-'CTACK_1		X	
604	TNF-a_1'-'DTK_1	X	X	X
605	TNF-a_1'-'EGF-R_1			
606	TNF-a_1'-'ENA-78_1			
607	TNF-a_1'-'FAS_1	X		
608	TNF-a_1'-'FGF-4_1			
609	TNF-a_1'-'FGF-9_1			
610	TNF-a_1'-'GITR_1			
611	TNF-a_1'-'GRO_1			
612	TNF-a_1'-'GRO-a_1	X	X	
613	TNF-a_1'-'HCC-4_1			
614	TNF-a_1'-'HGF_1			
615	TNF-a_1'-'ICAM-1_1	X	X	X
616	TNF-a_1'-'ICAM-3_1	X	X	X
617	TNF-a_1'-'IGF-1_SR			
618	TNF-a_1'-'IGFBP3_1	X	X	X
619	TNF-a_1'-'IGFBP-6_1			
620	TNF-a_1'-'IL-1_RI_1			
621	TNF-a_1'-'IL-11_1			
622	TNF-a_1'-'IL-12_p70_1			
623	TNF-a_1'-'IL-17_1	X	X	X
624	TNF-a_1'-'IL-1R4_/ST2_1	X		
625	TNF-a_1'-'IL-2_Ra_1	X	X	X
626	TNF-a_1'-'IL-6_R_1	X	X	X
627	TNF-a_1'-'IL-8_1			
628	TNF-a_1'-'Lymphotactin_1			
629	TNF-a_1'-'MIF_1			
630	TNF-a_1'-'MIP-1a_1	X	X	X
631	TNF-a_1'-'MIP-1b_1	X	X	X
632	TNF-a_1'-'MIP-3b_1			
633	TNF-a_1'-'NT-4_1			
634	TNF-a_1'-'OST_1	X	X	X
635	TNF-a_1'-'sTNF_RII_1	X	X	X
636	TNF-a_1'-'TPO_1			
637	TNF-a_1'-'TRAIL_R3_1			
638	TNF-a_1'-'TRAIL_R4_1			
639	TNF-a_1'-'uPAR_1	X	X	X
640	TNF-a_1'-'VEGF-B_1			
641	TNF-a_1'-'VEGF-D_1			X
642	TNF-b_1'-'IL-12_p40_1			
643	ANG-2_1'-'GCSF_1	X	X	X
644	ANG-2_1'-'I-TAC_1	X	X	X
645	ANG-2_1'-'TECK_1	X	X	X
646	AR_1'-'GITR-Light_1			
647	AXL_1'-'GRO-a_1			
648	b-NGF_1'-'BTC_1			
649	BTC_1'-'IL-12_p70_1	X	X	X
650	CCL-28_1'-'IGFBP3_1			
651	CTACK_1'-'ICAM-1_1			
652	EGF-R_1'-'GCSF_1			
653	ENA-78_1'-'IL-12_p70_1			
654	FAS_1'-'GCSF_1			
655	FAS_1'-'IL-17_1			
656	FAS_1'-'IL-1R4_/ST2_1			
657	GCSF_1'-'GITR_1	X	X	X
658	GCSF_1'-'GRO_1			
659	GCSF_1'-'GRO-a_1			
660	GCSF_1'-'HGF_1			
661	GCSF_1'-'IGF-1_SR			
662	GCSF_1'-'IGFBP3_1	X	X	X
663	GCSF_1'-'IGFBP-6_1			
664	GCSF_1'-'IL-1_RI_1	X	X	X
665	GCSF_1'-'IL-11_1			
666	GCSF_1'-'IL-1R4_/ST2_1			
667	GCSF_1'-'NT-4_1	X	X	X
668	GCSF_1'-'OST_1			
669	GCSF_1'-'TPO_1			
670	GCSF_1'-'uPAR_1			
671	GITR-Light_1'-'GRO_1			
672	GRO_1'-'IL-12_p70_1			
673	ICAM-1_1'-'ICAM-3_1	X	X	X
674	ICAM-1_1'-'IL-12_p70_1	X	X	X
675	ICAM-1_1'-'I-TAC_1			
676	ICAM-1_1'-'TECK_1			
677	IL-11_1'-'IL-12_p70_1	X	X	X
678	IL-11_1'-'IL-17_1	X	X	X
679	IL-11_1'-'IL-1R4_/ST2_1	X	X	X
680	IL-11_1'-'I-TAC_1	X	X	X
681	IL-11_1'-'TECK_1	X	X	X
682	IL-12_p70_1'-'OST_1	X	X	X
683	IL-17_1'-'IL-6_R_1	X	X	X
684	Lymphotactin_1'-'TRAIL_R4_1	X	X	X
685	PIGF_1'-'TECK_1			
686	TIMP-1_1'-'TRAIL_R3_1	X	X	X

at least α features must be in every feasible solution. Finally, Equation (3.6) ensures decision variables only take binary values.

An optimal solution for Model $\mathcal{IP}_{\mathcal{MCFSP}}$ includes a set of features $J^* \subseteq J$ with the total cost of $z^* = k^*$. In the unweighted or unicost variant of the problem (where all features have a unique cost) $z^* = |J^*| = k^*$, where J^* is the set of features in an optimal solution.

The reader may realize that the Min k (α, β)-k FSP is a variant of the Set k-Cover Problem (SkCP), where feature profiles may be represented by columns, and elements of set I_1 by rows; see Chapter 4 for more details. Because the SkCP is proven $\mathcal{NP}$-Hard, the Min k (α, β)-k FSP is also $\mathcal{NP}$-Hard.

3.5.2 An integer program for the Max β (α, β)-k Feature Set Problem

Step 3 of the four-step approach determines $\beta^* \in \mathbb{Z}^+$, such that a set of minimum cost features are selected to explain the dichotomy between the classes, and at least α^* features do so for each pair of entities of different classes. Paula (2012) calls this problem the Max β (α, β)-k Feature Set Problem (FSP). Indeed, this problem maximizes the internal consistency of the entities in the same class, and contributes to a more robust feature set. The Max β (α, β)-k FSP can mathematically be modeled as an IP, where α^* and k^* (obtained in Steps 1 and 2) are parameters. Model $\mathcal{IP}_{\mathcal{MBP}}$ shows this. The model has two types of integer decision variables: $x_j \in \{0, 1\}, \forall j \in J$, and $\beta \in \mathbb{Z}^+$.

Model $\mathcal{IP}_{\mathcal{MBP}}$

$$z = \max \beta \tag{3.7}$$

subject to

$$\sum_{j \in J} c_j x_j \leq k^* \tag{3.8}$$

$$\sum_{j \in J} a_{ij} x_j \geq \alpha, \forall i \in I_1, 1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+ \tag{3.9}$$

$$\sum_{j \in J} a_{ij} x_j \geq \beta, \forall i \in I_2 \tag{3.10}$$

$$x_j \in \{0, 1\}, \forall j \in J \tag{3.11}$$

$$\beta \in \mathbb{Z}^+ \tag{3.12}$$

In Model $\mathcal{IP}_{\mathcal{MBP}}$, the objective function (Equation (3.7)) maximizes $\beta \in \mathbb{Z}^+$. Equation (3.8) ensures always a minimum cost feature set will be selected, where the minimum cost for the set is determined through solving the Min k (α, β)-k FSP; see Section 3.5.1. Equation (3.9) ensures every element of I_1 is covered by at least α features, where $1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+$ is a parameter obtained by using Equation (3.3). Equation (3.10) ensures every element of I_2 is covered by at least β features. Finally, Equations (3.11) and (3.12) ensure that decision variables $x_j, \forall j \in J$ only take binary values, and β only takes positive integer values.

The optimal solution of Model $\mathcal{IP}_{\mathcal{MBP}}$ is a set of minimum cost features with the maximum value for β. By looking into Equation (3.7) and Equation (3.10) one may realize that the Max β (α, β)-k FSP can be considered as a variant of the well-known Maximum Satisfiability Problem (MAX-SAT), which is proven $\mathcal{NP}$-Hard. Therefore, the Max β (α, β)-k FSP is also $\mathcal{NP}$-Hard.

3.5.3 An integer program for the Max Cover (α, β)-k Feature Set Problem

The last step of the proposed four-step decomposition-based approach to solve the (α, β)-k FSP obtains a set of features such that the set provides more coverage. More precisely, among alternative minimum cost sets of features (each with the cost k^*) that cover every element of I_1 by at least α features, and every element of I_2 by at least β features, Step 4 obtains a set of features that provides more coverage ("explanations") in total, either to the differences between the classes or similarity within entities in the same class. This problem was previously called the Max Cover (α, β)-k Feature Set Problem (FSP) (Berretta et al., 2005), and can mathematically be modeled as an IP (Model $\mathcal{IP}_{\mathcal{MCP}}$). In the model, α^*, β^*, and k^* (obtained in Steps 1, 2, and 3, respectively) are optimal value of parameters α, β, and k.

Model $\mathcal{IP}_{\mathcal{MCP}}$

$$z = \max \sum_{j \in J} v_j x_j \tag{3.13}$$

subject to

$$\sum_{j \in J} c_j x_j \leq k^* \tag{3.14}$$

$$\sum_{j \in J} a_{ij} x_j \geq \alpha, \forall i \in I_1, 1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+ \tag{3.15}$$

$$\sum_{j \in J} a_{ij} x_j \geq \beta, \forall i \in I_2, 1 \leq \beta \leq \beta^*, \beta \in \mathbb{Z}^+ \tag{3.16}$$

$x_j \in \{0,1\}, \forall j \in J$ (3.17)

In Model $\mathcal{IP}_{\mathcal{MCP}}$, the objective function maximizes the total features value/degree. As we discussed in Section 3.3, this criterion can model the total differences between the classes or similarity within entities in the same class. The only decision variables of Model $\mathcal{IP}_{\mathcal{MCP}}$ are $x_j \in \{0,1\}, \forall j \in J$.

In fact, the Max Cover (α,β)-k FSP obtains a feature set that leads to the largest covering among all alternative solutions. That said, if there is a unique optimal solution for the previous steps, solving the Max Cover (α,β)-k FSP will result in the same set of features. Intuitively, one may realize that any solution for the Max Cover (α,β)-k FSP is feasible for the problems of Min k (α,β)-k FSP and Max β (α,β)-k FSP. Finally, the Max Cover (α,β)-k FSP is a variant of the Maximum Coverage Problem (MCP), which is $\mathcal{NP}$-Hard, so does the Max Cover (α,β)-k FSP.

3.6 Bounds

This section develops lower and upper bounds (LB and UB) for the Min k (α,β)-k Feature Set Problem (FSP) and Max β (α,β)-k Feature Set Problem (FSP). The bounds will be utilized in next chapters when designing and developing algorithms. In total, three bounds will be discussed here.

Lemma 3.1. ***A lower bound for the Min k $(\boldsymbol{\alpha},\boldsymbol{\beta})$-k Feature Set Problem (FSP).*** *Assume that the linear programming (LP) relaxation of Model $\mathcal{IP}_{\mathcal{MCFSP}}$ is given. Let Model $\mathcal{LP}_{\mathcal{MCFSP}}$ represents this. Model $\mathcal{LP}_{\mathcal{MCFSP}}$ is obtained by relaxing binary decision variables $x_j \in \{0,1\}, \forall j \in J$ of Model $\mathcal{IP}_{\mathcal{MCFSP}}$ (thus, x_j can take any non-negative values in the ranges $[0,1]$). Also, assume that the optimal objective function value of Model $\mathcal{LP}_{\mathcal{MCFSP}}$ is $\underline{k}^* \in \mathbb{R}^+$. If all features' costs are integer, then a tighter integer lower bound may be obtained by $\lceil \underline{k}^* \rceil \in \mathbb{Z}^+$. In other words, $\underline{k}^* \leq \lceil \underline{k}^* \rceil \leq k^*$, where $k^* \in \mathbb{Z}^+$ is the optimal objective function value of Model $\mathcal{IP}_{\mathcal{MCFSP}}$.*

Proof. We provide the proof for both weighted and unweighted variants of the Min k (α,β)-k FSP. Remember that in the unweighted variant $c_j = \mathcal{C}, \forall j \in J, \mathcal{C} \in \mathbb{R}^+$, and in the weighted variant $c_j \in \mathbb{R}^+, \forall j \in J$. Let us start by the weighted variant (the unweighted variant is a special case of the weighted one where all weights are unique).
Because the Min k (α,β)-k FSP is a minimization integer program we know that solving its linear programming relaxation, which is obtained by relaxing the integrality constraints on binary decision variables $x_j \in \{0,1\}, \forall j \in J$ and allowing them to take any non-negative values in the ranges $[0,1]$, results in a lower bound on the optimal objective function value of Model $\mathcal{IP}_{\mathcal{MCFSP}}$, i.e. $\underline{k}^* \leq k^*$, where $\underline{k}^* \in \mathbb{R}^+$ is the optimal objective function value of Model $\mathcal{LP}_{\mathcal{MCFSP}}$, and k^* is the optimal objective function value of Model $\mathcal{IP}_{\mathcal{MCFSP}}$.

Additionally, if $c_j \in \mathbb{Z}^+, \forall j \in J$, then any feasible solution to Model $\mathcal{IP}_{\mathcal{MCFSP}}$ must have an integer objective function value. Therefore, we can round up $\underline{k}^*$ to its nearest integer value. Thus, $\lceil \underline{k}^* \rceil \leq k^*$.

For the unweighted variant of the Min k (α, β)-k FSP, the proof is followed by observing that the objective function aims to minimize the number of features, which is always an integer value. Also, $\underline{k}^*$ can be rounded up as well. Note that if $\underline{k}^* \in \mathbb{Z}^+$, then we have obtained an optimal solution for unweighted Min k (α, β)-k FSP. ∎

Example 3.1 illustrates how a lower bound can be developed by using Lemma 3.1.

Example 3.1. *Assume for a given instance of the Min k (α, β)-k FSP $\underline{k}^* = 64.57$. Lemma 3.1 states that $k^* > 64.57$, where k^* is the optimal objective function value of Model $\mathcal{IP}_{\mathcal{MCFSP}}$. It also states that if $c_j \in \mathbb{Z}^+, \forall j \in J$, then $k^* \geq \lceil 64.57 \rceil$, i.e. $k^* \geq 65$.*

Lemma 3.2. ***A lower bound for the Max β (α, β)-k Feature Set Problem (FSP).*** *Assume an optimal set of features (J^*) is given for the Min k (α, β)-k FSP. An integer lower bound $\underline{\beta} \in \mathbb{Z}^+$ on the optimal objective function value of Max β (α, β)-k FSP may be derived by calculating the value of β for this solution.*

Proof. Proof is followed by observing that the Max β (α, β)-k FSP is to select the best solution, among all optimal solutions of the Min k (α, β)-k FSP, according to the criterion of maximizing the value of β. This is because according to Model $\mathcal{IP}_{\mathcal{MBP}}$, any feasible solution for Model $\mathcal{IP}_{\mathcal{MCFSP}}$ is indeed feasible to Model $\mathcal{IP}_{\mathcal{MBP}}$. Hence, the optimal solution of the Min k (α, β)-k FSP must also be a feasible solution for the Max β (α, β)-k FSP. Because the Max β (α, β)-k FSP is a maximization problem a feasible solution is always a lower bound solution. Therefore, the value of β for this solution, which we denote by $\underline{\beta} \in \mathbb{Z}^+$ is a lower bound for $\beta^* \in \mathbb{Z}^+$, where β^* is the optimal value of β. Equation (3.18) shows the calculation of $\underline{\beta}$.

$$\underline{\beta} = \min_{i \in I_2} \left(\sum_{j \in J^*} a_{ij} x_j \right) \tag{3.18}$$

∎

Example 3.2 shows how a lower bound on the value of β^* may be obtained by utilizing Lemma 3.2.

Example 3.2. *Given the data of Table 3.2, assume an optimal solution for the Min k (α, β)-k FSP is given as $J^* = \{1, 2, 3\}$, where $k^* = 3$ (features have a cost of 1). For this solution the value of $\beta_i, \forall i \in I_2$ can be calculated. This is shown in the right most column of Table 3.5. By using Equation (3.18), $\underline{\beta} = \min(4, 3, 4, 2, 3) = 2$. Thus, $\beta^* \geq 2$.*

Lemma 3.3. ***An upper bound for the Max β (α, β)-k Feature Set Problem (FSP).*** *Given the optimal objective function value for the linear programming (LP) relaxation of Max β (α, β)-k FSP, i.e. $\bar{\beta}^*$, a tighter integer upper bound may be obtained by $\lfloor \bar{\beta}^* \rfloor \in \mathbb{Z}^+$, i.e. $\lfloor \bar{\beta}^* \rfloor \geq \beta^*$, where β^* is the optimal objective function value for the Max β (α, β)-k FSP.*

Table 3.5: An illustrative example to explain obtaining a lower bound on the optimal objective function value of the Max β (α, β)-k Feature Set Problem (FSP). The lower bound is obtained through solving the Min k (α, β)-k Feature Set Problem.

Element	Profile					α_i
	P_1	P_2	P_3	P_4	P_5	
I_{11}	1	0	1	0	0	2
I_{12}	1	1	0	1	0	3
I_{13}	1	0	1	1	0	3
I_{14}	1	0	1	0	0	2
I_{15}	1	1	0	1	0	3
I_{16}	1	0	1	1	0	3
						β_i
I_{21}	1	1	1	0	1	4
I_{22}	1	1	0	0	1	3
I_{23}	1	0	1	1	1	4
I_{24}	1	0	0	0	1	2
I_{25}	1	1	0	1	0	3
Value of feature (v_j)	11	5	6	6	4	

Proof. Proof is exactly same as proof of Lemma 3.1. The only difference is that the Max β (α, β)-k FSP is a maximization problem, whereas the Min k (α, β)-k FSP is a minimization problem. Thus, we replace $\lceil\rceil$ by $\lfloor\rfloor$. ∎

Example 3.3 illustrates how an upper bound for the Max β (α, β)-k FSP may be obtained by solving the linear programming relaxation of Max β (α, β)-k FSP.

Example 3.3. *Presume we are given an instance of the Max β (α, β)-k FSP, which has an optimal solution to its linear programming relaxation wit the objective function value of $\bar{\beta}^* = 51.20$. Lemma 3.3 states that $\beta^* \leq 51$.*

3.7 Mathematical properties

We investigate and develop several properties and propositions of the Min k (α, β)-k Feature Set Problem (FSP), the Max β (α, β)-k Feature Set Problem (FSP), and the Max Cover (α, β)-k Feature Set Problem (FSP). Later in Chapters 4 and 5 we utilize those properties in order to design and develop algorithms and solution methods. According to the computational results reported in Chapters 4 and 5, those properties and propositions tremendously impact capability of the developed algorithms.

Proposition 3.1. *In the Min k (α, β)-k Feature Set Problem (FSP), if $\alpha = 1$ and there is a single feature $j \in J$ that has a coverage value (degree) of $|I_1|$ (i.e. feature j is capable* *of covering every element of I_1 exactly once, see Section 3.3), then an optimal solution only includes this feature. If more than one such a feature exists, the one with the least cost is chosen.*

Proof. Recall that in the Min k (α, β)-k FSP, every element of I_1 (the first set of elements) must be covered by at least α features. Given $\alpha = 1$, the optimal solution includes only one feature if and only if that feature covers all elements of I_1. If such a feature $j \in J$ exists it must have a value (degree) of $v_j = |I_1|$. Note that if more than one such a feature exists, the one with the minimum cost is chosen. Therefore, $k^* = \min_{j \in J^* | v_j = |I_1|}(c_j)$, and $J^* = \{j\}$. ■

Proposition 3.2. *An alternative optimal solution for the Min k (α, β)-k Feature Set Problem (FSP) can be obtained by iteratively optimizing Model $\mathcal{IP}_{\mathcal{MCFSP}}$ through including constraints of the form $\sum_{j \in J^*} c_j x_j \neq k^*, J^* \in P$, where k^* is the optimal objective function value for the Min k (α, β)-k FSP, and P is the set of so obtained optimal solutions (an optimal solution J^* is a set of selected features).*

Proof. Observe that including Equation (3.19) in Model $\mathcal{IP}_{\mathcal{MCFSP}}$, and re-optimizing the model ensures the most recent optimal solutions are not explored during the next re-optimization process.

$$\sum_{j \in J^*} c_j x_j \neq k^*, J^* \in P \tag{3.19}$$

After performing a re-optimization two outcomes are possible: 1) a new optimal solution for the Min k (α, β)-k FSP is obtained, in which P is updated to include this solution, or 2) an infeasible status is reported. If the former is the case, we may continue and obtain a pool P of optimal solutions for the Min k (α, β)-k FSP (one new optimal solution per each re-optimization) until the stopping condition (which may be an infeasiblity status) is met. If the latter is the case, we have the proof that all optimal solutions for the Min k (α, β)-k FSP are explored. ■

Proposition 3.3. *The Max β (α, β)-k Feature Set Problem (FSP) is the problem of selecting the solution with the largest value of β, among all optimal solutions of the Min k (α, β)-k Feature Set Problem (FSP).*

Proof. The proof is followed by observing that any feasible solution for the Min k (α, β)-k FSP is also feasible for the Max β (α, β)-k FSP. This is observed by having Equation (3.5) and Equation (3.6) in Model $\mathcal{IP}_{\mathcal{MBP}}$. Moreover, Equation (3.8) ensures that only optimal solutions (or the best obtained solutions) for the Min k (α, β)-k FSP are allowed to be in any feasible solution for the Max β (α, β)-k FSP. Therefore, the set of feasible solutions for Max β (α, β)-k FSP is indeed the set of all optimal solutions for the Min k (α, β)-k FSP. Intuitively, the solution with the largest value of β is selected for the Max β (α, β)-k FSP. ■

Proposition 3.4. *Given an optimal solution for the Min k (α, β)-k Feature Set Problem (FSP), a feasible solution for the Max β (α, β)-k Feature Set Problem (FSP) may be obtained in polynomial time.*

Proof. Assume an optimal solution for the Min k (α, β)-k FSP is available. By Proposition 3.3 we know that this solution is feasible for the Max β (α, β)-k FSP. Then, by using Equation (3.18) we can derive the value of β for this solution. Note that Equation (3.18) may easily be calculated in $O(n)$. ∎

Proposition 3.5. *Given all optimal solutions for the Min k (α, β)-k Feature Set Problem (FSP), the Max β (α, β)-k Feature Set Problem (FSP) will reduce to a sorting problem, and hence, can be solved in polynomial time.*

Proof. Proposition 3.3 states that the Max β (α, β)-k FSP includes selecting the solution with the largest value of β, among all optimal solutions of the Min k (α, β)-k FSP. Given all optimal solutions of the Min k (α, β)-k FSP, we have a pool of all feasible solutions for the Max β (α, β)-k FSP. By applying Equation (3.18) we can obtain the value of β for each solution. Obtaining the solution, out of this pool, with the largest value of β is indeed a sorting problem.
It is well known that the worst performance of the best sorting algorithm is $O(n \log n)$, where n is the total number of elements. Here, n is the total number of optimal solutions for the Min k (α, β)-k FSP. ∎

The importance of Proposition 3.5 is that it provides a polynomial time algorithm to solve the Max β (α, β)-k FSP, provided that we have all optimal solutions for the Min k (α, β)-k FSP. Notice that because the Min k (α, β)-k FSP can mathematically be modeled as an integer program (Model $\mathcal{IP}_{\mathcal{MCFSP}}$), practically, obtaining all of its optimal solutions is generally an $\mathcal{NP}$-Complete problem. Interestingly, if the Min k (α, β)-k FSP has a unique optimal solution, then this solution must be optimal for the Max β (α, β)-k FSP as well. This is discussed in Proposition 3.6.

Proposition 3.6. *If the Min k (α, β)-k Feature Set Problem (FSP) has a unique optimal solution, then this solution is also optimal for the Max β (α, β)-k Feature Set Problem (FSP).*

Proof. The proof is based on Proposition 3.5 in which the pool of all feasible solutions for the Max β (α, β)-k FSP can be constructed by obtaining all optimal solutions for the Min k (α, β)-k FSP. As a special case, if the Min k (α, β)-k FSP has only one optimal solution, then the pool includes only one feasible solution for the Max β (α, β)-k FSP, which is also the optimal solution. ∎

Proposition 3.7. *Given a lower bound on the optimal objective function value of the Max β (α, β)-k Feature Set Problem (FSP), an optimal solution may be obtained through solving a feasibility problem.*

Proof. Assume a lower bound on the objective function value of Max β (α, β)-k FSP is given. Let $\underline{\beta} \in \mathbb{Z}^+$ denotes this lower bound, and $\beta^* \in \mathbb{Z}^+$ denotes the optimal objective function value of the Max β (α, β)-k FSP; hence, $\underline{\beta} \leq \beta^*$. By iteratively increasing $\underline{\beta}$ until any further increase leads to infeasiblity we can obtain β^*. ■

One may notice that Proposition 3.7 may be implemented as an iterative exact algorithm in order to optimally solve the Max β (α, β)-k FSP. Particularly, because $\beta^* \in \mathbb{Z}^+$ the algorithm terminates in a countable number of iterations. Also notice that Proposition 3.7 solves the Max β (α, β)-k FSP through solving a feasibility problem, which has its own challenges if the instances are large.

Proposition 3.8. *The Max Cover (α, β)-k Feature Set Problem (FSP) is to select the solution, among all optimal solutions of the Max β (α, β)-k Feature Set Problem (FSP), that has the largest value of $\sum_{j \in J^*} v_j x_j$, where $J^* \in P$.*

Proof. Similar to the proof of Proposition 3.3, the proof is followed by observing that any feasible solution for the Max β (α, β)-k FSP is also feasible for the Max Cover (α, β)-k FSP. This may be verified by observing that Equations (3.8) to (3.12) appear in Model $\mathcal{IP}_{\mathcal{MCP}}$. Moreover, Equations (3.14) and (3.16) ensure that only optimal solutions for the Min k (α, β)-k FSP and Max β (α, β)-k FSP will be considered as feasible solutions for the Max Cover (α, β)-k FSP. Additionally, the Max Cover (α, β)-k FSP selects the solution, out of all these feasible solutions, that maximizes the total coverage value, that is $\sum_{j \in J^*} v_j x_j$, $J^* \in P$, where P is the set of all optimal solutions for the Max β (α, β)-k FSP. ■

3.8 Conclusion

This chapter discussed integer programs for the Min k (α, β)-k Feature Set Problem (FSP), Max β (α, β)-k FSP, and Max Cover (α, β)-k FSP. After establishing the definitions, notations, and mathematical models, we discussed lower and upper bounds for the Min k (α, β)-k FSP and Max β (α, β)-k FSP, in Section 3.6. Finally, in Section 3.7 we investigated the mathematical properties of those three problems of Min k (α, β)-k FSP, Max β (α, β)-k FSP, and Max Cover (α, β)-k FSP, and developed several propositions. Those propositions will be utilized in Chapters 4 and 5 to develop algorithms and solution methods for solving the Min k (α, β)-k FSP, Max β (α, β)-k FSP, and Max β (α, β)-k FSP.

Chapter 4

Solution Methods for the Min k (α, β)-k Feature Set Problem

The major outcome of this chapter entitled "Tight lower bounds and a hybrid heuristic for a problem of selecting features" was peer-reviewed, and accepted for oral presentation at the EURO 2016 international conference in Poznan, Poland, between 3 – 6 July 2016.

The second manuscript entitled "Efficient solution methods for the Min k (α, β)-k Feature Set Problem" is under preparation to be submitted for a top tier journal very soon.

Abstract

In Chapter 3 we discussed that the Min k (α, β)-k Feature Set Problem (FSP) is a variant of the well-known Set k-Cover Problem (SkCP), which itself is an extension of the classical Set Cover Problem (SCP). This chapter develops heuristics and exact-based algorithms for both weighted and unweighted Min k (α, β)-k FSP. While in the weighted variant there is a cost associated with selecting a feature, in the unweighted variant the cost is unique and equal across all features. The proposed heuristics include greedy construction and improvement algorithms, and a very efficient exact+heuristic (EH) algorithm, which combines both heuristic and exact algorithms, and obtains very high quality solutions for the Min k (α, β)-k FSP. The benchmark instances for evaluating the performance of algorithms include one set of 11 real-world unweighted instances ranging from medium to large, one set of 210 weighted instances of the SCP ranging from small to medium, which are available in the literature, and one set of 125 randomly generated large and unweighted instances. Computational results over a total of 346 settings show that the proposed EH algorithm competes well against the state-of-the-art algorithms. Moreover, the EH algorithm obtains several new best solutions for the standard instances of the SkCP and for the randomly generated instances.

4.1 Introduction

In Chapter 3 we discussed that the Min k (α, β)-k Feature Set Problem (FSP) is a variant of the well-known Set k-Cover Problem (SkCP). The SkCP, which has many applications including in computational biology, is an extension of the classical Set Cover Problem (SCP). While in the SCP each row (equivalently, an element in the Min k (α, β)-k FSP) is required to be covered by at least one column (equivalently, feature), in the SkCP each row is required to be covered by at least α columns, where $1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+$, such that the cost of selecting columns is minimized. The case of $\alpha = 1$ refers to the classical SCP. In the Min k (α, β)-k FSP, we are looking for a set of minimum cost features such that every element of I_1 (pairs of entities of different classes) is covered by at least α features, where $1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+$ is an instance-dependent parameter. In other words, α represents the minimum number of features that must explain the differences between any pair of entities of different classes.

The literature on the SCP is very rich. Several exact algorithms have been developed for the SCP that obtain optimal solutions for the medium sized instances in a reasonable amount of time (Balas and Ho (1980), Beasley (1987), Beasley (1990), Fisher and Kedia (1990), Beasley and Jrnsten (1992), Balas and Carrera (1996)). Nevertheless, the SCP still remains intractable in a general term, and hence, heuristics are of practical importance. One of the fundamental heuristic algorithms for the SCP was developed by Chvatal (1979). Chvatal's idea is based on the cost of a column j, i.e. c_j, and the number of currently uncovered rows that could be covered by column j, i.e. r_j. His greedy heuristic evaluates every column j by c_j/r_j, and then selects the column with the minimum c_j/r_j. This evaluation criterion has been used in many heuristic algorithms developed since. For example, Vasko (1984) improved the column selection mechanism of the Chvatal's greedy heuristic by adding a local search procedure, and Baker (1981) merged several solutions into a reduced cost solution. Randomized procedures have also been utilized along with the Chvatal's greedy heuristic. For example Feo and Resende (1989) created a list of columns that pass a certain criterion. Then a column is randomly selected from this list. Another randomized idea has been implemented by Lan et al. (2007); instead of selecting column j with the minimum c_j/r_j, their algorithm randomly selects column j while the total number of random selections is controlled by a parameter.

Probably one of the best heuristic algorithms for the SCP is due to Caprara et al. (1999). Their algorithm is a Lagrangian-based heuristic where the Lagrangian multipliers are obtained, and utilized in a greedy heuristic to obtain a solution for the SCP. Then, a subset of columns that have a high probability of being in an optimal solution is selected, and their corresponding variables are set to 1. In fact, this results in an SCP instance with a reduced number of columns and rows, on which the whole algorithm is iterated. Other less superior results were obtained by the Lagrangian-based procedures of Haddadi (1997) and Ceria et al. (1998). A review of the SCP algorithms has been brought in Caprara et al. (2000). Meta-heuristic have also been studied for the SCP. An effective genetic algorithm with improved genetic operators was developed by Beasley and Chu (1996). One efficient heuristic was developed by Yagiura et al. (2006). Their main idea is a "3–flip neighborhood", which obtains a set of solutions, from the current solution, by exchanging at most three subsets, followed by several procedures to reduce the size of the neighborhood. A Tabu Search algorithm was studied in Caserta (2007). Naji-Azimi et al. (2010) developed a meta-heuristic for the SCP where the construction and improvement phases have some degree of randomization.

Although, most of the literature is on weighted SCP, where c_j is the cost associated with column j, several studies targeted the unweighted or unicost SCP; see for example Bautista and Pereira (2007). The unweighted SCP is more difficult to solve than the weighted SCP (Vasko and Wilson, 1986). Notice that in the unicost SCP every column has the same cost, and thus, the optimal solution minimizes the total number of columns.

On the Set k-Cover Problem (SkCP), the literature is not as rich as on the SCP, although SCP is a special case of the SkCP, where $k = \alpha = 1$. The SkCP is more difficult to solve than the SCP because of the multi coverage requirement (i.e. $\alpha > 1$). Wang et al. (2016b) developed two randomized heuristic algorithms for the SkCP. The core of their algorithms is a column selection strategy. They tested their algorithms on 210 standard instances of the SCP. One of the heuristics for the SkCP is developed by Pessoa et al. (2011) and Pessoa et al. (2013). Their algorithm builds an initial solution by a Lagrangian-based heuristic, and then repairs it by using a randomized greedy algorithm combined with path relinking. Further improvement to this solution is obtained by two neighborhoods. The first neighborhood removes unnecessary columns while the second neighborhood replaces a more expensive column with a less expensive one. A dynamic programming framework has been discussed in Hua et al. (2010); the authors have not reported any computational results though.

The remaining of this chapter is organized as follows. Section 4.2 provides a short introduction on the SCP and SkCP. Section 4.3 explains the Min k (α, β)-k FSP. Section 4.4 discusses lower bound schemes for the Min k (α, β)-k FSP. Section 4.5 and Section 4.6 develop two greedy algorithms for the Min k (α, β)-k FSP. The algorithm of Section 4.5 is a constructive algorithm and aims to build an initial solution, and the algorithm of Section 4.6 is an improvement local search algorithm and aims to improve the solution by removing redundant features. Section 4.7 explains the proposed exact+heuristic (EH) algorithm for the Min k (α, β)-k FSP. Section 4.8 reports the computational experiments of the algorithms on three sets of 346 instances, including real-world, weighted, and randomly generated instances. Finally, the chapter ends with conclusions.

4.2 The Set k-Cover Problem

Given a set of elements (*rows*) $I = \{1, \ldots, m\}$ and a set $P = \{P_1, \ldots, P_n\}$ of n sets whose union equals I, where $P_j \subseteq I$, $j \in J = \{1, \ldots, n\}$, a subset $J^* \subseteq J$ defines a *cover* of I if $\bigcup_{j \in J^*} P_j = I$. Then, the Set Cover Problem or the SCP is to obtain a minimum cost cover. In fact, the SCP identifies the least expensive subset of P whose union equals I (Garfinkel and Nemhauser, 1972).

For example, consider $I = \{1, 2, 3, 4, 5\}$ and $P = \{P_1 = \{1, 2, 3\}, P_2 = \{2, 4\}, P_3 = \{3, 4\}, P_4 = \{4, 5\}, P_5 = \{1, 2\}, P_6 = \{1, 2, 5\}\}$, where $\bigcup_{j \in J} P_j = I$. Given $c_j = 1, \forall j \in J$ (equal cost for columns), the minimum cost subsets of P whose union is I has a cost of two, and the subsets are $P_1 = \{1, 2, 3\}$ and $P_4 = \{4, 5\}$. Thus, $J^* = \{1, 4\}$.

This definition implies that every element of I must be covered at least once. The Set k-Cover Problem or the SkCP is where every element of I must be covered by at least k (α) columns (for the purpose of being consistent across the chapter we use α to denote k). The SCP is a special case of the SkCP where $\alpha = 1$. Given $\alpha = 2$ in the above example, the minimum cost subsets of P would have a cost of four, and the subsets are $P_1 = \{1, 2, 3\}$, $P_3 = \{3, 4\}$, $P_4 = \{4, 5\}$, and $P_6 = \{1, 2, 5\}$. Thus, $J^* = \{1, 3, 4, 6\}$. Notice that in this example, we cannot have $\alpha \geq 3$ because we cannot cover every element of I more than two times, no matter how many subsets of J we select.

The SkCP can be modeled as an integer program (IP) and may be formulated as Model $\mathcal{IP}_{\mathcal{SKCP}}$ (Garfinkel and Nemhauser, 1972):

Model $\mathcal{IP}_{\mathcal{SKCP}}$

$$z = \min \sum_{j \in J} c_j x_j \tag{4.1}$$

$$\sum_{j \in J} a_{ij} x_j \geq \alpha, \forall i \in I, 1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+ \tag{4.2}$$

$$x_j \in \{0, 1\}, \forall j \in J \tag{4.3}$$

In Model $\mathcal{IP}_{\mathcal{SKCP}}$, the objective function (Equation (4.1)) minimizes the total cost of selecting columns, where $c_j \in \mathbb{R}^+, \forall j \in J$ is the cost of selecting column j. In an unweighted (unicost) SkCP, $c_j = \mathcal{C}, \forall j \in J$, where $\mathcal{C} \in \mathbb{R}^+$ is a scaler, and hence, the objective function minimizes the total number of columns. Equation (4.2) ensures a feasible solution is obtained, i.e. every element (row) is covered by at least α columns, where $1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+$ is a parameter, value of which is determined according to the instance. In this equation, parameter a_{ij} takes a value of 1 if column j is capable of covering row i, and 0 otherwise. Note that, the case of $\alpha = 1$ in the right hand side of Equation (4.2) results in the SCP. Finally, Equation (4.3) ensures decision variables are binary, and take 1 if column j is selected to be in the solution, and 0 otherwise.

4.3 The Min k (α, β)-k Feature Set Problem

As discussed earlier in Section 3.5.1, the Min k (α, β)-k Feature Set Problem (FSP) can mathematically be modeled as an integer program (see Model $\mathcal{IP}_{\mathcal{MCFSP}}$ in Section 3.5.1). The model obtains a minimum cost set of features (among alternative minimum cost sets of features) that explain the dichotomy between the classes, considering that at least α features do so for each pair of entities of different classes (elements of I_1).

An optimal solution to Model $\mathcal{IP}_{\mathcal{MCFSP}}$ is a vector $\mathbf{x}^* = \{x_j | x_j = 1, j \in J\}$, where J is the set of all features. That is, a set of binary decision variables whose values are 1. Additionally, an optimal set of features may be represented by set $J^* \subseteq J$. In the unweighted variant of the Min k (α, β)-k FSP (that is, all features have a unique cost) the objective function ensures the minimum number of features is selected. As discussed earlier, one may realize that the Min k (α, β)-k FSP is a variant of the Set k-Cover Problem (SkCP), where features represent columns, and elements of set I_1 represent rows.

4.4 Lower bounds

A lower bound (LB) would help to evaluate the quality of a given solution for the Min k (α, β)-k Feature Set Problem (FSP), i.e. in the worst case how far a feasible solution would be from the optimal solution. If the LB yields a feasible solution, then this solution is optimal for the Min k (α, β)-k FSP. In addition to this, the algorithm of Section 4.7 relies on an LB to construct a partially built solution for the Min k (α, β)-k FSP.

For the unweighted Min k (α, β)-k FSP, an intuitive LB on the optimal objective function value is the value of α. Because Equation (3.5) states that every element must be covered by at least α features. Thus, in order to have a feasible solution at least α features are needed, although the number of features in a feasible solution might be grater than this. Nevertheless, it guarantees no less than α features may construct a feasible solution. This LB, however, is often loose, and is far from an optimal solution.

Another LB for both weighted and unweighted Min k (α, β)-k FSP may be developed by solving a linear programming (LP) relaxation of Model $\mathcal{IP}_{\mathcal{MCFSP}}$. Linear programming relaxations have been studied for many integer and mixed integer programs including the Set Cover Problem (see Lovász (1975)). The LP relaxation model of Min k (α, β)-k FSP, which we named it Model $\mathcal{LP}_{\mathcal{MCFSP}}$, may be obtained by relaxing Equation (3.6); that is by setting $0 \leq x_j \leq 1, \forall j \in J$. Model $\mathcal{LP}_{\mathcal{MCFSP}}$ is illustrated in the following.

Model $\mathcal{LP}_{\mathcal{MCFSP}}$

$$z = \min \sum_{j \in J} c_j x_j \tag{4.4}$$

$$\sum_{j \in J} a_{ij} x_j \geq \alpha, i \in I_1, 1 \leq \alpha \leq \alpha^*, \alpha \in \mathbb{Z}^+ \tag{4.5}$$

$$0 \leq x_j \leq 1, \forall j \in J \tag{4.6}$$

Figure 4.1: Solution representation utilized in the heuristic algorithms for the Min k (α, β)-k Feature Set Problem (FSP). The set of features are listed in the first row, and their status (whether they are selected to be in a solution) are listed in the second row. According to the second row, five features have been selected to be in the solution. Those features are "A", "D", "G", "H", and "J".

Feature	A	B	C	D	E	F	G	H	I	J
Status	1	0	0	1	0	0	1	1	0	1

Presume the optimal objective function value of Min k (α, β)-k FSP (in other words, the optimal solution of Model $\mathcal{IP}_{\mathcal{MCFSP}}$) and that of its LP relaxation (the optimal solution of Model $\mathcal{LP}_{\mathcal{MCFSP}}$) are available, and let $z^* \in \mathbb{Z}^+$, and $\underline{z}^* \in \mathbb{R}^+$ denote those two, respectively. We know that $z^* \geq \underline{z}^*$, because the Min k (α, β)-k FSP is a minimization problem, and any solution to Model $\mathcal{IP}_{\mathcal{MCFSP}}$ is feasible for Model $\mathcal{LP}_{\mathcal{MCFSP}}$. We have observed that this LB is of better quality than that of the previous one. If $c_j \in \mathbb{Z}^+, \forall j \in J$, this LB may even be more tightened by rounding up $\underline{z}^*$ to its nearest integer value, i.e. $\lceil \underline{z}^* \rceil \in \mathbb{Z}^+$. This is fully discussed in Lemma 3.1. Notice that rounding up $\underline{z}^*$ to its nearest integer value does not impact the associated variables, because variables x_j are not enforced to take binary values. On the contrary, Equation (4.6) enforces them to take non-negative values between 0 and 1. Therefore, we may not still have a feasible solution for the Min k (α, β)-k FSP. Later in Section 4.7 we utilize this LB to construct a partially built solution for the EH algorithm.

4.5 A greedy construction algorithm

This section explains a greedy construction heuristic, which aims to construct initial solutions for the Min k (α, β)-k Feature Set Problem (FSP). We shall start by explaining the solution representation of the algorithm, which is utilized throughout this research thesis and in all heuristic algorithms.

Because the Min k (α, β)-k FSP is to select a subset of features, out of a larger set, intuitively we are interested in whether a specific feature should be selected or not. Thus, the decision is limited to only two choices. We represent a solution for the Min k (α, β)-k FSP in a form of a list, where the cardinality of the list (number of its elements) is equal to the total number of features. Each element of the list takes a value of either 1, if the associated feature is selected to be in a solution, or 0, if it is not. For instance, Figure 4.1 may represent an instance of the Min k (α, β)-k FSP including 10 features. The features are listed in the first row, and their status (whether they are selected to be in a solution) are listed in the second row. Here, features "A", "D", "G", "H", and "J" have been selected to be in the solution.

The observation behind the proposed greedy construction heuristic, which we name it multi

Column Row Cover Construction heuristic algorithm (or mCRCC for short), is that certain features must always be in any feasible solution. In other words, if those features are not included, certain elements may never be covered by exactly α features.

The mCRCC algorithm has two steps. In Step 1, it obtains the set of features that must be in any feasible solution. We perform this by looking for those elements that satisfy $\alpha_i = \alpha, \forall i \in I_1$ (notice that at least one such element exists because this is the way we determined α), and extracting the associated features. In Step 2, the mCRCC algorithm builds a feasible solution by iteratively adding a set of features to the partially built solution. The mCRCC heuristic is illustrated in Algorithm 4.1.

Upon adding a set of features (say α') into the partially built solution we may group all elements into two sets: those that have been covered by at least α features, and hence, we do not need to consider them for more coverage, and those that have been covered by less than α features, which we denote by $\tilde{I} \subset I_1$. Indeed, we can construct a smaller instance of the Min k (α, β)-k FSP over the sets of available features ($\tilde{J} \subset J$) and $\tilde{I}$. Equation (4.7) illustrates how α' is re-calculated.

$$\alpha' = \max_{i \in \tilde{I}} \alpha'_i \tag{4.7}$$

where, α'_i is the number of additional features required to cover $i \in \tilde{I}$, and can be derived by calculating the difference between α and the number of features that already covers i, and α' is the minimum number of features that must be added into the partially built solution. To select a set of α' features, out of the set of available features ($\tilde{J}$) we incorporate information regarding the *importance* of features. To do so, we calculate $v_j, \forall j \in \tilde{J}$ by using Equation (4.8). Then, α' features with the maximum value of v_j/c_j are added into the partially built solution.

$$v_j = \sum_{i \in \tilde{I}} \alpha'_i \tag{4.8}$$

Indeed, the features covering critical elements (those with the greatest value of $\alpha'_i, \forall i \in \tilde{I}, \tilde{I} \subset I_1$) are preferred the most.

4.6 A removal local search

Because the multi Column Row Cover Construction (mCRCC) heuristic is a construction algorithm, and iteratively adds features into a partially built solution, redundant features may be introduced into the solution. For this reason, we propose the Removal Local Search (RLS) algorithm that improves a feasible solution by removing redundant features. The stopping criterion of the algorithm is whenever removing features does not yield a feasible solution. Algorithm 4.2 illustrates the RLS algorithm for the Min k (α, β)-k Feature Set Problem (FSP).

Algorithm 4.2 starts by finding a set $\tilde{J} \subset J^*$ of redundant features. To do so, it checks whether removing feature $j, \forall j \in J^*$ (i.e. from the feasible solution) still keeps the solution

Algorithm 4.1: The multi Column Row Cover Construction (mCRCC) heuristic algorithm for constructing a feasible solution for the Min k (α, β)-k Feature Set Problem (FSP). The mCRCC algorithm builds such a solution in two steps. In Step 1, it obtains a set of features that must be in any feasible solution. In Step 2, it adds additional features into the partially built solution until a feasible solution is obtained.

Input: A set J of features each with a value $v_j, \forall j \in J$, and a cost $c_j, \forall j \in J$; a set $J^* = \{\}, J^* \subseteq J$ of selected features in a feasible solution; a set $I_1 = \{1, \ldots, m_1\}$ of elements; parameter α.

Output: A set $J^* \subseteq J$ of features (a feasible solution for the Min k (α, β)-k FSP).

Step 1. Obtaining a lower bound.

$J' \leftarrow$ a set of features that must be in any feasible solution;

$J^* = J^* \cup J'$;

if *the solution is feasible* **then**

  At least one optimal solution is obtained, where $J^* \subseteq J$ is the set of optimal features;

end

else

  Step 2. Obtaining a feasible solution.

  while *the solution is not feasible* **do**

    Obtain sets $\tilde{J} \subset J$ (the set of available features), and $\tilde{I} \subset I_1$ (the set of uncovered elements);

    Update α' by using Equation (4.7), and calculate $v_j, \forall j \in \tilde{J}$ by using Equation (4.8);

    Update J' (sorted features in descending order of $v_j/c_j, j \in \tilde{J}$);

    $J^* = J^* \cup \{J'_1, \ldots, J'_{\alpha'}\}$;

  end

end

Report J^*;

Algorithm 4.2: The Removal Local Search (RLS) algorithm, which obtains an improved solution for the Min k (α, β)-k Feature Set Problem (FSP) by removing redundant features from a given feasible solution. The RLS algorithm randomly removes redundant features until any further removal results in an infeasible solution.

Input: The set J^* (features in a feasible solution); a set $\tilde{J} = \{\}$, which keeps redundant features; a set $I_1 = \{1, \dots, m_1\}$ of elements; parameter α.

Output: An improved solution for the Min k (α, β)-k FSP (a set $J^* \subseteq J$ of features).

```
Step 1. Finding redundant features.
while the solution is feasible do
    if J* \ {j|∀j ∈ J*} is a feasible solution then
        J̃ = J̃ ∪ {j};
    end
end
Step 2. Removing redundant features.
while the solution is feasible or J̃ = {} do
    r ← random(j|∀j ∈ J̃);
    J* = J* \ {r};
    J̃ = J̃ \ {r};
end
Report J*;
```

feasible. Then, the algorithm iterates through $\tilde{J}$, and randomly selects feature $j, j \in \tilde{J}$, and removes it from J^*. The algorithm keeps removing redundant features as long as the solution remains feasible or $\tilde{J} = \{\}$.

Notice that the RLS algorithm sequentially removes redundant features, i.e. one at a time. This is essential because all redundant features are sequentially sought, and hence, independent of each other. As a result, we must sequentially remove them in order to ensure the improved solution remains feasible. In addition to this, the order in which the redundant features are removed impacts the solution's quality. For this purpose, as well as increasing the diversification of the removal procedure, the algorithm performs a *random* removal. This means at every iteration one redundant feature is randomly selected from the set of all redundant features, and is removed from the solution.

4.7 An exact+heuristic algorithm

In this section we propose an exact+heuristic (EH) algorithm for the Min k (α, β)-k Feature Set Problem (FSP). To the best of our knowledge and at the time of writing this thesis, this algorithm obtains superior results for the Min k (α, β)-k FSP on both real-world and randomly generated instances. Furthermore, we tested the algorithm on 210 standard instances (weighted) of the Set Cover Problem (SCP), and observed that the EH algorithm obtains new best solutions for several instances.

The EH algorithm for the Min k (α, β)-k FSP combines both exact and heuristic algorithms in order to solve the Min k (α, β)-k FSP. The EH algorithm starts by obtaining a lower bound. Because the solution associated with the lower bound may not always be feasible, the EH algorithm repairs the lower bound solution and obtains a feasible solution. Finally, the feasible solution is improved in two steps: an exact step and a heuristic step. The EH algorithm is summarized in Algorithm 4.3. The details of each step of the EH algorithm is discussed in the next sections.

Algorithm 4.3: The exact+heuristic (EH) algorithm for solving the Min k (α, β)-k Feature Set Problem (FSP). The EH algorithm has three steps. Step 1 obtains a lower bound solution. Step 2 repairs the lower bound solution and obtains a feasible solution, and improves the solution by performing a re-optimization. Step 3 further improves the best obtained solution by applying the Removal Local Search (RLS) algorithm.

Input: Models $\mathcal{IP}_{\mathcal{MCFSP}}$ and $\mathcal{LP}_{\mathcal{MCFSP}}$; a set J of features, a set $J^* = \{\}, J^* \subseteq J$ of selected features in a feasible solution; a set $I_1 = \{1, \ldots, m_1\}$ of elements; parameter α.

Output: A high quality solution (a set $J^* \subseteq J$) for the Min k (α, β)-k FSP.

Step 1. Obtaining a lower bound.

Solve Model $\mathcal{LP}_{\mathcal{MCFSP}}$ to optimality, and let $\underline{\mathbf{x}}^*$ be the optimal solution;

if $x_j \in \{0, 1\}, \forall j \in J$ **then**

  The lower bound solution is both feasible and optimal for the Min k (α, β)-k FSP;

  $J^* = \{j | x_j = 1, j \in J\}$;

end

else

  Step 2. Obtaining a feasible solution.

  Fix certain $0 < x_j < 1$ to 1, and enforce the remaining to take binary values;

  Apply Algorithm 4.5, and let J^* be the set of features;

  if *the solution is not optimal* **then**

    Step 3. Improving the best solution.

    Remove redundant feature(s) by using the RLS algorithm (Algorithm 4.2);

  end

end

Report J^*;

4.7.1 Obtaining a lower bound

In order to obtain a lower bound for the Min k (α, β)-k FSP, as well as a partially built solution, we solve the linear programming (LP) relaxation of the Min k (α, β)-k FSP. The procedure is summarized in Algorithm 4.4.

Algorithm 4.4 may result in a partially built solution for the Min k (α, β)-k FSP, which is built by including all features that have a value of 1 for their associated variables into the solution. After solving Model $\mathcal{LP}_{\mathcal{MCFSP}}$ not every x_j variable may have an integer value. If

Algorithm 4.4: A linear programming (LP) relaxation-based algorithm to obtain a lower bound and a partially built solution for the Min k (α, β)-k Feature Set Problem (FSP).

Input: Model $\mathcal{LP}_{\mathcal{MCFSP}}$; a set J of features; a set $J^* = \{\}, J^* \subseteq J$ of selected features in a partially built solution; a set I_1 of elements; parameter α.

Output: A partially built solution (a set of features) and a lower bound for the Min k (α, β)-k FSP.

Solve Model $\mathcal{LP}_{\mathcal{MCFSP}}$ to optimality; let $\underline{z}^* \in \mathbb{R}^+$ be the optimal objective function value, and $\underline{\mathbf{x}}^*$ the optimal solution;
if $x_j \in \{0, 1\}, \forall j \in J$ **then**
| $J^* = J^* \cup \{j | x_j = 1, j \in J\}$ (the optimal set of features);
end
else
| $J^* = J^* \cup \{j | x_j = 1, j \in J\}$ (a partially built solution);
end
Report J^*;

$0 < x_j < 1, \exists j \in J$, we have at least one fractional variable, which means the optimal solution of Model $\mathcal{LP}_{\mathcal{MCFSP}}$ is not feasible for the Min k (α, β)-k FSP. Section 4.7.2 explains how we repair this infeasible solution into a feasible one. On the other hand, if $x_j \in \{0, 1\}, \forall j \in J$, the optimal solution of Model $\mathcal{LP}_{\mathcal{MCFSP}}$ is both feasible and optimal for Model $\mathcal{IP}_{\mathcal{MCFSP}}$, and hence, we have the optimal set of features for the Min k (α, β)-k FSP.

4.7.2 Obtaining a feasible solution

Solving Model $\mathcal{LP}_{\mathcal{MCFSP}}$ will not always lead to a feasible solution for the Min k (α, β)-k FSP. We have developed and implemented a procedure, which repairs an infeasible solution into a feasible one by adjusting the values of non-negative decision variables. The procedure is guaranteed to obtain a feasible solution for the Min k (α, β)-k FSP. Also, it simultaneously improves the feasible solution. This procedure, which is summarized in Algorithm 4.5, performs three operations: it ensures that all x_j variables only take binary values (guarantee of obtaining a feasible solution); it fixes certain x_j variables to take a value of one, which results in a partially built solution (for this reason, we introduced constraints in the form of $x_j = 1, j \in J$ to Model $\mathcal{IP}_{\mathcal{MCFSP}}$); and it improves the feasible solution by performing a re-optimization. The outcome of this procedure is an upper bound solution for the Min k (α, β)-k FSP. It is worth mentioning that the partially built solution greatly impacts the termination/convergence of an exact solver. In fact, we observed that without introducing a partially built solution, particularly for large instances of the Min k (α, β)-k FSP, the solver CPLEX may not obtain a feasible solution even after 30 minutes of running.

Notice that the purpose of Algorithm 4.5 is twofold: obtaining a feasible solution for the Min k (α, β)-k FSP by only solving a smaller instance of the Min k (α, β)-k FSP, and improving the feasible solution by performing a re-optimization. Because we fix certain variables to take a

Algorithm 4.5: An algorithm to build a feasible solution for the Min k (α, β)-k Feature Set Problem (FSP). To do so, the algorithm ensures all x_j variables take binary values, fixes certain x_j variables to take a value of one, and further improves the feasible solution.

Input: A partially built solution J^* for the Min k (α, β)-k FSP (obtained by solving Model $\mathcal{LP}_{\mathcal{MCFSP}}$), a lower bound $\underline{z}^*$ on the optimal objective function value.

Output: An improved feasible solution for the Min k (α, β)-k FSP.

while *the stopping condition is not met* **do**
  Solve Model $\mathcal{IP}_{\mathcal{MCFSP}}$, where $x_j = 1, \forall j \in J^*$;
  Let k^* denotes the value of objective function, and J^* be the set of features;
  if $\underline{z}^* \neq k^*$ *(optimality check)* **then**
    An upper bound solution is obtained, where the set of features is J^*;
  end
end
Report J^*;

value of one, and we do not have a guarantee that this maintains solution's optimality, we may enforce redundant features into the solution. Therefore, the solution may further be improved. This is discussed in Section 4.7.3.

If the number of fractional variables is large, there is a possibility that this procedure slowly converges. In such a case one may initialize the EH algorithm by using the multi Column Row Cover Construction (mCRCC) heuristic (Algorithm 4.1).

4.7.3 Improving the feasible solution

After obtaining an improved feasible solution for the Min k (α, β)-k FSP, there is a possibility that redundant features have been entered into the solution. This is observed by the fact that Algorithm 4.5 forces certain features into a feasible solution without a proof on whether those features are part of an optimal set of features. Hence, we may further improve the solution by applying the Removal Local Search (RLS) algorithm presented in Algorithm 4.2 in order to remove redundant features.

4.8 Computational results

In this section we report the computational experiments of applying the exact+heuristic (EH) algorithm, which is presented in Algorithm 4.3, on three sets of instances. All presented algorithms have been implemented in the programming language Python 2.7, and all mathematical models were also implemented in the programming language Python 2.7 via the solver CPLEX 12.5.0 Python API. The computing resource has Linux Ubuntu 14.04 LTS operating system with 32 GB of memory and 12 cores of Intel®Xeon CPU E5-1650 at 3.5 GHz. However, only one thread has been used by the algorithms, in order to provide the most similar basis for comparing the results with the available studies.

The first set includes 11 real-world unweighted instances ranging from small to large. The computational results of those instances are discussed in Section 4.8.1. The second set includes 210 standard weighted instances of the Set Cover Problem (SCP) ranging from small to medium. In particular, several of those instances pose computational challenge for the exact solvers. We discuss the computational results of those instances in Section 4.8.2. The third set includes 125 randomly generated unweighted instances for the (α, β)-k Feature Set Problem (FSP). Those instances have the same size, however, due to the their generation framework they pose different computational challenge for the available solution methods. The computational results of those instances are discussed in Section 4.8.3.

4.8.1 Computational results of real-world instances

The first set of instances includes two sub-sets of 11 real-world unweighted (unicost, i.e. $c_j = 1, \forall j \in J$) instances. The first set includes six biological instances, and the second set includes five large face recognition instances. We chose the first six instances because the study of Paula (2012) has utilized the same instances to evaluate the performance of their Variable Neighborhood Search+Tabu Search (VNS+TS) algorithm. We selected the second five instances because they are large, and as the exact solvers are unable to solve them, they can truly reflect the performance of the EH algorithm.

The basic information regarding those 11 real-world instances is shown in Table 4.1. The first three columns are the name of the instances, number of features, which may represent protein, genes, probes, SNPs, etc., and total number of entities, e.g. samples (of both Class 1 and Class 2). Each instance includes two classes (groups) of data: Class 1 and Class 2 (see Chapter 2 for more details). The second three columns provide information on the associated Min k (α, β)-k FSP of each instance. Column "$|J|$" gives the number of features, which essentially is the same as the second column, and column "$|I_1|$" gives the total number of elements in the first set of elements. Recall from our earlier discussion in Section 2.2 that set I_1 includes pairs of entities of Class 1 and Class 2, and can be obtained by using Equation (2.1). Column "α^*", which is derived by using Equation (3.3), shows the optimal (maximum) value of α (recall that α is the minimum number of features that must explain the differences between any pair of entities of different classes). Obviously, α^* depends on the instance, and any value greater than α^* results in an infeasible solution. The last column provides additional references.

Table 4.2 summarizes the outcomes of CPLEX, EH, and VNS+TS algorithm of Paula (2012). Table 4.3 details those outcomes. Note that the outcomes of VNS+TS are only available for the first six instances, and that they are available in ranges. Five criteria of *percent of feasible solution*, *percent of best solution*, *percent of optimal solution*, *average computation time* (in second), and *average gap* (from the best known solution) were used to evaluate each solution method. Following those outcomes we can conclude,

- all methods obtain feasible solutions for all instances;

Table 4.1: 11 real-world unweighted instances of the Min k (α, β)-k Feature Set Problem (FSP), including six biological, and five large face recognition instances. In addition to the size of each instance, which is reported in the second and third columns, the table provides size of an instance of the Min k (α, β)-k FSP including "$|J|$" (number of features), "$|I_1|$" (pairs of entities of different classes), and α^* (optimal value of α).

Instance	No. of features	No. of entities	$\|J\|$	$\|I_1\|$	α^*	Reference
ADMF	686	83	686	1720	86	Paula et al. (2011)
DS	73	15	73	56	50	Lockstone et al. (2007)
PD1	17099	105	17097	2750	3970	Scherzer et al. (2007)
PD2	1674	25	1674	144	760	Lesnick et al. (2007)
PC	3556	171	3556	7290	229	Chandran et al. (2007)
SM	525	1219	525	273834	22	Charlesworth et al. (2010)
0_all	1969	450	1969	32400	354	Haque et al. (2016)
1_all	3304	450	3304	32400	683	Haque et al. (2016)
2_all	4243	450	4243	32400	1016	Haque et al. (2016)
3_all	5436	450	5436	32400	1394	Haque et al. (2016)
4_all	2005	450	2005	32400	387	Haque et al. (2016)

- the largest number of optimal solutions were obtained by CPLEX, and that for around 91% of instances, followed by the EH algorithm for around 73% of instances. The VNS+TS obtained optimal solution for only 33.3% on instances (over six instances);

- the greater number of optimal solutions reported by CPLEX paid its price by taking almost 15 times longer than the EH; and,

- both CPLEX and EH have excellent average gaps. We were not able to report the average gap of VNS+TS because only ranges for objective function values were reported in Paula (2012).

With respect to the computation time of the EH algorithm, which on average is less than five minutes and is around 15 times faster than CPLEX, its performance in obtaining optimal solution for around 73% of instances is quite promising. To conclude this section, several points can be highlighted:

- The percent of non-integer variables (column "Non-integer") is a tiny fraction of the total number of variables, in particular, for large instances. This is probably the argument behind strong performance of the EH algorithm.

- The lower bounds are of excellent quality, and very close to the outcomes of the EH algorithm. This is realized through values of column "Gap$_{LB}$". Furthermore, we observed that for all instances except for two instances of "SM" and "4_all", the value of lower

Table 4.2: Summary of the computational results of CPLEX, EH and VNS+TS for solving 11 real-world instances of Min k (α, β)-k Feature Set Problem (FSP).

Criterion	CPLEX	EH	VNS+TS
Percent of feasible solution	100%	100%	100%
Percent of best solution	90.9%	72.7%	33.3%
Percent of optimal solution	90.9%	72.7%	33.3%
Average computation time	3882.30	258.70	30.61
Average gap	0.01	0.02	-

bound is equal to the optimal solution. Additionally, for those two instances the lower bound is within 0.8% of optimality.

- As these 11 real-world instances are unweighted, they are more difficult to solve compared to weighted instances. This is well documented in the literature for the Set Cover Problem (Vasko and Wilson, 1986); inevitably, the same applies to the Min k (α, β)-k FSP. This positively contributes into the already strong role of the EH algorithm in tackling the Min k (α, β)-k FSP.

Table 4.3: Computational results of CPLEX, EH and VNS+TS algorithms for solving 11 real-world instances of Min k (α, β)-k Feature Set Problem (FSP), where $\alpha = \alpha^*$. Columns "CPLEX" refer to the outcomes of the solver CPLEX including the objective function value, computation time in seconds, and optimality gap in %. For the EH algorithm, column "z" is the best objective function value obtained by the algorithm, "Time" denotes the computation time in seconds, and "Gap" is calculated as $\frac{z-z^*}{z^*} \times 100$, where z^* is the best available solution for the Min k (α, β)-k FSP (the optimal solutions are recognized through a gap of zero for CPLEX; we obtained an optimal solution for instance "0_all" through solving a feasibility problem where the number of features were set to 1116). Also, column "Non-integer" shows the percent of fractional variables, and "Gap$_{LB}$" shows the gap between the lower bound, which is obtained by applying Lemma 3.1 and z, and is calculated as $\frac{z-LB}{LB} \times 100$. Columns "VNS+TS" report the outcomes of the Variable Neighborhood Search+Tabu Search proposed by Paula (2012). Because VNS+TS involves randomized elements, the study reported ranges for the objective function value and computation time, rather than single values. VNS+TS is able to obtain only two optimal solutions for the first six instances.

			CPLEX			EH					VNS+TS		
Instance	α^*	z^*	z	Time	Gap	z	Time	Gap	Non-integer	Gap$_{LB}$	z	Time	Gap
ADMF	86	292	**292**	0.57	0	**292**	1.29	0.00	3.06%	0.00%	294.8 ± 0.6	0.98 ± 0.19	-
DS	50	65	**65**	0	0	**65**	0.03	0.00	0.00%	0.00%	**65**	0.04	-
PD1	3970	9807	**9807**	106.59	0	9808	80.36	0.01	0.23%	0.01%	9853.9 ± 3.81	68.92 ± 2.30	-
PD2	760	1265	**1265**	0.11	0	**1265**	0.88	0.00	0.00%	0.00%	**1265**	0.96 ± 0.07	-
PC	229	725	**725**	118.41	0	726	21.94	0.14	1.10%	0.14%	735 ± 1.56	13.63 ± 3.16	-
SM	22	128	**128**	593.12	0	**128**	129.87	0.00	8.00%	0.79%	130.5 ± 0.82	84.57 ± 12.99	-
0_all	354	1116	1117	36000	0.11	**1116**	155.96	0.00	2.69%	0.00%	-	-	-
1_all	683	2220	**2220**	375.16	0	**2220**	284.75	0.00	1.15%	0.00%	-	-	-
2_all	1016	3154	**3154**	1998	0	3155	651.47	0.03	0.89%	0.03%	-	-	-
3_all	1394	4395	**4395**	3305.22	0	**4395**	1368.77	0.00	0.59%	0.00%	-	-	-
4_all	387	1324	**1324**	208.14	0	**1324**	150.33	0.00	1.64%	0.07%	-	-	-
Average				3882.30	0.01		258.70	0.02	1.76%	0.09%		≈30.61	

4.8.2 Computational results of standard instances of the Set Cover Problem

The performance of EH algorithm on 11 real-world instances of Section 4.8.1 is very promising. However, in terms of computational experiments 11 instances may not be enough for the evaluation purpose. Because the Min k (α, β)-k FSP is a variant of the Set k-Cover Problem (SkCP), we further evaluate the EH algorithm on 70 standard instances of the Set Cover Problem (SCP) available from OR Library (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html). We selected these 70 instances because they are standard instances of the SCP and because the studies of Pessoa et al. (2013) and Wang et al. (2016b) report on the same instances (we should mention that the study of Pessoa et al. (2013) included the first 45 instances). Table 4.4 shows basic information regarding these 70 instances.

To obtain an instance of the Min k (α, β)-k FSP per instance of the SCP, we must consider $\alpha \geq 2$. In particular, we considered the following three values for α. The same values were also used in the studies of Pessoa et al. (2013); Wang et al. (2016b)):

- $\alpha_{min} = 2$;
- α_{max} $(\alpha^*) = \min_{i \in I_1} \sum_{j \in J} a_{ij}$; and
- $\alpha_{med} = \lceil (\alpha_{min} + \alpha_{max})/2 \rceil$.

Note that α_{max} is exactly calculated as of Equation (3.3). Over all 70 instances, the first value of α, i.e. $\alpha_{min} = 2$, always results in a feasible solution for the Min k (α, β)-k FSP. In other words, we observed that every element can be covered by at least two features. The second coverage level, which is α_{max} (α^*), states the maximum value of α, which is an instance dependent parameter. Finally, we consider in-between values by setting $\alpha_{med} = \lceil (\alpha_{min} + \alpha_{max})/2 \rceil$. Given three values of α per instance, in total we have 210 settings.

Table 4.5 summarizes the outcomes of solver CPLEX and EH algorithm as well as LA-GRASP and DLL_CCSM algorithms of Pessoa et al. (2013); Wang et al. (2016b) as appeared in those studies. Five criteria of *percent of feasible solution*, *percent of best solution*, *percent of optimal solution*, *average computation time* (in second), and *average gap* (from the best known solution) were used to evaluate each solution method. Details of those computational experiments have been reported in Tables 4.9 to 4.11 (one table is reserved for each value of α). With respect to those outcomes the following observations may be concluded:

- Overall, in terms of solution quality CPLEX has a very promising performance across all tested values for α, in particular, when the value of α increases (the cases where obtaining the minimum number of features becomes more difficult). For example, CPLEX obtains the best solutions for 48.57% and 62.86% of instances for $\alpha = \alpha_{med}$, and $\alpha = \alpha_{max}$.
- While for smaller values of α, the DDL_CCSM outperforms the EH, for larger values of α, and hence, more challenging instances, the performance of EH algorithm is very promising

Table 4.4: Basic information for 70 standard instances of Set Cover Problem (SCP), which are available at OR Library. The table includes the size of each instance class (number of columns (features) and rows (elements)), along with the density and the number of instances in the class.

Class	No. of columns	No. of rows	Density (%)	Number of instances
scp4	1000	200	2	10
scp5	2000	200	2	10
scp6	1000	200	5	5
scpa	3000	300	2	5
scpb	3000	300	5	5
scpc	4000	400	2	5
scpd	4000	400	5	5
scpe	500	50	20	5
scpnre	5000	500	10	5
scpnrf	5000	500	20	5
scpnrg	10000	1000	2	5
scpnrh	10000	1000	5	5

Table 4.5: Summary of the computational results of CPLEX, EH, LAGRASP (Pessoa et al., 2013), and DLL_CCSM (Wang et al., 2016b) for solving 70 standard instances of the Set Cover Problem. The study of Pessoa et al. (2013) includes the first 45 instances.

α	Criterion	CPLEX	EH	LAGRASP	DDL_CCSM
α_{min}	Percent of feasible solution	100.00	100.00	100.00	100.00
	Percent of best solution	88.57	65.71	71.11	100.00
	Percent of optimal solution	85.71	65.71	71.11	85.71
	Average computation time	79.96	88.38	15.33	2.41
	Average gap	0.11	0.18	0.12	0.00
α_{med}	Percent of feasible solution	100.00	100.00	100.00	100.00
	Percent of best solution	48.57	22.86	0.00	51.43
	Percent of optimal solution	31.43	2.86	0.00	12.86
	Average computation time	355.02	318.92	148.22	323.40
	Average gap	0.07	0.11	0.31	0.08
α_{max}	Percent of feasible solution	100.00	100.00	100.00	100.00
	Percent of best solution	62.86	30.00	2.22	25.71
	Percent of optimal solution	35.71	4.29	2.22	12.86
	Average computation time	344.24	314.67	216.56	340.59
	Average gap	0.01	0.05	0.16	0.04

because it obtains best solutions for 30% of instances, as opposed to the DDL_CCSM, which obtains for 25.71% of instances. At the same time, EH is slightly faster than DDL_CCSM.

- When the value of α increases, the LAGRASP algorithm is not able to compete with the other three. Note that because the study of Pessoa et al. (2013) did not report the outcomes of their algorithm for the last 25 instances (starting from instance "scpe1"; note that presented outcomes show that those 25 instances are very challenging) we cannot compare the true performance of their LAGRASP against CPLEX, EH and DDL_CCSM methods. Nevertheless, the weak performance of the LAGRASP algorithm on the first 45 instances is probably a sign of another weak performance for the last 25 instances.

- The DDL_CCSM tends to take loner for larger instances, i.e. from instance "scpnre1" onward.

- Note that while the DDL_CCSM algorithm has a better performance for the first 45 instances, its performance deteriorates for the last 25 instances (see Table 4.10). Indeed, on average the EH algorithm is faster than the DDL_CCSM, particularly, over the last 25 instances. For these instances, the EH algorithm obtains the best solutions for more than 50% of instances.

- For the case of $\alpha = \alpha_{max}$, the EH algorithm competes well against the DDL_CCSM algorithm. Although the performance of both algorithms fluctuates over the first 45 instances, and hence, it is very difficult to compare their performance, for the last 25 instances, the EH algorithm competes well against the DDL_CCSM. For example, the EH obtains 15 best solutions and the DDL_CCSM only obtains 5 best solutions, while the EH has far shorter computation times. In addition to this, for the same instances the performance of the EH against the solver CPLEX is quite promising because CPLEX obtains fewer best solutions within almost the same computation time.

We should state that because the study of Pessoa et al. (2013) used different computation time limits, which are dependent on instance classes (see Table 4.6), and has a maximum of 580 seconds, and also because the maximum computation time of the DDL_CCSM algorithm is around 900 seconds, we used a maximum computation time of 500 seconds for the proposed EH algorithm, and for every instance class and every value of α. However, on the average, the computation time of the EH algorithm is much less than this, and is slightly more than five minutes (318.92 seconds). We believe this provides a fair basis in order to compare the computation time of the algorithms.

Figures 4.2 to 4.4 visualize gap and computation time of four solutions methods of CPLEX, EH, LAGRASP, and DDL_CCSM, and per each value of α. Those figures show that the performance of EH is improving when α takes larger values. Moreover, Figures 4.3 and 4.4

Table 4.6: Maximum computation times (in second), which were used in the LAGRASP algorithm of Pessoa et al. (2013). Their computation time limits depend on the instance classes.

Class	α_{min}	α_{med}	α_{max}
scp4	5	15	27
scp5	10	45	90
scp6	5	20	38
scpa	21	141	265
scpb	17	235	288
scpc	39	329	580
scpd	26	489	544

demonstrate that EH is performing superior than DDL_CCSM for the last 25 instances, particularly, longer computation time of DDL_CCSM does not contribute much into the solutions' quality, and even worse, it reports deteriorated solutions because the values of gap are larger than those of EH.

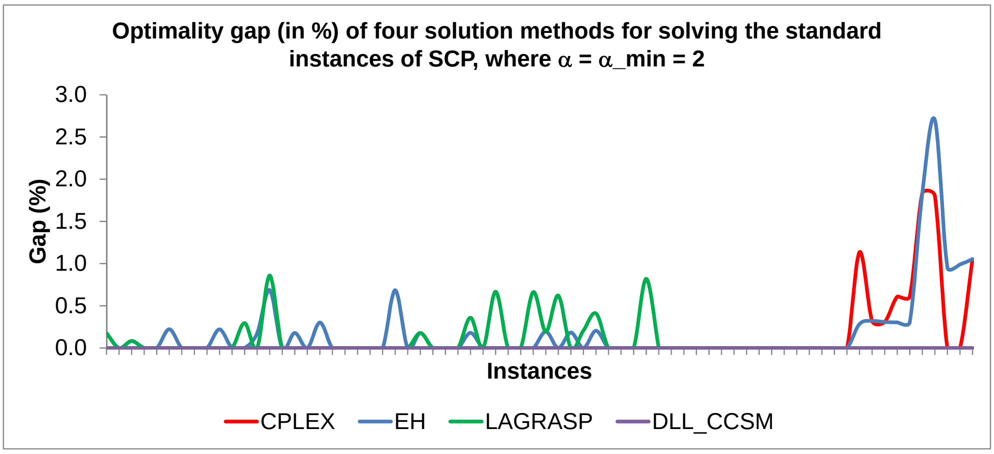
To have a better understanding of each method's individual and pair-wise performance we perform several statistical analysis tests in the following sections.

Statistical analyses of algorithms' gap

We perform a set of pair-wise statistical tests (Coffin and Saltzman, 2000) in order to compare the solution gap and time of four methods of CPLEX, EH, LAGRASP, and DDL_CCSM. When comparing the EH algorithm versus the LAGRASP we only use the outcomes of the first 45 instances of the SCP, out of 70, because the computational experiments of the LAGRASP algorithm are only available for these instances. For more details we refer the interested reader to (Pessoa et al., 2013).

We performed paired-sample t-tests to compare the mean of solution gap between EH and CPLEX, EH and LAGRASP, and EH and DDL_CCSM. The outcomes of those tests are reported in Table 4.7. The paired-sample t-test, which is also known the paired t-test or dependent t-test, determines whether there is a statistically significant difference in the mean of a dependent variable between two groups. Here, the dependent variable is the solution gap, the first group is the EH algorithm, and the second group is one of DDL_CCSM, LAGRASP or the solver CPLEX. Let D be the solution gap mean of pairwise differences, where the value of gap of LAGRASP, DDL_CCSM, or CPLEX is subtracted from that of the EH. Then $D = 0$ indicates that on a randomly chosen test problem two algorithms are likely to perform very closely. Because we have no a priori reason to suppose either algorithm obtains superior solutions, we test $H_0 : D = 0$ versus $H_1 : D \neq 0$. When H_0 is rejected we test H_0 versus either $H_1 : D < 0$ or $H_1 : D > 0$, where $H_1 : D < 0$ tests whether the EH performs better than the other algorithm (because $D < 0$ implies that the EH obtains lower gaps), and $H_1 : D > 0$ tests

Figure 4.2: Performance of gap and computation time of four solution methods of CPLEX, EH, LAGRASP and DDL_CCSM algorithms of Pessoa et al. (2013); Wang et al. (2016b) for solving 70 standard instances of the SCP, where $\alpha = \alpha_{min} = 2$. The study of Pessoa et al. (2013) reported the outcomes for the first 45 instances.



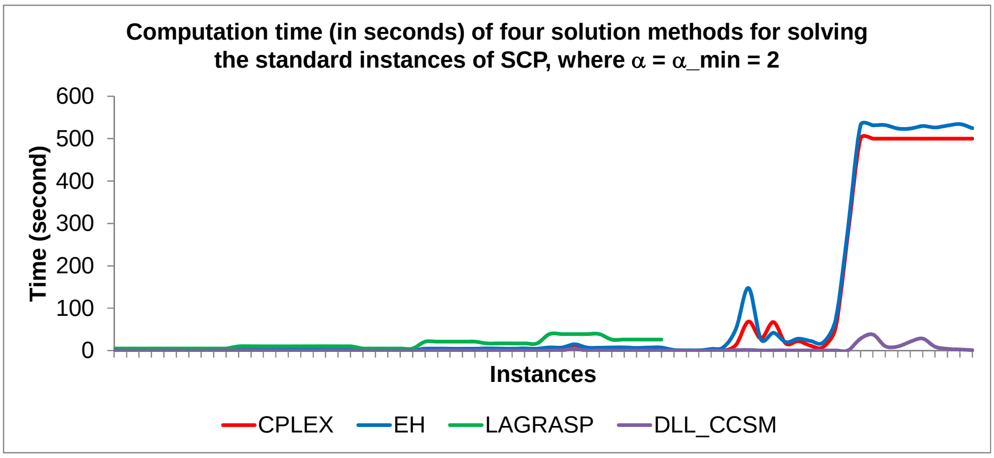
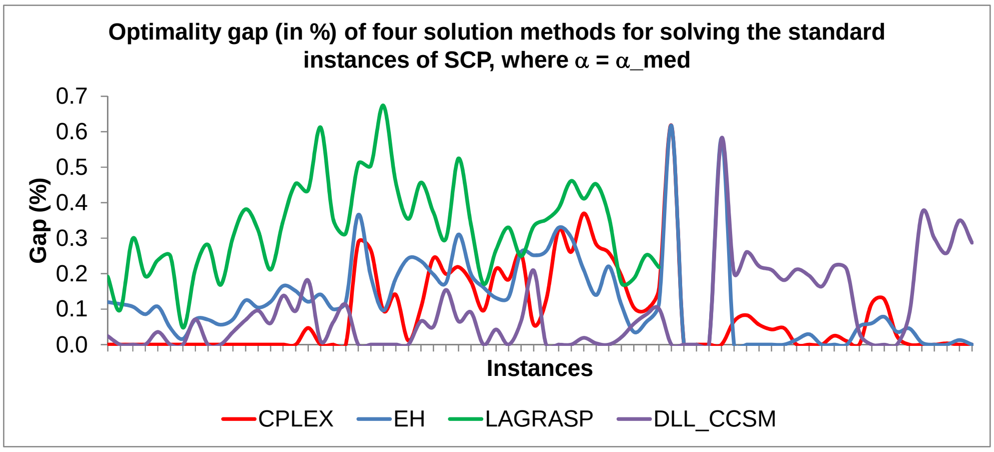


Figure 4.3: Performance of gap and computation time of four solution methods of CPLEX, EH, LAGRASP and DDL_CCSM algorithms of Pessoa et al. (2013); Wang et al. (2016b) for solving 70 standard instances of the SCP, where $\alpha = \alpha_{med}$. The study of Pessoa et al. (2013) reported the outcomes for the first 45 instances.



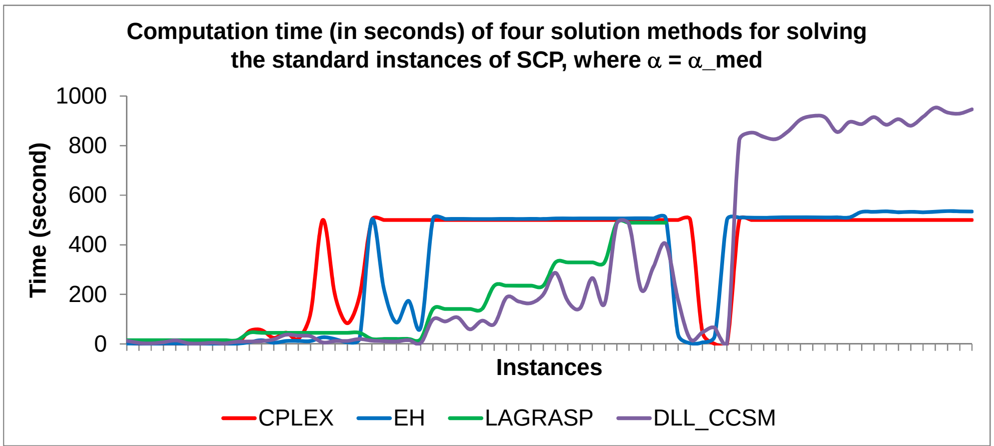
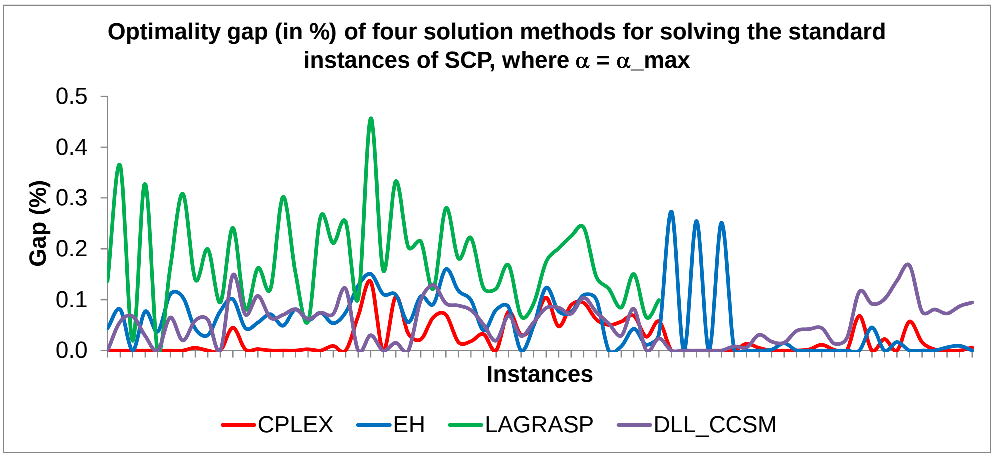
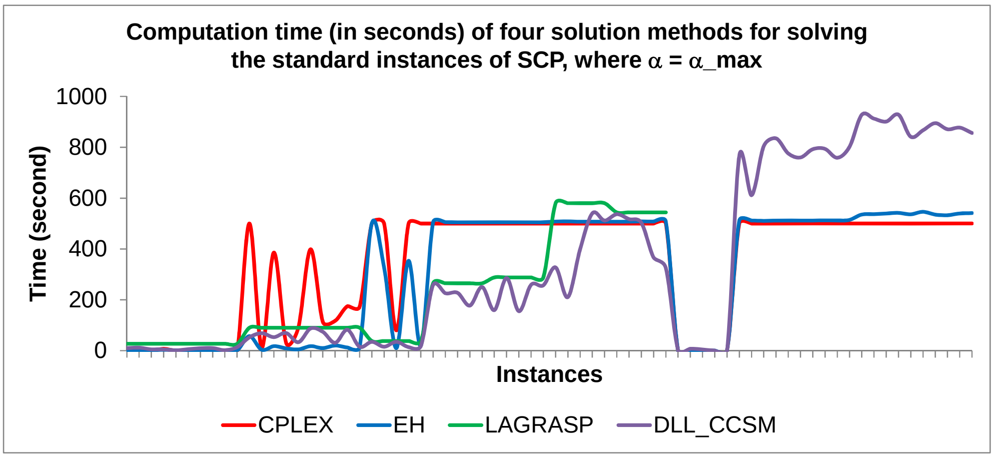


Figure 4.4: Performance of gap and computation time of four solution methods of CPLEX, EH, LAGRASP and DDL_CCSM algorithms of Pessoa et al. (2013); Wang et al. (2016b) for solving 70 standard instances of the SCP, where $\alpha = \alpha_{max}$ (α^*). The study of Pessoa et al. (2013) reported the outcomes for the first 45 instances.





whether the other method performs better. In all tests, we assumed a 95% confidence level; see Box et al. (2005); Neter (1996) for more details. The statistical tests and computations were performed by the software Minitab version 17.2.1 (Minitab, 2015).

Let us first discuss the outcomes where $\alpha = \alpha_{min}$. While Table 4.7 shows no statistically difference in performance between the EH and LAGRASP (p-values of the test shows that both performs equally good), there are differences between the EH and DDL_CCSM, and EH and CPLEX methods. Additional tests revealed that both DDL_CCSM algorithm and solver CPLEX obtain solutions with smaller solution gap mean. Because we suspected that when $\alpha = \alpha_{min}$, instances may be solved by less computational efforts, we performed several additional paired-sample t-tests to verify this hypothesis. The outcomes of those tests demonstrated that none of EH, DDL_CCSM, or LAGRASP perform better than the solver CPLEX. Therefore, the solver CPLEX can obtain very good solutions for these instances. We may conclude that instances with smaller values of α tend to solve by less computational efforts, and although we did not investigate the reason behind this, we believe it must be related to the requirement that every row is needed to be covered by only two columns.

For the case of $\alpha = \alpha_{med}$, the test between the EH and LAGRASP not only shows different performance between two methods, we observed that the EH has lower gap mean than the LAGRASP method because the test H_0 versus $H_1 : D < 0$ has a p-value of 0.000. Also, the test concluded that there is no statistically significant difference between solution gap mean of EH and DDL_CCSM methods, suggesting that the EH algorithm is just as likely as the DDL_CCSM to obtain good quality solutions, which can be interpreted as "the DDL_CCSM method is losing its previous advantages". Moreover, while the p-value of the test between the gap mean of EH and CPLEX is 0.005, we tested H_0 versus $H_1 : D > 0$ and obtained a p-value of 0.003, demonstrating that the solver CPLEX performs better than the EH because it has a lower gap mean. However, according to the Table 4.10 and Figure 4.3 their difference in performance is very small.

Finally, where $\alpha = \alpha_{max}$ (α^*), the test reveals that not only the EH and LAGRASP algorithms have statistically significant different gap means, the p-value associated with the test H_0 versus $H_1 : D < 0$ (which is 0.000) demonstrates that the EH method has a lower mean gap than the LAGRASP method. Also, the test shows that both EH and DDL_CCSM methods equally perform good. In contrast to this, the paired-sample t-test between the gap mean of the EH and CPLEX shows there is a statistically significant difference between the methods. Moreover, additional tests showed that the solver CPLEX obtains lower value for gap mean than both EH and DDL_CCSM methods.

In line with our findings, the tests acknowledge that, compared to our EH algorithm, the DDL_CCSM algorithm loses its performance when the value of α gets larger.

Table 4.7: Outcomes of paired-sample t-tests for comparing solution gap mean between EH and CPLEX, EH and LAGRASP, and EH and DDL_CCSM at a 95% confidence level. Because Pessoa et al. (2013) reported the outcome of the LAGRASP algorithm for the first 45 instances, we used these 45 instances to perform the test between EH and LAGRASP.

α	Method	N	Mean	Standard Deviation	Standard Error Mean	p-value
α_{min}	EH	70	0.1818	0.4378	0.0523	0.023
	CPLEX	70	0.1096	0.3634	0.0434	
	Difference	70	0.0721	0.2597	0.0310	
α_{min}	EH	45	0.0815	0.1588	0.0237	0.282
	LAGRAP	45	0.1229	0.2399	0.0358	
	Difference	45	-0.0413	0.2546	0.0380	
α_{min}	EH	70	0.1818	0.4378	0.0523	0.001
	DDL_CCSM	70	0.0000	0.0000	0.0000	
	Difference	70	0.1818	0.4378	0.0523	
α_{med}	EH	70	0.1211	0.1250	0.0149	0.005
	CPLEX	70	0.0856	0.1207	0.0144	
	Difference	70	0.0355	0.1029	0.0123	
α_{med}	EH	45	0.1544	0.0821	0.0122	0.001
	LAGRASP	45	0.1061	0.1149	0.0171	
	Difference	45	0.0483	0.0886	0.0132	
α_{med}	EH	70	0.1211	0.1250	0.0149	0.163
	DDL_CCSM	70	0.0910	0.1156	0.0138	
	Difference	70	0.0301	0.1786	0.0213	
α_{max}	EH	70	0.05866	0.06183	0.00739	0.000
	CPLEX	70	0.02362	0.03323	0.00397	
	Difference	70	0.03504	0.06371	0.00761	
α_{max}	EH	45	0.0718	0.0391	0.0058	0.000
	LAGRASP	45	0.1766	0.0946	0.0141	
	Difference	45	-0.1048	0.0788	0.0117	
α_{max}	EH	70	0.05866	0.06183	0.00739	0.706
	DDL_CCSM	70	0.05513	0.04177	0.00499	
	Difference	70	0.00353	0.07798	0.00932	

Statistical analyses of algorithms' time

Similar to the statistical tests for pair-wise comparison of solution gap of the methods, we perform a set of pair-wise statistical tests to compare the solution time of the four methods. However, by performing the Normality test, we realized that the computation time of the solution methods is not Normally distributed. Therefore, we chose the Wilcoxon Signed Rank test, which is a nonparametric and a distribution-free test, in order to compare solution times of different methods. The Wilcoxon Signed Rank test is a hypothesis test for the population median where the test statistic is based on counts of positive and negative values. Like before, in all tests we assumed a 95% confidence level, and we used the statistical software Minitab version 17.2.1 to execute the tests. Also, because Pessoa et al. (2013) reports the outcomes of their LAGRASP only for the first 45 instances, we considered this when comparing our EH algorithm versus the LAGRASP algorithm.

For the Wilcoxon Signed Rank test we calculate the pairwise differences in solution times as $d_i = y_i - x_i$, where y_i is the EH solution time on the i-th instance, and x_i is the solution time of one of DDL_CCSM, LAGRASP or CPLEX on the i-th instance. If we observe statistically significant difference between the solution times of two methods, we perform additional tests to investigate which method has a shorter solution time.

The details of the Wilcoxon Signed Rank tests were reported in Table 4.8. According to the table, where $\alpha = \alpha_{min}$, there is statistically difference in computation times between the EH and DDL_CCSM, EH and LAGRASP, and EH and CPLEX methods, and that additional tests showed that the DDL_CCSM and CPLEX are faster than our proposed EH, while the LAGRASP is slower; this is consistent with the previous findings, for example, see Figure 4.2. However, when the value of α increases, the DDL_CCSM method and the solver CPLEX start spending more time to obtain solutions. For example, where $\alpha = \alpha_{med}$, there is no statistically significant difference between the computation time of EH and DDL_CCSM, and EH and CPLEX (in contrast to where $\alpha = \alpha_{min}$). We also observed that when $\alpha = \alpha_{med}$ not only the EH and LAGRASP have different performance, the LAGRASP performs faster. We further investigated the latter and realized that the first 45 instances, out of the 70, are "easier" to solve than the last 25. This impression may be understood by analyzing Figures 4.2 to 4.4. Finally, when $\alpha = \alpha_{max}$, not only there is a statistically significant difference between the computation time of EH and DDL_CCSM methods, the DDL_CCSM method spends more time than the EH (recall that within these computation times we showed that both methods equally perform well, see Table 4.7, while the EH has a superior performance for the last 25 instances). Moreover, the Wilcoxon Signed Rank test used to compare the computation time between EH and the solver CPLEX resulted in a p-value of 0.051, although very close to the critical value of 0.05, it states that there is no statistically significant difference between the computation time of two methods. In spite of observing no statically significant difference between the solution times of EH and LAGRASP methods, comparing the performance of LAGRASP for smaller values of α, it seems that the computation time requirement of LAGRASP is following an incline

Table 4.8: Outcomes of pair-wise Wilcoxon Signed Rank tests for comparing the solution time differences between EH and DDL_CCSM, EH and CPLEX, and EH and LAGRASP at a 95% confidence level. Because Pessoa et al. (2013) reported the outcome of the LAGRASP for the first 45 instances, we used these 45 instances to perform the test between EH and LAGRASP.

α	Solution methods	N	Wilcoxon Statistic	Estimated Median	p-value
α_{min}	EH vs. DDL_CCSM	70	2485.0	7.330	0.000
α_{min}	EH vs. CPLEX	70	2398.5	4.365	0.000
α_{min}	EH vs. LAGRASP	45	0.0	-10.38	0.000
α_{med}	EH vs. DDL_CCSM	70	1184.0	-3.280	0.734
α_{med}	EH vs. CPLEX	70	1472.0	3.610	0.180
α_{med}	EH vs. LAGRASP	45	775.0	92.91	0.004
α_{max}	EH vs. DDL_CCSM	70	905.0	-18.08	0.049
α_{max}	EH vs. CPLEX	70	1576.0	4.015	0.051
α_{max}	EH vs. LAGRASP	45	507.0	-21.70	0.910

trend. Therefore, one may realize that the LAGRASP tends to be slower when the value of α increases.

Considering shorter solution time of EH algorithm compared to the DDL_CCSM (see Table 4.8) one may prefer the EH.

The performed statistical tests further validate our earlier conclusions about the performance of algorithms. In conclusion,

- for the smaller values of α, the DDL_CCSM and solver CPLEX are faster than the EH algorithm;
- when the value of α gets larger, pair-wise comparisons between the EH and DDL_CCSM, and the EH and CPLEX reveal that the DDL_CCSM and CPLEX lose either solution's quality or the computation time performance;
- when the value of α increases, the EH algorithm is superior than the DDL_CCSM and LAGRASP heuristics; and,
- for the last 25 instances of the SCP, out of 70, which seems to be very difficult, the EH algorithm obtains more new best solutions than both CPLEX, and the DDL_CCSM, and that in a shorter time.

Table 4.9: Computational results of CPLEX, EH, LAGRASP (Pessoa et al., 2013), and DDL_CCSM (Wang et al., 2016b) algorithms for solving 70 standard instances of the SkCP, where $\alpha = \alpha_{min} = 2$ (Pessoa et al. (2013) reported the outcomes for the first 45 instances; hence, "-" means no outcome is available). Each method reports the objective function value ("z"; the best values are highlighted), computation time in second, ad gap in % calculated as $\frac{z-z^*}{z^*} \times 100$, where z^* is the best known objective function value. The CPLEX and EH algorithm were allowed to run for 500 seconds, and the computation time of LAGRASP and DDL_CCSM were extracted from their studies.

			CPLEX			EH			LAGRASP			DDL_CCSM		
Instance	α	z^*	z	Time	Gap	z	Time	Gap	z	Time	Gap	z	Time	Gap
scp41	2	1148	**1148**	0.07	0.00	1150	1.03	0.17	1150	5	0.17	**1148**	0.35	0.00
scp42	2	1205	**1205**	0.01	0.00	**1205**	0.49	0.00	**1205**	5	0.00	**1205**	0.02	0.00
scp43	2	1213	**1213**	0.03	0.00	1214	1.09	0.08	1214	5	0.08	**1213**	0.12	0.00
scp44	2	1185	**1185**	0.02	0.00	**1185**	1	0.00	**1185**	5	0.00	**1185**	0.03	0.00
scp45	2	1266	**1266**	0.03	0.00	**1266**	1.05	0.00	**1266**	5	0.00	**1266**	0.38	0.00
scp46	2	1349	**1349**	0.04	0.00	1352	1.02	0.22	**1349**	5	0.00	**1349**	0.12	0.00
scp47	2	1115	**1115**	0.01	0.00	**1115**	1.01	0.00	**1115**	5	0.00	**1115**	0.02	0.00
scp48	2	1225	**1225**	0.09	0.00	**1225**	1.08	0.00	**1225**	5	0.00	**1225**	0.04	0.00
scp49	2	1485	**1485**	0.01	0.00	**1485**	0.98	0.00	**1485**	5	0.00	**1485**	0.06	0.00
scp410	2	1356	**1356**	0.02	0.00	1359	1.02	0.22	**1356**	5	0.00	**1356**	0.8	0.00
scp51	2	579	**579**	0.03	0.00	**579**	2.15	0.00	**579**	10	0.00	**579**	0.09	0.00
scp52	2	677	**677**	0.14	0.00	**677**	2.43	0.00	679	10	0.30	**677**	0.74	0.00
scp53	2	574	**574**	0.04	0.00	575	2.23	0.17	**574**	10	0.00	**574**	0.11	0.00
scp54	2	582	**582**	0.11	0.00	586	2.63	0.69	587	10	0.86	**582**	0.13	0.00
scp55	2	550	**550**	0.03	0.00	**550**	2.18	0.00	**550**	10	0.00	**550**	0.06	0.00
scp56	2	560	**560**	0.04	0.00	561	2.37	0.18	**560**	10	0.00	**560**	0.03	0.00

scp57	2	695	**695**	0.03	0.00	**695**	2.23	0.00	**695**	10	0.00	**695**	0.06	0.00
scp58	2	662	**662**	0.03	0.00	664	2.51	0.30	**662**	10	0.00	**662**	0.14	0.00
scp59	2	687	**687**	0.05	0.00	**687**	2.37	0.00	**687**	10	0.00	**687**	0.23	0.00
scp510	2	672	**672**	0.03	0.00	**672**	2.26	0.00	**672**	10	0.00	**672**	0.16	0.00
scp61	2	283	**283**	0.11	0.00	**283**	1.09	0.00	**283**	5	0.00	**283**	0.01	0.00
scp62	2	302	**302**	0.06	0.00	**302**	1.07	0.00	**302**	5	0.00	**302**	0.01	0.00
scp63	2	313	**313**	0.04	0.00	**313**	1	0.00	**313**	5	0.00	**313**	0.02	0.00
scp64	2	292	**292**	0.07	0.00	294	1	0.68	**292**	5	0.00	**292**	0.06	0.00
scp65	2	353	**353**	0.14	0.00	**353**	1.19	0.00	**353**	5	0.00	**353**	0.02	0.00
scpa1	2	562	**562**	0.58	0.00	563	4.36	0.18	563	21	0.18	**562**	0.16	0.00
scpa2	2	560	**560**	0.33	0.00	**560**	4.69	0.00	**560**	21	0.00	**560**	0.17	0.00
scpa3	2	524	**524**	0.26	0.00	**524**	4.25	0.00	**524**	21	0.00	**524**	0.16	0.00
scpa4	2	527	**527**	0.26	0.00	**527**	4.33	0.00	**527**	21	0.00	**527**	0.62	0.00
scpa5	2	557	**557**	0.16	0.00	558	4.4	0.18	559	21	0.36	**557**	0.13	0.00
scpb1	2	149	**149**	1.98	0.00	**149**	5.06	0.00	**149**	17	0.00	**149**	0.11	0.00
scpb2	2	150	**150**	0.47	0.00	**150**	4.81	0.00	151	17	0.67	**150**	0.1	0.00
scpb3	2	165	**165**	0.35	0.00	**165**	4.47	0.00	**165**	17	0.00	**165**	0.14	0.00
scpb4	2	157	**157**	0.64	0.00	**157**	5.09	0.00	**157**	17	0.00	**157**	0.09	0.00
scpb5	2	151	**151**	0.38	0.00	**151**	4.43	0.00	152	17	0.66	**151**	0.04	0.00
scpc1	2	514	**514**	1.06	0.00	515	7.55	0.19	515	39	0.19	**514**	0.31	0.00
scpc2	2	483	**483**	0.96	0.00	**483**	7.21	0.00	486	39	0.62	**483**	0.79	0.00
scpc3	2	544	**544**	12.84	0.00	545	14.91	0.18	**544**	39	0.00	**544**	3.99	0.00
scpc4	2	484	**484**	0.57	0.00	**484**	7.28	0.00	485	39	0.21	**484**	0.24	0.00
scpc5	2	488	**488**	1.32	0.00	489	6.61	0.20	490	39	0.41	**488**	0.29	0.00

scpd1	2	122	**122**	0.94	0.00	**122**	7.34	0.00	**122**	26	0.00	**122**	0.19	0.00
scpd2	2	127	**127**	0.69	0.00	**127**	7.58	0.00	**127**	26	0.00	**127**	0.06	0.00
scpd3	2	138	**138**	0.61	0.00	**138**	6.22	0.00	**138**	26	0.00	**138**	0.11	0.00
scpd4	2	122	**122**	1	0.00	**122**	7.24	0.00	123	26	0.82	**122**	0.1	0.00
scpd5	2	130	**130**	0.88	0.00	**130**	7.23	0.00	**130**	26	0.00	**130**	0.09	0.00
scpe1	2	9	**9**	0.84	0.00	**9**	1.32	0.00	-	-	-	**9**	0.01	0.00
scpe2	2	8	**8**	0.06	0.00	**8**	0.5	0.00	-	-	-	**8**	0.01	0.00
scpe3	2	8	**8**	0.15	0.00	**8**	0.6	0.00	-	-	-	**8**	0.01	0.00
scpe4	2	8	**8**	0.03	0.00	**8**	3.76	0.00	-	-	-	**8**	0.01	0.00
scpe5	2	8	**8**	0.06	0.00	**8**	8.24	0.00	-	-	-	**8**	0.01	0.00
scpnre1	2	49	**49**	13.97	0.00	**49**	52.22	0.00	-	-	-	**49**	1.14	0.00
scpnre2	2	51	**51**	68.52	0.00	**51**	147.61	0.00	-	-	-	**51**	1.56	0.00
scpnre3	2	47	**47**	28.38	0.00	**47**	26.09	0.00	-	-	-	**47**	0.14	0.00
scpnre4	2	49	**49**	67.22	0.00	**49**	42.04	0.00	-	-	-	**49**	0.22	0.00
scpnre5	2	49	**49**	16.51	0.00	**49**	19.87	0.00	-	-	-	**49**	0.18	0.00
scpnrf1	2	22	**22**	22.17	0.00	**22**	28.04	0.00	-	-	-	**22**	0.43	0.00
scpnrf2	2	24	**24**	11.38	0.00	**24**	22.5	0.00	-	-	-	**24**	0.25	0.00
scpnrf3	2	23	**23**	8.3	0.00	**23**	18.93	0.00	-	-	-	**23**	0.22	0.00
scpnrf4	2	22	**22**	54.03	0.00	**22**	72.51	0.00	-	-	-	**22**	0.28	0.00
scpnrf5	2	21	**21**	277.01	0.00	**21**	289.86	0.00	-	-	-	**21**	0.43	0.00
scpnrg1	2	352	356	500.05	1.14	353	531.49	0.28	-	-	-	**352**	28.12	0.00
scpnrg2	2	311	312	500.05	0.32	312	531.49	0.32	-	-	-	**311**	37.81	0.00
scpnrg3	2	325	326	500.05	0.31	326	531.97	0.31	-	-	-	**325**	10.3	0.00
scpnrg4	2	329	331	500.05	0.61	330	523.85	0.30	-	-	-	**329**	9.66	0.00

scpnrg5	2	327	329	500.05	0.61	328	523.7	0.31	-	-	-	**327**	20.98	0.00
scpnrh1	2	109	111	500.09	1.83	111	529.83	1.83	-	-	-	**109**	28.38	0.00
scpnrh2	2	111	113	500.09	1.80	114	526.43	2.70	-	-	-	**111**	9.09	0.00
scpnrh3	2	105	**105**	500.09	0.00	106	530.88	0.95	-	-	-	**105**	4.11	0.00
scpnrh4	2	101	**101**	500.09	0.00	102	534.32	0.99	-	-	-	**101**	2.47	0.00
scpnrh5	2	95	96	500.09	1.05	96	524.73	1.05	-	-	-	**95**	0.97	0.00
Average				79.96	0.11		88.38	0.18		15.33	0.12		2.41	0.00

Table 4.10: Computational results of CPLEX, EH, LAGRASP (Pessoa et al., 2013), and DDL_CCSM (Wang et al., 2016b) algorithms for solving 70 standard instances of the SkCP, where $\alpha = \alpha_{med}$ (Pessoa et al. (2013) reported the outcomes for the first 45 instances; hence, "-" means no outcome is available). Each method reports the objective function value ("z"; the best values are highlighted), computation time in second, and gap in % calculated as $\frac{z-z^*}{z^*} \times 100$, where z^* is the best known objective function value. The CPLEX and EH algorithm were allowed to run for 500 seconds, and the computation time of LAGRASP and DDL_CCSM were extracted from their studies.

			CPLEX			EH			LAGRASP			DDL_CCSM		
Instance	α	z^*	z	Time	Gap	z	Time	Gap	z	Time	Gap	z	Time	Gap
scp41	7	8350	**8350**	4.8	0.00	8360	1.69	0.12	8366	15	0.19	8352	12.61	0.02
scp42	6	6111	**6111**	0.62	0.00	6118	1.3	0.11	6117	15	0.10	**6111**	4.13	0.00
scp43	5	4676	**4676**	0.15	0.00	4681	1.23	0.11	4690	15	0.30	**4676**	2.28	0.00
scp44	5	4670	**4670**	0.39	0.00	4674	1.32	0.09	4679	15	0.19	**4670**	6.86	0.00
scp45	7	8389	**8389**	0.25	0.00	8398	1.37	0.11	8409	15	0.24	8392	14.15	0.04
scp46	6	6416	**6416**	1.47	0.00	6419	1.65	0.05	6432	15	0.25	**6416**	2.8	0.00
scp47	6	6281	**6281**	0.07	0.00	6282	1.16	0.02	6284	15	0.05	**6281**	1.54	0.00
scp48	7	8421	**8421**	1.09	0.00	8427	1.37	0.07	8439	15	0.21	8427	4.58	0.07
scp49	6	7101	**7101**	0.68	0.00	7106	1.42	0.07	7121	15	0.28	**7101**	2.27	0.00
scp410	5	5355	**5355**	0.17	0.00	5358	1.21	0.06	5364	15	0.17	**5355**	8.51	0.00
scp51	13	11205	**11205**	51.54	0.00	11213	6.61	0.07	11239	45	0.30	11209	9.77	0.04
scp52	14	14418	**14418**	55.64	0.00	14436	15	0.12	14473	45	0.38	14428	11.24	0.07
scp53	13	11476	**11476**	24.39	0.00	11488	4.87	0.10	11513	45	0.32	11487	18.2	0.10
scp54	12	9944	**9944**	45.74	0.00	9956	11.65	0.12	9965	45	0.21	9950	37.09	0.06
scp55	12	10880	**10880**	20.11	0.00	10898	12.19	0.17	10918	45	0.35	10895	33.41	0.14
scp56	12	10581	**10581**	123.4	0.00	10597	11.55	0.15	10629	45	0.45	10591	30.96	0.09

scp57	14	14919	14926	500.01	0.05	14937	26.86	0.12	14984	45	0.44	14946	6.05	0.18
scp58	12	10622	**10622**	196.16	0.00	10637	19.54	0.14	10687	45	0.61	10623	11.75	0.01
scp59	12	11042	**11042**	83.43	0.00	11053	7.09	0.10	11081	45	0.35	11049	12.07	0.06
scp510	13	12436	**12436**	193.33	0.00	12451	21.67	0.12	12475	45	0.31	12450	19.59	0.11
scp61	17	7653	7675	500	0.29	7681	501.16	0.37	7692	20	0.51	**7653**	13.24	0.00
scp62	16	6739	6757	500	0.27	6752	219.51	0.19	6773	20	0.50	**6739**	11.32	0.00
scp63	18	8309	8317	500	0.10	8317	86.55	0.10	8365	20	0.67	**8309**	9.74	0.00
scp64	18	8546	8558	500	0.14	8562	173.7	0.19	8585	20	0.46	**8546**	15.25	0.00
scp65	18	9038	9039	500.01	0.01	9060	67.79	0.24	9070	20	0.35	**9038**	4.01	0.00
scpa1	21	21227	21249	500.01	0.10	21277	504.53	0.24	21324	141	0.46	21241	99.36	0.07
scpa2	21	21739	21792	500.01	0.24	21782	503.94	0.20	21820	141	0.37	21750	90.84	0.05
scpa3	21	20095	20135	500.01	0.20	20130	504.36	0.17	20155	141	0.30	20126	107.08	0.15
scpa4	22	22865	22915	500.01	0.22	22936	503.89	0.31	22985	141	0.52	22880	59	0.07
scpa5	20	18643	18676	500.01	0.18	18680	503.94	0.20	18706	141	0.34	18660	93.74	0.09
scpb1	61	29184	29212	500.01	0.10	29231	503.96	0.16	29234	235	0.17	**29184**	79.85	0.00
scpb2	60	28112	28172	500.01	0.21	28149	504.6	0.13	28187	235	0.27	28124	187.65	0.04
scpb3	59	27852	27903	500.01	0.18	27889	504.08	0.13	27944	235	0.33	**27852**	171.15	0.00
scpb4	58	25678	25744	500.01	0.26	25745	504.32	0.26	25742	235	0.25	25695	164.97	0.07
scpb5	60	28203	28219	500.01	0.06	28274	504.26	0.25	28297	235	0.33	28262	199.5	0.21
scpc1	30	32648	32689	500.01	0.13	32734	506.39	0.26	32763	329	0.35	**32648**	286.86	0.00
scpc2	31	32745	32851	500.01	0.32	32853	506.43	0.33	32871	329	0.38	**32745**	172.7	0.00
scpc3	31	34451	34541	500.01	0.26	34555	506.46	0.30	34610	329	0.46	**34451**	144	0.00
scpc4	30	31366	31482	500.01	0.37	31432	506.1	0.21	31495	329	0.41	31372	265.82	0.02
scpc5	29	30060	30145	500.01	0.28	30102	506.38	0.14	30196	329	0.45	30061	161.68	0.00

scpd1	82	38991	39092	500.02	0.26	39077	506.56	0.22	39132	489	0.36	**38991**	484.71	0.00
scpd2	83	39030	39106	500.02	0.19	39074	506.38	0.11	39098	489	0.17	39038	482.25	0.02
scpd3	81	39198	39239	500.02	0.10	39212	507.09	0.04	39271	489	0.19	39221	218.23	0.06
scpd4	82	38781	38819	500.02	0.10	38806	506.56	0.06	38879	489	0.25	38814	311.55	0.09
scpd5	83	40321	40384	500.02	0.16	40367	506.9	0.11	40409	489	0.22	40362	403.47	0.10
scpe1	40	162	163	500	0.62	163	39.12	0.62	-	-	-	**162**	178.44	0.00
scpe2	40	158	**158**	500	0.00	**158**	2.69	0.00	-	-	-	**158**	19.58	0.00
scpe3	42	166	**166**	46.16	0.00	**166**	6.23	0.00	-	-	-	**166**	46.87	0.00
scpe4	44	177	**177**	0.18	0.00	**177**	32.19	0.00	-	-	-	**177**	65.42	0.00
scpe5	43	172	**172**	0.02	0.00	173	500.46	0.58	-	-	-	173	4.13	0.58
scpnre1	224	59249	59288	500.04	0.07	**59249**	509.8	0.00	-	-	-	59369	821.12	0.20
scpnre2	225	57866	57914	500.04	0.08	**57866**	509.64	0.00	-	-	-	58017	852.59	0.26
scpnre3	224	57041	57073	500.04	0.06	**57041**	509.06	0.00	-	-	-	57167	835.23	0.22
scpnre4	223	56086	56110	500.05	0.04	**56086**	510.4	0.00	-	-	-	56204	826.51	0.21
scpnre5	224	56189	56215	500.04	0.05	**56189**	510.73	0.00	-	-	-	56291	858.28	0.18
scpnrf1	460	57034	**57034**	500.08	0.00	57042	510.76	0.01	-	-	-	57155	905.25	0.21
scpnrf2	458	58174	**58174**	500.08	0.00	58191	511.06	0.03	-	-	-	58287	919.42	0.19
scpnrf3	461	58575	**58575**	500.08	0.00	**58575**	509.94	0.00	-	-	-	58671	914.18	0.16
scpnrf4	463	59780	59795	500.08	0.03	**59780**	510.68	0.00	-	-	-	59913	854.98	0.22
scpnrf5	462	61548	61554	500.08	0.01	**61548**	510.56	0.00	-	-	-	61678	895.49	0.21
scpnrg1	78	89903	**89903**	500.06	0.00	89949	532.75	0.05	-	-	-	89933	887.01	0.03
scpnrg2	78	88380	88483	500.05	0.12	88433	532.76	0.06	-	-	-	**88380**	915.04	0.00
scpnrg3	78	89303	89417	500.05	0.13	89373	535.11	0.08	-	-	-	**89303**	884.36	0.00
scpnrg4	79	94045	94068	500.05	0.02	94079	531.32	0.04	-	-	-	**94045**	906.75	0.00

scpnrg5	79	92262	**92262**	500.05	0.00	92304	532.98	0.05	-	-	-	92342	880.56	0.09
scpnrh1	220	110892	**110892**	500.09	0.00	110898	531.15	0.01	-	-	-	111303	915.91	0.37
scpnrh2	220	112584	**112584**	500.09	0.00	**112584**	533.39	0.00	-	-	-	112921	953.42	0.30
scpnrh3	220	110262	110266	500.09	0.00	**110262**	536.13	0.00	-	-	-	110547	933.63	0.26
scpnrh4	220	109963	**109963**	500.09	0.00	109977	534.97	0.01	-	-	-	110348	929.54	0.35
scpnrh5	219	109275	**109275**	500.09	0.00	**109275**	534.09	0.00	-	-	-	109589	946.25	0.29
Average				355.02	0.09		318.92	0.12		148.22	0.33		323.40	0.09

Table 4.11: Computational results of CPLEX, EH, LAGRASP (Pessoa et al., 2013), and DDL_CCSM (Wang et al., 2016b) algorithms for solving 70 standard instances of the SkCP, where $\alpha = \alpha_{max}$ (α^*) (Pessoa et al. (2013) reported the outcomes for the first 45 instances; hence, "-" means no outcome is available). Each method reports the objective function value ("z"; the best values are highlighted), computation time in second, and gap in % calculated as $\frac{z-z^*}{z^*} \times 100$, where z^* is the best known objective function value. The CPLEX and EH algorithm were allowed to run for 500 seconds, and the computation time of LAGRASP and DDL_CCSM were extracted from their studies.

			CPLEX			EH			LAGRASP			DDL_CCSM		
Instance	α	z^*	z	Time	Gap	z	Time	Gap	z	Time	Gap	z	Time	Gap
scp41	11	18265	**18265**	0.09	0.00	18273	1.16	0.04	18290	27	0.14	**18265**	9.76	0.00
scp42	9	12360	**12360**	4.1	0.00	12370	1.68	0.08	12405	27	0.36	12367	11.54	0.06
scp43	8	10396	**10396**	0.06	0.00	**10396**	1.12	0.00	10398	27	0.02	10403	5.25	0.07
scp44	8	10393	**10393**	7.18	0.00	10401	3.38	0.08	10427	27	0.33	10396	5.21	0.03
scp45	11	18856	**18856**	0.06	0.00	18863	1.22	0.04	**18856**	27	0.00	**18856**	1.02	0.00
scp46	10	15394	**15394**	3.81	0.00	15411	1.76	0.11	15419	27	0.16	15404	5.87	0.06
scp47	10	15233	**15233**	1.09	0.00	15249	1.33	0.11	15280	27	0.31	15236	9.37	0.02
scp48	11	18602	18603	0.59	0.01	18610	1.24	0.04	18628	27	0.14	18613	9.9	0.06
scp49	10	16558	**16558**	0.77	0.00	16563	1.36	0.03	16591	27	0.20	**16568**	2.05	0.06
scp410	8	11607	**11607**	0.37	0.00	11616	1.34	0.08	11618	27	0.09	**11607**	13.99	0.00
scp51	24	35663	35679	500.01	0.04	35699	57.08	0.10	35749	90	0.24	35716	52.27	0.15
scp52	26	45396	45397	12.65	0.00	45416	3.57	0.04	45433	90	0.08	45428	68.76	0.07
scp53	24	36329	36330	385.63	0.00	36349	17.56	0.06	36388	90	0.16	36368	53.16	0.11
scp54	21	28017	**28017**	30.25	0.00	28037	9	0.07	28051	90	0.12	28035	69.43	0.06
scp55	22	32779	**32779**	91.12	0.00	32795	5.08	0.05	32878	90	0.30	32802	33.24	0.07
scp56	21	29608	**29608**	398.6	0.00	29632	17.6	0.08	29653	90	0.15	29632	87.43	0.08

scp57	25	41930	41931	112.55	0.00	41955	10.06	0.06	41954	90	0.06	41956	73.8	0.06
scp58	22	32320	**32320**	117.43	0.00	32344	20.24	0.07	32405	90	0.26	32344	30.81	0.07
scp59	22	33584	33587	174.65	0.01	33602	12.58	0.05	33655	90	0.21	33608	82.14	0.07
scp510	24	38709	**38709**	172.75	0.00	38738	13.05	0.07	38807	90	0.25	38756	15.36	0.12
scp61	31	23510	23526	500	0.07	23540	501.31	0.13	23534	38	0.10	**23510**	34.62	0.00
scp62	29	19934	19961	500	0.14	19964	326.61	0.15	20025	38	0.46	19940	15.44	0.03
scp63	34	27983	**27983**	79.63	0.00	28014	10	0.11	28027	38	0.16	**27983**	34.78	0.00
scp64	33	26442	26470	500	0.11	26471	353.42	0.11	26530	38	0.33	26446	13.89	0.02
scp65	33	27069	27078	500	0.03	27084	20.48	0.06	27124	38	0.20	**27069**	18.39	0.00
scpa1	40	68522	68537	500.01	0.02	68595	504.98	0.11	68669	265	0.21	68590	258.14	0.10
scpa2	39	65842	65885	500.01	0.07	65902	505.24	0.09	65922	265	0.12	65927	226.02	0.13
scpa3	40	66829	66876	500.01	0.07	66936	504.78	0.16	67016	265	0.28	66891	227.35	0.09
scpa4	41	72334	72346	500.01	0.02	72419	504.72	0.12	72465	265	0.18	72398	177.24	0.09
scpa5	38	60491	60502	500.01	0.02	60551	504.55	0.10	60625	265	0.22	60539	250.3	0.08
scpb1	119	105506	105540	500.01	0.03	105548	505.35	0.04	105636	288	0.12	105560	159.32	0.05
scpb2	118	102921	**102921**	500.01	0.00	103003	504.55	0.08	103046	288	0.12	102941	285.94	0.02
scpb3	115	98280	98355	500.01	0.08	98364	505.38	0.09	98445	288	0.17	98347	155.19	0.07
scpb4	114	93773	93802	500.01	0.03	**93773**	505.16	0.00	93836	288	0.07	93800	259.11	0.03
scpb5	118	102810	102862	500.01	0.05	102860	505.03	0.05	102905	288	0.09	102867	256.99	0.06
scpc1	58	112471	112588	500.01	0.10	112610	508.03	0.12	112667	580	0.17	112565	327.52	0.08
scpc2	59	113916	113970	500.01	0.05	114004	508.4	0.08	114145	580	0.20	114012	210.31	0.08
scpc3	59	117416	117521	500.01	0.09	117505	507.08	0.08	117680	580	0.22	117501	398.16	0.07
scpc4	58	110823	110927	500.01	0.09	110944	507.25	0.11	111091	580	0.24	110938	540.09	0.10
scpc5	56	104439	104503	500.01	0.06	104544	507.27	0.10	104591	580	0.15	104518	511.67	0.08

scpd1	162	144884	144957	500.02	0.05	**144884**	507.28	0.00	145060	544	0.12	144961	536.63	0.05
scpd2	163	144096	144178	500.02	0.06	144108	507.46	0.01	144218	544	0.08	144138	517.41	0.03
scpd3	159	140474	140570	500.02	0.07	140534	508.3	0.04	140685	544	0.15	140589	503.39	0.08
scpd4	162	143488	143527	500.02	0.03	143504	507.72	0.01	143582	544	0.07	**143488**	366.89	0.00
scpd5	163	146307	146391	500.02	0.06	146354	508.17	0.03	146452	544	0.10	146342	325.45	0.02
scpe1	77	366	**366**	1.02	0.00	367	0.56	0.27	-	-	-	**366**	2.34	0.00
scpe2	78	364	**364**	0.01	0.00	**364**	1.28	0.00	-	-	-	**364**	7.69	0.00
scpe3	82	393	**393**	0.57	0.00	394	0.5	0.25	-	-	-	**393**	4.66	0.00
scpe4	85	421	**421**	0.03	0.00	**421**	0.51	0.00	-	-	-	**421**	0.09	0.00
scpe5	83	398	**398**	0.16	0.00	399	0.51	0.25	-	-	-	**398**	0.81	0.00
scpnre1	445	224062	**224062**	500.04	0.00	224068	511.27	0.00	-	-	-	224080	766.27	0.01
scpnre2	447	224973	225003	500.05	0.01	**224973**	511.76	0.00	-	-	-	224987	611.23	0.01
scpnre3	445	220353	220364	500.04	0.00	**220353**	510.6	0.00	-	-	-	220421	804.61	0.03
scpnre4	444	218233	**218233**	500.04	0.00	218237	511.53	0.00	-	-	-	218271	835.09	0.02
scpnre5	445	218361	**218361**	500.04	0.00	218392	511.93	0.01	-	-	-	218395	775.7	0.02
scpnrf1	918	226774	**226774**	500.08	0.00	**226774**	511.47	0.00	-	-	-	226862	759.96	0.04
scpnrf2	914	227573	227578	500.08	0.00	**227573**	511.74	0.00	-	-	-	227669	792.27	0.04
scpnrf3	919	231214	231240	500.08	0.01	**231214**	512.32	0.00	-	-	-	231316	793.58	0.04
scpnrf4	924	235921	235924	500.08	0.00	**235921**	512.07	0.00	-	-	-	235953	758.83	0.01
scpnrf5	922	237911	237913	500.08	0.00	**237911**	514.25	0.00	-	-	-	237969	801.87	0.02
scpnrg1	153	324968	325189	500.05	0.07	**324968**	535.36	0.00	-	-	-	325344	927.32	0.12
scpnrg2	154	328003	**328003**	500.05	0.00	328152	536.78	0.05	-	-	-	328304	912.46	0.09
scpnrg3	154	329440	329514	500.05	0.02	**329440**	539.51	0.00	-	-	-	329773	901.09	0.10
scpnrg4	155	338373	**338373**	500.05	0.00	338429	541.87	0.02	-	-	-	338838	928.29	0.14

scpnrg5	155	334710	334901	500.05	0.06	**334710**	536.47	0.00	-	-	-	335270	841.44	0.17
scpnrh1	438	429560	429631	500.09	0.02	**429560**	546.1	0.00	-	-	-	429888	866.54	0.08
scpnrh2	437	428063	428072	500.1	0.00	**428063**	535.11	0.00	-	-	-	428409	894.69	0.08
scpnrh3	437	424290	**424290**	500.09	0.00	424316	532.75	0.01	-	-	-	424599	870.84	0.07
scpnrh4	437	423090	**423090**	500.09	0.00	423129	539.6	0.01	-	-	-	423458	877.23	0.09
scpnrh5	436	422102	422127	500.09	0.01	**422102**	541.11	0.00	-	-	-	422501	856.12	0.09
Average			344.24	0.02		314.67	0.06		216.56	0.18		340.59	0.06	

Table 4.12: Summary of the performance of four solution methods of VNS+TS, LS, EH, and CPLEX for solving Min k (α, β)-k Feature Set Problem over 125 randomly generated instances. Number of optimal solutions was derived by comparing the solutions obtained by the algorithms with those proven optimal obtained by CPLEX.

Criterion	VNS+TS	LS	EH	CPLEX
Percent of feasible solution	100%	100%	100%	100%
Percent of best solution	20.80%	0.00%	71.20%	65.60%
Percent of optimal solution	0.00%	0.00%	20.00%	20.00%
Average computation time	10.57	70.52	243.52	243.24
Average gap	1.06	13.31	0.80	0.80

4.8.3 Computational results of random instances

Following successful application of the EH algorithm on 11 real-world, and 210 standard instances of the Set k-Cover Problem (SkCP), we are determined to apply the EH algorithm on 125 randomly generated instances by Paula (2012). This allows us to further evaluate the performance of the EH algorithm on large instances, and also to compare the EH algorithm with the Variable Neighborhood Search+Tabu Search (VNS+TS) algorithm proposed by Paula (2012).

This set include 125 randomly generated instances each with 2000 features and two disjoint sets (I_1 and I_2) of 20000 sample pairs (total sample pairs is 40000) and an edge density of 20% (recall that the density of the standard instances of the SkCP is 5% and 10%). Therefore, they can represent case-control datasets with 2000 features and 200 samples. Due to incorporating several parameters to generate the instances, they pose different solution challenges: while some of them can be optimally solved by an exact solver in a few seconds, for majority of them obtaining high quality solutions in a reasonable amount of time is a challenge. Indeed, as we will see in Chapter 5, for these instances exact solvers are unable to obtain even feasible solutions for Max β (α, β)-k Feature Set Problem and Max Cover (α, β)-k Feature Set Problem in a reasonable amount of time.

Table 4.12 summarizes the computational experiments of EH and CPLEX as well as VNS+TS and LS (Local Solver) algorithms as reported in Paula et al. (2016) and Paula (2012) for solving 125 randomly generated instances, where $\alpha = \alpha_{max}$. Here, we only considered $\alpha = \alpha_{max}$ because the previous studies have only considered the same value of α, and therefore, a direct comparison is possible. We set the maximum computation time of both CPLEX and EH to 300 seconds (five minutes).

According to Table 4.12, the EH algorithm outperforms all three methods in terms of solution quality. In particular,

- the EH algorithm obtains best known solutions for more than 71% of instances, while CPLEX obtains for 65.60% of instances, and the VNS+TS algorithm obtains for less than 21% of instances. Interestingly, the LS is quite unable to report any best known solution;

- the EH algorithm has obtained optimal solutions for 20% of instances, while the VNS+TS, and LS failed to report any (this was derived by comparing the solutions obtained by the algorithms with those of proven optimal obtained by CPLEX);

- the worst average gap (from the best known solution) is due to the LS, and the best is due to the EH (and also CPLEX), which are less than 1%. The average gap of the VNS+TS algorithm is greater than 1%;

- while the computation time of EH and CPLEX is almost identical, the EH outperforms CPLEX because it obtains 7 more best known solutions than CPLEX within the same computation time; and,

- although the VNS+TS requires far less computation time than the EH, this does not impact the superiority of the EH. This is because the VNS+TS algorithm stops when no improvement is attainable by the algorithm. Note that we limited the computation time of the EH algorithm to 300 seconds (five minutes), which is quite short with respect to the size of instances.

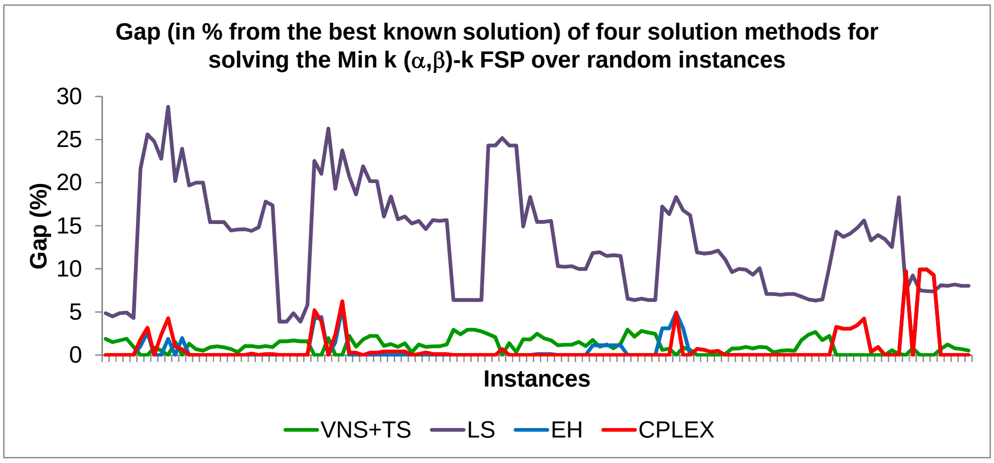
It is also interesting to see how the solutions obtained by EH, VNS+TS, and LS are matched with those obtained by CPLEX, i.e. they have exactly the same objective function values. Evidenced by Table 4.13, solutions obtained by the EH are exactly matched with those obtained by CPLEX for more than 70% of instances. This is however, 0% and less than 1% for the LS and VNS+TS.

Table 4.13: Percent of solutions obtained by EH, VNS+TS, and LS that are exactly matched (have the same objective function value) with CPLEX.

Criterion	VNS+TS	LS	EH
Matched with CPLEX	0.80%	0.00%	70.40%

Figure 4.5 illustrates the values of gap (over the best known solution) of CPLEX, EH, LS, and VNS+TS algorithms for solving those 125 randomly generated instances. From the figure it is not difficult to see that the EH algorithm has the best performance in this regard; the worst performance is due to the LS. Table 4.14 details these outcomes. Here, we reported the objective function value ("z"; the best values are highlighted), computation time in second, and gap (from the best) in % calculated as $\frac{z-z^*}{z^*} \times 100$, where z^* is the best known objective function value. The CPLEX and EH algorithm were allowed to run for 300 seconds, and the computation time of VNS+TS and LS were extracted from Paula (2012) and Paula et al. (2016).

Figure 4.5: Gap of four solution methods of VNS+TS, LS, EH, and CPLEX for solving 125 randomly generated instances of Min k (α, β)-k Feature Set Problem where $\alpha = \alpha_{max}$.



We should add that in contrary to Section 4.8.2 we did not perform the statistical analysis tests for the algorithm's performance because the capability of EH algorithm in comparison to the three methods of VNS+TS, LS, and CPLEX has been well demonstrated through Tables 4.12 and 4.13 as well the detailed computational results depicted in Figure 4.5 and reported in Table 4.14.

Table 4.14: Computational results of VNS+TS algorithm and LS (Local Solver) method as reported in Paula et al. (2016) and Paula (2012), and EH and CPLEX for solving 125 randomly generated instances of Min k (α,β)-k Feature Set Problem, where $\alpha = \alpha_{max}$. Each method reports the objective function value ("z"; the best values are highlighted), computation time in second, and gap in % calculated as $\frac{z-z^*}{z^*} \times 100$, where z^* is the best known objective function value. The CPLEX and EH algorithm were allowed to run for 300 seconds, and the computation time of LS and VNS+TS were extracted from Paula et al. (2016) and Paula (2012). In the outcomes of the CPLEX, "Nodes" refers to the number of nodes left to be explored in the Branch-and-Bound tree, and "Gap_0" is reported by the CPLEX.

		VNS+TS			LS			EH			CPLEX				
Instance	z^*	z	Time	Gap	z	Time	Gap	z	Time	Gap	z	Time	Nodes	Gap_0	Gap
1	535	545	9.76	1.87	561	67.22	4.86	**535**	3.41	0.00	**535**	4.60	0	0.00	0.00
2	535	543	7.11	1.50	559	70.67	4.49	**535**	3.68	0.00	**535**	3.36	0	0.00	0.00
3	535	544	7.92	1.68	561	67.26	4.86	**535**	3.26	0.00	**535**	3.12	0	0.00	0.00
4	529	539	6.97	1.89	555	68.96	4.91	**529**	3.45	0.00	**529**	3.13	0	0.00	0.00
5	535	540	9.82	0.93	558	70.05	4.30	**535**	3.20	0.00	**535**	3.20	0	0.00	0.00
6	384	**384**	13.82	0.00	467	70.84	21.61	388	303.17	1.04	391	303.40	1	21.10	1.82
7	379	**379**	10.52	0.00	476	66.85	25.59	389	303.10	2.64	391	304.83	1	21.10	3.17
8	234	236	4.92	0.85	292	66.66	24.79	**234**	303.38	0.00	**234**	303.02	65	35.07	0.00
9	382	384	10.09	0.52	469	72.96	22.77	**382**	303.13	0.00	391	303.03	1	21.10	2.36
10	375	**375**	11.66	0.00	483	68.28	28.80	382	303.15	1.87	391	303.06	1	21.10	4.27
11	604	613	10.98	1.49	726	68.74	20.20	**604**	303.17	0.00	610	303.07	161	8.42	0.99
12	305	**305**	11.05	0.00	378	71.84	23.93	311	303.21	1.97	307	303.07	28	23.43	0.66
13	610	618	8.97	1.31	730	71.01	19.67	**610**	303.14	0.00	**610**	303.37	151	8.42	0.00
14	610	614	9.61	0.66	732	70.91	20.00	**610**	303.16	0.00	**610**	303.08	151	8.42	0.00
15	610	613	13.51	0.49	732	71.02	20.00	**610**	303.12	0.00	**610**	303.10	150	8.42	0.00

16	882	890	8.14	0.91	1018	72.23	15.42	**882**	303.19	0.00	**882**	303.07	207	4.06	0.00
17	882	891	6.97	1.02	1018	71.53	15.42	**882**	303.28	0.00	**882**	303.04	214	4.06	0.00
18	882	890	9.57	0.91	1018	71.81	15.42	**882**	303.16	0.00	**882**	303.10	201	4.06	0.00
19	885	891	7.46	0.68	1013	72.07	14.46	**885**	303.44	0.00	**885**	303.06	228	4.39	0.00
20	885	888	10.75	0.34	1014	71.65	14.58	**885**	303.23	0.00	**885**	303.06	240	4.39	0.00
21	870	879	7.72	1.03	997	72.29	14.60	**870**	303.13	0.00	**870**	303.10	324	4.10	0.00
22	1055	1066	7.89	1.04	1207	72.26	14.41	**1055**	303.20	0.00	1057	303.10	392	2.92	0.19
23	870	878	7.97	0.92	999	72.37	14.83	**870**	303.17	0.00	**870**	303.08	324	4.10	0.00
24	870	879	6.38	1.03	1025	71.57	17.82	**870**	303.19	0.00	871	303.41	301	4.21	0.11
25	870	878	8.68	0.92	1021	72.28	17.36	**870**	303.15	0.00	871	303.09	308	4.21	0.11
26	568	577	6.29	1.58	590	70.97	3.87	**568**	3.30	0.00	**568**	3.13	0	0.00	0.00
27	568	577	7.84	1.58	590	71.12	3.87	**568**	3.50	0.00	**568**	3.16	0	0.00	0.00
28	536	545	9.44	1.68	562	70.43	4.85	**536**	3.32	0.00	**536**	3.17	0	0.00	0.00
29	568	577	7.03	1.58	590	70.59	3.87	**568**	3.27	0.00	**568**	3.12	0	0.00	0.00
30	568	577	8.72	1.58	601	71.23	5.81	**568**	3.33	0.00	**568**	3.16	0	0.00	0.00
31	404	**404**	11.59	0.00	495	68.10	22.52	421	303.18	4.21	425	303.32	87	23.32	5.20
32	409	**409**	14.32	0.00	495	68.13	21.03	427	304.35	4.40	425	303.16	87	23.32	3.91
33	255	260	9.06	1.96	322	70.33	26.27	**255**	303.51	0.00	**255**	303.06	41	35.93	0.00
34	415	**415**	8.72	0.00	495	70.31	19.28	421	303.44	1.45	425	303.12	88	23.32	2.41
35	400	**400**	20.25	0.00	495	71.94	23.75	422	303.49	5.50	425	303.13	88	23.32	6.25
36	724	740	20.76	2.21	874	71.61	20.72	**724**	303.24	0.00	726	303.10	228	6.10	0.28
37	724	731	12.03	0.97	859	72.02	18.65	**724**	303.30	0.00	726	303.08	235	6.10	0.28
38	726	739	11.02	1.79	885	71.45	21.90	**726**	303.79	0.00	**726**	303.06	234	6.10	0.00
39	724	740	15.68	2.21	870	71.37	20.17	**724**	303.20	0.00	726	303.12	235	6.10	0.28

40	724	740	9.49	2.21	870	72.38	20.17	**724**	303.15	0.00	726	303.09	234	6.10	0.28
41	952	962	9.14	1.05	1105	71.92	16.07	**952**	303.44	0.00	956	303.08	386	3.97	0.42
42	952	964	8.64	1.26	1127	69.30	18.38	**952**	303.20	0.00	956	303.35	391	3.97	0.42
43	952	961	10.90	0.95	1102	71.70	15.76	**952**	303.38	0.00	956	303.08	389	3.97	0.42
44	952	965	9.07	1.37	1105	72.20	16.07	**952**	303.21	0.00	956	303.10	388	3.97	0.42
45	956	959	8.81	0.31	1102	72.14	15.27	**956**	303.55	0.00	**956**	303.10	392	3.97	0.00
46	990	1002	5.58	1.21	1144	71.81	15.56	**990**	303.10	0.00	991	303.08	311	4.08	0.10
47	1053	1063	11.25	0.95	1207	71.63	14.62	**1053**	303.11	0.00	1056	303.05	306	3.44	0.28
48	990	1000	4.99	1.01	1145	71.89	15.66	**990**	303.10	0.00	991	303.15	308	4.08	0.10
49	990	1000	6.87	1.01	1144	71.74	15.56	**990**	303.10	0.00	991	303.36	307	4.08	0.10
50	990	1002	5.83	1.21	1145	71.90	15.66	**990**	303.09	0.00	991	303.15	307	4.08	0.10
51	581	598	11.28	2.93	618	67.78	6.37	**581**	3.31	0.00	**581**	3.19	0	0.00	0.00
52	581	595	12.20	2.41	618	68.34	6.37	**581**	3.54	0.00	**581**	3.22	0	0.00	0.00
53	581	598	10.23	2.93	618	68.10	6.37	**581**	3.33	0.00	**581**	3.18	0	0.00	0.00
54	581	598	12.29	2.93	618	67.83	6.37	**581**	3.33	0.00	**581**	3.22	0	0.00	0.00
55	581	597	13.31	2.75	618	66.65	6.37	**581**	3.36	0.00	**581**	3.20	0	0.00	0.00
56	288	295	7.46	2.43	358	66.72	24.31	**288**	303.13	0.00	**288**	303.17	43	36.90	0.00
57	288	294	12.55	2.08	358	67.77	24.31	**288**	303.16	0.00	**288**	303.10	43	36.90	0.00
58	286	286	12.83	0.00	358	70.33	25.17	**288**	303.47	0.70	**288**	303.16	42	36.90	0.70
59	288	292	10.91	1.39	358	70.48	24.31	**288**	305.17	0.00	**288**	303.13	42	36.90	0.00
60	288	289	15.91	0.35	358	70.08	24.31	**288**	303.52	0.00	**288**	303.34	43	36.90	0.00
61	932	949	13.87	1.82	1071	71.06	14.91	**932**	303.35	0.00	**932**	303.12	201	5.72	0.00
62	834	849	22.62	1.80	987	68.40	18.35	**834**	303.37	0.00	**834**	303.11	138	7.36	0.00
63	932	955	10.03	2.47	1076	70.56	15.45	933	303.70	0.11	**932**	303.09	201	5.72	0.00

64	932	950	14.77	1.93	1076	67.29	15.45	933	303.42	0.11	**932**	303.11	201	5.72	0.00
65	932	948	12.26	1.72	1077	67.31	15.56	933	303.33	0.11	**932**	304.03	199	5.72	0.00
66	1251	1265	12.17	1.12	1380	66.49	10.31	**1251**	303.28	0.00	**1251**	303.11	334	3.80	0.00
67	1251	1266	8.84	1.20	1379	66.58	10.23	**1251**	303.40	0.00	**1251**	303.33	334	3.80	0.00
68	1251	1266	10.17	1.20	1380	71.68	10.31	**1251**	303.44	0.00	**1251**	303.14	333	3.80	0.00
69	1251	1270	8.74	1.52	1376	72.29	9.99	**1251**	303.43	0.00	**1251**	303.13	336	3.80	0.00
70	1251	1264	7.79	1.04	1376	72.12	9.99	**1251**	303.39	0.00	**1251**	303.12	334	3.80	0.00
71	1148	1168	9.14	1.74	1284	71.99	11.85	1161	303.68	1.13	**1148**	303.33	129	5.19	0.00
72	1148	1159	7.94	0.96	1285	71.81	11.93	1161	303.50	1.13	**1148**	303.58	129	5.19	0.00
73	1148	1162	9.10	1.22	1280	72.43	11.50	1161	303.47	1.13	**1148**	303.09	129	5.19	0.00
74	1148	1157	7.65	0.78	1281	71.99	11.59	1161	303.52	1.13	**1148**	303.25	129	5.19	0.00
75	1148	1163	6.16	1.31	1280	72.15	11.50	1161	303.48	1.13	**1148**	303.09	129	5.19	0.00
76	611	629	8.61	2.95	651	68.64	6.55	**611**	3.54	0.00	**611**	3.44	0	0.00	0.00
77	611	624	16.26	2.13	650	69.41	6.38	**611**	3.71	0.00	**611**	3.41	0	0.00	0.00
78	611	628	15.69	2.78	651	67.18	6.55	**611**	3.95	0.00	**611**	3.68	0	0.00	0.00
79	611	627	11.40	2.62	650	69.52	6.38	**611**	3.88	0.00	**611**	3.51	0	0.00	0.00
80	611	626	19.65	2.45	650	69.79	6.38	**611**	3.51	0.00	**611**	3.41	0	0.00	0.00
81	679	683	12.90	0.59	796	70.61	17.23	700	303.36	3.09	**679**	303.12	90	14.18	0.00
82	679	684	16.77	0.74	790	68.25	16.35	700	303.41	3.09	**679**	303.13	85	14.18	0.00
83	507	**507**	12.30	0.00	600	72.73	18.34	532	303.91	4.93	532	303.41	60	26.96	4.93
84	679	685	9.25	0.88	793	67.28	16.79	700	303.52	3.09	**679**	303.12	91	14.18	0.00
85	679	683	16.61	0.59	789	70.49	16.20	**679**	307.27	0.00	**679**	303.36	91	14.18	0.00
86	831	**831**	13.24	0.00	930	68.68	11.91	837	303.38	0.72	837	303.16	81	13.71	0.72
87	832	**832**	14.44	0.00	930	70.89	11.78	837	303.29	0.60	837	303.18	81	13.71	0.60

88	834	**834**	15.35	0.00	933	70.87	11.87	837	304.76	0.36	837	303.16	81	13.71	0.36
89	833	**833**	15.17	0.00	934	66.49	12.12	837	303.98	0.48	837	303.18	81	13.71	0.48
90	837	838	13.84	0.12	930	72.30	11.11	**837**	303.29	0.00	**837**	303.17	81	13.71	0.00
91	1051	1059	11.67	0.76	1152	71.75	9.61	**1051**	304.85	0.00	**1051**	303.07	17	10.52	0.00
92	1051	1059	12.83	0.76	1156	72.21	9.99	**1051**	304.23	0.00	**1051**	303.19	17	10.52	0.00
93	1051	1061	7.54	0.95	1155	71.77	9.90	**1051**	303.77	0.00	**1051**	303.14	17	10.52	0.00
94	1051	1059	12.13	0.76	1149	72.27	9.32	**1051**	303.82	0.00	**1051**	303.09	17	10.52	0.00
95	1051	1061	7.13	0.95	1157	72.30	10.09	**1051**	303.72	0.00	**1051**	303.16	17	10.52	0.00
96	1260	1271	5.52	0.87	1349	71.87	7.06	**1260**	303.59	0.00	**1260**	303.10	17	8.40	0.00
97	1260	1264	6.33	0.32	1349	71.72	7.06	**1260**	303.34	0.00	**1260**	303.14	17	8.40	0.00
98	1260	1266	5.49	0.48	1348	72.26	6.98	**1260**	304.60	0.00	**1260**	303.11	17	8.40	0.00
99	1260	1267	5.88	0.56	1349	72.16	7.06	**1260**	303.90	0.00	**1260**	303.08	17	8.40	0.00
100	1260	1266	9.89	0.48	1349	71.57	7.06	**1260**	303.81	0.00	**1260**	303.13	17	8.40	0.00
101	635	646	12.04	1.73	678	68.13	6.77	**635**	3.66	0.00	**635**	3.21	0	0.00	0.00
102	635	650	10.18	2.36	676	68.07	6.46	**635**	3.29	0.00	**635**	3.23	0	0.00	0.00
103	633	650	13.08	2.69	673	67.35	6.32	**633**	4.67	0.00	**633**	3.65	0	0.00	0.00
104	635	646	11.23	1.73	676	70.50	6.46	**635**	4.23	0.00	**635**	3.22	0	0.00	0.00
105	633	647	16.35	2.21	698	71.17	10.27	**633**	3.83	0.00	**633**	3.47	0	0.00	0.00
106	524	**524**	10.31	0.00	599	70.80	14.31	541	303.40	3.24	541	303.26	50	28.23	3.24
107	525	**525**	9.46	0.00	597	70.77	13.71	541	303.61	3.05	541	303.21	50	28.23	3.05
108	525	**525**	23.34	0.00	599	70.81	14.10	541	303.39	3.05	541	303.14	50	28.23	3.05
109	523	**523**	8.90	0.00	600	67.55	14.72	541	303.37	3.44	541	303.14	50	28.23	3.44
110	519	**519**	12.41	0.00	600	70.16	15.61	541	303.47	4.24	541	303.23	50	28.23	4.24
111	549	**549**	8.76	0.00	622	72.32	13.30	551	304.09	0.36	551	303.80	14	25.17	0.36

112	546	**546**	9.99	0.00	622	71.25	13.92	551	304.04	0.92	551	304.02	14	25.17	0.92
113	551	**551**	9.35	0.00	625	70.83	13.43	**551**	303.73	0.00	**551**	303.46	14	25.17	0.00
114	551	554	7.18	0.54	620	68.15	12.52	**551**	303.32	0.00	**551**	303.80	14	25.17	0.00
115	847	848	15.43	0.12	1002	68.70	18.30	**847**	303.05	0.00	**847**	303.05	20	14.43	0.00
116	1160	1160	18.58	0.00	1247	72.70	7.50	**1273**	303.55	9.74	**1273**	303.22	22	18.84	9.74
117	1138	1147	9.17	0.79	1243	73.58	9.23	**1138**	303.33	0.00	**1138**	303.08	12	10.20	0.00
118	1158	**1158**	10.98	0.00	1245	73.27	7.51	1273	303.94	9.93	1273	303.32	22	18.84	9.93
119	1158	**1158**	8.97	0.00	1244	72.56	7.43	1273	304.03	9.93	1273	303.17	22	18.84	9.93
120	1165	**1165**	6.71	0.00	1251	71.37	7.38	1273	303.68	9.27	1273	303.36	22	18.84	9.27
121	1148	1156	6.89	0.70	1241	73.14	8.10	**1148**	303.34	0.00	**1148**	303.58	0	10.94	0.00
122	1148	1162	4.83	1.22	1240	72.95	8.01	**1148**	303.34	0.00	**1148**	303.40	0	10.94	0.00
123	1148	1157	6.10	0.78	1242	72.04	8.19	**1148**	303.35	0.00	**1148**	303.35	0	10.94	0.00
124	1148	1156	7.12	0.70	1240	71.97	8.01	**1148**	303.33	0.00	**1148**	303.37	0	10.94	0.00
125	1148	1154	6.41	0.52	1240	71.90	8.01	**1148**	303.36	0.00	**1148**	303.37	0	10.94	0.00

4.9 Conclusion

This chapter proposed an efficient exact+heuristic (EH) algorithm for both weighted and unweighted Min k (α, β)-k Feature Set Problem (FSP). The algorithm includes a greedy construction heuristic (mCRCC), an exact component, and one removal local search algorithm (RLS). It is worth emphasizing that not only the EH algorithm is distinct from those used in the previous studies, it utilizes problem-driven local searches, which were developed by exploring mathematical properties of the Min k (α, β)-k FSP, and delivers high quality solutions.

Over a set of 346 tested instances including real-word, weighted and randomly generated instances, the proposed EH algorithm has a very competitive performance. For example, it obtains several new best solutions for the weighted instances of the Set k-Cover Problem. This is further supported by the statistical tests that proved that for larger values of α, and larger and more challenging instances, CPLEX and state-of-the-art algorithms lose either solution's quality or computation time superiority, whereas the EH algorithm obtains larger number of best solutions (about 4% more), and that within the same or less computation time. Also, the fact that the EH algorithm obtains best solutions for more than 50% of large and more challenging instances proves its superiority compared to the state-of-the-art algorithms.

For unweighted and randomly generated instances, the EH algorithm has an excellent solution quality: it has an average gap of less than 1%, and obtains best known solutions for more than 71% of instances, about 6% more than CPLEX, and 50% more than the state-of-the-art algorithms. Moreover, the EH obtains optimal solutions for 20% of instances, while the state-of-the-art algorithms fail to report any optimal solution.

With respect to these outcomes one may conclude that the proposed EH algorithm competes well against the state-of-the-art algorithms, and is capable of delivering high quality solutions for the Min k (α, β)-k FSP, and that in a reasonable amount of time.

Chapter 5

Solution Methods for the Max β and Max Cover (α, β)-k Feature Set Problems

The major outcome of this chapter entitled "An optimization approach towards selecting features in biological datasets" was peer reviewed and accepted for oral presentation at the ASOR 2016 conference in Canberra, Australia, between 16 – 18 November 2016.
The second manuscript entitled "A heuristic algorithm for the (α, β)-k Feature Set Problem" is under preparation to be submitted for an international journal very soon.

Abstract

This chapter develops algorithms and solution methods to solve the Max β and Max Cover (α, β)-k Feature Set Problems (FSPs). We explore both exact and heuristic solution methods for the Max β (α, β)-k FSP. On the exact approaches, an integer programming formulation is solved by utilizing available solvers. Because of the computational complexity of the Max β (α, β)-k FSP and incapability of exact solvers in obtaining even feasible solutions, in particular, for large instances, we propose two pre-processing methods. These methods greatly facilitate exact solvers to obtain feasible, and even optimal solutions for medium sized instances of the Max β (α, β)-k FSP. However, even by utilizing those pre-processing methods solving large instances is still a challenge. Therefore, we propose an exact+heuristic algorithm, which combines both heuristic and exact algorithms in order to solve the Max β (α, β)-k FSP. To the best of our knowledge, the algorithm obtains the best results for the Max β (α, β)-k FSP, and far better than the exact solvers; in particular, it has a very promising performance for large instances. On solving the Max Cover (α, β)-k FSP, we obtain feasible lower bound solutions, and show that those lower bound solutions are within close proximity to upper

bounds. According to the computational experiments of solving 136 instances, which are reported in this chapter, the proposed solution procedures outperform the exact solvers.

5.1 Introduction

Given a set of features and two classes of data, the Max β (α,β)-k Feature Set Problem (FSP) selects a set of minimum cost features, out of a larger set, to explain the dichotomy between the classes, and at least α features do so for each pair of entities of different classes. Indeed, the Max β (α,β)-k FSP maximizes the internal consistency of the entities in the same class. This problem has a broad range of applications including in computational biology. We refer the interested reader to Chapter 3 for a detailed discussion on the Max β (α,β)-k FSP.

Earlier we discussed that the Max β (α,β)-k FSP can be modeled as a variant of the well-known Maximum Satisfiability Problem (MAX-SAT). Because the MAX-SAT is $\mathcal{NP}$-Hard (Karp, 1972; Battiti and Protasi, 1999), the Max β (α,β)-k FSP is $\mathcal{NP}$-Hard as well. In addition to this, the large size of the datasets associated with the Max β (α,β)-k FSP applications adds more difficulty to the problem solving. Therefore, designing and developing efficient algorithms and solution methods for solving the Max β (α,β)-k FSP is of particular importance.

This chapter designs and develops an exact+heuristic (EH) algorithm to solve the Max β (α,β)-k FSP. We first attempt to solve the integer programming formulation of the problem (Model $\mathcal{IP}_{\mathcal{MBP}}$ in Section 3.5.2) by exact solvers, in particular, the solver CPLEX. As expected, exact solvers are not very efficient in solving the Max β (α,β)-k FSP. Nevertheless, the obtained outcomes provide a figure on the difficulty of the Max β (α,β)-k FSP. We improve the incapability of exact solvers by proposing two pre-processing methods, which obtain high quality feasible (initial) solutions for the Max β (α,β)-k FSP, as well as starting solutions for the exact solvers. Those pre-processing methods are direct product of applying the properties discussed in Chapter 3. Also, we will utilize the pre-processing methods in the EH algorithm.

The proposed EH algorithm utilizes both exact and heuristic methods to develop high quality solutions for the Max β (α,β)-k FSP. The EH obtains a set of features that have a high probability of being in an optimal solution of the Max β (α,β)-k FSP, and ensures those features will be included in a feasible solution. Then, the algorithm builds a feasible solution by solving a sub-problem of the original Max β (α,β)-k FSP, which has less features and elements. To the best of our knowledge, and as evidenced by the computational results, this is a very efficient algorithm and is able to deliver high quality solutions over all 136 solved instances of the Max β (α,β)-k FSP, and that in a reasonable amount of time.

This chapter is organized as follows. We provide a short review of the available and relevant studies to this research in Section 5.2. Section 5.3 discusses exact solution methods for the Max β (α,β)-k FSP. Section 5.4 discusses the EH algorithm. As evidenced by the computational results, the EH algorithm outperforms available methods. In Section 5.6 we discuss solution methods for solving the Max Cover (α, β)-k Feature Set Problem (FSP). Finally, the chapter ends with several conclusions.

5.2 Literature review

Cotta et al. (2004) discussed the fundamental ideas of the Max β (α, β)-k Feature Set Problem (FSP). Earlier applications of the Max β (α, β)-k FSP were studied in the work of Berretta et al. (2007), and Berretta et al. (2008). They developed an integer programming formulation for the problem, and utilized the commercial solver CPLEX to solve it. Due to the computational difficulty of solving the Max β (α, β)-k FSP, as well as lack of efficient methods in their studies, they could only solve small and medium sized instances. Therefore, they did not investigate their model on large instances of the Max β (α, β)-k FSP. Several applications of the problem in computational biology and Bioinformatics were investigated by Ravetti and Moscato (2008); Ravetti et al. (2009); Paula et al. (2011). Their major contributions include identifying biomarkers for certain diseases, rather than on the computational side. Later, Paula (2012) developed the first heuristic algorithm for the Max β (α, β)-k FSP. To the best of our knowledge, the study of Paula (2012) is the only available work on developing heuristic solution algorithms for the Max β (α, β)-k FSP.

Because the Max β (α, β)-k FSP can be modeled as a variant of the Maximum Satisfiability Problem (MAX-SAT or MaxSAT) with certain additional constraints, we shall briefly review the most relevant studies on the MAX-SAT, and focus on recent advances, in particular, those related to the algorithm development. Partial MAX-SAT is a generalization of the MAX-SAT with both hard and soft clauses. Several local searches were developed in Cai et al. (2016) for this variant of the MAX-SAT. The main idea of these local searches is separation between hard and soft clauses. Bouhmala (2015) proposed a multilevel learning algorithm for the MAX-SAT. The multilevel paradigm creates a hierarchy of increasingly smaller sub-problems (of the original problem) until the size of the smallest sub-problem falls below a specified threshold. The algorithm generates a solution for the smallest sub-problem, and then projects it back onto each of the intermediate sub-problems. Martins et al. (2015) improved the performance of linear search algorithms for the MAX-SAT. The linear search algorithms start by adding a new relaxation variable to each soft clause and solving the resulting model with a solver. A new constraint on the relaxation variables is added such that models with a less value are kept. Instead of adding a new constraint over variables, the authors added a constraint on a subset of relaxation variables. This approach avoids large number of relaxation variables.

Ansótegui et al. (2016) have extended some of the best performing algorithms for the MAX-SAT by introducing certain improvements. Their main ideas include solving sub-problems of the original MAX-SAT by developing heuristics. Goffinet and Ramanujan (2016) applied Monte-Carlo Tree Search in combination with local search algorithms. Their hybrid algorithm overcame the drawback of the available algorithms, mainly by moving to different areas in the search space. Poloczek and Williamson (2016) reported a thorough analysis on the performance of approximation algorithms for the MAX-SAT. Their major findings are twofold: greedy algorithms often obtain very good solutions at low computational cost, and performance of the deterministic algorithms is better than the randomized greedy algorithms. Following these observations, they proposed a new algorithm that combines greedy and stochastic local searches, and obtained very high quality solutions. Many algorithms for solving the MAX-SAT has a core of a stochastic local search algorithm; Cai et al. (2015) discusses several of these algorithms. Also, Lu and Vasko (2015) applied a stochastic local search algorithm for the weighted MAX-SAT.

Golovnev and Kutzkov (2014) proposed new exact algorithms for a variant of the MAX-SAT, where the input instance is not very sparse, and improved the best known bound. Poloczek et al. (2014), and Escoffier et al. (2012) discussed several approximation algorithms for the MAX-SAT. One approach to solve the MAX-SAT is via solving a sequence of satisfiability/feasibility problems (that is, finding feasible solutions). As expected, these algorithms heavily rely on a solver because they use the solver to obtain feasible solutions. With regard to this, the approaches investigated by Ignatiev et al. (2014) reduce the number of times a satisfiability problem is solved. Also, many variants of the MAX-SAT have been investigated. See for example Petkovska et al. (2016), and Zhang et al. (2016).

According to these approaches for solving the MAX-SAT, one may group them based on their similarities of the applied solution methods, some of which are decomposition-based methods (Cai et al., 2016; Bouhmala, 2015; Martins et al., 2015), heuristic and approximation algorithms (Ansótegui et al., 2016; Poloczek and Williamson, 2016; Poloczek et al., 2014; Escoffier et al., 2012), and solving a sequence of feasibility problems (Ignatiev et al., 2014). Some of these frameworks were used in this research to develop and implement algorithms and solution methods for the Max β (α, β)-k FSP.

5.3 Pre-processing methods

The focus of this section is to utilize available exact solvers, for example CPLEX, to solve the Max β (α, β)-k Feature Set Problem (FSP). This is because exact solvers have tremendously been advanced in the last decade, and such advancements should well be utilized.

We first attempt to solve the Max β (α, β)-k FSP through solving an integer program (IP) by an exact solver. In Section 3.5.2 we discussed such an IP model, and named it Model $\mathcal{IP}_{\mathcal{MBP}}$. Our initial attempt to solve Model $\mathcal{IP}_{\mathcal{MBP}}$ over large instances failed; exact solvers, including the CPLEX are unable to obtain optimal solutions, and even feasible solutions, for large instances of the Max β (α, β)-k FSP. Therefore, we overcome this limitation by developing two pre-processing methods in order to facilitate exact solvers. These methods obtain very good quality feasible solutions for the Max β (α, β)-k FSP, and allow exact solvers to converge much faster.

The proposed pre-processing methods are based on Proposition 3.3, in which we showed that any solution to the Max β (α, β)-k FSP must also be feasible for the Min k (α, β)-k Feature Set Problem (FSP). Moreover, Proposition 3.3 proves that the Max β (α, β)-k FSP is to select the best solution among all optimal solutions of the Min k (α, β)-k FSP, according to the criterion of maximizing β (recall that multiple optimal solutions for the Min k (α, β)-k FSP have the same objective function value, however, as they include different sets of features, they may have different values of β). Lemma 3.2 shows that this solution is also a lower bound solution for the Max β (α, β)-k FSP. Thus, an optimal solution for the Min k (α, β)-k FSP is a feasible lower bound for the Max β (α, β)-k FSP.

Those two processing methods are different in the number of optimal solutions (pool) for the Min k (α, β)-k FSP they utilize. While in the first method $|P| = 1$, and a single feasible lower bound solution for the Max β (α, β)-k FSP is generated, in the second method $|P| \geq 2$, and multiple feasible lower bound solutions are generated. We call the former Initial Method 1 (IM 1), and the latter Initial Method 2 (IM 2). Note that because IM 2 obtains more than one optimal solution for the Min k (α, β)-k FSP, it is more capable of delivering a higher quality feasible lower bound solution for the Max β (α, β)-k FSP than the IM 1.

Algorithm 5.1 summarizes a procedure, in which IM 1 and IM 2 are applied in order to deliver feasible lower bound solutions for the Max β (α, β)-k FSP. Such a feasible lower bound solution may be supplied into exact solvers for optimization.

Algorithm 5.1: The procedure of generating feasible lower bound solution for the Max β (α, β)-k Feature Set Problem (FSP). The procedure works by obtaining a pool of high quality (ideally optimal) solutions for the Min k (α, β)-k Feature Set Problem (FSP). The solution with the maximum value of β is chosen as the feasible lower bound solution.

Input: Model $\mathcal{IP}_{\mathcal{MCFSP}}$, and parameter $|P|$ (cardinality of the pool).

Output: A feasible lower bound solution J^* (a set of features) for the Max β (α, β)-k FSP.

while *stopping condition is not met* **do**

  Construct a pool $P = \{P_1, \ldots, P_p\}$ of high quality (ideally optimal) solutions for the Min k (α, β)-k FSP;

end

Let $J^* \in P$ denote the solution (a set of features) that has the maximum value of β;

Report J^*;

Algorithm 5.1 will be implemented in the EH algorithm in order to generate feasible initial solutions for the Max β (α, β)-k FSP. Several points are worth discussing regarding Algorithm 5.1:

- In order to obtain multiple optimal solutions for the Min k (α, β)-k FSP, Proposition 3.2 can be applied. That is, by iteratively solving the Min k (α, β)-k FSP and ensuring that the so obtained optimal solutions will not be explored one may obtain new optimal solutions for the Min k (α, β)-k FSP, if they exist. Termination with an infeasible status (in a reasonable amount of time) is the proof that all optimal solutions for the Min k (α, β)-k FSP have been explored. This is important because it implies that we are able to obtain the optimal solution for the Max β (α, β)-k FSP (see Proposition 3.5).

- In general, we may not be able to obtain all optimal solutions of the Min k (α, β)-k FSP because obtaining all optimal solutions of an integer program is an $\mathcal{NP}$-Hard problem in its own right. For this reason, we may obtain only p optimal solutions (i.e. $|P| = p$). Another stopping criterion may be the total computation time allocated to obtaining multiple optimal solutions.

- Following the computational difficulty of the Min k (α, β)-k FSP, particularly for large instances, we may not be able obtain even a single optimal solution. In such a case, we can generate a pool of best obtained solutions for the Min k (α, β)-k FSP to construct a feasible lower bound solution for the Max β (α, β)-k FSP. This solution does not lead to the optimal solution for the Max β (α, β)-k FSP because this is not constructed out of an optimal solution of the Min k (α, β)-k FSP.

- In practice, providing a feasible solution for an integer program is very beneficial since it provides a starting point for an exact solver. This is very useful because when the size of an instance of the Max β (α, β)-k FSP is large, even obtaining a feasible solution is very resource demanding, and may take hours; not to mention that the exact solvers may even fail to obtain such a feasible solution. According to the computational results of Sections 5.5.1 and 5.5.2, CPLEX is not able to deliver feasible solutions for large instances of the Max β (α, β)-k FSP.

- For integer programs, exact solvers spend considerable amount of resources on the root relaxation of the Branch-and-Bound algorithm. A feasible solution greatly facilitates the root relaxation process, and decreases the demand for resources.

5.4 An exact+heuristic algorithm

Proposition 3.3 proves that the Max β (α, β)-k Feature Set Problem (FSP) is the problem of selecting the best solution, among all optimal solutions of the Min k (α, β)-k Feature Set Problem (FSP), according to the objective function of maximizing β. Therefore, the optimal solution for the Max β (α, β)-k FSP must lie in the pool of all optimal solutions of the Min k (α, β)-k FSP. We already utilized this property to generate feasible solutions for the Max β (α, β)-k FSP (see Section 5.3). This section utilizes this proposition to design and implement a very efficient heuristic algorithm for the Max β (α, β)-k FSP, which is capable of solving large instances and obtaining very high quality solutions. This algorithm, which we name it the exact+heuristic (EH) algorithm, combines both exact and heuristics to solve the Max β (α, β)-k FSP. The EH algorithm has two major steps. Step 1 generates a feasible (initial) solution, and Step 2 improves this solution. While the first step is heuristically performed, the second step is exactly solved. Thus, the algorithm has the potential to obtain proven optimal solutions for the Max β (α, β)-k FSP. Algorithm 5.2 summarizes the EH algorithm. Both Initial Method 1 (IM 1) and Initial Method 2 (IM 2) have been implemented in Algorithm 5.2.

Algorithm 5.2: The exact+heuristic (EH) algorithm to solve the Max β (α, β)-k Feature Set Problem (FSP). The algorithm starts by constructing a feasible solution, and proceeds with improving this solution until the stopping condition is met.

Input: Models $\mathcal{IP}_{\mathcal{MCFSP}}$ and $\mathcal{IP}_{\mathcal{MBP}}$; a set J of features; sets I_1 and I_2 of elements; parameters α and p.

Output: An improved solution (a set $J^* \subseteq J$ of features) for the Max β (α, β)-k FSP.

Step 1. Constructing a feasible solution.

Apply either Initial Method 1 (IM 1) or Initial Method 2 (IM 2) to construct a feasible solution. Alternatively, apply Algorithm 5.3 for the same purpose.

Step 2. Improving the feasible solution.

while *the stopping condition is not met* **do**

  Apply an exact solver to solve the original Max β (α, β)-k FSP, given the set J^* of features as a starting solution;

  Update J^*;

end

Report J^*;

5.4.1 Initial solutions

Initial solutions for the EH algorithm are constructed through two major procedures. The first procedure is Algorithm 5.1. In the second procedure a pool of $p \in \mathbb{Z}^+$ optimal solutions for the Min k (α, β)-k FSP is obtained. Then, the algorithm utilizes these optimal solutions to construct a partially built solution for the Max β (α, β)-k FSP. This is illustrated in Algorithm 5.3.

From the pool of p optimal solutions for the Min k (α, β)-k FSP, those features that are common across all optimal solutions are extracted, because these features may have a high probability to be in an optimal solution of the Max β (α, β)-k FSP, or at least it can be argued that they are part of a very good quality feasible solution. By obtaining common features across all solutions in the pool, we will have a set of features that can appear in a feasible solution of the Max β (α, β)-k FSP. Let $\tilde{J} \subset J$ denotes this set of features. Set $\tilde{J}$ leads to a partially built solution for the Max β (α, β)-k FSP, which may not be feasible. Therefore, we need to repair it in order to ensure that at least one feasible solution is available for the Max β (α, β)-k FSP.

The feasibility of the partially built solution may be restored by solving a sub-problem of the original Max β (α, β)-k FSP, which has a reduced number of features and elements because a set of features has already been chosen to be in a solution. Indeed, the sub-problem is generated by including the set of available features, i.e. $J \setminus \tilde{J}$, and uncovered elements. The union of the set of features obtained through solving this sub-problem and $\tilde{J}$ forms a feasible

Algorithm 5.3: The procedure of generating a partially built solution for the Max β (α, β)-k Feature Set Problem (FSP). The procedure starts by obtaining a pool of p optimal solutions for the Min k (α, β)-k Feature Set Problem (FSP). From this pool, a partially built solution is constructed by including the set $\tilde{J} \subset J$ of features that are common across all solutions of the pool. If this solution is not feasible for the Max β (α, β)-k FSP, then a sub-problem of the original problem (over the sets of available features and uncovered elements) is solved.

Input: Models $\mathcal{IP}_{\mathcal{MCFSP}}$ and $\mathcal{IP}_{\mathcal{MBP}}$; a set J of features; sets I_1 and I_2 of elements; parameters α and p.

Output: An improved solution (a set $J^* \subseteq J$ of features) for the Max β (α, β)-k FSP.

while *the stopping condition is not met* **do**
    Obtain a pool $P = \{P_1, \ldots, P_p\}$ of optimal solutions for the Min k (α, β)-k FSP, where $|P| = p$;
    Let $\tilde{J} \subset J$ be the set of all features that appear in every solution of pool;
    Construct a partially built solution for the Max β (α, β)-k FSP: $J^* = \tilde{J}$;
end
if J^* *is not feasible* **then**
    Build a sub-problem of the original Max β (α, β)-k FSP by including the set of available features, and uncovered elements;
    Solve the sub-problem; let $\tilde{J}$ be the set of features in the optimal solution of the sub-problem;
    $J^* = J^* \cup \tilde{J}$;
end
Report J^*;

(initial) set of features for the Max β (α, β)-k FSP. Remember from our earlier discussion in Section 5.3 that having a feasible solution is very important when attempting to exactly solve integer and mixed-integer programs, particularly, when the instances are large.

Note that if the size of the sub-problem in Algorithm 5.3 is still very large, and therefore cannot be solved in a short time, recursive applications of the EH algorithm can be performed.

5.4.2 Improved solutions

After generating a feasible solution for the Max β (α, β)-k FSP we improve the solution. The improvement procedure of the EH algorithm exactly solves the original Max β (α, β)-k FSP by providing the so obtained feasible solution as the starting point to an exact solver.

Note that because improvement phase solves the original Max β (α, β)-k FSP it may yield proven optimal solution. Indeed, according to the computational experiments of Sections 5.5.1 and 5.5.2, the EH algorithm delivers very high quality solutions, including optimal, for the real-world and randomly generated instances of the Max β (α, β)-k FSP. Finally, note that by only using an exact solver, for example CPLEX, to solve the Max β (α, β)-k FSP we may not obtain such high quality solutions (this incapability of CPLEX is further discussed in Sections 5.5.1 and 5.5.2). On the other hand, the EH algorithm extensively contributes into solving the Max β (α, β)-k FSP, and improving incapability of CPLEX.

5.5 Computational results

This section reports the computational experiments of applying the exact+heuristic (EH) algorithm, which is presented in Algorithm 5.2, on two sets of instances. All algorithms were implemented in the programming language Python 2.7 via the standard solver CPLEX 12.5.0 Python API. The computing resource has Linux Ubuntu 14.04 LTS operating system with 32 GB of memory and 12 cores of Intel®Xeon CPU E5-1650 at 3.5 GHz. Unless otherwise stated, for all computational experiments we utilize only one thread (processor).

The first set includes 11 real-world unweighted instances ranging from small to large. Section 5.5.1 discusses the computational results of those instances. The second set includes 125 randomly generated unweighted and large instances for the (α, β)-k Feature Set Problem (FSP). Those instances have the same size, however, due to the their generation framework they pose computational challenge for the available solution methods of the Max β (α, β)-k Feature Set Problem (FSP). The computational results of those instances are discussed in Section 5.5.2.

5.5.1 Computational results of real-world instances

This section reports the computational results of the EH algorithm on 11 real-world instances ranging from small to large. Two sets of real-world instances were considered to evaluate the performance of the EH algorithm. The first set, which includes six biological instances ranging from small to large, was previously studied by Paula (2012), and the second set, which includes

Table 5.1: 11 real-world instances of the Max β (α, β)-k Feature Set Problem (FSP), including the size of instances, number of features, and number of entities/samples (of both classes). Columns "$|J|$", "I_1", "I_2", and "α^*" show parameters of the associated Max β (α, β)-k FSP, including number of features, number of elements (pairs of entities) belonging to different classes and to the same class, and the optimal value of parameter α.

Instance	No. of features	No. of entities	$\lvert J\rvert$	$\lvert I_1\rvert$	$\lvert I_2\rvert$	α^*	Reference
ADMF	686	83	686	1720	1683	86	Paula et al. (2011)
DS	73	15	73	56	49	50	Lockstone et al. (2007)
PD1	17099	105	17097	2750	2710	3970	Scherzer et al. (2007)
PD2	1674	25	1674	144	156	760	Lesnick et al. (2007)
PC	3556	171	3,556	7290	7245	229	Chandran et al. (2007)
SM	525	1,219	525	273834	468537	22	Charlesworth et al. (2010)
0_all	1969	450	1969	32400	68625	354	Haque et al. (2016)
1_all	3304	450	3304	32400	68625	683	Haque et al. (2016)
2_all	4243	450	4243	32400	68625	1016	Haque et al. (2016)
3_all	5436	450	5436	32400	68625	1394	Haque et al. (2016)
4_all	2005	450	2005	32400	68625	387	Haque et al. (2016)

five large instances of face recognition, may truly represent actual dimension of the datasets we may encounter in applications of the Max β (α, β)-k FSP, particularly in computational biology. Obtaining optimal solution for the instances of the second set, or even high quality solutions has been a challenge for the exact solvers. Therefore, these instances provide a good test bed to evaluate the performance of the EH algorithm for solving the Max β (α, β)-k FSP. All instances are unweighted (unicost).

The basic information regarding those 11 real-world instances is shown in Table 5.1. The first three columns show the instance name, number of features (which may represent protein, genes, probes, SNPs, etc.), and total number of entities (of both Class 1 and Class 2). In each dataset, we have two classes (groups) of data: Class 1 (e.g. Healthy or Control) and Class 2 (e.g. Disease or Case, see Chapter 2 for more details). The second four columns provide parameters of the Max β (α, β)-k FSP associated with each instance. Here, column "$|J|$" gives the total number of features, column "$|I_1|$" is total number of pairs of entities of different classes, and column "$|I_2|$" is total number of pairs of entities of the same class. Recall from our earlier discussion in Section 2.2 that an element $i \in I_1$ may be obtained by considering every combination of size two of entities of Class 1 and Class 2, and an element $i \in I_2$ may be obtained by considering every combination of size two of entities of the same class. Sets I_1 and I_2 can be obtained by using Equation (2.1) and Equation (2.2), respectively. Column "α^*" shows the optimal value of parameter α, and is derived by using Equation (3.3). Finally, the last column of the table provides additional references for instances.

Table 5.2: Summary of the computational results of EH and CPLEX for solving 11 real-world instances.

Criterion	CPLEX	EH
Percent of feasible solution	81.8%	100%
Percent of best solution	63.6%	100%
Percent of optimal solution	63.6%	90.9%
Average computation time	13823.47	3093.52
Average gap	0.08	0.06

Table 5.2 summarizes the outcomes of CPLEX and EH on those 11 real-world instances of Table 5.1 (we did not report the outcomes of the Variable Neighborhood Search+Tabu Search (VNS+TS) by Paula (2012) because the study obtained slightly larger values for the objective function of the Min k (α,β)-k Feature Set Problem (FSP), and hence, larger values for β, which does not allow us to compare our results against). Five criteria of *percent of feasible solution*, *percent of best solution*, *percent of optimal solution*, *average computation time* (in second), and *average gap* (from the best known solution) were used to evaluate each solution method. As the table illustrates, the EH algorithm outperforms CPLEX in every criterion. Table 5.3 reports the details of these results. With respect to the results reported in Tables 5.2 and 5.3 several points are worth discussing:

- While the EH algorithm obtained feasible and best known solutions for all 11 instances, i.e. for 100% of instances, and optimal solutions for 10 instances (90.9% of instances), CPLEX rates are 81.8%, 63.6%, and 63.6%, respectively;
- observe that despite CPLEX obtains the optimal solution for the first set of six instances, it has a very weak performance for the instances "PC" and "SM", where the computation times are around 10 and 3 hours. Moreover, it only obtains the optimal solution for one instance in the second set ("4_all"), and cannot obtain feasible solution for two instances ("0_all" and "2_all") within 36,000 seconds (10 hours) of computation time due to the huge size of the Branch-and-Bound tree; and,
- the EH algorithm obtains feasible solution for all 11 instances, and superior than CPLEX, and that in a quarter of computation time (on average). Additionally, the EH algorithm delivered optimal solution for all instances in the first set, as well as for the last four instances of the second set (except "0_all"). Note that the instances of the second set are those that the exact solvers including CPLEX have a great difficulty in solving them.

Those arguments demonstrate the efficiency of the EH algorithm, particularly, for large instances of the Max β (α,β)-k FSP.

Table 5.3 details the computational results of CPLEX and EH algorithm on 11 real-world instances of Table 5.1. The presented results were obtained by incorporating the minimum

Table 5.3: Computational results of the exact+heuristic (EH) algorithm for solving 11 real-world instances of the Max β (α, β)-k Feature Set Problem (FSP), where $\alpha = \alpha^*$, and $p = 20$ (20 optimal solutions for each instance of the Min k (α, β)-k Feature Set Problem (FSP) were obtained). As before, we reported the outcomes of the solver CPLEX (here, "-" denotes the CPLEX was stopped at the time limit of 36,000 seconds without obtaining even a feasible solution). Column "$|\tilde{J}|$" is the number of common features across all solutions in the pool, β_0 is the best value of β for the feasible solution, and β is the maximum value obtained. Columns "Time" and "Gap" denote the computation time in second, and gap in % calculated as $\frac{\beta - \beta^*}{\beta^*} \times 100$, where β^* is best values of β.

			CPLEX			EH				
Instance	α^*	β^*	β	Time	Gap	$\lvert\tilde{J}\rvert$	β_0	β	Time	Gap
ADMF	86	118	**118**	5.84	0.00	101	114	**118**	13.11	0.00
DS	50	51	**51**	0.02	0.00	51	51	**51**	0.16	0.00
PD1	3970	4325	**4325**	2013.78	0.00	8858	4324	**4325**	3489.94	0.00
PD2	760	645	**645**	0.4	0.00	1265	645	**645**	14.56	0.00
PC	229	233	**233**	18539.13	0.00	225	233	**233**	3657.76	0.00
SM	22	40	**40**	10577.19	0.00	37	39	**40**	6579.08	0.00
0_all	354	471	-	36000	-	998	471	**471**	2466.08	0.63
1_all	683	989	987	36000	0.41	2120	982	**989**	2428.11	0.00
2_all	1016	1394	-	36000	-	3005	1394	**1394**	3801.78	0.00
3_all	1394	1965	1964	11044.41	0.33	4215	1962	**1965**	7642.15	0.00
4_all	387	549	**549**	1877.36	0.00	501	536	**549**	3935.98	0.00
Average				13823.47	0.08				3093.52	0.06

number of features, which we obtained by the algorithms of Chapter 4. Therefore, the values of β in the table are the greatest values obtained given the minimum number of features. Moreover, we set $p = 20$, i.e. 20 optimal solutions for each instance of the Min k (α, β)-k FSP were obtained. The first three columns show instance name, optimal values of α, the best values for β, which are available as of the time (the optimal values of β are recognized through a value of zero for the gap, either CPLEX or EH). Columns "CPLEX" refer to the outcomes of solving Model $\mathcal{IP}_{\mathcal{MBP}}$ by the solver CPLEX. The outcomes include the best objective function value ("β"), if CPLEX is able to solve, the computation time in second, and the optimality gap in %. The remaining columns show the outcomes of the EH algorithm. Column "β_0" shows the value of β for the solution obtained by the initial methods, "β" shows the best obtained value of β, "Time" is the computation time in second, and "Gap" is calculated as $\frac{\beta - \beta^*}{\beta^*} \times 100$. Across the table "-" denotes the solver CPLEX was stopped at the time limit of 36,000 seconds without reporting even a feasible solution. Best solution obtained by each method were highlighted.

Table 5.4: Summary of the computational results of EH and CPLEX for solving 125 randomly generated instances.

Criterion	CPLEX	EH
Percent of feasible solution	31.20%	100%
Percent of best obtained	31.20%	100%
Percent of optimal solution	31.20%	21.60%
Average computation time	667.72	196.82
Average gap	0.00	0.00

5.5.2 Computational results of random instances

The computational outcome of the EH algorithm on 11 real-world instances are very promising, in particular, several of those instances are very large and may truly reflect the effectiveness of the EH algorithm in solving large instance of Max β (α, β)-k FSP. Nevertheless, it would be interesting to further evaluate the performance of the EH algorithm on a larger number of instances. Following this, we applied the EH algorithm on 125 randomly generated instances by Paula (2012). Recall that the standard instances of the Set k-Cover Problem (SkCP), which were discussed in Chapter 4, do not include set I_2, and therefore, we cannot create an instance of the Max β (α, β)-k FSP.

As discussed earlier in Chapter 4, each of these randomly generated instances include 2000 features and two disjoint sets (I_1 and I_2) of 20000 sample pairs and an edge density of 20%, and may represent case-control datasets with 2000 features and 200 samples. Also, due to incorporating different parameters for generating instances they pose different computational challenges: some of them can be optimally solved by an exact solver in a few seconds, however, for the majority of them even a feasible solution cannot be obtained in a reasonable amount of time. Moreover, those instances pose computational challenge for the Max Cover (α, β)-k Feature Set Problem (FSP) as well. The latter will be discussed in Section 5.6.

Despite applying the EH algorithm on those 125 instances of Paula (2012), we may not be able to directly compare the EH with the Variable Neighborhood Search+Tabu Search (VNS+TS) algorithm proposed in Paula (2012) because the objective function values obtained by the VNS+TS were not reported in Paula (2012). Instead, that study reported the gap between the VNS+TS and the CPLEX, which are not helpful to our study because CPLEX versions, programming language, and platforms, among others, are different between two studies, and hence, one may not benefit from those reported values of gap. In addition to this, Paula (2012) used slightly larger values for the number of features, which in turn leads to quite different solutions for the Max β (α, β)-k FSP and Max Cover (α, β)-k FSP.

Table 5.4 summarizes the computational experiments of the EH and CPLEX for solving 125 randomly generated instances, where $\alpha = \alpha_{max}$ (α^*), and the number of features are extracted from Table 4.14. Here, we compared the outcomes of CPLEX and EH across five criteria of *percent of feasible solution*, *percent of best solution*, *percent of optimal solution*, *average computation time* (in second), and *average gap* (from the best known solution). Both methods were allowed to run for 900 seconds (15 minutes). The last two criteria were calculated over the number of instances solved to feasibility (only for CPLEX). Over those instances,

- the EH algorithm obtains feasible solutions for 100% of instances (i.e. for all 125 instances), whereas CPLEX is able to obtain feasible solutions for slightly more than 31% of instances;
- in addition to this, while the EH algorithm obtains best solutions for 100% of instances, CPLEX rate is 31.2%; and,
- the average computation time of the EH algorithm is less than a third of that of CPLEX.

The above observations demonstrate the superiority and effectiveness of the EH algorithm in solving randomly generated instances of the Max β (α, β)-k FSP. Note that the average gap of CPLEX is calculated over the instances solved to feasibility, i.e. over only 39 instances. With respect to this, having a value of zero for the average gap does not tell us much about the CPLEX performance, neither does it impact the superiority of the EH algorithm. This along with the outcomes of the EH algorithm on solving 11 real-world instances (Section 5.5.1) is a testament to the effectiveness of EH algorithm in solving the Max β (α, β)-k FSP, and obtaining high quality solutions, including for large instances, in a reasonable amount of time.

5.6 The Max Cover (α, β)-k Feature Set Problem

Recall from our earlier discussion in Chapter 3 that in order to solve the (α, β)-k Feature Set Problem (FSP) as well as determining the optimal values of parameters, we implemented a four-stage approach. The last step of this approach involves solving the Max Cover (α, β)-k Feature Set Problem (FSP). The Max Cover (α, β)-k FSP aims to obtain a set of minimum cost features, among alternative sets of features, that provides more explanations (coverage) in total, either to the differences between the classes or similarity within entities in the same class. In other words, the solution to the Max Cover (α, β)-k FSP is a minimum cost set of features that maximizes the similarities between entities of the same class and the differences between entities of different classes, and has more explanations (coverage) in total. Therefore, in this section we discuss solving the Max Cover (α, β)-k FSP. One may realize that the set of features obtained by solving the Max Cover (α, β)-k FSP provides a very robust feature set because it has the maximum coverage among any alternative sets of features.

5.6.1 Proposed solution method

The proposed method for solving the Max Cover (α, β)-k FSP obtains a feasible solution by utilizing Proposition 3.8. As we showed in Proposition 3.8, an optimal solution for both Min k (α, β)-k Feature Set Problem (FSP) and Max β (α, β)-k Feature Set Problem (FSP) must be feasible for the Max Cover (α, β)-k FSP. For this solution we can calculate the value of its objective function by using Equation (5.1):

$$z^* = \sum_{j \in J^*} v_j \tag{5.1}$$

where, J^* is a set of features in the optimal solution of Max β (α, β)-k FSP, v_j is the value of feature j (a parameter), and $z^* \in \mathbb{Z}^+$ is the optimal objective function value of the Max Cover (α, β)-k FSP. Note that if the optimal solutions to Min k (α, β)-k FSP and Max β (α, β)-k FSP are not available, then z^* is not optimal anymore, and J^* is a feasible solution for the Max Cover (α, β)-k FSP.

The so obtained feasible solution may be further improved by the solver CPLEX. In spite of the simplicity of the proposed method the computational results demonstrate that this method is very effective in solving even large instances of the Max Cover (α, β)-k FSP.

In order to evaluate the quality of z^*, that is how far it is from optimality, we solve a Linear Programming (LP) relaxation of Model $\mathcal{IP}_{\mathcal{MCP}}$ (this model is discussed in Section 3.5.3) by the solver CPLEX. An LP relaxation of Model $\mathcal{IP}_{\mathcal{MCP}}$ may be obtained by relaxing Equation (3.17) into $0 \leq x_j \leq 1, \forall j \in J$. Because the Max Cover (α, β)-k FSP is a maximization problem, and any feasible solution for the Max Cover (α, β)-k FSP is indeed feasible for its LP relaxation, the optimal objective function value of its LP relaxation, which we denote it by $\bar{z}^* \in \mathbb{R}^+$, is an upper bound for z^*, hence, $\bar{z}^* \geq z^*$. In addition to this, the Max Cover (α, β)-k FSP is an integer program (IP) and $v_j \in \mathbb{Z}^+, \forall j \in J$; therefore, we may round down $\bar{z}^*$ to its nearest integer value, thus, $\lfloor \bar{z}^* \rfloor \geq z^*$. This upper bound has been denoted as "UB" in Figure 5.1 and Table 5.6.

We also solve the Max Cover (α, β)-k FSP by the solver CPLEX (i.e. solving Model $\mathcal{IP}_{\mathcal{MCP}}$, where possible). However, the CPLEX may not be able to solve large instances of the Max Cover (α, β)-k FSP. We discuss this in details in Sections 5.6.2 and 5.6.3.

5.6.2 Computational results of real-world instances

This section reports the computational results of the proposed method for solving the Max Cover (α, β)-k FSP, i.e. by obtaining a feasible solution through utilizing Proposition 3.8. Here we did not improve the feasible solution by using the solver CPLEX because we are motivated to show that even feasible solutions obtained through Proposition 3.8 are within reasonable proximity to the CPLEX results, while they require much less computational efforts. All computational experiments were implemented in the programming language Python 2.7 via the solver CPLEX 12.5.0 Python API. Our set of instances and the computing facility are the same as those we discussed in Section 5.5.1.

Table 5.5 summarizes the major outcomes of the experiment across four criteria of *percent of feasible solution*, *percent of best solution*, *percent of optimal solution*, and *average computation*

Table 5.5: Summary of the computational results of the proposed method and CPLEX for solving 11 real-world instances of the Max Cover (α, β)-k Feature Set Problem.

Criterion	CPLEX	The proposed method
Percent of feasible solution	63.60%	100%
Percent of best solution	63.60%	45.50%
Percent of optimal solution	63.60%	9.09%
Average computation time	8388.55	1228.00

time (in second). Table 5.6 details the outcomes. It is very interesting to observe that CPLEX is only able to obtain feasible solutions for 63.60% of instances, while the proposed method generates feasible solutions for all instances. At the same time, CPLEX obtains best solutions for 63.60% of instances, and that in more than 2 hours, while the proposed method obtains for 45.50% of instances, and that in about 20 minutes, i.e. at least six times faster. Note that here we did not improve the feasible solutions generated by the proposed method. In Section 5.6.3, we show that when those feasible solutions are further improved, they present very high quality solutions, and far superior than those obtained by CPLEX.

Table 5.6: Computational results of the proposed method for solving 11 real-world instances of Max Cover (α, β)-k Feature Set Problem (FSP), where $\alpha = \alpha^*$, and β^* is the maximum value of β obtained through the EH algorithm. Column "Max Cover" shows the best available objective function values for the Max Cover (α, β)-k FSP, where the optimal values were highlighted. The outcomes of CPLEX include the best obtained objective function value, and computation time in second (we did not report the optimality gap because either it was 0 or it was not reported by the CPLEX). Also, "-" denotes the solver CPLEX was stopped at the time limit of 18,000 seconds without obtaining a feasible solution). Columns "Proposed method" show feasible solutions for the Max Cover (α, β)-k FSP obtained by utilizing Proposition 3.8, and the computation time in second. Columns "UB" refer to the upper bounds for the Max Cover (α, β)-k FSP. Finally, two gaps were reported: optimality gap, which is between the feasible and optimal solutions, and the upper bound gap, which is between the feasible and upper bound solutions.

				CPLEX		Proposed method		UB		Gap	
Instance	α^*	β^*	Max Cover	z	Time	z	Time	UB	Time	Optimality	Upper bound
ADMF	86	118	**581,608**	581,608	10.07	581,328	9.11	581,755	1.90	0.05	0.07
DS	50	51	**5,341**	5,341	0.24	5,341	0.10	5,341	0.07	0.00	0.00
PD1	3970	4325	**26,863,408**	26,863,408	275.86	26,792,538	831.02	26,863,848	112.76	0.26	0.27
PD2	760	645	**262,248**	262,248	1.52	260,268	3.02	262,248	1.42	0.76	0.76
PC	229	233	**5,699,721**	5,699,721	686.97	5,646,036	612.00	5,707,403	451.48	0.94	1.08
SM	22	40	**50,214,804**	50,214,804	7498.72	50,149,938	755.00	50,548,001	777.59	0.13	0.79
0_all	354	474	64,001,349	-	18000	64,001,349	4,007.23	64,259,990	883.88	0.00	0.40
1_all	683	989	121,737,413	-	18000	121,737,413	710.75	121,862,814	1969.02	0.00	0.10
2_all	1016	1382	173,107,143	-	18000	173,107,143	3,573.06	173,751,149	3715.30	0.00	0.37
3_all	1394	1965	252,717,489	-	18000	252,717,489	2,671.33	252,816,753	6830.99	0.00	0.04
4_all	387	549	**76,052,688**	76,052,688	3413.62	76,028,538	335.41	76,104,982	1154.00	0.03	0.10
Average					8388.55		1228.00		1445.31	0.21	0.39

Figure 5.1: Comparison of optimality gap versus upper bound gap for real-world instances of Max Cover (α, β)-k Feature Set Problem (FSP). The optimality gap is between feasible and optimal solutions and is calculated as $\frac{z-z^*}{z^*} \times 100$, and the upper bound gap is between feasible and upper bound solutions and is calculated as $\frac{z-UB}{UB} \times 100$.

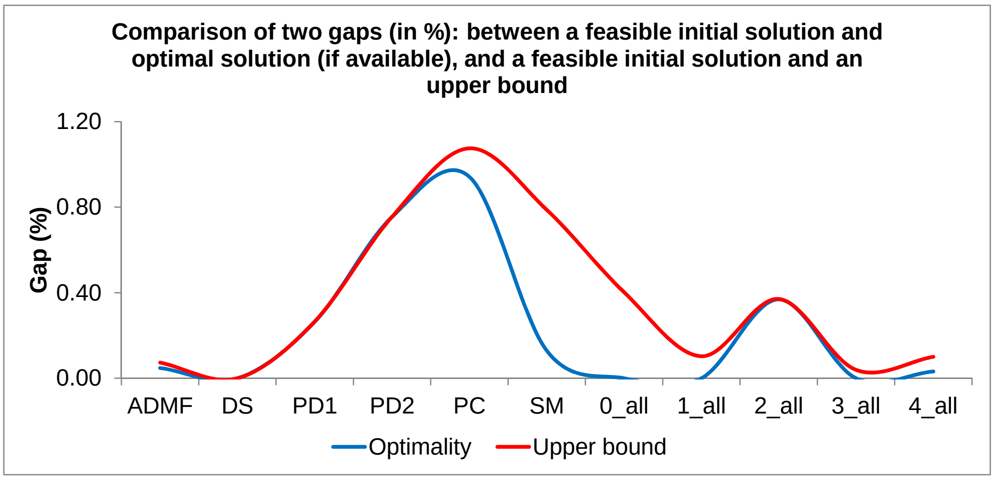


Figure 5.1 illustrates two gaps (optimality and upper bound) have very close values. This implies that z, which is obtained as the result of applying Proposition 3.8, is of very high quality. This may be observed by looking into the optimality gap ($\frac{z-z^*}{z^*} \times 100$, where z^* is the optimal value of objective function, where available), and the upper bound gap ($\frac{z-UB}{UB} \times 100$). We observed that the average of the optimality gap over all instances is 0.21%, and that of the upper bound gap is 0.39%. Therefore, in the worst case the obtained solutions are within 0.39% of optimality.

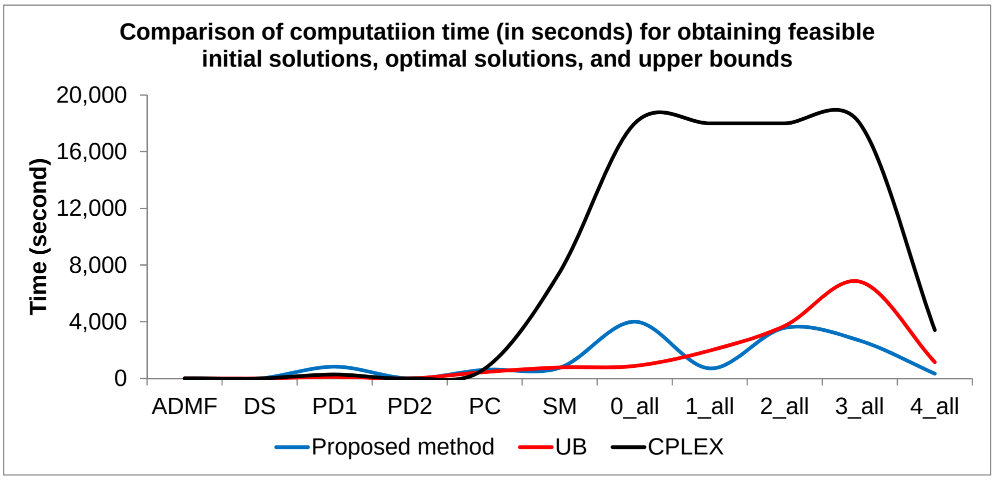
In addition to this, the proposed method obtains feasible solutions faster than UB, and far faster than CPLEX. This is illustrated in Figure 5.2. Taking into account the values of gap, computation time, and more importantly, the size of instances, particularly of the second set, we may conclude that Proposition 3.8 is efficiently capable of solving the Max Cover (α, β)-k FSP.

Therefore, we believe spending additional resources to obtain optimal solutions for the Max Cover (α, β)-k FSP does not justify its cost while high quality solutions and that very close to optimality, in particular for large instances, can be obtained in a short time. We further verify this argument in Section 5.6.3.

5.6.3 Computational results of random instances

To further validate the effectiveness of the proposed solution method for solving the Max Cover (α, β)-k FSP (Section 5.6.1) we solved 125 randomly generated instances by Paula (2012) by using the proposed solution method and CPLEX. Here, we improved the feasible solutions by supplying them to CPLEX for re-optimization. Our set of instances are the same as those we

Figure 5.2: Computation time of obtaining feasible, optimal, and upper bound solutions for 11 real-world instances of Max Cover (α, β)-k Feature Set Problem (FSP). The computation time limit of the standard solver CPLEX is set to 18,000 seconds.



discussed in Section 5.5.1.

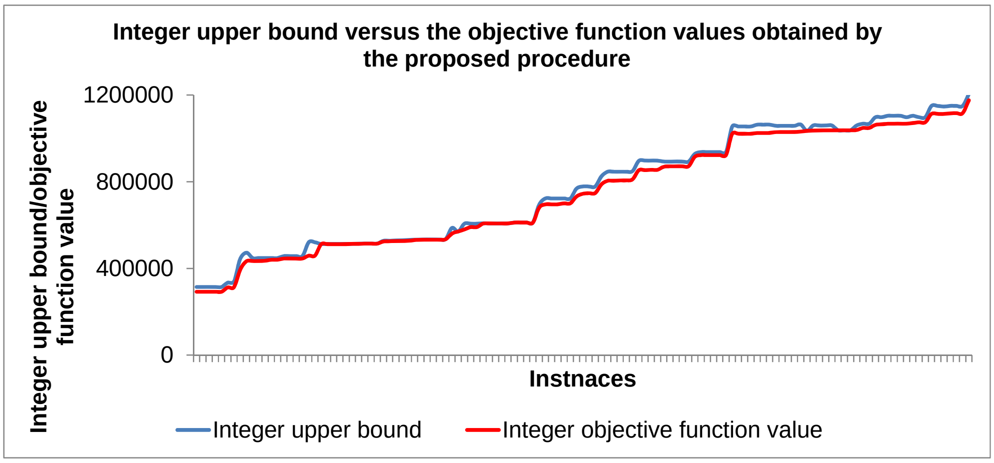
Table 5.7: Summary of the computational results of EH and CPLEX for solving 125 randomly generated instances of Max Cover (α, β)-k Feature Set Problem (FSP).

Criterion	CPLEX	Propose method
Percent of feasible solution	32.00%	100%
Percent of best solution	29.60%	88.80%
Percent of optimal solution	12.00%	12.00%
Average computation time	705.14	706.31
Average gap	0.00	0.06

Table 5.7 summarizes the outcomes of the computational experiments of CPLEX and EH algorithm for solving those instances. For evaluation purpose we considered five criteria of *percent of feasible solution*, *percent of best solution*, *percent of optimal solution*, *average computation time* (in second), and *average gap* (from the best known solution). According to the table the followings may be observed:

- CPLEX is only able to obtain feasible solution for 32% of instances. In contrast, the proposed method of Section 5.6.1 obtains feasible solutions for all instances.
- While CPLEX obtains best solutions for only 29.6% of instances, that of the proposed method is 88.8%, which is three times greater than CPLEX.
- Both methods have almost identical average computation times, and both were allowed to run for 900 seconds (15 minutes).

Figure 5.3: The gap between the integer upper bound and objective function value of the Max Cover (α, β)-k FSP over 125 randomly generated instances. The values were sorted in ascending order. The graph is in line with the earlier observation that the proposed method of solving the Max Cover (α, β)-k FSP obtains very high quality solutions within close proximity to the optimal solution.



- The values of gap for the CPLEX were averaged over those 40 instances solved to feasibility, whereas those for the proposed method were averaged over all 125 instances. Therefore, this value of gap for CPLEX does not tell us much because of the small sample size, and also, this does not negatively impact the effectiveness of the proposed method in solving the Max Cover (α, β)-k FSP.

Earlier we observed that the proposed method of solving the Max Cover (α, β)-k FSP obtains very high quality solutions within close proximity to the integer upper bound (Figure 5.1). To further validate that observation, Figure 5.3 depicts the objective function value of 125 instances, which were obtained by the proposed method, and the integer upper bound, which were obtained by the procedure discussed in Section 5.6.1. As the figure shows, these two values are very close to each other, and given that the objective function value of the Max Cover (α, β)-k FSP cannot be greater than the integer upper bound, therefore, even in the worst case there is not that much space for improvement. This is in line with our earlier observations regarding the performance of the proposed method.

5.7 Conclusion

In this chapter, we discussed two pre-processing and an exact+heuristic (EH) algorithm for solving the Max β (α, β)-k Feature Set Problem (FSP). These methods are developed by utilizing the properties and propositions discussed in Chapter 3. Because our attempt to solve the Max β (α, β)-k FSP by exact solvers within reasonable amount of time failed, we developed and implemented the EH algorithm. To the best of our knowledge, and at the time of writing this thesis, the proposed EH algorithm, which was tested on 136 instances of both real-world and randomly generated, obtains the best solution for the Max β (α, β)-k FSP, including for large instances, which posed computational challenges for the exact solvers prior to this research. In a testament to this, the EH algorithm delivered feasible and best solutions for all 136 instances (i.e. for 100% of instances), whereas the CPLEX rates are 35.3% for delivering feasible solutions, and 33.8% for delivering the best solutions. Moreover, CPLEX requires more computation time than the EH algorithm, and that in the magnitude of three times.

Finally, we solved the Max Cover (α, β)-k Feature Set Problem (FSP) by solving the Max β (α, β)-k FSP, and using its solution as a feasible solution for the Max Cover (α, β)-k FSP, and improving this solution. According to the computational results, this method outperforms the outcomes of the solver CPLEX, particularly, for large instances. In particular, we obtained feasible and best solutions for 100% and 85.2% of instances, whereas the CPLEX rates are 34.5% and 32.3% (over 136 instances). Moreover, we showed that the obtained solutions are within close proximity to the integer upper bounds implying the proposed method is capable of obtaining very high quality solutions.

Chapter 6

Concluding Remarks and Future Research

In this research thesis, we investigated the (α, β)-k Feature Set Problem (FSP), explored its mathematical properties and characteristics, and proposed efficient solution methods, which are able to obtain high quality solutions, and evaluated those methods against the state-of-the-art methods. This chapter aims to answer the research question and goals presented in Section 1.4 by using the theoretical and computational outcomes of the thesis. In addition to this, Section 6.2 reviews limitations of this research and proposes several future research directions.

6.1 Theoretical and computational contributions and outcomes

The research problem of the presented thesis is to develop solution methods for the (α, β)-k Feature Set Problem (FSP), which aims to select a minimum cost/cardinality set of features maximizing the similarities between entities of the same class and the differences between entities of different classes. The (α, β)-k FSP has applications in computational biology and Bioinformatics. As we discussed in Chapter 2, our work is motivated by limitations of the previous studies, some of which are discussed in the followings.

- Lack of efficient solution methods for large instances. The exact methods are only capable of solving small and medium instances. Furthermore, by comparing the state-of-the-art algorithms with exact methods, it seems that the available heuristics do not have competitive performance, particularly, for large instances. Hence, studying large datasets and solving large instances pose a challenge to these methods. This is particularly important because almost all applications of the (α, β)-k FSP in computational biology and Bioinformatics involve dealing with large datasets, and hence, there is a huge demand for efficient methods.

- Lack of performance guarantee. To the best of our knowledge, the only heuristic algorithms for the (α, β)-k FSP are due to the study of Paula (2012). His study developed algorithms that can obtain good quality solutions for medium sized instances. We realized that his algorithms do not have good performance for large instances. Apart from this, that study has several limitations. Firstly, there is no guarantee on the algorithms' performance. Secondly, the algorithms benefit from general and randomized local searches originally developed for the traditional combinatorial optimization problems, whereas it is well accepted in the literature that problem-driven local searches usually lead to superior outcomes.

- Lack of modeling the feature's cost. Previous studies did not consider the cost associated with selecting features. The cost may model distinguishing factors, for example, importance, correlation with other features, dependency on other features, etc.

To overcome those limitations we defined the research question:

- Research question. Can we develop efficient combinatorial optimization-based algorithms and methods for the (α, β)-k Feature Set Problem (FSP), in order to select a subset of features, out of a larger set, and that in a reasonable amount of time?

We answered the research question by developing several algorithms for the (α, β)-k FSP, in particular two very efficient exact+heuristic algorithms (Algorithm 4.3 and Algorithm 5.2) that deliver high quality solutions, including feasible solutions for all 346 instances, and the best known solutions for more than 50% of instances, and that in a reasonable of time. Many of those instances still pose computational challenges for exact solvers and methods.

Our major goal of this research thesis has been to "design, develop and implement modeling techniques, and efficient and advanced optimization-based algorithms and methods for the (α, β)-k FSP". This goal has successfully been achieved and accomplished in this research thesis, more extensively,

- For the first time, we investigated and explored important properties and characteristics of the (α, β)-k FSP in Sections 3.6 and 3.7, and utilized those in Chapters 4 and 5 in order to design and develop algorithms and methods to solve the (α, β)-k FSP (connected to Research goal 1; Section 1.4).

- Our developed algorithms and solution methods (Algorithms 4.3 and 5.2) can efficiently solve large instances of the (α, β)-k FSP. Prior to our research exact algorithms could not obtain optimal solution for these instances, and even these algorithms have difficulty in delivering feasible solutions in a reasonable amount of time. However, the solution methods of this research overcame these limitations by delivering high quality solutions, and very close to optimal, even for large instances of the (α, β)-k FSP. Also, the proposed algorithms of this research outperform state-of-the-art algorithms of Paula (2012). Our focus has been on developing exact+heuristic algorithms, which benefit from both exact methods, and heuristic procedures. In addition to those, we also developed certain heuristics and problem-driven local searches in order to facilitate and speed-up exact algorithms. Our contributions have thoroughly been discussed in Chapters 4 and 5 along with the outcomes and computational results (connected to Research goal 2; Section 1.4).

- We extended the basic modeling, algorithms and solution methods to the weighted variant of the (α, β)-k FSP by considering features costs in all modeling, algorithms, and analyses of Chapters 3 to 5. This allows analyzing the impact of each feature. This is valuable because in practical analyses and applications some features may be preferred over others, or some must always be selected by any set of features (connected to Research goal 1; Section 1.4).

- Finally, we studied the usefulness of the developed algorithms and methods by applying them on biological datasets ranging from medium to large instances, on weighted instances, and on unweighted randomly generated instances, and in total on 346 instances (we discussed these in Chapters 4 and 5). To this end, our algorithms and methods efficiently tackled large instances, and delivered new best solutions. Such an achievement is not available prior to this research, and is accomplished in this research thesis (connected to Research goals 1 and 2; Section 1.4).

The major contributions and achievements of this research thesis have been summarized in Table 6.1. We should emphasize that in this research thesis our focus has been on developing and implementing exact-based algorithms and methods, which can deliver global optimal solutions or at least very good quality solutions. This has been considered in every aspect of this research, for example, when developing bounds and propositions in Chapter 3, developing pre-processing methods for the exact algorithms, and developing exact+heuristic algorithms in Chapters 4 and 5.

In Chapter 3 we investigated and developed fundamental mathematical concepts and characteristics for the (α, β)-k FSP and that for the first time. In order to solve the (α, β)-k FSP, we followed a four-stage decomposition-based approach proposed in the previous studies (Berretta et al., 2007; Paula, 2012), where we determined the optimal value of α (the minimum number of features that must explain the differences between any pair of entities of different classes), obtained optimal value for k (the optimal cost/cardinality of a set of features necessary to explain the dichotomy between the classes, considering that at least α features do so for each pair of entities of different classes) through solving the Min k (α, β)-k FSP, determined the optimal value of β (explaining the dichotomy between the classes, and at least α features do so for each pair of entities of different classes) through solving the Max β (α, β)-k FSP, and finally solved the Max Cover (α, β)-k FSP to obtain a set of features with the maximum explanation either

Table 6.1: A summary of contributions and outcomes of the presented research thesis.

Area	Major contributions
Mathematics and properties	Bounds for the Min k (α, β)-k FSP and Max β (α, β)-k FSP (Section 3.6). Properties and characteristics of the Min k (α, β)-k FSP, Max β (α, β)-k FSP, and Max Cover (α, β)-k FSP (Section 3.7). Feasibility conditions for the Max β (α, β)-k FSP and Max Cover (α, β)-k FSP (Section 3.7). Optimality condition for the Max β (α, β)-k FSP and Max Cover (α, β)-k FSP (Section 3.7).
Algorithms and methods	The greedy construction heuristic of multi Column Row Cover Construction (mCRCC) for the Min k (α, β)-k FSP (Section 4.5). The Removal Local Search (RLS) improvement heuristic for the Min k (α, β)-k FSP (Section 4.6). The exact+heuristic algorithm for the Min k (α, β)-k FSP (Section 4.7). Pre-processing methods for the Max β (α, β)-k FSP (Section 5.3). The exact+heuristic algorithm for the Max β (α, β)-k FSP (Section 5.4). The solution method for the Max Cover (α, β)-k FSP (Section 5.6.1).
Computational outcomes	Optimal solutions for the Min k (α, β)-k FSP over all instances of set 1, and new best solutions over sets 2 and 3 (Section 4.8). Feasible solutions for all instances of the Max β (α, β)-k FSP, including large instances (Section 5.5). Optimal solutions for all instances of the Max β (α, β)-k FSP, for set 1, and for several large instances of set 3 (Section 5.5). High quality solutions for all instances of the Max Cover (α, β)-k FSP (Section 5.6.3).
Tested instances	Set 1: six real-world biological instances (unweighted) and five real-world face recognition datasets (unweighted) (Sections 4.8 and 5.5). Set 2: 210 standard instances of the Set Cover Problem (weighted) (Section 4.8). Set 3: 125 randomly generated instances (weighted) for the (α, β)-k FSP (Sections 4.8 and 5.5).

to the differences between the classes or similarity within entities in the same class, and that with the desired characteristics, i.e. satisfying k, α, and β. Those mathematical properties and propositions discussed in Chapter 3 are particularly important because they are building blocks of the algorithms and solution methods proposed in Chapter 4 and Chapter 5. Several of those properties derive bounds on the optimal objective function value of Min k (α, β)-k FSP and Max β (α, β)-k FSP. Others investigate connections among the Min k (α, β)-k FSP, Max β (α, β)-k FSP and Max Cover (α, β)-k FSP. Particularly, these properties and propositions greatly advance solving the Max β (α, β)-k FSP; this is very beneficial because the Max β (α, β)-k FSP is a very challenging problem. We believe the most important propositions include those establishing a connection between optimal solutions of the Min k (α, β)-k FSP and feasible solutions of the Max β (α, β)-k FSP.

Chapter 4 is devoted to the algorithms and solution methods for both weighted and unweighted Min k (α, β)-k FSP. These algorithms include greedy construction (mCRCC) and improvement (RLS) heuristics. In particular, we developed a very efficient exact+heuristic algorithm, which combines both exact and heuristics, and obtains very high quality solution for the Min k (α, β)-k FSP, including several new best solutions. The core idea of this algorithm is variable fixation and iterative optimization. By testing the algorithm over three sets of 346 instances we showed that it outperforms state-of-the-art algorithms.

Chapter 5 designs and develops exact and heuristic solution methods for solving the Max β (α, β)-k FSP. While the standard solvers fail to solve the Max β (α, β)-k FSP, particularly, for large instances, we facilitated them by obtaining feasible initial solutions and providing these solutions to the solvers. The computational experiments demonstrated the impact of those initial solutions. We also utilized properties and propositions developed earlier in Chapter 3 in an exact+heuristic algorithm, which builds very good quality initial solutions for the Max β (α, β)-k FSP by using multiple optimal solutions of the Min k (α, β)-k FSP. Through this we showed that good quality solutions for the Max β (α, β)-k FSP can be obtained in a reasonable amount of time. Note that due to difficulty of the Max β (α, β)-k FSP exact solvers are unable to generate feasible solutions for large instances. Therefore, obtaining good quality solutions for 100% of instances, as our proposed EH algorithm does, indeed is a great breakthrough towards solving the Max β (α, β)-k FSP. We tested the EH algorithm on a set of 136 instances, majority of which are large instances, and concluded that the EH algorithm obtains the best results for the Max β (α, β)-k FSP among available algorithms and exact solvers, and has a very competitive performance for large instances.

Section 5.6 of Chapter 5 proposes a simple but effective solution method for the Max Cover (α, β)-k FSP, which is able to obtain very good quality solutions in a short time. The proposed method utilizes the propositions discussed in Section 3.7, and builds upon solutions of the the Max β (α, β)-k FSP. We showed that this method has a very good performance, and while it is much faster than the exact solvers, it delivers solutions, which are in close proximity to the upper bounds. More importantly, the proposed method outperforms exact solvers.

We conducted extensive computational experiments to evaluate the performance of the proposed algorithms. More precisely, we considered three sets of 346 instances ranging from small to large, both weighted and unweighted. Among those are 11 real-world unweighted instances, 210 weighted instances of the Set Cover Problem, and 125 randomly generated unweighted instances proposed by Paula (2012) for the (α, β)-k FSP.

6.2 Future research directions

In its own right this research has greatly advanced the mathematics, algorithms and solution methods of the (α, β)-k Feature Set Problem (FSP), and that to a great extent, and has led to delivering high quality solutions for large instances in a reasonable amount of time. Despite this, there are several areas that may benefit from further investigation.

The major technique used in the exact+heuristic algorithm of Section 4.7 is variable fixation, which allows the original Min k (α, β)-k FSP to be reduced into smaller sub-problems. In order to perform the variable fixation, the algorithm utilizes the information obtained during solving a linear programming relaxation of the Min k (α, β)-k FSP, mainly, by rounding the values of variables that have fractional values into certain integer values. Another scheme is to benefit from the reduced costs of the relaxed variables. Also, those rounding techniques proposed for the Set Cover Problem may shed light into other variable fixation techniques for the Min k (α, β)-k FSP.

The exact+heuristic for the Max β (α, β)-k FSP, which was discussed in Section 5.4, can be subject to further improvement. The algorithm implements certain mathematical properties and propositions to obtain good quality solutions for the Max β (α, β)-k FSP. The major operation of the algorithm is variable fixation, and the performance of the algorithm heavily depends on it, in particular, because a pool of optimal solutions for the Min k (α, β)-k FSP is utilized in order to fix certain variables. Firstly, any improvement in obtaining multiple optimal solutions for the Min k (α, β)-k FSP will benefit the algorithm. Secondly, the algorithm implements a deterministic strategy to fix variables. Despite its outcomes, investigation into other schemes may lead to solutions with less computational efforts. One stream for this is to investigate implementing a probabilistic strategy, which decides upon fixing a variable on the basis of the number of times it appears in the pool. This may lead to obtaining feasible solutions in the earlier stages of the algorithm because features than those *common* across the pool may be selected.

Clearly, the proposed solution methods can be applied in other domains where there are many features and comparatively few samples. Such domains include, but not limited to, analysis of written texts (text mining), image analysis, social media, and email spam filtering. For example, website Twitter produces more than 250 millions tweets per day, which includes many new words and abbreviations (features). Li et al. (2017) argues that when selecting features for tweets, one cannot wait until all features have been generated. Therefore, an online feature selection is indeed a preferred option. Contexts such as financial analysis, online trading, and medical testing also include very large datasets, implying large number of features are inevitable, and selecting features may therefore be needed (Li et al., 2017).

References

Aini, Asghar and Amir Salehipour (2012). “Speeding up the Floyd–Warshall algorithm for the cycled shortest path problem”. In: *Applied Mathematics Letters* 25(1), pp. 1–5.

Albrecht, Andreas A (2006). “Stochastic local search for the feature set problem, with applications to microarray data”. In: *Applied Mathematics and Computation* 183(2), pp. 1148–1164.

Alizadeh, Ash A, Michael B Eisen, R Eric Davis, Chi Ma, Izidore S Lossos, Andreas Rosenwald, Jennifer C Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, et al. (2000). “Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling”. In: *Nature* 403(6769), pp. 503–511.

Ansótegui, Carlos, Joel Gabàs, and Jordi Levy (2016). “Exploiting subproblem optimization in SAT-based MaxSAT algorithms”. In: *Journal of Heuristics* 22(1), pp. 1–53.

Baker, Edward K. (1981). “Efficient heuristic algorithms for the weighted set covering problem”. In: *Computers and Operations Research* 8(4), pp. 303 –310.

Balas, Egon and Maria C. Carrera (1996). “A Dynamic Subgradient-Based Branch-and-Bound Procedure for Set Covering”. In: *Operations Research* 44(6), pp. 875–890.

Balas, Egon and Andrew Ho (1980). “Combinatorial Optimization”. In: ed. by M. W. Padberg. Springer Berlin Heidelberg: Berlin, Heidelberg. Chap. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study, pp. 37–60.

Banga, Julio R. (2008). “Optimization in computational systems biology”. In: *BMC Systems Biology* 2(1), p. 47.

Bar, Yaniv, Idit Diamant, Lior Wolf, Sivan Lieberman, Eli Konen, and Hayit Greenspan (2018). “Chest pathology identification using deep feature selection with non-medical training”. In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 6(3), pp. 259–263.

Battiti, Roberto and Marco Protasi (1999). “Approximate Algorithms and Heuristics for MAX-SAT”. In: *Handbook of Combinatorial Optimization: Volume1–3*. Ed. by Ding-Zhu Du and Panos M. Pardalos. Springer US: Boston, MA, pp. 77–148.

Batun, Sakine and Mehmet A. Begen (2013). “Optimization in Healthcare Delivery Modeling: Methods and Applications”. In: *Handbook of Healthcare Operations Management: Methods and Applications*. Ed. by Brian T. Denton. Springer New York: New York, NY, pp. 75–119.

Bautista, Joaqun and Jordi Pereira (2007). "A GRASP algorithm to solve the unicost set covering problem". In: *Computers & Operations Research* 34(10), pp. 3162 –3173.

Beasley, J E (1987). "An algorithm for set covering problem". In: *European Journal of Operational Research* 31(1), pp. 85 –93.

Beasley, J E (1990). "A lagrangian heuristic for set-covering problems". In: *Naval Research Logistics* 37(1), pp. 151–164.

Beasley, J E and P C Chu (1996). "A genetic algorithm for the set covering problem". In: *European Journal of Operational Research* 94(2), pp. 392 –404.

Beasley, J.E. and K. Jrnsten (1992). "Practical Combinatorial Optimization Enhancing an algorithm for set covering problems". In: *European Journal of Operational Research* 58(2), pp. 293 –300.

Berretta, Regina, Alexandre Mendes, and Pablo Moscato (2005). "Integer programming models and algorithms for molecular classification of cancer from microarray data". In: *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*. Australian Computer Society, Inc., pp. 361–370.

Berretta, Regina, Alexandre Mendes, and Pablo Moscato (2007). "Selection of discriminative genes in microarray experiments using mathematical programming". In: *Journal of Research and Practice in Information Technology* 39(4), pp. 287–299.

Berretta, Regina, Wagner Costa, and Pablo Moscato (2008). "Combinatorial Optimization Models for Finding Genetic Signatures from Gene Expression Datasets". In: *Bioinformatics: Structure, Function and Applications. Series: Methods in Molecular Biology* 453(01), pp. 363–377.

Blanco, Rosa, Pedro Larrañaga, Iñaki Inza, and Basilio Sierra (2004). "Gene selection for cancer classification using wrapper approaches". In: *International Journal of Pattern Recognition and Artificial Intelligence* 18(08), pp. 1373–1390.

Bouhmala, Noureddine (2015). "A multilevel learning automata for MAX-SAT". In: *International Journal of Machine Learning and Cybernetics* 6(6), pp. 911–921.

Box, G.E.P., J.S. Hunter, and W.G. Hunter (2005). *Statistics for experimenters: design, innovation, and discovery*. Wiley series in probability and statistics. Wiley-Interscience.

Brandeau, Margaret L, François Sainfort, and William P Pierskalla (2004). *Operations research and health care: a handbook of methods and applications*. Vol. 70. Springer Science & Business Media.

Brandeau, Margaret L., Sainfort Francois, and William P. Pierskalla (2005). *Operations Research and Health Care A Handbook of Methods and Applications*. Vol. 70. International Series in Operations Research & Management Science. Springer US.

Breitling, Rainer, Patrick Armengaud, Anna Amtmann, and Pawel Herzyk (2004). "Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments". In: {*FEBS*} *Letters* 573(13), pp. 83 –92.

Brinza, Dumitru and Alexander Zelikovsky (2006). “Combinatorial Methods for Disease Association Search and Susceptibility Prediction”. In: *Algorithms in Bioinformatics: 6th International Workshop, WABI 2006, Zurich, Switzerland, September 11-13, 2006. Proceedings.* Ed. by Philipp Bücher and Bernard M. E. Moret. Springer Berlin Heidelberg: Berlin, Heidelberg, pp. 286–297.

Brotcorne, Luce, Gilbert Laporte, and Frédéric Semet (2003). “Ambulance location and relocation models”. In: *European Journal of Operational Research* 147(3), pp. 451–463.

Cai, Shaowei, Zhong Jie, and Kaile Su (2015). “An effective variable selection heuristic in SLS for weighted Max-2-SAT”. In: *Journal of Heuristics* 21(3), pp. 433–456.

Cai, Shaowei, Chuan Luo, Jinkun Lin, and Kaile Su (2016). “New local search methods for partial MaxSAT”. In: *Artificial Intelligence* 240, pp. 1 –18.

Caprara, Alberto, Matteo Fischetti, and Paolo Toth (1999). “A Heuristic Method for the Set Covering Problem”. In: *Operations Research* 47(5), pp. 730–743.

Caprara, Alberto, Paolo Toth, and Matteo Fischetti (2000). “Algorithms for the Set Covering Problem”. In: *Annals of Operations Research* 98(1-4), pp. 353–371.

Caserta, Marco (2007). “Tabu Search-Based Metaheuristic Algorithm for Large-scale Set Covering Problems”. In: *Metaheuristics: Progress in Complex Systems Optimization.* Ed. by Karl F. Doerner, Michel Gendreau, Peter Greistorfer, Walter Gutjahr, Richard F. Hartl, and Marc Reimann. Springer US: Boston, MA, pp. 43–63.

Ceria, Sebastián, Paolo Nobili, and Antonio Sassano (1998). “A Lagrangian-based heuristic for large-scale set covering problems”. In: *Mathematical Programming* 81(2), pp. 215–228.

Chandran, Uma R, Changqing Ma, Rajiv Dhir, Michelle Bisceglia, Maureen Lyons-Weiler, Wenjing Liang, George Michalopoulos, Michael Becich, and Federico A Monzon (2007). “Gene expression profiles of prostate cancer reveal involvement of multiple molecular pathways in the metastatic process”. In: *BMC cancer* 7(1), p. 64.

Chandrashekar, Girish and Ferat Sahin (2014). “A survey on feature selection methods”. In: *Computers & Electrical Engineering* 40(1). 40th-year commemorative issue, pp. 16 –28.

Charlesworth, Jac C, Joanne E Curran, Matthew P Johnson, Harald HH Göring, Thomas D Dyer, Vincent P Diego, Jack W Kent, Michael C Mahaney, Laura Almasy, Jean W MacCluer, et al. (2010). “Transcriptomic epidemiology of smoking: the effect of smoking on gene expression in lymphocytes”. In: *BMC medical genomics* 3(1), p. 29.

Chormunge, Smita and Sudarson Jena (2018). “Correlation based feature selection with clustering for high dimensional data”. In: *Journal of Electrical Systems and Information Technology* 5(3), pp. 542 –549.

Chuang, Li-Yeh, Cheng-Huei Yang, and Cheng-Hong Yang (2009). “Tabu Search and Binary Particle Swarm Optimization for Feature Selection Using Microarray Data”. In: *Journal of Computational Biology* 16(12), pp. 1689–1703.

Chuang, Li-Yeh, Sheng-Wei Tsai, and Cheng-Hong Yang (2011). “Improved binary particle swarm optimization using catfish effect for feature selection”. In: *Expert Systems with Applications* 38(10), pp. 12699–12707.

Chvatal, V (1979). “A greedy heuristic for the set covering problem”. In: *Mathematics of Operations Research* 4(3), pp. 233–235.

Coffin, Marie and Matthew J. Saltzman (2000). “Statistical Analysis of Computational Tests of Algorithms and Heuristics”. In: *INFORMS Journal on Computing* 12(1), pp. 24–44.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2009). *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press.

Cotta, C, C Sloper, and P Moscato (2004). “Evolutionary search of thresholds for robust feature set selection: Application to the analysis of microarray data”. In: *Applications of Evolutionary Computing*. Ed. by GR Raidl. Vol. 3005. Lecture Notes in Computer Sscience. EvoWorkshops Conference, Coimbra, Portugal, April 05-07, 2004, pp. 21–30.

Dasgupta, S., C. H. Papadimitriou, and U. V. Vazirani (2008). *Algorithms*. 1st. McGraw Hill.

Dashtban, M, Mohammadali Balafar, and Prashanth Suravajhala (2018). “Gene selection for tumor classification using a novel bio-inspired multi-objective approach”. In: *Genomics* 110(1), pp. 10–17.

Díaz-Uriarte, Ramón and Sara Alvarez de Andrés (2006). “Gene selection and classification of microarray data using random forest”. In: *BMC Bioinformatics* 7(1), p. 3.

Dos Santos, Bruno César, Cristiane Neri Nobre, and Luis Enrique Zárate (2018). “Multi-objective genetic algorithm for feature selection in a protein function prediction context”. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1–6.

Duval, Batrice and Jin-Kao Hao (2009). “Advances in metaheuristics for gene selection and classification of microarray data”. In: *Briefings in Bioinformatics* 11(1), p. 127.

Emura, Takeshi, Shigeyuki Matsui, and Hsuan-Yu Chen (2019). “compound. Cox: univariate feature selection and compound covariate for predicting survival”. In: *Computer methods and programs in biomedicine* 168, pp. 21–37.

Escoffier, Bruno, Vangelis Th. Paschos, and Emeric Tourniaire (2012). “Approximating MAX SAT by Moderately Exponential and Parameterized Algorithms”. In: *Theory and Applications of Models of Computation: 9th Annual Conference, TAMC 2012, Beijing, China, May 16-21, 2012. Proceedings*. Ed. by Manindra Agrawal, S. Barry Cooper, and Angsheng Li. Springer Berlin Heidelberg: Berlin, Heidelberg, pp. 202–213.

Fan, Ya-Ju and Wanpracha Art Chaovalitwongse (2010). “Optimizing feature selection to improve medical diagnosis”. In: *Annals of Operations Research* 174(1), pp. 169–183.

Farahani, Reza Zanjirani, Nasrin Asgari, Nooshin Heidari, Mahtab Hosseininia, and Mark Goh (2012). “Covering problems in facility location: A review”. In: *Computers & Industrial Engineering* 62(1), pp. 368–407.

Fayyad, Usama M and Keki B Irani (1993). “Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning”. In: *IJCAI*, pp. 1022–1029.

Feo, Thomas A. and Mauricio G. C. Resende (1989). “A probabilistic heuristic for a computationally difficult set covering problem”. In: *Operations Research Letters* 8(2), pp. 67–71.

Ferchichi, Sabra El, Kaouther Laabidi, and Salah Zidi (2009). “Genetic Algorithm and Tabu Search for Feature Selection”. In: *Studies in Informatics and Control* 18(2), pp. 181–187.

Fisher, M. L. and P. Kedia (1990). “Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics”. In: *Manage. Sci.* 36(6), pp. 674–688.

Fox, Richard J and Matthew W Dimmic (2006). “A two-sample Bayesian t-test for microarray data”. In: *BMC bioinformatics* 7(1), p. 126.

Garfinkel, Robert S. and George L. Nemhauser (1972). *Integer Programming.* Wiley & Sons, Inc.

Ghosh, Manosij, Shemim Begum, Ram Sarkar, Debasis Chakraborty, and Ujjwal Maulik (2019). “Recursive Memetic Algorithm for gene selection in microarray data”. In: *Expert Systems with Applications* 116, pp. 172–185.

Goffinet, Jack and Raghuram Ramanujan (2016). “Monte-Carlo Tree Search for the Maximum Satisfiability Problem”. In: *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings.* Ed. by Michel Rueher. Springer International Publishing: Cham, pp. 251–267.

Golovnev, Alexander and Konstantin Kutzkov (2014). “New exact algorithms for the 2-constraint satisfaction problem”. In: *Theoretical Computer Science* 526, pp. 18 –27.

González, Jesús, Julio Ortega, Miguel Damas, John Q Gan, et al. (2019). “A new multi-objective wrapper method for feature selection–Accuracy and stability analysis for BCI”. In: *Neurocomputing.*

Guyon, Isabelle and Andre Elisseeff (2003). “An introduction to variable and feature selection”. In: *The Journal of Machine Learning Research* 3, pp. 1157–1182.

Haddadi, Salim (1997). “Simple Lagrangian heuristic for the set covering problem”. In: *European Journal of Operational Research* 97(1), pp. 200 –204.

Haque, Mohammad Nazmul, Nasimul Noman, Regina Berretta, and Pablo Moscato (2016). “Heterogeneous Ensemble Combination Search Using Genetic Algorithm for Class Imbalanced Data Classification”. In: *PLoS ONE* 11(1), e0146116.

Hua, Qiang-Sheng, Yuexuan Wang, Dongxiao Yu, and Francis C.M. Lau (2010). “Dynamic programming based algorithms for set multicover and multiset multicover problems”. In: *Theoretical Computer Science* 411(2628), pp. 2467 –2474.

Huang, Jun, Guorong Li, Qingming Huang, and Xindong Wu (2018). “Joint feature selection and classification for multilabel learning”. In: *IEEE transactions on cybernetics* 48(3), pp. 876–889.

Ignatiev, A., A. Morgado, V. Manquinho, I. Lynce, and J. Marques-Silva (2014). “Progression in Maximum Satisfiability”. In: *Proceedings of the Twenty-first European Conference on Artificial Intelligence.* ECAI’14. IOS Press: Prague, Czech Republic, pp. 453–458.

Inostroza-Ponta, Mario, University of Newcastle (N.S.W.). School of Electrical Engineering, and Computer Science (2008). "An integrated and scalable approach based on combinatorial optimization techniques for the analysis of microarray data". English. School of Electrical Engineering and Computer Science. Thesis.

Inostroza-Ponta, Mario, Regina Berretta, and Pablo Moscato (2011). "QAPgrid: A Two Level QAP-Based Approach for Large-Scale Data Analysis and Visualization". In: *PLOS ONE* 6(1), pp. 1–18.

Inza, Iñaki, Pedro Larrañaga, Rosa Blanco, and Antonio J. Cerrolaza (2004). "Filter Versus Wrapper Gene Selection Approaches in DNA Microarray Domains". In: *Artif. Intell. Med.* 31(2), pp. 91–103.

Jafari, Peyman and Francisco Azuaje (2006). "An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors". In: *BMC Medical Informatics and Decision Making* 6(1), p. 27.

Jain, Indu, Vinod Kumar Jain, and Renu Jain (2018). "Correlation feature selection based improved-Binary Particle Swarm Optimization for gene selection and cancer classification". In: *Applied Soft Computing* 62, pp. 203–215.

Jiang, Hongying, Youping Deng, Huann-Sheng Chen, Lin Tao, Qiuying Sha, Jun Chen, Chung-Jui Tsai, and Shuanglin Zhang (2004). "Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes". In: *BMC bioinformatics* 5(1), p. 81.

Jirapech-Umpai, Thanyaluk and Stuart Aitken (2005). "Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes". In: *BMC Bioinformatics* 6(1), p. 148.

Kabir, Monirul, Shahjahan, and Kazuyuki Murase (2012). "A new hybrid ant colony optimization algorithm for feature selection". In: *Expert Systems with Applications* 39(3), pp. 3747–3763.

Karp, Richard M. (1972). "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department.* Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Springer US: Boston, MA, pp. 85–103.

Kong, Yunchuan and Tianwei Yu (2018). "A graph-embedded deep feedforward network for disease outcome classification and feature selection using gene expression data". In: *Bioinformatics* 1, p. 11.

Kuncheva, Ludmila I. and Juan J. Rodriguez (2018). "On feature selection protocols for very low-sample-size data". In: *Pattern Recognition* 81, pp. 660 –673.

Lai, Chyh-Ming (2018). "Multi-objective simplified swarm optimization with weighting scheme for gene selection". In: *Applied Soft Computing* 65, pp. 58–68.

Lan, Guanghui, Gail W. DePuy, and Gary E. Whitehouse (2007). "An effective and simple heuristic for the set covering problem". In: *European Journal of Operational Research* 176(3), pp. 1387 –1403.

Lancia, G. (2008). "Mathematical Programming in Computational Biology: an Annotated Bibliography". In: *Algorithms* 1(4), pp. 100–129.

Lesnick, Timothy G, Spiridon Papapetropoulos, Deborah C Mash, Jarlath Ffrench-Mullen, Lina Shehadeh, Mariza de Andrade, John R Henley, Walter A Rocca, J. Eric Ahlskog, and Demetrius M Maraganore (2007). "A Genomic Pathway Approach to a Complex Disease: Axon Guidance and Parkinson Disease". In: *PLoS Genet* 3(6), e98.

Li, Yun, Tao Li, and Huan Liu (2017). "Recent advances in feature selection and its applications". In: *Knowledge and Information Systems* 53(3), pp. 551–577.

Liu, H. and H. Motoda (2007). *Computational Methods of Feature Selection*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press.

Liu, Huan and Hiroshi Motoda (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers: Norwell, MA, USA.

Lockstone, H.E., L.W. Harris, J.E. Swatton, M.T. Wayland, A.J. Holland, and S. Bahn (2007). "Gene expression profiling in the adult Down syndrome brain". In: *Genomics* 90(6), pp. 647–660.

Lovász, L. (1975). "On the ratio of optimal integral and fractional covers". In: *Discrete Mathematics* 13(4), pp. 383 –390.

Lu, Yun and Francis J. Vasko (2015). "An OR Practitioner's Solution Approach for the Set Covering Problem". In: *Int. J. Appl. Metaheuristic Comput.* 6(4), pp. 1–13.

Luscombe, N. M., D. Greenbaum, and M. Gerstein (2001). "What is bioinformatics? A proposed definition and overview of the field". In: *Methods of Information in Medicine* 40(4), pp. 346–358.

Ma, Shuangge and Jian Huang (2005). "Regularized ROC method for disease classification and biomarker selection with microarray data". In: *Bioinformatics* 21(24), pp. 4356–4362.

Martins, Ruben, Vasco Manquinho, and Ins Lynce (2015). "Improving linear search algorithms with model-based approaches for MaxSAT solving". In: *Journal of Experimental & Theoretical Artificial Intelligence* 27(5), pp. 673–701.

Meiri, R and J Zahavi (2006). "Using simulated annealing to optimize the feature selection problem in marketing applications". In: *European Journal of Operational Reserach* 171(3). 21st Euro Summer Institute (ESI), Nida Neringa, LITHUANIA, JUL 25-AUG 07, 2003, pp. 842–858.

Minitab, Inc (2015). *Minitab 17 Statistical Software [Computer software]*. www.minitab.com. State College, PA.

Mount, David W. (2004). *Bioinformatics: Sequence and Genome Analysis*. G - Reference,Information and Interdisciplinary Subjects Series. Cold Spring Harbor Laboratory Press.

Naji-Azimi, Zahra, Paolo Toth, and Laura Galli (2010). “An electromagnetism metaheuristic for the unicost set covering problem”. In: *European Journal of Operational Research* 205(2), pp. 290 –300.

Neter, John (1996). *Applied linear statistical models*. Irwin series in statistics v. 1. Irwin.

Pati, Soumen K, Subhankar Mallick, Aruna Chakraborty, and Ankur Das (2019). “Informative Gene Selection Using Clustering and Gene Ontology”. In: *Emerging Technologies in Data Mining and Information Security*. Springer, pp. 417–427.

Paula, Mateus Rocha de (2012). “Efficient Methods of Feature Selection Based on Combinatorial Optimization Motivated by the Analysis of Large Biological Datasets”. PhD thesis. School of Electrical Engineering and Computer Science, The University of Newcastle, Australia.

Paula, Mateus Rocha de, Martn Gmez Ravetti, Regina Berretta, and Pablo Moscato (2011). “Differences in Abundances of Cell-Signalling Proteins in Blood Reveal Novel Biomarkers for Early Detection Of Clinical Alzheimer’s Disease”. In: *PLoS ONE* 6(3), e17481.

Paula, Mateus Rocha de, Regina Berretta, and Pablo Moscato (2016). “A fast meta-heuristic approach for the (α, β)-k feature set problem”. In: *Journal of Heuristics* 22(2), pp. 199–220.

Pessoa, Luciana S., Mauricio G. C. Resende, and Celso C. Ribeiro (2011). “Experiments with LAGRASP heuristic for set k-covering”. In: *Optimization Letters* 5(3), pp. 407–419.

Pessoa, Luciana S., Mauricio G. C. Resende, and Celso C. Ribeiro (2013). “A Hybrid Lagrangean Heuristic with GRASP and Path-relinking for Set K-covering”. In: *Comput. Oper. Res.* 40(12), pp. 3132–3146.

Petkovska, Ana, Alan Mishchenko, Mathias Soeken, Giovanni De Micheli, Robert Brayton, and Paolo Ienne (2016). “Fast Generation of Lexicographic Satisfiable Assignments: Enabling Canonicity in SAT-based Applications”. In: *Proceedings of the 35th International Conference on Computer-Aided Design*. ICCAD ’16. ACM: Austin, Texas, 4:1–4:8.

Polanski, Andrzej and Marek Kimmel (2007). *Bioinformatics*. Springer, pp. I–XVII, 1–376.

Poloczek, Matthias and David P. Williamson (2016). “An Experimental Evaluation of Fast Approximation Algorithms for the Maximum Satisfiability Problem”. In: *Experimental Algorithms: 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings*. Ed. by Andrew V. Goldberg and Alexander S. Kulikov. Springer International Publishing: Cham, pp. 246–261.

Poloczek, Matthias, David P. Williamson, and Anke van Zuylen (2014). “On Some Recent Approximation Algorithms for MAX SAT”. In: *LATIN 2014: Theoretical Informatics: 11th Latin American Symposium, Montevideo, Uruguay, March 31–April 4, 2014. Proceedings*. Ed. by Alberto Pardo and Alfredo Viola. Springer Berlin Heidelberg: Berlin, Heidelberg, pp. 598–609.

Rais, Abdur and Ana Viana (2011). “Operations Research in Healthcare: a survey”. In: *International Transactions in Operational Research* 18(1), pp. 1–31.

Ramsden, Jeremy (2009). *Bioinformatics: An Introduction*. Springer.

Ravetti, M. Gómez and P. Moscato (2008). "Identification of a 5-protein biomarker molecular signature for predicting Alzheimer's disease". In: *PLoS One* 3(9), e3111.

Ravetti, M Gómez, R Berretta, and P Moscato (2009). "Novel Biomarkers for Prostate Cancer Revealed by (α,β)-k-Feature Sets". In: vol. 5. Foundations of Computational Intelligence. Springer Berlin / Heidelberg. Chap. 7, in Foundations of Computational Intelligence, pp. 149–175.

Ravetti, Martn Gómez, Osvaldo A. Rosso, Regina Berretta, and Pablo Moscato (2010). "Uncovering Molecular Biomarkers That Correlate Cognitive Decline with the Changes of Hippocampus' Gene Expression Profiles in Alzheimer's Disease". In: *PLoS ONE* 5(4), e10153.

Saeys, Yvan, Iaki Inza, and Pedro Larraaga (2007). "A review of feature selection techniques in bioinformatics". In: *Bioinformatics* 23(19), p. 2507.

Salehipour, Amir and Mohammad Mehdi Sepehri (2012). "Exact and Heuristic Solutions to Minimize Total Waiting Time in the Blood Products Distribution Problem". In: *Advances in Operations Research* 2012.

Sánchez-Maroño, Noelia, Amparo Alonso-Betanzos, and María Tombilla-Sanromán (2007). "Filter Methods for Feature Selection – A Comparative Study". In: *Intelligent Data Engineering and Automated Learning - IDEAL 2007: 8th International Conference, Birmingham, UK, December 16-19, 2007. Proceedings.* Ed. by Hujun Yin, Peter Tino, Emilio Corchado, Will Byrne, and Xin Yao. Springer Berlin Heidelberg: Berlin, Heidelberg, pp. 178–187.

Scherzer, Clemens R., Aron C. Eklund, Lee J. Morse, Zhixiang Liao, Joseph J. Locascio, Daniel Fefer, Michael A. Schwarzschild, Michael G. Schlossmacher, Michael A. Hauser, Jeffery M. Vance, Lewis R. Sudarsky, David G. Standaert, John H. Growdon, Roderick V. Jensen, and Steven R. Gullans (2007). "Molecular markers of early Parkinson's disease based on gene expression in blood". In: *Proceedings of the National Academy of Sciences* 104(3), pp. 955–960.

Şeref, Onur, Ya-Ju Fan, Elan Borenstein, and Wanpracha A Chaovalitwongse (2018). "Information-theoretic feature selection with discrete k-median clustering". In: *Annals of Operations Research* 263(1-2), pp. 93–118.

Shukla, Alok Kumar, Pradeep Singh, and Manu Vardhan (2019). "DNA Gene Expression Analysis on Diffuse Large B-Cell Lymphoma (DLBCL) Based on Filter Selection Method with Supervised Classification Method". In: *Computational Intelligence in Data Mining.* Springer, pp. 783–792.

Tsai, Chih-Fong and Yu-Chieh Hsiao (2010). "Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches". In: *Decision Support Systems* 50(1), pp. 258–269.

Unler, Alper and Alper Murat (2010). "A discrete particle swarm optimization method for feature selection in binary classification problems". In: *European Journal of Operational Research* 206(3), pp. 528–539.

Urbanowicz, Ryan J, Randal S Olson, Peter Schmitt, Melissa Meeker, and Jason H Moore (2018). "Benchmarking relief-based feature selection methods for bioinformatics data mining". In: *Journal of biomedical informatics* 85, pp. 168–188.

Vasko, Francis J. (1984). "An efficient heuristic for large set covering problems". In: *Naval Research Logistics Quarterly* 31(1), pp. 163–171.

Vasko, Francis J. and George R. Wilson (1986). "Hybrid heuristics for minimum cardinality set covering problems". In: *Naval Research Logistics Quarterly* 33(2), pp. 241–249.

Vieira, Susana M, Joao M C Sousa, and Thomas A Runkler (2010). "Two cooperative ant colonies for feature selection using fuzzy models". In: *Expert Systems with Applications* 37(4), pp. 2714–2723.

Vieira, Susana M, Joao M C Sousa, and Uzay Kaymak (2012). "Fuzzy criteria for feature selection". In: *Fuzzy Sets and Systems* 189(1), pp. 1–18.

Wagner, Michael, Jarosław Meller, and Ron Elber (2004). "Large-scale linear programming techniques for the design of protein folding potentials". In: *Mathematical Programming* 101(2), pp. 301–318.

Wang, Chia-Ming and Yin-Fu Huang (2009). "Evolutionary-based feature selection approaches with new criteria for data mining: A case study of credit approval data". In: *Expert Systems with Applications* 36(3), pp. 5900–5908.

Wang, L., A. Ngom, and R. Gras (2008). "Non-unique oligonucleotide microarray probe selection method based on genetic algorithms". In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1004–1010.

Wang, Lipo (2012). "Feature selection in bioinformatics". In: *SPIE Defense, Security, and Sensing.* International Society for Optics and Photonics, pp. 840113–840113.

Wang, Lipo, Yaoli Wang, and Qing Chang (2016a). "Feature selection methods for big data bioinformatics: A survey from the search perspective". In: *Methods* 111. Big Data Bioinformatics, pp. 21 –31.

Wang, Yiyuan, Minghao Yin, Dantong Ouyang, and Liming Zhang (2016b). "A novel local search algorithm with configuration checking and scoring mechanism for the set k-covering problem". In: *International Transactions in Operational Research.*

Wang, Yu, Igor V. Tetko, Mark A. Hall, Eibe Frank, Axel Facius, Klaus F.X. Mayer, and Hans W. Mewes (2005). "Gene selection from microarray data for cancer classificationa machine learning approach". In: *Computational Biology and Chemistry* 29(1), pp. 37 –46.

Waterman, Michael S (1995). *Introduction to Computational Biology: Sequences, Maps and Genomes.* CRC Press.

Xiong, Momiao, Xiangzhong Fang, and Jinying Zhao (2001). "Biomarker identification by feature wrappers". In: *Genome Research* 11(11), pp. 1878–1887.

Xu, Ying, Dong Xu, and Harold N Gabow (2000). "Protein domain decomposition using a graph-theoretic approach". In: *Bioinformatics* 16(12), pp. 1091–1104.

Yagiura, Mutsunori, Masahiro Kishida, and Toshihide Ibaraki (2006). “A 3-flip neighborhood local search for the set covering problem”. In: *European Journal of Operational Research* 172(2), pp. 472 –499.

Yang, J and S Olafsson (2009). “Near-optimal feature selection for large databases”. In: *Journal of the Operational Research Society* 60(8), pp. 1045–1055.

Yongming, Li, Zhang Sujuan, and Zeng Xiaoping (2009). “Research of multi-population agent genetic algorithm for feature selection”. In: *Expert Systems with Applications* 36(9), pp. 11570–11581.

Zhang, Xin, Ravi Mangal, Aditya V. Nori, and Mayur Naik (2016). “Query-guided Maximum Satisfiability”. In: *SIGPLAN Not.* 51(1), pp. 109–122.

Zhao, Zheng, Fred Morstatter, Shashvata Sharma, Salem Alelyani, Aneeth Anand, and Huan Liu (2010). *title*. Tech. rep. Arizona State University Feature Selection Repository.