6-30-2014

Novel Models and Efficient Algorithms for Network-based Optimization in Biomedical Applications

Seyed Javad Sajjadi
University of South Florida, sjsajjadi@yahoo.com

Novel Models and Efficient Algorithms for Network-based Optimization in Biomedical Applications

by

Seyed Javad Sajjadi

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Industrial Engineering
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Date of Approval:
June 30, 2014

Keywords: Integer Programming, Combinatorial Optimization, Column Generation,
Maximum Clique Problem, Biomarker Identification

DEDICATION

To my loving parents and brother for their endless love, devotion and support.

ACKNOWLEDGMENT

It is a great pleasure to appreciate a number of people who have made this work possible. First and foremost, I would like to gratefully thank my co-advisors Dr. Bo Zeng and Dr. Xiaoning Qian for their tremendous encouragement and support during the course of my Ph.D. study. They have been my mentors and friends from whom I have learned so much regarding research and life. I also owe a great deal to Dr. Jose Zayas-Castro, Dr. Tapas Das and Dr. Kendra Vehik for graciously serving on my doctoral committee and their constructive comments, and to Amin Ahmadi Adl, without whom chapter 5 would not exist, and to Dr. Yao Liu for chairing my dissertation defense. My deepest gratitude goes to my family for their lifelong kindness and support. Lastly, I offer my regards to all of those who supported me in any respect during the completion of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

We introduce and study a novel graph optimization problem to search for multiple cliques with the maximum overall weight, to which we denote as *the Maximum Weighted Multiple Clique Problem (MWMCP)*. This problem arises in research involving network-based data mining, specifically, in bioinformatics where complex diseases, such as various types of cancer and diabetes, are conjectured to be triggered and influenced by a combination of genetic and environmental factors. To integrate potential effects from interplays among underlying candidate factors, we propose a new network-based framework to identify effective biomarkers by searching for "groups" of synergistic risk factors with high predictive power to disease outcome. An interaction network is constructed with vertex weight representing individual predictive power of candidate factors and edge weight representing pairwise synergistic interaction among factors. This network-based biomarker identification problem is then formulated as a MWMCP. To achieve near optimal solutions for large-scale networks, an analytical algorithm based on column generation method as well as a fast greedy heuristic have been derived. Also, to obtain its exact solutions, an advanced branch-price-and-cut algorithm is designed and solved after studying the properties of the problem. Our algorithms for MWMCP have been implemented and tested on random graphs and promising results have been obtained. They also are used to analyze two biomedical datasets: a Type 1 Diabetes (T1D) dataset from the Diabetes Prevention Trial-Type 1 (DPT-1) Study, and a breast cancer genomics dataset for metastasis prognosis. The results demonstrate that our network-based methods can identify important biomarkers with better prediction accuracy compared to the conventional feature selection that only considers individual effects.

CHAPTER 1: INTRODUCTION[1]

Modern high-throughput technologies have generated unprecedented amounts of large-scale high-dimensional "-omics" data for better understanding complex diseases, which have been commonly believed to result from complicated interactions between both genetic risk factors and environmental exposures [2]. Analyzing these high-dimensional heterogeneous data to identify effective biomarkers for better disease prognosis and diagnosis has been a critical challenge in computational biology [3, 4, 5]. Previous methods have focused on either greedy or penalized feature selection including LASSO [4, 5, 6, 7], which typically do not explicitly consider interactions among different candidate risk factors. These methods have shown limited power to identify stable and effective biomarkers with high predictive power in complex disease studies as interactive effects may be essential to understand these systems impairments, including cancer and diabetes [2, 8, 9, 10].

To bridge this discrepancy, in a previous study [11], an interaction network representation scheme has been developed to capture both the individual effects from candidate risk factors and the pairwise interactive effects among them. Network representations have been effectively used in various research studies in bioinformatics, specifically, in [12] a molecular interaction network is used to analyze changes in expression for yeast galactose utilization pathway, and in [13] an approach is introduced for screening this network to identify

[1]This chapter was partially published in [1]. Permission is included in Appendix A.

connected subnetworks which lead to significant changes in expression. A model of genetic network interactions is presented in [14] to identify drug targets, and in [15], this method is combined with expression profiles to identify the genetic mediators and mediating pathways associated with prostate cancer. In [9] a protein-based-network approach is applied to identify markers of metastasis for breast cancer.

In our interaction network, each vertex represents a candidate risk factor and its assigned vertex weight describes its individual predictive power for the outcome of interest. An edge between any pair of vertices also has an assigned edge weight corresponding to the synergistic power of the interaction between two corresponding factors. There are different ways to estimate synergy between two risk factors, the estimation we present is based on regression models. In this interaction network framework, we then formulate the biomarker identification problem as a network optimization problem to search for a Maximum Weighted Clique (MWC) that has the maximum total weight from both constituent vertices and induced edges. The identified MWC is a complete subnetwork with selected risk factors that have the highest predictive power with the most synergistic interactions among them. Therefore, interactive effects among risk factors are integrated together with individual effects for the most effective biomarker identification. It has been known that complex diseases may be triggered and affected by multiple factors (genetic as well as environmental) [2, 16, 17, 18], which indicates a single clique may not be sufficient to fully explain the cause or development of disease. So, a more comprehensive model should be developed and employed to identify a set of highly synergistic cliques in a systematic way. However, such a task imposes a big computational challenge, given the fact that computing a single MWC is already NP-hard [19, 20]. Actually, to the best of our knowledge, there has been no analytical study on selecting a set of cliques whose total weight for both vertices and edges is maximized.

To achieve the goal of identifying effective biomarkers for accurate disease prognosis and diagnosis, we aim to address this challenge by first developing advanced mathematical models and algorithms to identify multiple cliques from our interaction network representation. Specifically, a discrete optimization model, which seeks for a collection of non-overlapping (disjoint) cliques with maximum total weight, and its top-K extension, which restricts the cardinality of that collection to K, are constructed. We observe that although those formulations are compact, the state-of-the-art professional solvers cannot even deal with very small instances with tens of candidate risk factors. Therefore, a sophisticated computational strategy, i.e., the *branch-price-and-cut* method [21], is adopted and customized to identify those disjoint cliques simultaneously. Branch-price-and-cut is an advanced solution framework to solve complex integer programs to optimality and can be described as an algorithm that incorporates three different tools: *branch-and-bound*, *column generation* and *cutting planes*. Branch-and-bound (B&B) allows to partition the original problem into smaller and easier problems. The information obtained by solving some of these problems, is used to assure that many of the remaining problems do not need to be solved as they do not contain the optimal solution. The smaller problems are solved by column generation (CG) algorithm that is a two-level master-subproblem computing framework [22, 23] and is suitable to solve large-scale optimization problems. In the master problem, which is a computationally friendly linear program (LP), a solution is derived based on a *restricted* set of feasible solutions. Then, the dual information of the LP problem is applied to populate the subproblem to generate high quality potential solutions, which will be used to augment the feasible solution set of the master problem. By iteratively computing the master and subproblems, a global optimal solution of a large-scale instance can be obtained. Recently, this computational method has been adopted in the study of predicting HIV-1 drug resistance [24] and protein fold prediction [25] problems. To strengthen the LP relaxation in a master problem, cutting planes can be generated and added. Cutting planes are valid inequalities for the integer programming formulation that cut out non-integral optimal solutions from its LP relaxation. It leads to achieving more useful information to be used in branch-and-bound.

In addition to the branch-price-and-cut algorithm, to further improve our solution capability for large-scale problems with -omics data, a heuristic method based on CG, as well as a fast greedy heuristic method are also designed to identify highly weighted cliques from the network. These heuristic algorithms allow us to handle networks at different scales in a reasonable time with desired quality.

A set of random networks are generated to test the performance of the algorithms. The experimental results have shown that the algorithms are capable to achieve optimal or near optimal solutions for various networks efficiently. We also have performed a set of experiments on real datasets to demonstrate the significance of considering synergistic interactions for biomarker identification as well as the effectiveness of identified biomarkers for disease prognosis. Two real datasets are studied: Type 1 Diabetes (T1D) and breast cancer. Our experimental results with constructed interaction networks from both T1D and breast cancer datasets have shown that our network-based biomarker identification methods can effectively identify critical biomarkers for better prediction accuracy.

In chapter 2, a background of the MWCP is presented and its generalization to the MWMCP formulation is introduced. In chapter 3, the CG reformulation and its benefits are discussed and its performance is tested. Chapter 4 delineates the incorporation of branch-and-bound and cutting planes to the CG, i.e., the branch-price-and-cut algorithm together with the implementation techniques and computational results on random graphs. In chapter 5, a real-life application of the MWMCP in biomarker identification is explained and compared to the conventional methods. Finally, chapter 6 outlines the possible future work on this research in various directions.

CHAPTER 2: CLIQUE-BASED NETWORK OPTIMIZATION BACKGROUND

In a weighted undirected graph $G(V, E)$ where $V = \{v_1, ..., v_n\}$ and $E \subseteq V \times V$ are sets of vertices and edges respectively, a subgraph $C \subseteq V$ is a clique if all pairs of vertices in C are directly connected by edges in E. All vertices and edges are assigned weights by a function $w : V \cup E \to \mathbb{R}$. Let e_{ij} denote the edge between v_i and v_j and $E_C \subseteq E$ denote the set of edges induced by clique C. Then the weight of clique C, denoted by w_C, is defined as $w_C = \sum_{v_i \in C} w(v_i) + \sum_{e_{ij} \in E_C} w(e_{ij})$. In the *Maximum Weighted Clique Problem (MWCP)* the objective is to find a clique C with maximum w_C. See Figure 1 for a simple example. Note that the MWCP is a generalization of the classical *Maximum Clique Problem (MCP)* in which $w(v_i) = 1$ for all $v_i \in V$ and $w(e_{ij}) = 0$ for all $e_{ij} \in E$. Consequently, in the MCP, $w_C = |C|$ and the objective is to find a clique C of largest size. The MCP is a well known NP-hard problem [19]. It is shown that this problem is actually not even approximable [26] . As a consequence, the MWCP is also NP-hard. The MCP and MWCP have been central problems in graph theory and have been used in numerous applications, including sociology [27, 28], computational chemistry [29], computer vision [30], coding theory [31], fault diagnosis [32], geometric tiling [33], and especially bioinformatics [34, 35, 36]. We refer the reader to the thorough survey in [20] for more details on the tremendous effort that has been made on devising solution algorithms for both weighted and unweighted maximum clique problems over years. Furthermore, because it is computationally equivalent to the *Maximum Independent Set Problem (MISP)* and the *Minimum Vertex Cover Problem*, the study of this problem is very important.

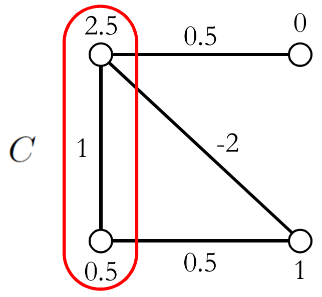


Figure 1: A solution of the Maximum Weighted Clique Problem. All nodes and edges are weighted and $w_C = 2.5 + 1 + 0.5 = 4$.

2.1 Problem Formulation and Computational Complexity

In this dissertation, we introduce the *Maximum Weighted Multiple Clique Problem (MWMCP)* as an expansion of the MWCP. In the weighted undirected graph $G(V, E)$, the objective of the MWMCP is to find a collection of disjoint (mutually exclusive) cliques with maximum total weight, i.e., to find $\mathbb{C}^* = \{C_1, ..., C_K\}$ such that $\sum_{C_i \in \mathbb{C}^*} w_{C_i}$ is maximized and $C_i \cap C_j = \varnothing$ for all $C_i, C_j \in \mathbb{C}^*, i \neq j$. A schematic example of $\mathbb{C}^*$ and G is illustrated in Figure 2. Because this is the first formal study of the MWMCP, we determine its computational complexity first.

Theorem 1 *The Maximum Weighted Multiple Clique Problem (MWMCP) is NP-hard.*

Proof. We provide a proof by reducing the Exact Cover problem, a well-known NP-complete problem [37], to a MWMCP. Given an instance $\langle \mathbb{V}, \mathbb{S} \rangle$ of Exact Cover, we construct an

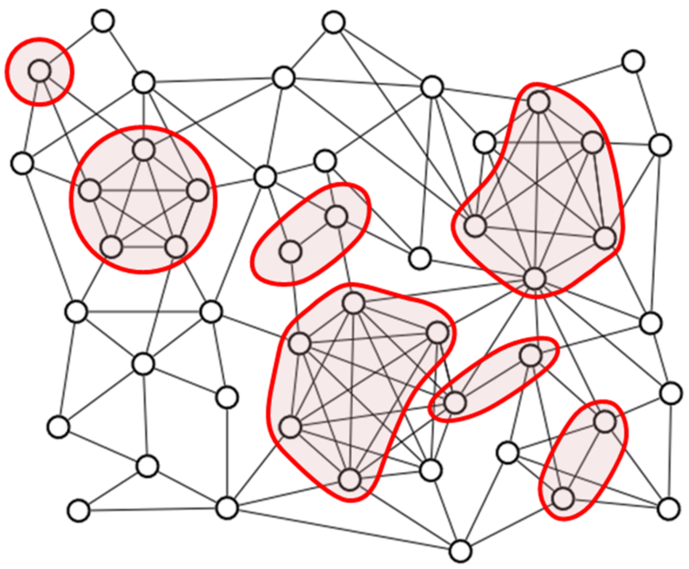


Figure 2: A solution of the Maximum Weighted Multiple Clique Problem. All the nodes and edges are weighted in this graph.

instance $\langle G', w \rangle$ of MWMCP in polynomial time such that there exists an exact cover $\mathcal{S}^* \subseteq \mathbb{S}$ *if and only if* $\mathcal{S}^*$ corresponds to an optimal solution for MWMCP in G' with weight equal to $|\mathbb{V}|$. Given $\mathbb{S} = \{S_1, ..., S_m\}$ a collection of subsets of $\mathbb{V} = \{v_1, ..., v_n\}$, the exact cover problem can be represented by a bipartite graph (See Figure 3 for an example). Vertices of this graph consist of two groups, one representing m subsets in $\mathbb{S}$ and another representing n elements in $\mathbb{V}$. If an element is contained in a subset, an edge connects the corresponding vertices in the graph. Let $G = (\mathbb{S} \cup \mathbb{V}, E)$ denote this graph. Then, an exact cover is a selection of vertices $\mathcal{S}^* \subseteq \mathbb{S}$ such that any vertex in $\mathbb{V}$ is connected to exactly one element in $\mathcal{S}^*$ by an edge in E. To construct $\langle G', w \rangle$, let $G' = (\mathbb{S} \cup \mathbb{V}, E \cup E')$ where E' is a set of $\binom{n}{2}$ edges connecting all pairs of vertices in $\mathbb{V}$, and let w be a function assigning real-valued weights to vertices and edges in G'. Zero weights are assigned to all elements in $\mathbb{V}$ and E' while weight of edges in E is $1 + \epsilon$ and weight of vertex $S_j \in \mathbb{S}$ is $-d_j \epsilon$ where d_j is the degree of vertex S_j for all $j = 1, ..., m$ and $\epsilon \in (0, \frac{1}{n})$. Note that degree of vertex S_j is the same in both G and G' for all j.

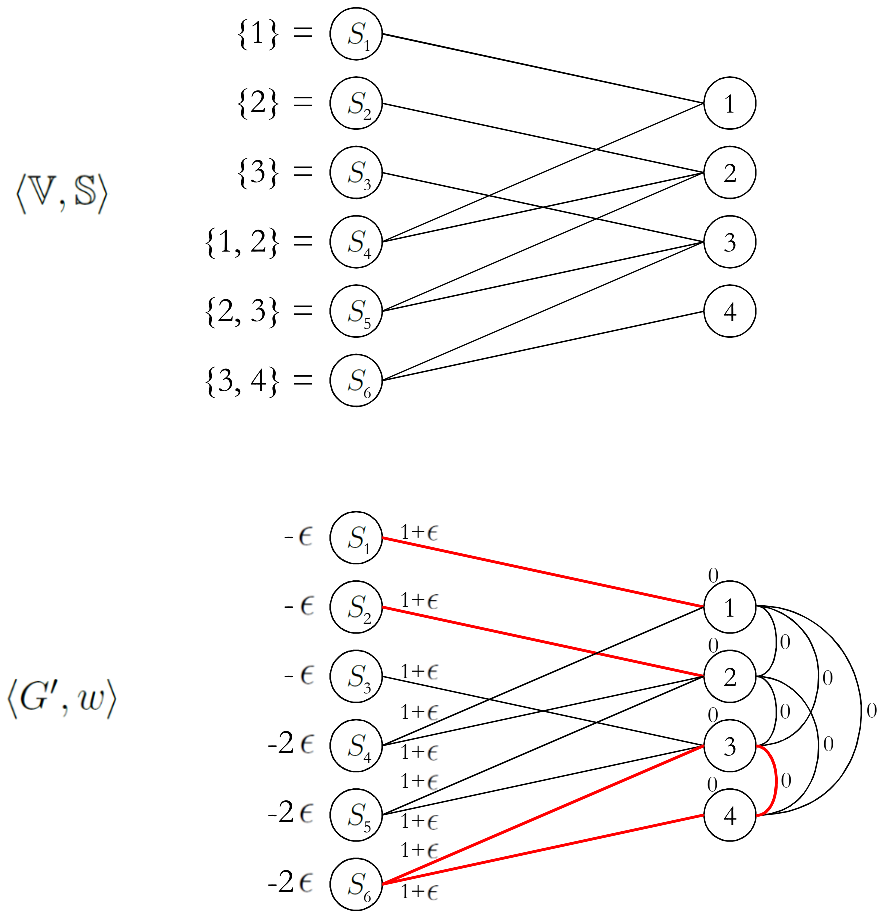


Figure 3: Reduction of an exact cover problem (top) to a MWMCP (bottom).

We now show that if there exists an exact cover $\mathcal{S}^* \subseteq \mathbb{S}$, then it corresponds to an optimal solution for MWMCP in G' with total weight of $|\mathbb{V}|$. For any $S_j \in \mathcal{S}^*$, let Q_j be the maximal clique that includes vertex S_j. Q_j clearly includes all neighbors of S_j as they all are directly connected. Let $\mathcal{Q}^*$ be the collection of such cliques and corresponding to $\mathcal{S}^*$. Since $\mathcal{S}^*$ is an exact cover, cliques in $\mathcal{Q}^*$ are disjoint and therefore $\mathcal{Q}^*$ is a feasible solution to MWMCP. Denote the weight of Q_j by $W_{Q_j} = d_j(1 + \epsilon) - d_j\epsilon = d_j$ and let $W_{\mathcal{Q}^*}$ denote the total weight of $\mathcal{Q}^*$. We have

$$W_{\mathcal{Q}^*} = \sum_{j:Q_j \in \mathcal{Q}^*} W_{Q_j} = \sum_{j:Q_j \in \mathcal{Q}^*} d_j = \sum_{j:S_j \in \mathcal{S}^*} d_j = |\mathbb{V}|.$$

We claim that the total weight of any feasible solution for MWMCP in G' cannot exceed $|\mathbb{V}|$ and hence $\mathcal{Q}^*$ is an optimal solution. We provide the proof for this claim later on.

We now show that if the total weight of the optimal solution for MWMCP in G' is equal to $|\mathbb{V}|$, then there exists an exact cover $\mathcal{S}^* \subseteq \mathbb{S}$. Let $\mathcal{Q}^*$ be an optimal solution for MWMCP with $W_{\mathcal{Q}^*} = |\mathbb{V}| = n$ and let k denote the number of vertices in $\mathbb{V}$ that are connected to vertices in $\mathbb{S}$ with selected edges in $\mathcal{Q}^*$. If $k < n$, we have

$$\begin{aligned} W_{\mathcal{Q}^*} &= \sum_{j:Q_j \in \mathcal{Q}^*} W_{Q_j} = k(1+\epsilon) - \sum_{j:Q_j \in \mathcal{Q}^*} d_j \epsilon \\ &< k(1+\epsilon) < k + \frac{k}{n} \\ &\leq n. \end{aligned}$$

Hence, k must be equal to n. Given $\mathcal{Q}^*$, we construct a corresponding $\mathcal{S}^*$ and show that it is an exact cover. For any $Q_j \in \mathcal{Q}^*$ let $S_j = Q_j \cap \mathbb{S}$ be its corresponding element in $\mathcal{S}^*$. Note that S_j is nonempty since $k = n$. Denote by l_j the number of vertices in $\mathbb{V}$ that are selected in Q_j. Note that $l_j \leq d_j$ since $k = n$, also, $l_j \geq 1$ since vertices in $\mathbb{S}$ cannot be selected as single-vertex cliques due to their negative weight. Then

$$\begin{aligned}
W_{\mathcal{Q}^*} = n \ &\Rightarrow \sum_{j:Q_j \in \mathcal{Q}^*} W_{Q_j} = n \\
&\Rightarrow \sum_{j:Q_j \in \mathcal{Q}^*} [l_j(1+\epsilon) - d_j\epsilon] = n \\
&\Rightarrow \epsilon \sum_{j:Q_j \in \mathcal{Q}^*} (l_j - d_j) = n - \sum_{j:Q_j \in \mathcal{Q}^*} l_j \qquad (2.1)
\end{aligned}$$

The left hand side of the equation (2.1) is nonpositive while the right hand side is nonnegative, thus both are equal to zero. Then

$$\begin{aligned}
\sum_{j:Q_j \in \mathcal{Q}^*} (l_j - d_j) = 0 \ &\Rightarrow l_j = d_j \ \forall j : Q_j \in \mathcal{Q}^* \\
&\Rightarrow l_j = d_j \ \forall j : S_j \in \mathcal{S}^* \\
&\Rightarrow \mathcal{S}^* \subseteq \mathbb{S}.
\end{aligned}$$

Since each vertex in $\mathbb{V}$ is in at most one clique in $\mathcal{Q}^*$, it can be connected to at most one element in $\mathcal{S}^*$. Also from (2.1) we have

$$\sum_{j:Q_j \in \mathcal{Q}^*} l_j = n \ \Rightarrow \sum_{j:S_j \in \mathcal{S}^*} l_j = n.$$

This shows that $\mathcal{S}^*$ is an exact cover. Note that if the weight of the optimal solution for MWMCP in G' is less than $|\mathbb{V}|$, $\mathbb{S}$ contains no exact cover.

Lastly, we prove the claim we made earlier. We need to show that the total weight of any feasible solution for MWMCP in G' does not exceed $|\mathbb{V}| = n$. Let $\mathcal{Q}'$ be an arbitrary feasible solution for MWMCP in G'. Let k be the number of vertices in $\mathbb{V}$ that are connected to vertices in $\mathbb{S}$ by selected edges in $\mathcal{Q}'$. Denote by $W_{\mathcal{Q}'}$ the total weight of $\mathcal{Q}'$. Then

$$\begin{aligned} W_{\mathcal{Q}'} &= \sum_{j:Q_j\in\mathcal{Q}'} \big[l_j(1+\epsilon) - d_j\epsilon\big] \\ &= \sum_{j:Q_j\in\mathcal{Q}'} l_j - \epsilon \sum_{j:Q_j\in\mathcal{Q}'} (d_j - l_j) \\ &= k - \epsilon \sum_{j:Q_j\in\mathcal{Q}'} (d_j - l_j) \\ &\le k \le n. \end{aligned}$$

□

As a preprocessing step, we identify certain conditions to remove some of the vertices from G and make the problem smaller and easier to solve. Any vertex that cannot be in the optimal solution or only can be selected within a single-vertex clique is removed in this step. These conditions are depicted in Theorem 2 and 3 in detail.

THEOREM 2 *Suppose* $w(v_i) + \sum_{j:e_{ij}\in E} max\{0, w(e_{ij})\} < 0$ *for some* $v_i \in \mathbb{V}$*, then* v_i *is not in any optimal solution of the MWMCP.*

Proof. Assume v_i is in the optimal solution within clique C. By substituting C by $C' = C \setminus \{v_i\}$ in the solution, we can form a feasible solution with a better objective value since we have

$$\begin{aligned} w_C &= w_{C'} + w(v_i) + \sum_{j:e_{ij}\in E} w(e_{ij}) \\ &\le w_{C'} + w(v_i) + \sum_{j:e_{ij}\in E} max\{0, w(e_{ij})\} \\ &< w_{C'}. \end{aligned}$$

This contradicts with the optimality of the solution. Notice if $C = \{v_i\}$, then $w_{C'} = w_{\varnothing} = 0$.

□

Theorem 3 *Suppose $w(v_i) > 0$ and $w(e_{ij}) < 0$ $\forall e_{ij} \in E$ for some $v_i \in \mathbb{V}$, then any optimal solution of MWMCP contains v_i within a single-vertex clique.*

Proof. First we show by contradiction that v_i must be in any optimal solution, and second, we proof it is selected within a single-vertex clique.

Assume v_i is not in the optimal solution for MWMCP. By adding a single-vertex clique $C_1 = \{v_i\}$ to the solution, we form a feasible solution with a better objective value because we have $w_{C_1} = w(v_i) > 0$. This contradicts with the optimality of the solution.

Now, assume v_i is in the optimal solution within clique C that is not single-vertex ($|C| > 1$). Let $C_1 = \{v_i\}$ and $C_2 = C \setminus \{v_i\}$. By substituting C by C_1 and C_2 in the solution, we can form a feasible solution with a better objective value because:

$$\begin{aligned} w_C &= w_{C_1} + w_{C_2} + \sum_{j:e_{ij} \in E} w(e_{ij}) \\ &< w_{C_1} + w_{C_2}. \end{aligned}$$

This contradicts with the optimality of the solution. $\square$

The MWMCP can be formulated as an quadratic integer programming model in **QIP**.

$$\textbf{QIP}: \quad max \quad \sum_k \Big[\sum_i w(v_i) X_{ik} + \sum_i \sum_{j>i} w(e_{ij}) X_{ik} X_{jk} \Big]$$

$$s.t. \quad \sum_k X_{ik} \le 1 \qquad \forall i \tag{2.2}$$

$$X_{ik} + X_{jk} \le 1 \qquad \forall i, j, k : j > i, \ e_{ij} \notin E \tag{2.3}$$

$$X_{ik} \in \{0,1\} \qquad \forall i,k$$

$$i = 1,...,n \ , \ j = 1,...,n \ , \ k = 1,...,\mathbb{K}$$

where X_{ik} is a binary variable equal to 1 if vertex v_i is selected in k-th clique and 0 otherwise and parameter $\mathbb{K}$ is an upper bound for the number of selected cliques, e.g., $\mathbb{K} = n$. There are n constraints of type (2.2) to guarantee that no vertex will be selected in multiple cliques, while there are $\big(\binom{n}{2} - |E|\big)\mathbb{K}$ constraints of type (2.3) to avoid selection of non-adjacent vertices in the same clique. Also, there are $n\mathbb{K}$ variables in the model. Note that **QIP** can be easily linearized by introducing $\binom{n}{2}\mathbb{K}$ new variables and $2\binom{n}{2}\mathbb{K}$ new constraints as in **LIP**.

$$\begin{aligned}
\textbf{LIP}: \quad max \quad & \sum_k \Big[\sum_i w(v_i)X_{ik} + \sum_i \sum_{j>i} w(e_{ij})Z_{ijk}\Big] & & \\
s.t. \quad & \sum_k X_{ik} \leq 1 & \forall i & \\
& X_{ik} + X_{jk} \leq 1 & \forall i,j,k : j > i, e_{ij} \notin E & \\
& Z_{ijk} \leq \frac{1}{2}(X_{ik} + X_{jk}) & \forall i,j,k : j > i & \quad (2.4) \\
& Z_{ijk} \geq X_{ik} + X_{jk} - 1 & \forall i,j,k : j > i & \quad (2.5) \\
& X_{ik}, Z_{ijk} \in \{0,1\} & \forall i,j,k & \\
& i = 1,...,n, j = 1,...,n, k = 1,...,\mathbb{K}, & &
\end{aligned}$$

where variable Z_{ijk} equals to 1 if edge e_{ij} is selected in clique k and 0 otherwise, note that we let $Z_{ijk} = X_{ik}X_{jk}$ $\forall i, j > i, k$ by adding constraints (2.4) and (2.5).

These compact formulations are not suitable to solve the MWMCP since their linear relaxations are weak. Furthermore, their symmetric structures make any solution algorithm inefficient. Finally, solving an integer quadratic programming or its linearized version that has many constraints is a daunting task. However, due to the special problem structure, it can be decomposed and reformulated to obtain a stronger relaxation and also to reduce symmetry. This reformulation leads to a linear problem with many variables. Nonetheless, dynamic addition of such variables entailed by a column generation approach. In chapter 3 and 4 we explain how this algorithm is adopted and then combined with branch-and-bound and cutting planes to solve the MWMCP efficiently.

CHAPTER 3: A COLUMN GENERATION FORMULATION [1]

In many large linear programs, it is intractable to consider all the variables explicitly and only a subset of columns are needed to solve the problem. This approach is based on the observation that in any optimal solution of large problems, most columns will be nonbasic with their corresponding variables equal to zero. Therefore, a large majority of columns are irrelevant for solving the problem. Leveraging this idea, column generation (CG) [22, 23, 38] only generates the variables that potentially improve the objective function, i.e., finding variables with positive reduced cost (negative in minimization problems).

Besides the advantage of solving a problem with a huge number of variables implicitly, there are several reasons that a column generation formulation is considered. It provides stronger linear programming (LP) relaxation than the compact formulation (**LIP**). Also, a compact formulation may have symmetric structure resulting in a poor performance of branch-and-bound. Further, CG decomposes the problem into a master problem and a subproblem, and may provide an interpretation for the nature of the problem. This may allow for further structural investigation, e.g., incorporation of special constraints. In the MWMCP, we benefit from all of these reformulation properties.

In general, a column generation algorithm has two parts in implementation: a master problem and a subproblem. The master problem is the original problem consisting of only a

[1] This chapter was partially published in [1]. Permission is included in Appendix A.

subset of variables and hence is called a *restricted master problem (RMP)*. The subproblem on the other hand is a new problem created and solved to identify new variables that will be added to the master problem. In the subproblem, the objective function is the reduced cost of the new variable with respect to the current dual prices, and the constraints require that the variable obey the naturally occurring constraints. The column generation algorithm works as follows. The master problem is solved and dual prices are obtained for each of the constraints. Then, this dual information is passes into the subproblem and utilized in the objective function. The subproblem is solved and if the objective value is positive (negative for minimization problems), an improving variable has been identified. This new variable is added to the master problem. The master problem is re-solved and the new set of dual prices are obtained. This process is repeated until no positive reduced cost variable is identified by subproblem, i.e., the master problem is optimal.

The column generation algorithm was first used by Ford and Fulkerson [39] for a multi-commodity maximum flow problem. However, its formulation of integer programs was first proposed by Gilmore and Gomory [40, 41] on the cutting stock problem. The implementation of column generation on integer problems, arises several technical considerations [42].

In the case of integer problems, LP relaxation is used to form the master problem. An exact solution is obtained only if the column generation procedure terminates with an integer solution to the master problem. However, when this solution is not integral, further steps are necessary to be taken in order to find a feasible solution. In this study we take two different routes to obtain an integral solution. First, we include the integrality restriction in the master problem and resolve it. We term this method **CG-IP**. Although it does not guarantee to obtain the optimal solution, the integer solution of this method is of high quality in general because the column generation procedure only generates most needed columns. Second, we employ a branch-and-price algorithm [43, 44] to guarantee the optimality. The branch-and-price algorithm is a branch-and-bound method that at each vertex of the tree, columns may be generated and added to the master problem. In this chapter we discuss the reformulation of the MWMCP to perform a column generation algorithm as well as the **CG-IP** method. In chapter 4, we discuss how the branch-and-price algorithm can be utilized to obtain optimal solutions for a MWMCP.

3.1 Problem Reformulation

Given a graph $G = (V, E)$, any feasible solution for the MWMCP is a collection of disjoint cliques in G. Let $\mathbb{C} = \{C \subseteq \mathbb{V} : C \text{ is a clique in } G\}$ be the set of all cliques in G and for any $C \in \mathbb{C}$, w_C denotes the weight of clique C. As $\mathbb{C}$ is a finite set and the objective of the MWMCP is to find a solution with maximum total weight, one can formulate the MWMCP as the following integer (binary) programming model:

$$\begin{aligned}
\textbf{MWMCP-IP}: \quad & \mathit{max} && \sum_{C \in \mathbb{C}} w_C X_C && \\
& \mathit{s.t.} && \sum_{\{C \in \mathbb{C} : i \in C\}} X_C \leq 1 \quad \forall i \in V && (3.1) \\
& && X_C \in \{0, 1\} \qquad \forall C \in \mathbb{C} && (3.2)
\end{aligned}$$

where X_C is a binary variable equal to 1 if clique C is selected in the solution and 0 otherwise. In this formulation, the objective function is to maximize the total weight of the selected cliques and constraints (3.1) are to guarantee that selected cliques do not overlap.

As the size of $\mathbb{C}$ can exponentially grow with respect to the size of $\mathbb{V}$, the column generation algorithm is suitable here as it allows us to solve this model without the need of knowing set $\mathbb{C}$. Let $\overline{\mathbb{C}} \subseteq \mathbb{C}$ be a subset of cliques in G, substituting $\mathbb{C}$ in **MWMCP-IP** by $\overline{\mathbb{C}}$, gives a restricted version of the problem. Theorem 4 shows that this restricted version is still NP-hard.

THEOREM 4 *Let $\overline{\mathbb{C}}$ be an arbitrary set of weighted cliques in graph $G = (V, E)$, the problem of finding a collection of disjoint cliques $C^* \subseteq \overline{\mathbb{C}}$ with maximum total weight is NP-hard.*

Proof. We provide a proof by reducing the Exact Cover problem, a well-known NP-complete problem [37] to this problem.

Given an instance $\langle \mathbb{U}, \mathbb{S} \rangle$ of Exact Cover, we construct an instance $\langle \overline{\mathbb{C}}, w \rangle$ of this problem in polynomial time where $\overline{\mathbb{C}}$ is the set of cliques and w is a function assigning weights to cliques. Let $\overline{\mathbb{C}} = \mathbb{S}$ be a representation of every set in $\mathbb{S}$ with a clique in $\overline{\mathbb{C}}$, and $w(C_j) = |C_j| \; \forall C_j \in \overline{\mathbb{C}}$. Now, it is easy to see that there exists an exact cover $\mathcal{S}^* \subseteq \mathbb{S}$ *if and only if* the total weight of maximum multiple cliques in $C^* = \mathcal{S}^*$ equals to $|\mathbb{U}|$. $\square$

To perform the column generation algorithm, a master problem and a subproblem need to be formulated properly.

3.1.1 Master Problem

MWMCP-IP is an integer programming model and its linear relaxation is considered by relaxing the integrality constraints (3.2) to form the master problem. Moreover, the column generation algorithm can start with a subset $\overline{\mathbb{C}}$ of all cliques. Finally, to show that cliques are actually columns in this formulation, we define parameter $a_{i,C} = 1$ if $i \in C$ and 0 otherwise for all $i \in \mathbb{V}$ and $C \in \overline{\mathbb{C}}$. Hence, the master problem (**MP**) is as follows.

$$
\begin{aligned}
\textbf{MP}: \quad & max \quad && \sum_{C \in \overline{\mathbb{C}}} w_C X_C && \\
& s.t. && \sum_{C \in \overline{\mathbb{C}}} a_{i,C} X_C \leq 1 && \forall i \in V \\
& && 0 \leq X_C \leq 1 && \forall C \in \overline{\mathbb{C}}
\end{aligned}
$$

Note that every column is the incidence vector of the corresponding clique. By solving this linear program, we obtain the dual solution to each constraint. Such dual information provides us an improving direction to seek (*price out*) a new clique, i.e., a column denoting the vertices belonging to the clique.

To create the initial subset $\overline{\mathbb{C}}$, different primal heuristics can be performed. A simple initiation is to consider all single-vertex cliques to start with, i.e., letting $\overline{\mathbb{C}} = \{C_1, ..., C_n\}$ such that $C_i = \{v_i\}$ for all $i \in \mathbb{V}$. This simple heuristic method has showed satisfactory results comparing to some other heuristics we have tested. Especially, having all single-vertex cliques present in the column pool, guarantees an important property of the model, i.e., all inequalities are face-defining. This property is demonstrated in Theorem 5.

Theorem 5 *Let $Conv(S)$ denote the MWMC polytope of a given graph $G = (V, E)$. Then for any given $i = 1, ..., |V|$, if $\{v_i\} \in \overline{\mathbb{C}}$ then inequality $\sum_{C_j \in \overline{\mathbb{C}}} a_{i,C_j} X_{C_j} \leq 1$ induces a facet of $Conv(S)$.*

Proof. Let $S = \{x \in \mathbb{B}^{|\overline{\mathbb{C}}|} : \sum_{j=1}^{|\overline{\mathbb{C}}|} a_{i,C_j} x_j \leq 1 \ \forall i = 1, ..., |V|\}$ define the feasible space of the restricted **MWMCP-IP**, where $\overline{\mathbb{C}} = \{C_1, ..., C_{|\overline{\mathbb{C}}|}\}$ is the set of all columns in the model, C_j denotes the j-th column which corresponds to variable x_j and $a_{i,C_j} = 1$ if $v_i \in C_j$ and 0 otherwise, for $i = 1, ..., |V|$ and $j = 1, ..., |\overline{\mathbb{C}}|$. Moreover, for a given $i = 1, ..., |V|$, let $R_i = \{j : v_i \in C_j \ \forall j = 1, ..., |\overline{\mathbb{C}}|\}$ denote the set of all columns that include v_i (R_i represents the i-th row of the model), and let $I_i = \{v_i\}$ denote a unit column with i-th vertex only. Also, let $e_k \in \mathbb{R}^{|\overline{\mathbb{C}}|}$ denote the unit vector with k-th element one and all others zero.

To show that $dim(Conv(S)) = |\overline{\mathbb{C}}|$, we construct $|\overline{\mathbb{C}}| + 1$ affinely independent vectors as $x^0 = 0$, $x^k = e_k$ for $k = 1, ..., |\overline{\mathbb{C}}|$. Now, for any given $i = 1, ..., |V|$, let $F_i = \{x \in Conv(S) : \sum_{j \in R_i} x_j = 1\}$. We show if $I_i \in \overline{\mathbb{C}}$, we can construct $|\overline{\mathbb{C}}|$ affinely independent vectors in F_i and therefore F_i is a facet of $Conv(S)$. Without loss of generality, assume $C_i = I_i$ (then $I_i \in \overline{\mathbb{C}}$ is equivalent to $i \in R_i$). For every column C_k, $k = 1, ..., |\overline{\mathbb{C}}|$, let $x^k = e_k$ if $v_i \in C_k$ and $x^k = e_k + e_i$ otherwise. Except x^i, all these vectors have a different element equal to one. It follows that if $\sum_k \lambda_k x^k = 0$, then $\lambda_k = 0 \ \forall k$. Therefore $x^1, ..., x^{|\overline{\mathbb{C}}|}$ are affinely independent. Every x^k is a solution consisting of either one or two columns. This solution can be infeasible only if two intersecting columns C_i and C_k are selected. It is clearly not possible since if $v_i \notin C_k$ then $C_i \cap C_k = \{v_i\} \cap C_k = \varnothing$. Hence $x^k \in Conv(S)$ for all k. Similarly, every x^k satisfies the inequality in equality form since exactly one of k or i is in R_i. □

This is important because it shows that the problem formulation is tight and provides good bounds. Also, the structure of columns may not be altered since no lifting procedure is necessary to tighten the constraints.

3.1.2 Subproblem

In the subproblem (also called *the pricing problem*) we seek for a column (clique) with a positive reduced cost to enter the basis. Ideally, we want to find the most positive one, however, it is not necessary. We need to solve a MWCP that can be formulated as **QIP** with $\mathbb{K} = 1$ (constraints (2.2) become redundant), where vertex weights are penalized by the dual values from the master problem. Specifically, let π_i denote the dual value corresponding to the i-th constraint in **MP**. The weight of v_i in the subproblem is updated as $w(v_i) - \pi_i$ from the original vertex weight, see Figure 4 for an illustration. Then, in this updated graph, we solve the subproblem and identify clique C'. If $w_{C'}$ is positive, we expand $\overline{\mathbb{C}}$ to $\overline{\mathbb{C}} \cup \{C'\}$, i.e., include one more column in **MP**. If there is no clique with a positive weight, the master problem is solved to optimality and we stop the procedure.

Note that finding the optimal column is NP-hard as a MWCP needs to be solved. Considering that any column with a positive reduced cost can enter the basis, we use a two-phase approach. We call a quick approximation algorithm to find a clique (column) with a positive weight. In case this algorithm fails to find such a clique, we call an exact algorithm to either find a positive-weight clique or prove optimality. Let us also remark that adding non-optimal columns to **MP** may result in an increase in the number of iterations in the column generation. Therefor, there is a trade-off between the subproblem solution time and the number of times it is solved. Finally, it is worthwhile to consider adding more than one column to **MP** at each iteration. This may help to reduce the number of iterations but does increase the size of **MP**.

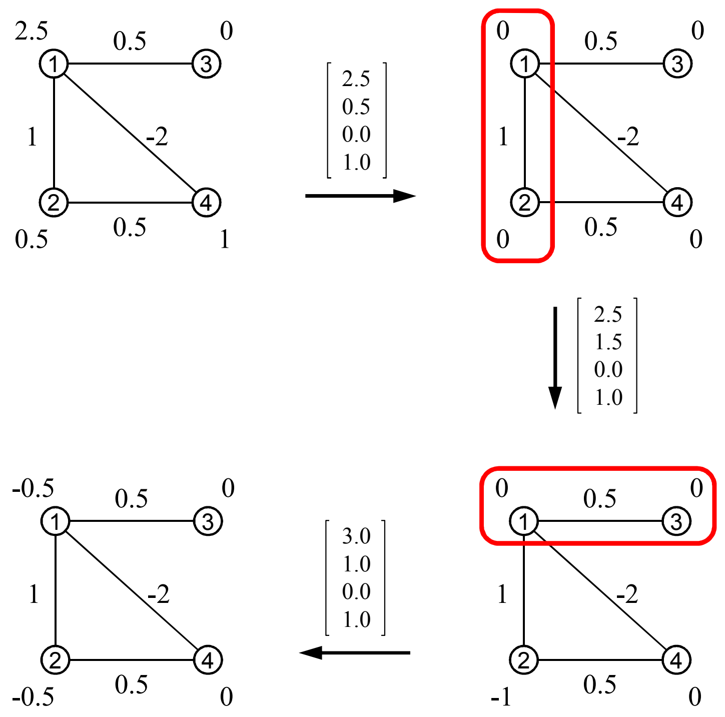


Figure 4: Illustration of the subproblem in CG in three iterations. Each time, the dual values (represented in vectors) are updated by solving the master problem and selected cliques are added to the master problem. We stop when there is no clique with a positive total weight.

3.1.2.1 Exact Algorithm for MWCP

The generalized case of the MWCP, in which both vertices and edges could have unrestricted real-valued weights, has not received much attention in the existing literature. Thus, we first develop an exact algorithm [11] for this general case. It can be considered as an extension of the algorithm in [45] for unweighted cases, which is also modified to solve vertex-weighted cases [46, 47]. Our algorithm adopts a branch-and-bound framework to obtain the exact solution. It maintains three sets: S, which denotes the current forming clique; U, which is the working set and stores the prospective members to S; and P, which stores the updated weights of vertices in S. Initially and at the root vertex of the branch-and-bound tree (level 0), $S_0 = \varnothing$, $U_0 = V$ and $P_0 = \{w(v_j)|v_j \in U_0\}$. A branch at level $d+1$ is created by selecting a vertex v_{new} from U_d. This new member is added to the forming clique S_{d+1} and removed from the previous working set. All vertices in U_d that are adjacent to v_{new} will form U_{d+1} and vertex weights are updated by shifting the respective weights of the edges connecting these vertices to v_{new}. Equivalently, $U_d = U_d \setminus \{v_{new}\}$, $S_{d+1} = S_d \cup \{v_{new}\}$, $U_{d+1} = U_d \cap N(v_{new})$ and $P_{d+1} = \{w(v_j) + \sum_{v_k \in S_{d+1}} w(e_{jk})|v_j \in U_{d+1}\}$ are updated, where $N(v_{new})$ is the set of all neighbors of v_{new}. The algorithm adopts a depth-first search for branch-and-bound: it goes in depth of the search tree first; and whenever $U_d = \varnothing$ it steps back to level $d-1$ to branch again. This procedure keeps track of the updated total weight of the forming clique while forming all possible cliques recursively. Eventually, the clique with the maximum weight is obtained when $U_0 = \varnothing$, i.e., the search tree is traversed completely. Note that the updated vertex weights in P are used for all weight calculations.

To enhance the search process, we embed a pruning strategy during the depth-first branch-and-bound. Specifically, assuming that we are in level d, before expanding the forming clique S_d, we calculate an upper bound, for the weight of the best clique in U_d. If this upper bound together with the weight of the forming clique w_{S_d} is lower than the weight of the current best clique, we do not need to explore U_d, thus, the current branch is pruned and the algorithm directly steps back to level $d-1$. To obtain strong bounds on updated weight estimation for fast pruning, we adopt a quick heuristic coloring technique [48] which assigns distinct colors to any two adjacent vertices. Given that all vertices in a clique will have all different colors, we can obtain an upper bound for the weight of the maximum clique in a working set U_d, by adding the maximum vertex weights in each color class with the summation of all edge weights induced by U_d. We further have adopted another strategy to improve efficiency: we have employed an ordered set of vertices V, given that the vertex ordering may only affect the efficiency of the algorithm due to the induced branching order but not the correctness of the solution. In our implementation, we order vertices increasingly by their degrees for more efficient pruning, because intuitively fewer branches on lower levels of the search tree may help to prune higher degree vertices in higher levels. Experimental observations have confirmed the effectiveness of this ordering strategy.

3.2 Top-K-Vertex Model

It is common nowadays that we collect high-dimensional measurements by including all candidate risk factors that may contribute to disease development, especially due to the advancement of high-throughput-omic profiling technologies [49]. By analyzing these high-dimensional data, we hope to identify critical risk factors as biomarkers to better understand the disease of interest. It is often the case that only a limited number of measured variables are associated with disease outcome. Motivated by identifying a small number of biomarkers that are most effective, we consider a variety of **MWMCP-IP** model, i.e., the *top-K* model. It finds a collection of cliques with maximum total weight such that they contain up to K vertices. Towards this direction, the top-K extension is modeled by adding a knapsack constraint:

$$\sum_{C \in \overline{\mathbb{C}}} |C| \, X_C \leq K$$

to **MWMCP-IP**. Parameter K can be set based on the percentage of vertices in G that we want to include for prediction. For example, if we take 25% of vertices in G, $K = \lceil 0.25 \, n \rceil$.

3.3 Heuristic Sequential Method

In case of very large-scale problems with more than tens of thousands of vertices, we observe that it may be challenging to solve **MWMCP-IP** or its top-K extension by our column generation method in a reasonable time. Clearly, the computational complexity of this problem necessitates the development of a faster method to obtain solutions with high quality within a shorter time frame. Hence, we develop a fast greedy heuristic method to handle such problems. We mention that this heuristic procedure can also be applied to solve the top-K-vertex variant with minor modifications.

In the greedy method, we sequentially solve the MWCP in G to find a single maximum weighted clique C, update G by removing the selected clique from G, and repeat those operations until there exists no clique with a positive weight. Obviously, the collection of obtained cliques is a feasible solution to **MWMCP-IP** as those cliques are disjoint. To further improve the solution quality, we design and implement a perturbation procedure that can avoid removing a clique that may hurt other potentially highly-weighted cliques. Specifically, before removing C from G, we check whether there exists a vertex $v_i \in C$ that could be removed from C to form a separate clique C' (consisting of v_i only, or v_i with one of its neighbors outside C) such that the total weight of the resulting two cliques ($C \setminus \{v_i\}$ and C') is greater than the weight of C. If such a vertex exists, we perturb C by removing v_i. The idea of perturbation is demonstrated in Figure 5 where all vertex weights are assumed to be zero. The solution to the MWCP for the network given in Figure 5, is the triangle in the middle with a total weight of 2.1. If we remove this clique, there would be no clique with a positive weight in the remaining network. Hence, this triangle clique would be a solution to MWMCP based on the greedy sequential procedure while it is not optimal as there are three pairwise cliques with a greater total weight of 6. Now with perturbation, we can perturb any vertex in the triangle clique C by removing it from C, forming another clique with weight of 2 by connecting it to its adjacent vertex and obtaining two cliques with 2.7 as their total weight, greater than $w_C = 2.1$. By repeating those steps, we will have three cliques as shown in Figure 5 (right), which actually consists the optimal solution for the MWMCP in the given network.

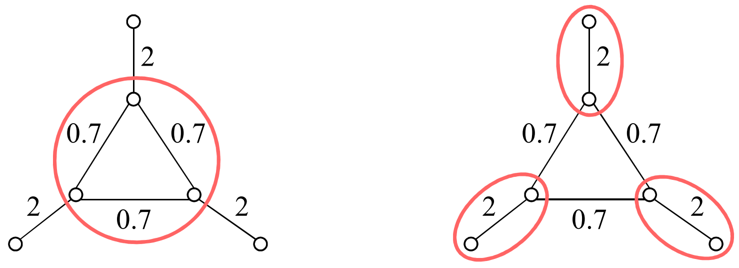


Figure 5: The solution to MWMCP without perturbation (left) may vary dramatically from the solution with perturbation (right).

3.4 Computational Results on Random Graphs

The performance of our **CG-IP** and greedy heuristic methods is evaluated on both Erdős-Rényi (ER) random graphs [50] and constructed interaction networks based on data collected from the DPT-1 study for T1D as well as the breast cancer microarray dataset. The results for real-world data sets are presented in chapter 5. The algorithms are implemented in C++ on a standard PC with a 2.2 GHz CPU and 2 GB of RAM. The state-of-the-art integer programming solver IBM ILOG CPLEX 12.1 [51, 52] is adopted to solve **MP** within the CG method, as well as the compact integer programming formulation (**LIP**). Results of the latter can be used to benchmark the developed **CG-IP** and the heuristic sequential methods.

To generate an ER random network with a given number of edges (or equivalently the density), a pair of disjoint vertices are randomly chosen and connected by adding an edge, this process is repeated until we get the desired number of edges. Vertex and edge weights are also assigned randomly. Vertex weights are random numbers between -1 and 1 following a uniform distribution, while edge weights are uniformly distributed between -0.5 and 0.5. For the top-K extensions, we set K such that the solution involves 25% of the vertices, i.e., $K = \lceil 0.25\, n \rceil$.

Experimental results on ER networks are presented in Table 1. It is obvious that solving the compact integer programming formulation by CPLEX is not practically feasible as it is extremely hard to solve instances with only tens of vertices. On the contrary, the **CG-IP** method can solve instances up to a thousand vertices in a reasonable time, which drastically improves our solution capability to larger instances. Indeed, for instances that their optimal solutions can be computed by CPLEX, **CG-IP** solves them to optimality with negligible computational time. For those instances that CPLEX fails to derive their optimal solutions within two hours, the **CG-IP** method always generates significantly better solutions. Hence, this observation confirms the capability of **CG-IP** method to obtain optimal or near optimal solutions. Actually, the experiments on random networks have shown empirically that for **CG-IP** method, the gap between the integral solution obtained after adding the integrality constraint, and the optimal integral solution is no more than 10%.

We further note for instances with thousands of vertices, that **CG-IP** method may take a long time to compute. Nevertheless, the heuristic sequential method is much less sensitive to the size of instances. It can quickly generate high-quality solutions for large-scale networks. In fact, it generally generates solutions of high quality and sometimes the difference is marginal, compared to those derived by **CG-IP** method. Hence, we believe that these two algorithms allow us to handle networks at different scales in a reasonable time with desired quality.

Finally, for the top-K model, both **CG-IP** and the sequential methods can complete within a shorter time. One explanation is that the cardinality constraint made the problems easier to solve by cutting out a significant amount of feasible solutions.

Table 1: Experimental results on random graphs

			CG-IP		CG-IP top-K		Seq		Seq top-K		CPLEX-IP		
n	%	$\lvert\overline{\mathbb{C}}\rvert$	obj	t	obj	t	obj	t	obj	t	obj	t	gap%
20	50	27	4.65	0.02	3.6	0.01	3.97	0.01	2.99	0.01	4.65	11.3	0
20	80	35	13.02	0.02	6.6	0.02	12.29	0.01	6.3	0.01	13.02	67.3	0
50	5	58	13.73	0.03	9.65	0.02	13.51	0.01	8.62	0.01	13.73	31.1	0
50	20	75	14.15	0.03	8.9	0.02	13.48	0.01	8.12	0.01	14.15	7200	12.8
50	30	91	22.61	0.05	13.22	0.04	21.12	0.01	11.26	0.01	22.03	7200	42.3
50	40	86	17.86	0.06	12.29	0.05	16.78	0.01	10.2	0.01	11.58	7200	193.3
50	60	129	29.81	0.34	15.16	0.26	26.92	0.01	14.33	0.01	19.59	7200	261.4
50	80	162	36.11	5.43	19.53	5.37	32.2	0.03	17.8	0.02	17.24	7200	453.0
100	40	295	173.1	7.88	64.47	1.42	169.6	0.03	65.36	0.01	-	-	-
200	40	525	120.0	66.8	56.75	44.8	108.4	0.59	51.97	0.37	-	-	-
500	25	1261	270.5	578.2	132.2	563.2	247.2	7.76	125.4	4.8	-	-	-
1000	10	2873	1535.8	1385.9	554.2	1026.1	1520.7	11.8	523.6	8.15	-	-	-
2000	5	5730	2355.0	4674.1	888.2	4630.1	2213.3	57.0	849.3	33.4	-	-	-

n—graph size;

%—density in percent;

$\lvert\overline{\mathbb{C}}\rvert$—number of generated columns;

obj—total weight of selected cliques;

t—computing time in seconds;

gap%—relative gap between the best integer solution and the best upper bound in percent;

CG-IP—column generation algorithm (**CG-IP**);

Seq— heuristic sequential algorithm;

CPLEX-IP— compact formulation (**LIP**) solved by solver;

CHAPTER 4: CUTTING PLANES AND BRANCHING METHODS

In this chapter, we describe our exact solution algorithm to solve a MWMCP to optimality. As mentioned earlier, in order to obtain the exact solution, the column generation can be combined with a branch-and-bound algorithm to search the feasible region thoroughly. This combination is called the *branch-and-price* algorithm [43, 44]. Yet to improve the performance of this method and the efficiency of the branch-and-bound procedure, one may tighten the linear programming formulation by introducing cutting planes [53] to the model. Namely, a *branch-price-and-cut* algorithm can be developed. We first explain how to generate efficient cutting planes, and then, we delineate how a branch-and-price-and-cut algorithm can be tailored for the MWMCP.

4.1 Cutting Planes

When the column generation algorithm terminates, the optimal solution to the **MP** is obtained. In case of integer programs, if this solution is fractional (non-integral and hence infeasible), it can be used as an upper bound (UB) for the optimal objective value. This bound is important in the implementation of the branch-and-price, as it helps to prune the branches of the search tree more effectively. In other words, it helps to decrease the size of the tree to explore by terminating the branches with the UBs lower then any known lower bound. Hence, the lower the obtained UB is, the more branches we can terminate. With regard to achieving tighter bounds, cutting planes can be added to the **MP**. A cutting plane (a *cut*) is a valid inequality for the integer formulation that is designated to cut off (make infeasible) the current fractional optimal solution from the feasible region, while remaining all the feasible integral solutions in the model. After adding a cutting plane, a new solution for this revised problem can be obtained. It may still be fractional, however, it provides a tighter upper bound. The idea of cutting planes to solve integer programming and mixed-integer programming problems was first proposed by Gomory [54] in 1950s. Despite their mathematical elegance, cutting planes were believed to be impractical and ineffective until 1990s when Balas et al. [55] showed them to be very effective if combined with branch-and-bound.

In order to generate cutting planes, special problems (often called *separation problems*) might be formulated and solved. As this separation step increases the solution time, we need to ensure that it will be compensated by a significant reduction in the tree search time. There are different general and special types of cutting planes that can be generated for a given integer programming problem. Namely, Gomory cuts, cover inequalities, clique inequalities, odd hole inequalities, etc. can be formulated (see [53, 56] for surveys).

As the MWMCP is a maximization problem containing constraints with only 0-1 coefficients and its right-hand-side is 1, it is in the form of a *weighted set packing* problem [53]. Any set packing problem has an interesting graph theoretic interpretation. Let $H(V_H, E_H)$ be a graph where each vertex represents a column in $\overline{\mathbb{C}}$, i.e., a clique in G. Two vertices are connected by an edge only if their corresponding columns have a common non-zero element in some row (equivalently, the corresponding cliques overlap). More formally, let

$V_H = \{c_1, ..., c_K\}$ where vertex c_j represents $C_j \in \overline{\mathbb{C}}$ for $j = 1, ..., K$ and $K = |\overline{\mathbb{C}}|$, and $E_V = \{e_{ij} : C_i \cap C_j \neq \varnothing \; \forall C_i, C_j \in \overline{\mathbb{C}}\}$. The weights are also mapped onto graph H by assigning clique weights to vertices while edge weights are zero, $w(c_j) = w_{C_j} \; \forall c_j \in V_H$, $w(e_{ij}) = 0 \; \forall e_{ij} \in E_H$. Graph H is called a *(column) intersection graph.* Then, observe that any integral feasible solution to **MP**, is an *independent set* in H. Because of this polynomial-time one-to-one reduction between these problems, by studying the independent set problem in H, valid inequalities can be generated for **MP**. Here we focus on the most common and suitable families of cutting planes for problems with a 0-1 coefficient matrix: clique inequalities and odd hole inequalities.

4.1.1 Clique Inequalities

A clique inequality is formed with a group of binary variables that at most one of which can be non-zero in any feasible solution. Note that any clique in graph H defines a valid clique inequality for **MP**. Fulkerson and Padberg [57, 58] showed that if such a clique is maximal, the corresponding inequality is facet-defining.

THEOREM 6 (Fulkerson and Padberg) *Let $Q \subseteq V_H$ be a clique in graph H. The inequality $\sum_{c_j \in Q} X_{C_j} \leq 1$ defines a facet for **MP** if and only if Q is maximal in H.*

Proof. See [57] and [58].

COROLLARY 1 *Let $Q^* \subseteq V_H$ be the maximum clique in graph H. The inequality $\sum_{c_j \in Q^*} X_{C_j} \leq 1$ defines a facet for **MP** if and only if Q^* is maximal in H.*

Let $X^* = (X^*_{C_1}, ..., X^*_{C_K})$ denote the non-integral optimal solution to **MP**. To cut out this solution by introducing a valid inequality, new weights are properly assigned to graph

H. Let new vertex weights be the optimal solution for the corresponding variable, i.e., let $w(c_j) = X^*_{C_j}$ for all $c_j \in V_H$. Then, a clique in H with a total weight greater than 1 represents a valid inequality that is violated by X^*. The most violated clique inequality is obtained by solving a maximum weighted clique problem in H. As this problem is NP-hard, one may use approximation algorithms here. If the weight of the maximum clique is less than or equal to 1, no clique inequality is violated by X^* (it satisfies all clique inequalities). Because the column pool $\overline{\mathbb{C}}$ grows within the column generation algorithm, the intersection graph H grows as well. To ensure that clique inequalities remain facet-defining, we need to ensure they are maximal in H. Therefore, whenever a new column is generated and added to $\overline{\mathbb{C}}$, we try to lift all the clique inequalities by adding the new variable corresponding to the new column. Moreover, we utilize the dual information associated with these inequalities to adjust the weights in the subproblem as follows. Let π_i denote the dual value corresponding to the i-th original constraint in **MP** and μ denote the dual value corresponding to a clique inequality $\sum_{c_j \in Q} X_{C_j} \leq 1$. In the subproblem, we adjust the vertex weight of vertices as $w(v_i) - \pi_i - \mu$ for $v_i \in C_j$ if $c_j \in Q$.

4.1.2 Other Inequalities

We have also tested odd hole inequalities and cover inequalities for **MP**. An odd hole S in H is an odd cycle that has no chords. Odd holes inequalities are in form:

$$\sum_{j \in S} X_j \leq \frac{|S| - 1}{2},$$

and are facet-defining and can be generated in polynomial time. However, their improvement on **MP** solution was limited and we did not use them in our algorithm. Cover inequalities are not directly applicable to **MP** since all original inequalities are generalized upper bound (GUB) constraints. Therefore they dominate any cover inequality. However, to find a cover inequality, a temporary constraint can be created by adding up some of the constraints in the model. We took the corresponding dual value for each constraint as its weight to create a weighted summation of all the original constraints. Then, we solve a separation problem to find the most violated cover inequality based on this temporary constraint. A cover inequality is in form:

$$\sum_{j \in S} X_j \leq |S| - 1,$$

where S is a cover for the temporary constraint. This method has showed negligible improvement in our experiments and hence is not used for MWMCP in this study. One potential reason these classes of inequalities do not perform well, could be the already tight formulation of **MP** where all original constraints are in the form of clique inequalities and define facets for its polytope.

4.2 Branch-Price-and-Cut

The idea of the branch-and-bound algorithm for finding the optimal solution of discrete problems was first proposed in 1960 [59, 60]. This algorithm systematically enumerates all possible solutions (typically within a search tree), while discarding a large subset of useless solutions. Two procedures are required for this method: *branching* and *bounding*. In the branching procedure, the problem is split into several smaller problems under different branches of the tree. These smaller problems are called children of the current (parent) node and typically are easier to solve. The tree size grows by adding new branches. In the bounding procedure, on the other hand, we try to prune the unfruitful branches of the tree. In general, there are three criteria under which the current node (and thereafter branch) will be pruned:

1. *pruning by bound* occurs when the solution at the current node is dominated by the best known feasible solution. The current node can be safely discarded as it does not have a better solution than the best known one. In a maximization problem, this is true when the upper bound of the current node is less than the global lower bound.

2. *pruning by infeasibility* occurs when the problem at the current node is not feasible. This is possible as child nodes are often more restricted than their parent nodes.

3. *pruning by optimality* occurs when the solution at the current node is feasible and can be used to update the best known feasible solution.

If the current node cannot be pruned, branching is performed and the search algorithm proceeds to another node in the tree. This procedure repeats until the whole tree is traversed and an optimal solution is obtained. To design an effective branch-price-and-cut framework that integrates column generation and cutting plane methods with a branch-and-bound algorithm, the structure of the search tree and the way it is traversed, are of crucial importance. In the following sections we describe how this framework is designed to solve the MWMCP to optimality.

4.2.1 Branching Criteria

On a given node of the tree, there are different ways to separate the feasible search space into smaller regions and create child nodes. However, it is known that the branch-and-bound algorithm is more likely to be effective if the feasible region is partitioned evenly [61]. In other words, an unbalanced tree is not desirable [62]. Moreover, since children problems are created by adding new restrictions, it is imperative to ensure that they are not difficult to solve.

For the MWMCP, due to the fact that there are many columns in **MP**, direct branching on columns may not be effective as it exhaustively enumerates branches that result in an unbalanced tree. Instead, we need an intelligent and special approach toward branching to identify more important feasible regions of the problem. As the number of vertices is drastically fewer than columns, by branching based on vertices, i.e., fixing them in or out of the optimal solution, an effective branching procedure can be designed. Moreover, observe that cliques can be naturally grouped by their common vertices. This is of our special interest because at most one clique from each group can be selected in the optimal solution. This provides a strong branching scheme for the problem.

To identify a good vertex for branching, we first need to define a measure to rank vertices. Several measures are defined and tested and the following measure with the best performance is chosen. Let $X^* = (X^*_{C_1}, ..., X^*_{C_K})$ be the current optimal solution to **MP** in a given node of the tree. Assume this branch is not pruned and branching needs to take place (obviously, X^* is not integral).

Let $\mathbb{F}_i = \{C_j \in \overline{\mathbb{C}} : v_i \in C_j \ \forall j = 1, ..., K\}$ be the set of columns in $\overline{\mathbb{C}}$ that include v_i, and $\mathbb{F}'_i = \{C_j \in \mathbb{F}_i : 0 < X^*_{C_j} < 1 \ \forall j = 1, ..., K\}$ be a subset of such columns that their corresponding variables are fractional. Clearly, we have $\mathbb{F}'_i \subseteq \mathbb{F}_i \subseteq \overline{\mathbb{C}}$ for all $i = 1, ..., n$. Now we take vertex i^* as the next vertex for branching, where:

$$i^* = \arg\min_i \Big\{ \sum_{C_j \in \mathbb{F}'_i} \Big| X^*_{C_j} - \frac{1}{|\mathbb{F}'_i|} \Big| \Big\}. \tag{4.1}$$

This intuitive procedure identifies the vertex that is most equally selected in multiple cliques. Now, note that there are only three possibilities for this specific vertex:

- (Type 1) v_{i^*} is not in the optimal solution.
- (Type 2) v_{i^*} is in the optimal solution but within a column (clique) that is not yet generated.
- (Type 3) v_{i^*} is in the optimal solution within a column (clique) that is already generated, and therefore is in $\mathbb{F}_i$.

Based on this observation, three types of branches can be created. Theorem 7 implies that the type 1 branch is not always necessary.

THEOREM 7 *Suppose $w(v_i) > 0$ for some $v_i \in V$, then any optimal solution of MWMCP contains v_i.*

Proof. Assume that the optimal solution does not contain v_i. By selecting v_i within a single-vertex clique and adding it to the solution, we can form a better feasible solution and it contradict with the optimality of the solution. $\square$

Consequently, when $w(v_{i^*}) \leq 0$, a single type 1 child is created by removing v_{i^*} from G and changing the right-hand side of the constraint associated with v_{i^*} to zero. These two adjustments make it impossible to select vertex v_{i^*} in the optimal solution under this branch. For types 2 and 3, we have considered two general branching rules: *non-binary branching* and *binary branching* that are described and compared in detail in the following sections.

4.2.1.1 Non-Binary Branching

In the *non-binary branching* method, we first create the type 2 branch. We do not allow to select vertex v_{i^*} within already-generated cliques by fixing all variables associated with $\mathbb{F}_{i^*}$ to zero. Therefore, we create a single type 2 child by adding constraint (4.2) to the current model.

$$\sum_{C_j \in \mathbb{F}_{i^*}} X_{C_j} \leq 0 \tag{4.2}$$

For type three branches, we create a child node for every clique in $\mathbb{F}_{i^*}$ to fix that clique selected in the optimal solution. As a result, $|\mathbb{F}_{i^*}|$ child nodes are created. To fix clique $C_j \in \mathbb{F}_{i^*}$, constraint (4.3) can be added to the model.

$$X_{C_j} \geq 1 \tag{4.3}$$

However, rather than explicitly adding such constraints to the model that may increase the problem complexity, clique C_j is removed from G and its weight w_{C_j} is fixed in the objective value. Also, all other cliques that overlap with C_j are eliminated because selected cliques cannot overlap in any optimal solution. In general, this elimination makes a significant change in the model and is done by changing the right-hand side of all the constraints associated with vertices in C_j to zero. This non-binary branching method is easy to implement, however, it may increase the tree size significantly. See Figure 6 for an illustration of this method.

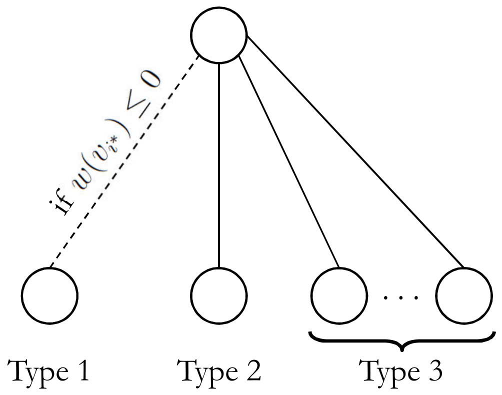


Figure 6: Illustration of the non-binary branching procedure

4.2.1.2 Binary Branching

Most commonly, on a given node of a branch-and-bound tree, the problem is divided into two subproblems. This two-way approach was first proposed in [63] and we term it *binary branching* as it results in a binary tree. To construct a balanced tree, we need to split $\mathbb{F}_{i^*}$ into two partitions as evenly as possible. We first rank cliques in $\mathbb{F}_{i^*}$ decreasingly based on the value of their corresponding elements in X^*. Then we create set $\mathbb{F}_1$ by taking every other clique in this ranking. Also, we let $\mathbb{F}_2 = \mathbb{F}_{i^*} \setminus \mathbb{F}_1$. Clearly $\mathbb{F}_1$ and $\mathbb{F}_2$ form a partition for $\mathbb{F}_{i^*}$. Note that the sizes of $\mathbb{F}_1$ and $\mathbb{F}_2$ are as close as possible, i.e., $\big||\mathbb{F}_1| - |\mathbb{F}_2|\big| \leq 1$. Also, given that v_{i^*} is the most equally selected vertex, this procedure provides a meaningful an effective partitioning of the cliques. Consequently, we create two child nodes by adding constraints (4.4) to one and constraint (4.5) to the other.

$$\sum_{C_j \in \mathbb{F}_1} X_{C_j} \leq 0 \tag{4.4}$$

$$\sum_{C_j \in \mathbb{F}_1} X_{C_j} \geq 1 \tag{4.5}$$

Observe that in the child node with constraint (4.5), v_{i^*} can be removed from G. Also, notice that constraint (4.4) contains the type 2 child where none of the existing columns in $\mathbb{F}_{i^*}$ are in the optimal solution. This branching method is shown in Figure 7.

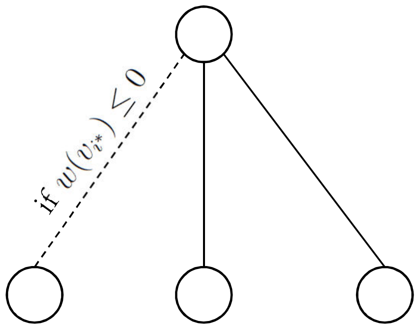


Figure 7: Illustration of the binary branching procedure

Constraining a group of variables in a branch is known as *constraint branching.* In contrast to non-binary branching, in this method, the tree grows more slowly, however, the added constraints are less restricting. Note that one can develop a hybrid branching rule, that is to use both binary and non-binary rules to split the search space. Figure 8 depicts such a hybrid branching rule. A decision needs to be made at any node of the tree for choosing the branching rule. The following sections provide details on the implementation, results, and discussions of the branch-price-and-cut algorithm.

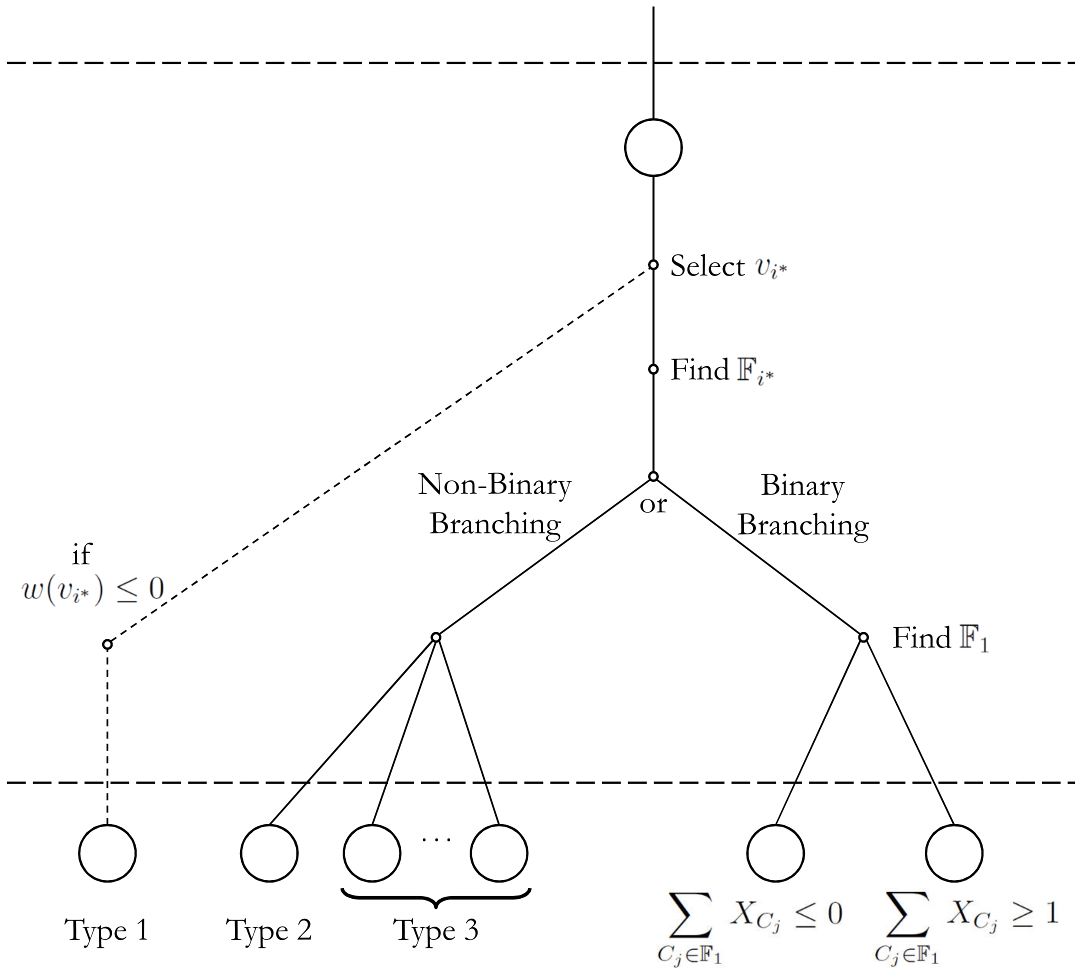


Figure 8: Illustration of a hybrid branching procedure

4.2.2 Implementation

On the root node of the tree, we start with **MP**, the linear relaxation of the restricted model. The column generation is performed and **MP** is solved to optimality. If the solution is not integral, cutting planes are generated and added to the model to reduce the global upper bound (UB). If the solution is still non-integral, the branching procedure is started and some child nodes are created. The current (root) node is considered visited and an unvisited node of the tree is chosen. At the chosen node, the corresponding **MP** with additional restrictions is solved and the UB is updated. Then the bounding procedure is performed to prune the current branch if possible. If bounding fails, the branching procedure is performed. The current node is marked as visited and the search process continues by choosing an unvisited node. This process continues to explore the tree until all nodes are visited and therefore an optimal solution is obtained. Figure 9 outlines the algorithm in a flowchart. The details for each step of this algorithm are described as follows.

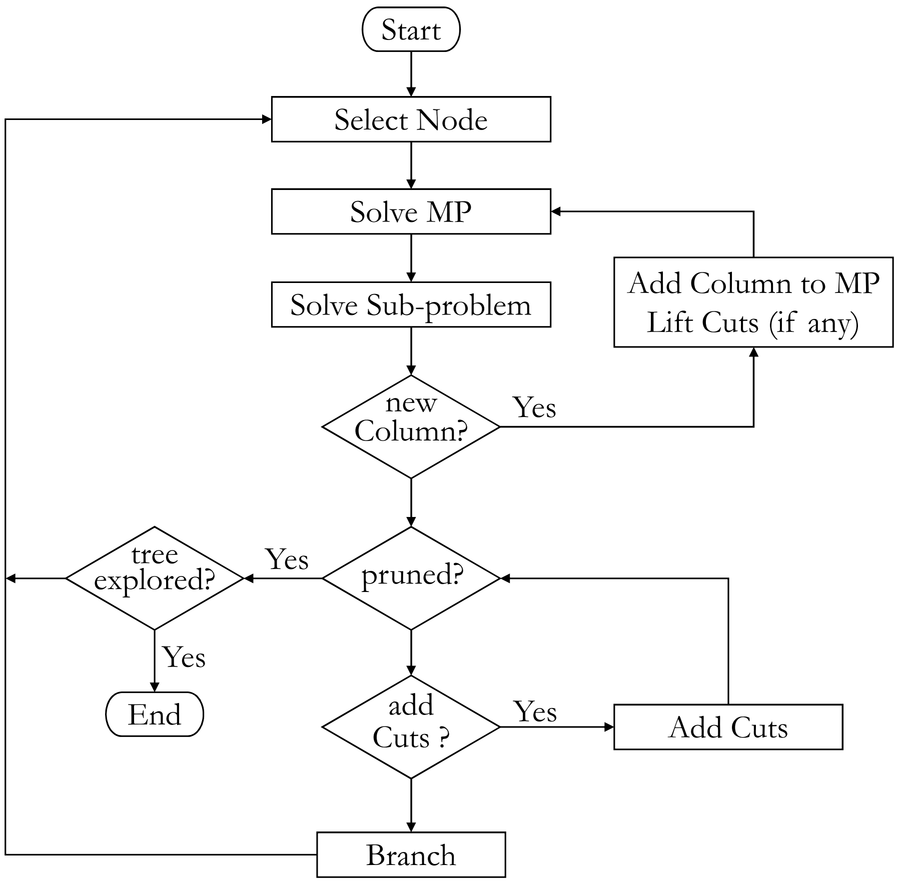


Figure 9: Outline of the Branch-price-and-cut algorithm

4.2.2.1 Lower and Upper Bounds

Initially, a trivial UB is used (e.g. $\sum_i \max\{0, w(v_i)\} + \sum_i \sum_{j>i} \max\{0, w(e_{ij})\}$), however, it is updated as the maximum of the upper bounds of all unvisited nodes in the tree. Note that the upper bound of an unvisited node is its parent's optimal value. The UB is updated when a branch is pruned or added to the tree. The global lower bound (LB) on the other hand, is the best feasible solution found so far. Considering that a good LB is critical in any branch-and-bound algorithm, feasible solutions are frequently found within the tree. A fast heuristic algorithm based on [64] is developed to quickly solve the MWISP in H. As mentioned earlier, this is equivalent to solving the restricted **MWMCP-IP**. Moreover, when this heuristic algorithm fails to obtain good feasible solutions, CPLEX solver is used to solve the restricted **MWMCP-IP** while the time limit is set to 20 seconds. The search for a feasible solution is performed only when the local upper bound (optimal value for current **MP**) is greater than a certain value. Intuitively, a node with greater local upper bound is more likely to contain a good feasible solution and our experiments have approved it. This value is set to $\text{LB} + 0.8(\text{UB} - \text{LB})$. The LB is updated if the new feasible solution provides a better lower bound. The new stronger LB is used to possibly prune the unpruned branches of the tree.

4.2.2.2 Priority of Nodes to Branch

The goal here, is to minimize the number of evaluated nodes in the tree. Hence, a *best-node first* strategy is taken to traverse the tree. That is to choose an unvisited node with the maximum local upper bound to be visited next. Further, because this maximum value is actually the UB, by visiting this node, the UB can be reduced. In case of multiple nodes with the same priority, the deepest node is selected.

4.2.2.3 Uniqueness of Columns

At every node of the tree, the column generation algorithm is carried out on a different **MP**. This leads to generating identical columns in different nodes. However, to avoid this redundancy in computation, we store all generated columns in a pool. Then, before solving the subproblem, this column pool is searched for a column with maximum reduced cost. If the reduced cost is positive, the column is introduced to **MP**, otherwise, the subproblem is solved. The implementation of the column pool is essential for an efficient algorithm to avoid regenerating the existing columns.

4.2.2.4 Merging Constraints

As we go deeper in the tree, more constraints are added to the problem and it becomes bigger and possibly more complicated. However, it is easy to see that constraints of type (4.2) can be merged together without changing the structure of the model. This is true for constraints of type (4.4) as well. Similarly, for constraints of type (4.5), Theorem 8 implies that they can also be added together without imposing any relaxation. As a result, at any node of the tree, at most two additional constraints are needed in the **MP**, one in ≤ 0 form and one in ≥ 1 form. However, instead of adding the ≤ 0 inequality, the variables in this inequality are dropped from the model. Note that adding the single constraint (≥ 1) to the model does not make it more difficult to solve.

THEOREM 8 *Let* $F_A = \{x \in Conv(S) : \sum_{j \in S_k} a_{i_k j} x_j \geq 1 \; \forall k = 1, ..., K\}$ *be a subspace of* $Conv(S)$ *restricted by* K *inequalities and* $F_B = \{x \in Conv(S) : \sum_{k=1}^{K} \sum_{j \in S_k} a_{i_k j} x_j \geq K\}$

be a subspace of $Conv(S)$ restricted by a single inequality which is the summation of all K inequalities in F_A where $S_k \subseteq \{1, ..., |\mathbb{C}|\}$ and $i_k \in \{1, ..., |V|\}$ for all $k = 1, ..., K$, then we have $F_A = F_B$.

Proof. Clearly $F_A \subseteq F_B$ since if x satisfies all K inequalities in F_A then it satisfies the single inequality in F_B as well. We need to show $F_B \subseteq F_A$. To prove by contradiction, assume $F_B \nsubseteq F_A$ and there exists $x \in F_B$ such that $x \notin F_A$. Then:

$$x \notin F_A \quad \Rightarrow \quad \exists\, k_0 \in \{1, ..., K\} : \sum_{j \in S_{k_0}} a_{i_{k_0} j} x_j < 1 \tag{4.6}$$

$$x \in F_B \quad \Rightarrow \quad x \in Conv(S) \quad \Rightarrow \sum_{j=1}^{|\mathbb{C}|} a_{i_k j} x_j \leq 1 \;\forall k \;\Rightarrow \sum_{j \in S_k} a_{i_k j} x_j \leq 1 \;\forall k \tag{4.7}$$

$$(4.6),\ (4.7) \quad \Rightarrow \quad \sum_{k=1}^{K} \sum_{j \in S_k} a_{i_k j} x_j < K \quad \Rightarrow x \notin F_B$$

Thus, $F_B \subseteq F_A$ and therefore $F_A = F_B$. □

4.2.2.5 Solving Master Problem and Subproblem

In the branch-price-and-cut algorithm, the master problem needs to be solved numerous times. It is therefore necessary to have a simple formulation in order to solve the model quickly. In the case of MWMCP, the (**MP**) is fairly easy to solve. The sophisticated commercial solver, IBM ILOG CPLEX is used for this purpose. It is important specially due to its built-in reoptimization algorithm, that starts from the optimal base to reoptimize the model after adding a new column. This feature significantly reduces the total number of iterations of the simplex algorithm for individual **MP**s [62].

A great advantage of our proposed branching scheme, is that the structure of the subproblem remains intact. It is to solve the MWCP in a given graph. At the root node of the tree, it is graph G, and as we go in depth, more of its vertices are eliminated. It makes the subproblem easier to solve. To solve the subproblem, we use the algorithm described in section 3.1.2. Note that by solving the subproblem, we only introduce one column (usually the optimal column) to the **MP**, rather than multiple columns.

4.3 Computational Results on Random Graphs

Our exact branch-price-and-cut algorithm is evaluated on variants of randomly generated graphs. The experiments presented in this chapter have been performed on a workstation with two 3.46 GHz processors and 24GB of memory. ER random graphs with four different sizes and densities are considered. Also, both non-binary and binary branching rules are tested on every graph. For each size and density, 10 graphs are randomly generated and their average results are presented in Table 2. We also have limited the solution time to 2 hours. The results demonstrate that our algorithm is capable to solve instances of reasonable sizes and densities. Note that none of the branching rules is consistently better than the other. Evidently, the non-binary branching rule performs better in dense graphs while the binary branching is more effective in sparse ones. It also became evident that the linear relaxation of the model is quite tight and provides very good upper bounds. On the other hand, note that the lower bounds we obtained for the problems are of high quality as well. It is interesting to notice that even with such small gaps between the UB and LB, many nodes in the search tree is created and evaluated to obtain the optimal solution. This observation verifies that the MWMCP is a very difficult problem to tackle. Finally, the number of improving columns found in the column pool shows how valuable it is, considering that the average computation time to find a column in the pool is much shorter than of solving the subproblem.

Table 2: Experimental results on random graphs

n	%	Rule	t	UB	obj	LB	#Cuts	#Cols	#Cols Reused	#Nodes	gap%
100	50	B	540.8	102.579	101.394	100.216	124.5	1313.4	27604.5	3136.6	0
100	50	N	233.6	102.579	101.394	100.259	124.3	1604.6	31464.0	2070.5	0
200	25	B	328.4	168.405	167.51	166.672	33.2	1095.5	10752.4	1962.8	0
200	25	N	332.5	168.405	167.51	166.672	33.2	1433.3	30973.1	3690.0	0
500	10	B	1936.4	357.854	357.18	356.407	10.1	1946.0	21210.5	3877.7	0
500	10	N	2812.5	357.854	357.18	356.407	10.1	2428.4	80388.8	11032.4	0.0005
1000	5	B	3470.6	658.096	657.747	657.316	4.8	3066.3	18812.8	2869.1	0.0022
1000	5	N	3889.1	658.096	657.734	657.316	4.8	3822.2	32915.2	4566.0	0.0092

n—graph size;

%—density in percent;

Rule—branching rule (B:binary, N:non-binary);

t— computing time in seconds;

UB—initial upper bound;

obj—optimal total weight of selected cliques;

LB—initial lower bound;

#Cuts—number of cuts in **MP**;

#Cols—number of all columns generated;

#Cols Reused—number of columns added from column pool;

#Nodes—number of all nodes in tree;

gap%—relative gap between LB and UB in percent;

CHAPTER 5: APPLICATIONS OF BIOMEDICAL DATA SETS [1]

In this chapter we discuss in details how our developed methods can be applied on real world problems to identify effective biomarkers. The network construction phase and the implementation details are described. Then the results for two biomedical data sets are presented and compared with the results from the conventional method.

5.1 Construction of Interaction Networks

In order to evaluate our network-based biomarker identification methods, we first construct a weighted network for all included candidate risk factors in the analysis. We define the node weight $w(v_i) = -\log(p_i)$, in which p_i is the coefficient p-value for β_1 by fitting a logistic regression model:

$$\log(\frac{g}{1-g}) = \beta_0 + \beta_1 v_i,$$

with the corresponding candidate factor v_i. Here, g denotes the posterior probability of a certain disease outcome y given measurement of v_i: $g = Pr(y|v_i)$; and $\log(g/(1-g))$ is the link function of the logistic regression model. Similarly, we can define the edge weight $w(e_{ij})$ between candidate factors v_i and v_j as $w(e_{ij}) = -\log(p_{ij})$ based on the coefficient p-value p_{ij} for β_3 in the logistic regression model integrating with the interaction term between v_i and v_j:

$$\log(\frac{g}{1-g}) = \beta_0 + \beta_1 v_i + \beta_2 v_j + \beta_3 v_i v_j,$$

[1]This chapter was partially published in [1]. Permission is included in Appendix A.

in which $g = Pr(y|v_i, v_j)$. In the constructed network, to focus on strong interactions, low-weighted edges (i.e., $w(e_{ij}) \leq Threshold$) can be removed. It also makes the problem easier to solve for large scales.

5.2 Network-based Biomarker Identification and Performance Evaluation

We implement our network-based methods and compare them with a traditional forward feature selection algorithm [4] that only considers the discriminating power of individual candidate biomarkers. Such a comparison demonstrates that our network-based biomarker identification approach can achieve better prediction accuracy due to the integration of interactive effects among candidate factors. We first apply both CG [2] and heuristic sequential algorithms to solve MWMCP on the interaction networks. As a result, we obtain multiple cliques which capture both individual and interactive effects among candidate factors. To evaluate and compare the performance of biomarker identification, we adopt a Support Vector Machine (SVM) with polynomial kernel of degree two as our classifier. The choice of kernels in SVM is to ensure that interactions among biomarkers are considered for classification while controlling model complexity at the same time. In our experiments, we have adopted the LIBSVM [65] implementation of SVM in Matlab. For both network-based and individual biomarker identification, the same forward feature selection procedure has been applied to select the best group of biomarkers with the highest classification accuracy.

As there are several steps during our classifier training stage, we perform the following "embedded" cross-validation to appropriately estimate the classification performance for both network-based and individual biomarker identification. In this cross-validation proce-

[2]In this chapter, CG refers to the **CG-IP** method (See chapter 3).

dure, we first randomly divide the dataset into five folds. Then, four folds of data are used as the *training set* to select biomarkers and build the classifier; and the remaining fold is used as the *testing set* to estimate the classification accuracy of selected biomarkers. This procedure is repeated five times for each fold as the *testing set*.

For each *training set*, we perform a feature selection algorithm. For individual feature selection, we rank candidate biomarkers based on a descending order of their individual discriminating power measured by the coefficient p-values from fitted regression models. For network-based methods, we rank the identified cliques based on a descending order of their corresponding total weights. Then, the same forward selection procedure is implemented to sequentially add individual biomarkers or cliques, in the ranked order to the set of selected biomarkers. If adding a new individual factor or clique improves the estimated classification accuracy, it will be selected in the final biomarker set. Otherwise, we move on to the next ranked factor or clique to iterate the same procedure.

The classification accuracy for feature selection is estimated by traditional three-fold cross-validation using the *training set*, in which two folds of the *training set* are used to train the SVM classifier and one fold is used for testing. The procedure is repeated three times to estimate the accuracy based on the currently selected biomarkers. Finally, the *testing set* is used to estimate the testing classification accuracy based on the selected final biomarker set. The overall evaluation procedure is repeated 100 times and the average accuracy is reported as the estimated classification accuracy.

5.2.1 DPT-1

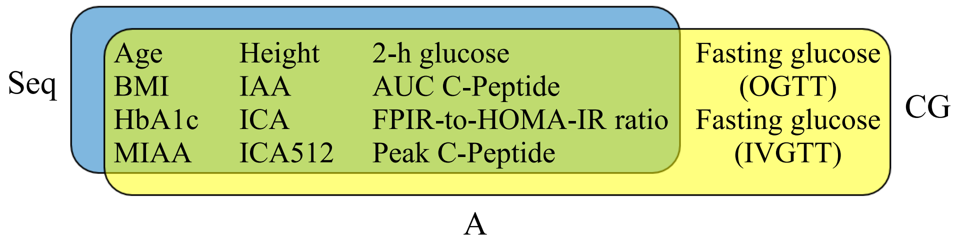
We first test and compare the performance of different biomarker identification methods using a relatively small dataset studying Type 1 Diabetes (T1D). We study baseline characteristics including immunologic and metabolic indices with respect to T1D development in subjects with high risk using the collected data from the Diabetes Prevention Trial-Type 1 (DPT-1) study. In DPT-1, there are $3,483$ subjects positive for islet cell autoantibodies (ICA) among the total $103,391$ screened subjects. The projected five-year risk of diabetes for these subjects is estimated according to genetic susceptibility; age; immunologic indices from different autoantibodies, including ICA, IAA (insulin autoantibodies), GAD (glutamic acid decarboxylase), ICA512 (insulinoma-associated protein 2), and MIAA (micro-insulin autoantibodies); and metabolic indices, including 2-hour glucose, fasting glucose, glycated hemoglobin (HbA1c), fasting insulin, first-phase insulin response (FPIR), C-peptide measurements in the fasting state, and then 30, 60, 90, and 120 minutes after oral glucose. As in the previous univariate analysis [66], we compute Homeostasis model assessment of insulin resistance (HOMA-IR = fasting insulin $(m\mu/l)$ $\times$ fasting glucose $(mmol/l)$ /22.5), FPIR-to-HOMA-IR ratio, peak C-peptide as the maximum point of all measurements, and AUC (area under the curve) C-peptide using the trapezoid rule based on the given metabolic indices. Furthermore, we include age and Body Mass Index (BMI) in our network-based multivariate analysis as important confounding factors.

In this study, we focus on DPT-1 study subjects staged to the "high risk" group [67, 68, 66], which contains 339 subjects in total. Within this high risk group, 169 subjects received parenteral insulin supplement and we refer this set as the "Treatment" group while the other 170 subjects received placebo as a control group, which is referred as "Placebo". We are interested in identifying the most predictive group of biomarkers from the previously described candidates to predict the outcome y—the development of T1D at the end of DPT-1 study. Within both the "Treatment" and "Placebo" groups, there are 80 subjects diagnosed with T1D at the end of the study with $y = 1$. We have tested both the individual and network-based methods using both groups of data. We have computed the classification accuracies from different biomarker identification methods based on the previous cross-validation procedure. These estimated classification accuracies are reported in Table 3.

Comparing both column generation and sequential network-based methods with the individual-based feature selection, the reported results clearly show that both network-based biomarker identification methods are performing significantly better (with p-values $< 1e-6$) than the traditional individual-based feature selection. These results verify our expectation that network-based biomarker selection methods are able to find biomarkers with higher predictive accuracies by integrating interactive effects among biomarkers.

Table 3: Classification accuracies of methods based on T1D datasets. (Ind—Individual predictive power based feature selection; CG—Column Generation algorithm; Seq— heuristic sequential algorithm)

Dataset	Ind	Seq	CG
T1D Treatment	62.39	65.69	65.60
T1D Placebo	59.74	62.57	62.45



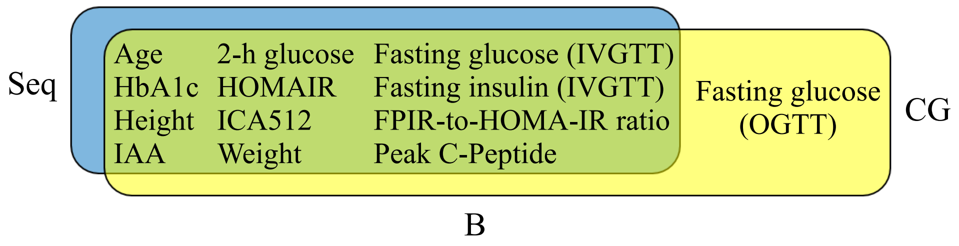


Figure 10: Features that appeared in at least 70% of 500 feature selections done in experiments for Treatment (A) and Placebo (B) groups in T1D dataset.

In the previous cross-validation experiments, we have implemented 500 (100 repeated five-fold "embedded" cross validation) feature selections, each time based on a randomly sampled training subset. As a result, we have found 500 different subsets of final biomarker sets. To ensure that we have obtained reliable results without overfitting, we provide in Figures 10 (A) and (B) the lists of frequently selected final biomarkers that appeared in at least 70% of 500 different feature selection runs from different biomarker identification methods for the "Treatment" and "Placebo" groups respectively. When comparing selected features by our CG and sequential methods, we find that the additional features selected by CG are Fasting glucose levels from either OGTT or IVGTT. As discussed in the recent position statement of the American Diabetes Association (ADA) [69], these indices are main diagnosis criteria for clinical diabetes. We also have tested the performance of those final biomarkers based on 100 repeated five-fold cross validation (without feature selection) and their corresponding estimated testing accuracies are given in Table 4. The results further verify that within these commonly conjectured important biomarkers for T1D [67, 68, 66], network-based biomarker selection can provide better biomarkers with higher predictive power for T1D, which may lead to better prognosis models.

Table 4: Estimated testing classification accuracies of final biomarkers based on 100 repeated five-fold cross validation for different methods. (Ind—Individual predictive power based feature selection; CG—Column Generation algorithm; Seq— heuristic sequential algorithm)

Dataset	Ind	Seq	CG
T1D Treatment	62.1	68.21	68.02
T1D Placebo	57.51	65.36	65.18
Breast Cancer	74.56	75.26	76.43

5.2.2 Breast Cancer

We further evaluate our proposed network-based biomarker identification methods on a large genomic dataset for breast cancer metastasis study [70], which is referred as the "USA" dataset as in the literature [9, 10]. The USA dataset contains the gene expression profiles for 22,283 genes of 286 breast cancer patients from which 107 are detected with metastasis and the remaining 179 are metastasis-free. An extremely large amount of time needs to be spent, especially for CG method, if we apply the previous embedded cross validation procedure with 500 repeats of network construction and clique finding with such a large number of candidate genes. In order to perform a comparison between CG and the other methods in a reasonable time, we have adopted a preprocessing step to filter out a large number of genes. To obtain a smaller set of important genes as potential biomarkers, they are ranked by their individual predictive power, again based on the coefficient p-value in logistic regression using all the samples. Then, the top 1% of genes (222 genes in total) with the highest individual predictive power are kept for performance comparison for the USA dataset. Table 5 provides the estimated classification accuracies for all the methods. The results clearly show that our network-based biomarker identification methods which incorporate the interactions among candidate genes, select markers with significantly (p-values $< 1e-7$) better classification accuracy than the traditional feature selection based on only individual power.

Table 5: Classification accuracies of methods based on breast cancer dataset. (Ind— Individual predictive power based feature selection; CG—Column Generation algorithm; Seq— heuristic sequential algorithm)

Dataset	Ind	Seq	Seq top-K	CG	CG top-K
Breast Cancer	65.54	70.89	68.65	71.02	67.82

To check the consistency of selected genes among 500 repeated runs in cross validation, we draw a frequency curve for selected genes. Each gene would appear from 0 to 500 times among 500 final biomarker sets of genes. We compute the ratio of the number of genes that have appeared at least f times ($1 \leq f \leq 500$) over the total number of genes that are selected at least once. As illustrated in Figure 11, the ratio of repeatedly selected genes for our network-based methods are consistently higher than the corresponding ratio for individual-based feature selection method. This demonstrates that the selected genes by network-based methods are more stable towards different training sets.

Finally, we provide in Figure 12 the list of frequently selected genes as final biomarkers that have been selected in at least 30% of 500 repeated runs from different biomarker identification methods. According to a recent study [71], protein RNF19A has been identified

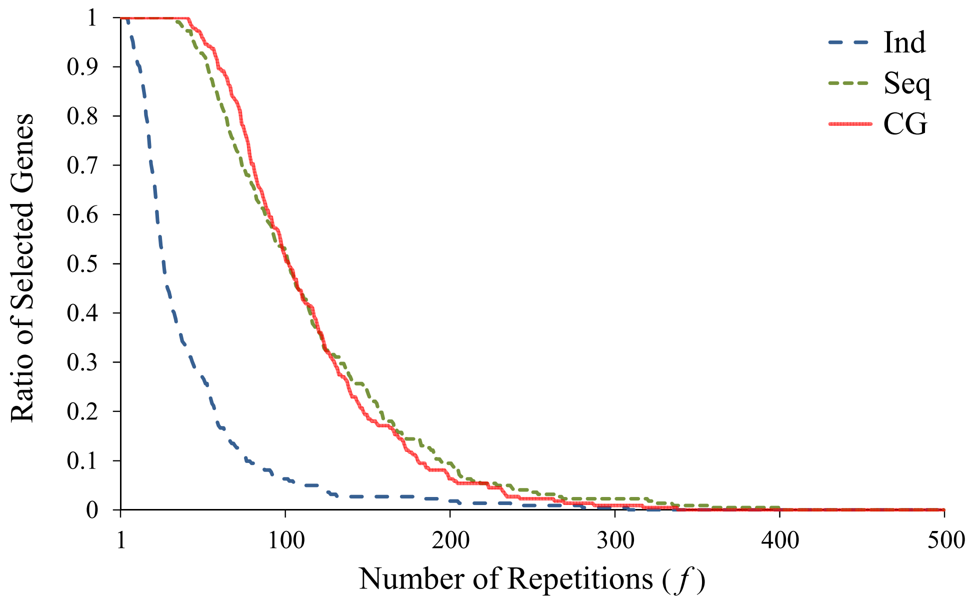


Figure 11: Stability curves for breast cancer (USA) dataset.

as a differentially expressed marker for breast cancer. The authors in [71] have shown that RNF19A is one of functional molecules in cancer-associated fibroblasts. Based on our proposed feature selection results shown in Figure 12, the CG method has successfully selected this marker, which demonstrates its promising potential for accurate identification of discriminating biomarkers. We also have tested the performance of those final biomarkers based on 100 repeated five-fold cross validation (without feature selection) and the corresponding estimated testing accuracies are given in Table 4. Although the difference of the obtained accuracies by different feature selection methods is relatively small, the improvement by our new methods considering interactions among biomarkers in the synergy network is consistent and indeed statistically significant with p-values smaller than 0.01 based on two sample t-tests.

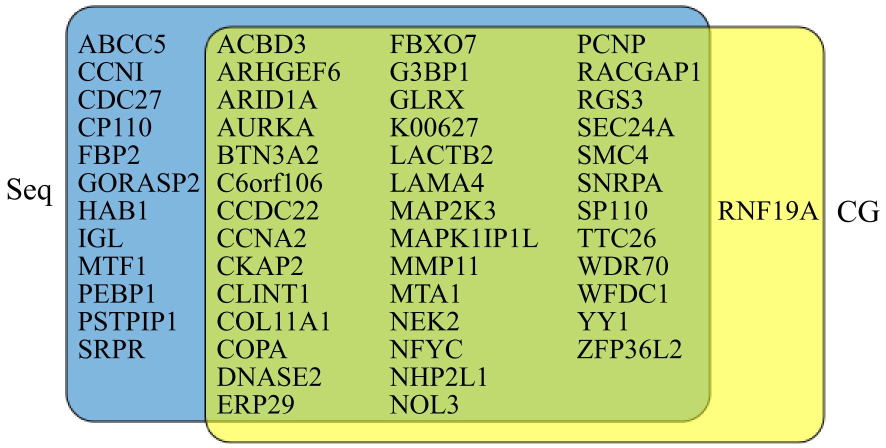


Figure 12: Features that appeared in at least 30% of 500 feature selections done in experiments for breast cancer dataset.

CHAPTER 6: FUTURE WORK

6.1 Biomedical Insight

The proposed network-based biomarker identification by solving MWMCP has been shown to be able to help identify important biomarkers for more accurate prognosis and diagnosis for complex diseases, such as cancer and diabetes. Due to the underlying complex disease mechanisms, there are still many aspects that we may need to mathematically model them faithfully for effective biomarker identification, such as the non-linearity of potential interaction (rather than simply adding up the individual and interactive power) as well as the possible noise and uncertainty from the available data. Moreover, it is indeed imperative to further investigate the responsibility of the identified biomarkers for the initiation, progression and onset of diseases, as well as the establishment of a methodology to systematically utilize such insights in patient stratification and classification.

6.2 Applications of the MWMCP

The MWMCP introduced in this study has several potential applications. From the problem definition, we can see that in any network that multiple disjoint and complete sub-networks are of interest, this problem can be applied to identify such sub-networks. For example, in network interdiction problem where critical elements of a network need to be identified. Also, in power grids, where the contingency analysis and network reliability are essential, the MWMCP can be utilized. The vulnerability of other infrastructure networks in general, is of vital national interests.

6.3 Variant Network Structures

The MWMCP can be formulated in generalized networks to solve broader problems. For instance, one may think of networks with multiple layers, that is a network with edges of different types. Such a network could be utilized to model different parameters within same elements (vertices). A two-layer network formulation has applications in biomarker identification, drug discovery, etc.

6.4 Variant Sub-network Structures

The solution approach in this research can be viewed as a general framework to simultaneously find multiple sub-networks with some specific property in a given network. A wide range of problems then can be formulated and solved within this framework. The sub-network of interest is application-specific and could have any common or special graph theoretic structure. One may consider a tree, a k-plex, a hub-spoke or a path as the sub-network and develop similar solution algorithms as in the MWMCP. Note that to solve some of these new problems, the only necessary modification is to substitute the sub-problem algorithm of the MWMCP while the rest of the model remains unchanged. This property alone clearly shows the flexibility of our solution framework. Especially, recall that the sub-problem does not need to be solved to optimality and as long as new improving columns are generated, the algorithm works.

6.5 Robust and Stochastic Extensions

Another direction to expand the MWMCP is for the case that we deal with special uncertainties. More sophisticated formulations are needed to model such uncertainties. For example, consider a data set of poor quality given as an input of a network-based optimization model. Robust optimization can be involved to expand the problem and provide solution techniques in this case. Also, to deal with probabilistic data and network entries, stochastic version of the model can be considered and studied.

6.6 Implementation

Several methodologies and techniques are considered within our solution algorithms to solve the MWMCP. From the branch-price-and-cut algorithm to the fast heuristic algorithms, there are many opportunities to introduce new techniques and methods to improve the algorithms. For instance, stronger cutting planes may be formulated for the problem or more effective branching procedures may be developed. Also, different solution methods for the sub-problem may be considered to quickly generate good columns.

CHAPTER 7: CONCLUDING REMARKS

Motivated by the biomarker identification problem in computational biology, we represent the biomarkers in a network and formulate their individual and interaction effects in this network simultaneously. Vertex and edge weights are used to represent these effects respectively. To identify highly discriminating and interacting markers, multiple complete sub-networks, i.e., cliques with maximum total weights are found and studied. To find such maximum weighted cliques, we introduce a novel optimization problem called the *Maximum Weighted Multiple Clique Problem (MWMCP).* After proving its NP-hardness, its integer programming formulations are presented and compared within two different approaches. Namely, compact formulations and column generation (CG) reformulation are discussed. Benefiting from the advantages of CG reformulation, a solution framework is developed to obtain the exact solutions for the MWMCP by combining CG and the branch-and-bound algorithm that is known as branch-and-price. By formulating cutting planes to strengthen the linear relaxation of the model, the branch-price-and-cut algorithm is formed to solve the problem efficiently. Also, fast heuristic algorithms are developed to solve problem of larger scales. The preliminary results on a set of random networks show that the developed algorithms can handle instances of different scales with high quality solutions. Finally, the algorithms are utilized on interaction networks created based on real-world datasets for Type 1 Diabetes and breast cancer to obtain important biomarkers. The results show that the interaction network framework and our MWMCP solution approach are capable to identify more accurate biomarkers in comparison to individual-based feature ranking.

REFERENCES

[1] S. J. Sajjadi, X. Qian, B. Zeng, and A. A. Adl, "Network-based methods to identify highly discriminating subsets of biomarkers," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on.* DOI: 10.1109/TCBB.2014.2325014, in press.

[2] D. Thomas, "Gene–environment-wide association studies: Emerging approaches," *Nature Reviews Genetics*, vol. 11, no. 4, pp. 259–272, 2010.

[3] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, 1999.

[4] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[5] Z. Q. Tang, L. Y. Han, H. H. Lin, J. Cui, J. Jia, B. C. Low, B. W. Li, and Y. Z. Chen, "Derivation of stable microarray cancer-differentiating signatures using consensus scoring of multiple random sampling and gene-ranking consistency evaluation," *Cancer Research*, vol. 67, no. 20, pp. 9996–10003, 2007.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] S. Ma and J. Huang, "Penalized feature selection and classification in bioinformatics," *Briefings in Bioinformatics*, vol. 9, no. 5, pp. 392–403, 2008.

[8] J. Watkinson, X. Wang, T. Zheng, and D. Anastassiou, "Identification of gene interactions associated with disease from gene expression data using synergy networks," *BMC Systems Biology*, vol. 2, no. 1, p. 10, 2008.

[9] H.-Y. Chuang, E. Lee, Y.-T. Liu, D. Lee, and T. Ideker, "Network-based classification of breast cancer metastasis," *Molecular Systems Biology*, vol. 3, no. 1, 2007.

[10] J. Su, B.-J. Yoon, and E. R. Dougherty, "Accurate and reliable cancer classification based on probabilistic inference of pathway activity," *PLoS ONE*, vol. 4, no. 12, p. e8161, 2009.

[11] S. J. Sajjadi, A. A. Adl, B. Zeng, and X. Qian, "Finding the most discriminating sets of biomarkers by maximum weighted clique," in *Proceedings of the 6th INFORMS Workshop on Data Mining and Health Informatics*, vol. 1500, 2011.

[12] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold, and L. Hood, "Integrated genomic and proteomic analyses of a systematically perturbed metabolic network," *Science*, vol. 292, no. 5518, pp. 929–934, 2001.

[13] T. Ideker, O. Ozier, B. Schwikowski, and A. F. Siegel, "Discovering regulatory and signalling circuits in molecular interaction networks," *Bioinformatics*, vol. 18, no. suppl 1, pp. S233–S240, 2002.

[14] D. di Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. Eastwood, A. P. Wojtovich, S. J. Elliott, S. E. Schaus, and J. J. Collins, "Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks," *Nature biotechnology*, vol. 23, no. 3, pp. 377–383, 2005.

[15] A. Ergün, C. A. Lawrence, M. A. Kohanski, T. A. Brennan, and J. J. Collins, "A network biology approach to prostate cancer," *Molecular systems biology*, vol. 3, no. 1, 2007.

[16] W. F. Symmans, J. Liu, D. M. Knowles, and G. Inghirami, "Breast cancer heterogeneity: Evaluation of clonality in primary and metastatic lesions," *Human Pathology*, vol. 26, no. 2, pp. 210–216, 1995.

[17] L. Ein-Dor, I. Kela, G. Getz, D. Givol, and E. Domany, "Outcome signature genes in breast cancer: Is there a unique set?," *Bioinformatics*, vol. 21, no. 2, pp. 171–178, 2005.

[18] A. H. Bild, G. Yao, J. T. Chang, Q. Wang, A. Potti, D. Chasse, M.-B. Joshi, D. Harpole, J. M. Lancaster, A. Berchuck, J. A. Olson, J. R. Marks, H. K. Dressman, M. West, and J. R. Nevins, "Oncogenic pathway signatures in human cancers as a guide to targeted therapies," *Nature*, vol. 439, no. 7074, pp. 353–357, 2005.

[19] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," 1979.

[20] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, "The maximum clique problem," in *Handbook of Combinatorial Optimization*, pp. 1–74, Springer, 1999.

[21] J. Desrosiers and M. E. Lübbecke, "Branch-price-and-cut algorithms," *Wiley encyclopedia of operations research and management science*, 2011.

[22] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[23] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*, vol. 5. Springer, 2005.

[24] H. Saigo, T. Uno, and K. Tsuda, “Mining complex genotypic features for predicting HIV-1 drug resistance,” *Bioinformatics*, vol. 23, no. 18, pp. 2455–2462, 2007.

[25] Y. Ying, K. Huang, and C. Campbell, “Enhanced protein fold recognition through a novel data integration approach,” *BMC Bioinformatics*, vol. 10, no. 1, p. 267, 2009.

[26] J. Håstad, “Clique is hard to approximate within $n^{1-\varepsilon}$,” *Acta Mathematica*, vol. 182, no. 1, pp. 105–142, 1999.

[27] R. D. Luce and A. D. Perry, “A method of matrix analysis of group structure,” *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949.

[28] F. Harary and I. C. Ross, “A procedure for clique detection using the group matrix,” *Sociometry*, pp. 205–215, 1957.

[29] F. S. Kuhl, G. M. Crippen, and D. K. Friesen, “A combinatorial algorithm for calculating ligand binding,” *Journal of Computational Chemistry*, vol. 5, no. 1, pp. 24–34, 1984.

[30] R. Horaud and T. Skordas, “Stereo correspondence through feature grouping and maximal cliques,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 11, pp. 1168–1180, 1989.

[31] N. Sloane, “Unsolved problems in graph theory arising from the study of codes,” *Graph Theory Notes of New York*, vol. 18, pp. 11–20, 1989.

[32] P. Berman and A. Pelc, “Distributed probabilistic fault diagnosis for multiprocessor systems,” in *20th International Symposium on Fault-Tolerant Computing (FTCS-20)*, pp. 340–346, IEEE, 1990.

[33] K. Corrádi and S. Szabó, “A combinatorial approach for Keller’s conjecture,” *Periodica Mathematica Hungarica*, vol. 21, no. 2, pp. 95–100, 1990.

[34] A. Ben-Dor, R. Shamir, and Z. Yakhini, “Clustering gene expression patterns,” *Journal of Computational Biology*, vol. 6, no. 3-4, pp. 281–297, 1999.

[35] S. Zhang, X. Ning, and X.-S. Zhang, “Identification of functional modules in a ppi network by clique percolation clustering,” *Computational Biology and Chemistry*, vol. 30, no. 6, pp. 445–451, 2006.

[36] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek, “Cfinder: locating cliques and overlapping modules in biological networks,” *Bioinformatics*, vol. 22, no. 8, pp. 1021–1023, 2006.

[37] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 1972.

[38] A. Mehrotra and M. A. Trick, “Cliques and clustering: A combinatorial approach,” *Operations Research Letters*, vol. 22, no. 1, pp. 1–12, 1998.

[39] L. Ford and D. R. Fulkerson, *Flows in networks*, vol. 3. Princeton Princeton University Press, 1962.

[40] P. C. Gilmore and R. E. Gomory, “A linear programming approach to the cutting-stock problem,” *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

[41] P. Gilmore and R. Gomory, “A linear programming approach to the cutting-stock problem-part II,” *Operations Research*, vol. 11, no. 6, pp. 863–888, 1963.

[42] F. Vanderbeck, “Implementing mixed integer column generation,” in *Column Generation*, pp. 331–358, Springer, 2005.

[43] F. Vanderbeck and L. A. Wolsey, “An exact algorithm for IP column generation,” *Operations Research Letters*, vol. 19, no. 4, pp. 151–159, 1996.

[44] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, “Branch-and-price: Column generation for solving huge integer programs,” *Operations Research*, vol. 46, no. 3, pp. 316–329, 1998.

[45] R. Carraghan and P. M. Pardalos, “An exact algorithm for the maximum clique problem,” *Operations Research Letters*, vol. 9, no. 6, pp. 375–382, 1990.

[46] P. R. Östergård, “A new algorithm for the maximum-weight clique problem,” *Nordic Journal of Computing*, vol. 8, no. 4, pp. 424–436, 2001.

[47] D. Kumlander, “A new exact algorithm for the maximum-weight clique problem based on a heuristic vertex-coloring and a backtrack search,” in *Proc. 5th Intl Conf. on Modelling, Computation and Optimization in Information Systems and Management Sciences*, pp. 202–208, Citeseer, 2004.

[48] D. Brélaz, “New methods to color the vertices of a graph,” *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[49] R. Chen, G. I. Mias, J. Li-Pook-Than, L. Jiang, H. Y. Lam, R. Chen, E. Miriami, K. J. Karczewski, M. Hariharan, F. E. Dewey, Y. Cheng, M. J. Clark, H. Im, L. Habegger, S. Balasubramanian, M. O’Huallachain, J. T. Dudley, S. Hillenmeyer, R. Haraksingh, D. Sharon, G. Euskirchen, P. Lacroute, K. Bettinger, A. P. Boyle, M. Kasowski, F. Grubert, , S. Seki, M. Garcia, M. Whirl-Carrillo, M. Gallardo, M. A. Blasco, P. L. Greenberg, P. Snyder, T. E. Klein, R. B. Altman, A. J. Butte, E. A. Ashley, M. Gerstein, K. C. Nadeau, H. Tang, and M. Snyder, “Personal omics profiling reveals dynamic molecular and medical phenotypes,” *Cell*, vol. 148, no. 6, pp. 1293–1307, 2012.

[50] P. Erdős and A. Rényi, “On the evolution of random graphs,” *Publ. Math. Inst. Hungar. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[51] I. I. CPLEX, "V12. 1: Users manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[52] `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[53] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey, "Cutting planes in integer and mixed integer programming," *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 397–446, 2002.

[54] R. E. Gomory *et al.*, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, no. 5, pp. 275–278, 1958.

[55] E. Balas, S. Ceria, and G. Cornuéjols, "A lift-and-project cutting plane algorithm for mixed 0–1 programs," *Mathematical programming*, vol. 58, no. 1-3, pp. 295–324, 1993.

[56] G. Cornuéjols, "Valid inequalities for mixed integer linear programs," *Mathematical Programming*, vol. 112, no. 1, pp. 3–44, 2008.

[57] D. R. Fulkerson, "Blocking and anti-blocking pairs of polyhedra," *Mathematical programming*, vol. 1, no. 1, pp. 168–194, 1971.

[58] M. W. Padberg, "On the facial structure of set packing polyhedra," *Mathematical programming*, vol. 5, no. 1, pp. 199–215, 1973.

[59] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[60] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[61] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser, "Solving binary cutting stock problems by column generation and branch-and-bound," *Computational optimization and applications*, vol. 3, no. 2, pp. 111–130, 1994.

[62] L. A. Wolsey, *Integer programming*, vol. 42. Wiley New York, 1998.

[63] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," *The Computer Journal*, vol. 8, no. 3, pp. 250–255, 1965.

[64] S. Sakai, M. Togasaki, and K. Yamazaki, "A note on greedy algorithms for the maximum weighted independent set problem," *Discrete Applied Mathematics*, vol. 126, no. 2, pp. 313–322, 2003.

[65] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[66] P. Xu, Y. Wu, Y. Zhu, G. Dagne, G. Johnson, D. Cuthbertson, J. P. Krischer, J. M. Sosenko, J. S. Skyler, and on behalf of the Diabetes Prevention TrialType 1 (DPT-1) Study Group, "Prognostic performance of metabolic indexes in predicting onset of type 1 diabetes," *Diabetes Care*, vol. 33, no. 12, pp. 2508–2513, 2010.

[67] J. P. Krischer, D. D. Cuthbertson, L. Yu, T. Orban, N. Maclaren, R. Jackson, W. E. Winter, D. A. Schatz, J. P. Palmer, G. S. Eisenbarth, and the Diabetes Prevention TrialType 1 Study Group, "Screening strategies for the identification of multiple antibody-positive relatives of individuals with type 1 diabetes," *Journal of Clinical Endocrinology & Metabolism*, vol. 88, no. 1, pp. 103–108, 2003.

[68] J. M. Sosenko, J. P. Palmer, C. J. Greenbaum, J. Mahon, C. Cowie, J. P. Krischer, H. P. Chase, N. H. White, B. Buckingham, K. C. Herold, D. Cuthbertson, J. S. Skyler, and the Diabetes Prevention Trial-Type 1 Study Group, "Increasing the accuracy of oral glucose tolerance testing and extending its application to individuals with normal glucose tolerance for the prediction of type 1 diabetes the diabetes prevention trial-type 1," *Diabetes Care*, vol. 30, no. 1, pp. 38–42, 2007.

[69] American Diabetes Association, "Diagnosis and classification of diabetes mellitus," *Diabetes Care*, vol. 36, no. Suppl 1, pp. S67–S74, 2013.

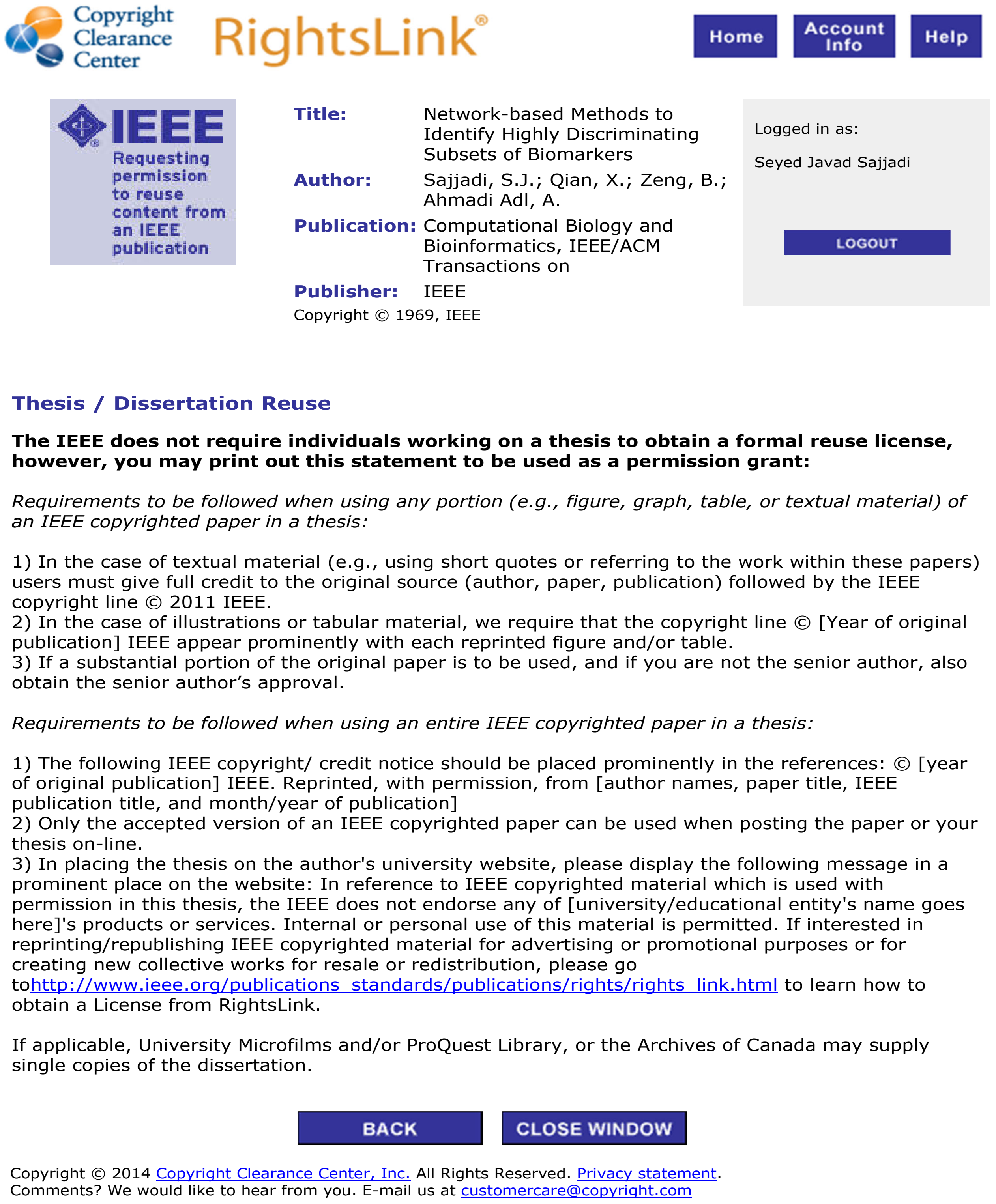
[70] Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu, T. Jatkoe, E. M. Berns, D. Atkins, and J. A. Foekens, "Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer," *The Lancet*, vol. 365, no. 9460, pp. 671–679, 2005.

[71] B. Bozóky, A. Savchenko, P. Csermely, T. Korcsmáros, Z. Dúl, F. Pontén, L. Székely, and G. Klein, "Novel signatures of cancer-associated fibroblasts," *International Journal of Cancer*, 2013.

APPENDICES

Appendix A Copyright Permissions

A.1 Permission from IEEE to Reuse [1].


Copyright Clearance Center
RightsLink®
Home
Account Info
Help
IEEE
Requesting permission to reuse content from an IEEE publication
LOGOUT

Title: Network-based Methods to Identify Highly Discriminating Subsets of Biomarkers
Author: Sajjadi, S.J.; Qian, X.; Zeng, B.; Ahmadi Adl, A.
Publication: Computational Biology and Bioinformatics, IEEE/ACM Transactions on
Publisher: IEEE
Copyright © 1969, IEEE

Logged in as:
Seyed Javad Sajjadi

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go tohttp://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

ABOUT THE AUTHOR

Seyed Javad Sajjadi received the B.S. degree in statistics from the University of Tehran and the M.S. degree in industrial engineering from Sharif University of Technology. He earned his Ph.D. degree in industrial engineering at the University of South Florida in 2014. His research interests include Operations Research Applications in Data Mining and Data Analysis, Large-Scale Network Optimization in Computational Biology and Bioinformatics and Integer Programming and Combinatorial Optimization.