"Finding submatrices of maximal sum : applications to the analysis of gene expression data"

Branders, Vincent

ABSTRACT

An important aspect of cancer research is the development of better tools to understand underlying cellular processes. These tools are crucial as they help clinicians choose the best treatment strategy for each patient or develop new treatment strategies. Gene expression data is typically represented as a large matrix of gene expression levels across various samples. The study of such data is a valuable tool to improve the understanding of biological processes. Therefore, grouping genes according to their expression under certain conditions or group conditions based on the expression of some genes is a frequent objective of gene expression analysis. Biclustering, also known as co-clustering, is one of the most common approaches for such a task. It identifies specific subsets of rows and columns that jointly form homogeneous entries. However, relevant gene/sample combinations can be missed when they lack the assumed homogeneity of expression values. It is a growing concern as cancer is a heterogeneous disease. Thus, there is an ongoing trend for the study of cellular processes by combining heterogeneous data sources. This thesis is centered around the development of approaches that find patterns of high values in large data matrices. It encompasses the definition of optimization problems and algorithmic solutions to find such patterns. The relevance of these contributions is evaluated through implementation and comparative experiments on biological data.

CITE THIS VERSION

Branders, Vincent. *Finding submatrices of maximal sum : applications to the analysis of gene expression data.* Prom. : Dupont, Pierre ; Schaus, Pierre http://hdl.handle.net/2078.1/250200

Finding submatrices of maximal sum: applications to the analysis of gene expression data

Vincent Branders

Thesis submitted in partial fulfillment of the requirements for the Degree of Doctor in Engineering and Technology Sciences

August 2021

ICTEAM
Louvain School of Engineering
Université catholique de Louvain
Louvain-la-Neuve
Belgium

Finding submatrices of maximal sum: applications to the analysis of gene expression data

by Vincent Branders

ICTEAM
Université catholique de Louvain
Place Sainte-Barbe, 2
1348 Louvain-la-Neuve
Belgium

Preamble

An important aspect of cancer research is the development of better tools to understand underlying cellular processes. These tools are crucial as they help clinicians choose the best treatment strategy for each patient or develop new treatment strategies.

Gene expression data is typically represented as a large matrix of gene expression levels across various samples. The study of such data is a valuable tool to improve the understanding of biological processes. Grouping genes according to their expression under certain conditions or group conditions based on the expression of some genes is a frequent objective of gene expression analysis. Biclustering, also known as co-clustering, is one of the most common approaches for such a task. It identifies specific subsets of rows and columns that jointly form homogeneous entries. However, relevant gene/sample combinations can be missed when they lack the assumed homogeneity of expression values. It is a growing concern as cancer is a heterogeneous disease. Moreover, there is an ongoing trend for the study of cellular processes by combining heterogeneous data sources.

This thesis is centered around the development of approaches that find patterns of high values in large data matrices. It encompasses the definition of optimization problems and algorithmic solutions to find such patterns. The relevance of these contributions is evaluated through implementation and comparative experiments on biological data. The main focus of this thesis is not methodological but rather consists in solving difficult optimization problems to guide biologists. In particular, the main contributions of this thesis are the following.

- *Proposition of new optimization problems to model biological problems.*

  Three problems related to the extraction of patterns in biological data are considered. It includes the search for a single pattern of globally high value, the search for multiple patterns that do not overlap, and the search for multiple patterns that may overlap.

- *Proposition of algorithmic solutions to address the new optimization problems.*

It encompasses the contributions aiming at providing simple and efficient algorithms to solve the combinatorial problems presented earlier. These contributions include the study of upper and lower bounds, the definition of filtering and dominance rules, variable heuristics, and large neighborhood searches.

- *Implementation of the algorithmic contributions.*

  It encompasses the contributions aiming to test, evaluate, and share the implementations to solve the optimization problems that are addressed. These implementations include several *constraint programming approaches*, *new constraints* implementation, a *column generation* approach, *integer programming* approaches, and an *R package*.

- *Illustration of the relevance of the previous contributions to biological data.*

  It encompasses the evaluation of the relevance of the mathematical objectives and the time performance of our implementations, as compared to existing algorithms. It includes a rigorous statistical validation protocol to assess and prove the quality of the patterns found in gene expression data.

The thesis is organized as follows.

Chapter 1, *The maximal sum submatrix problem*, presents the problem of finding a subset of rows and columns that maximizes the sum of covered entries from an input matrix. This chapter can be seen as the intersection of the work presented in all publications done during the thesis.

Chapter 2, *Mining a submatrix of maximal sum*, presents a simple yet efficient constraint programming approach to solve the *maximal sum submatrix* problem. This chapter corresponds to the work published in [BSD17; BSD18].

Then, Chapter 3, *Mining K disjoint submatrices of maximal sum*, and Chapter 4, *Mining k overlapping submatrices of maximal sum*, presents two extensions to the *maximal sum submatrix* problem that concern the identification of k submatrices that cannot and may overlap, respectively. Chapter 3 corresponds to the work published in [Bra+19b]. Chapter 4 corresponds to the work published in [Der+19]. My main contributions, as a second author, of that last publication consist of the bounds and dominance rules propositions, the experiments, and the evaluation of the results.

Subsequently, Chapter 5, *Mining submatrices of maximal sum in gene expression data*, presents an application of the *maximal sum submatrix* problem to biological data. This chapter corresponds to the work published in [BSD19].

Chapter 6, *Mining submatrices of maximal sum quicker*, presents an improved solution to the *maximal sum submatrix* problem to find submatrices in difficult instances. This chapter is an ongoing work not yet published.

Finally, Chapter 7, *Conclusions and perspectives*, summarises the main conclusions and results obtained and investigates some directions for future works.

Bibliographic notes

Journal Publication

1. V. Branders, P. Schaus, and P. Dupont. "Identifying gene-specific subgroups: an alternative to biclustering". In: *BMC bioinformatics* 20.1 (Dec. 2019), pp. 1–13. DOI: 10.1186/s12859-019-3289-0.

Book Chapter

1. V. Branders, P. Schaus, and P. Dupont. "Combinatorial Optimization Algorithms to Mine a Sub-Matrix of Maximal Sum". In: *New Frontiers in Mining Complex Patterns.* Ed. by A. Appice, C. Loglisci, G. Manco, E. Masciari, and Z. Ras. Vol. 10785. Lecture Notes in Artificial Intelligence. Cham: Springer International Publishing, Jan. 2018, pp. 65–79. DOI: 10.1007/978-3-319-78680-3_5.

Conference Papers

1. V. Branders, G. Derval, P. Schaus, and P. Dupont. "Mining a maximum weighted set of disjoint submatrices". In: *Discovery Science.* Springer International Publishing, 2019, pp. 18–28. DOI: 10.1007/978-3-030-33778-0_2.

2. G. Derval, V. Branders, P. Dupont, and P. Schaus. "The Maximum Weighted Submatrix Coverage Problem: A CP Approach". In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Ed. by W.-J. van Hoeve. Springer International Publishing, 2019, pp. 258–274. DOI: 10.1007/978-3-030-19212-9_17.

Workshop Paper

1. V. Branders, P. Schaus, and P. Dupont. "Mining a Sub-Matrix of Maximal Sum". In: *Proceedings of the 6th International Workshop on New Frontiers in Mining Complex Patterns in conjunction with ECML-PKDD 2017* (Sept. 25, 2017). arXiv: 1709.08461.

Miscellaneous Contribution

1. V. Branders. *vbranders/mssm: Release as used in "Identifying gene-specific subgroups: an alternative to biclustering"*. Version v0.1-beta. July 2020. DOI: 10.5281/zenodo.4607363.

Acknowledgments

Le travail présenté dans ce document est le résultat d'une cascade de discussions, d'interactions et d'efforts. Par ces lignes, j'aimerai faire part de ma gratitude à toutes les personnes qui m'ont permis d'aboutir ce travail.

Mes premiers remerciements vont à mes deux promoteurs. Je souhaite remercier Pierre (Dupont) qui m'a fait confiance en m'accueillant dans le département. Pendant ces années de recherche et d'enseignement, tu m'as appris énormément de choses et m'as fait surmonter des difficultés (telles que le respect de deadlines...) que je n'imaginais pas pouvoir surmonter. Je souhaite à tous les doctorants d'avoir un promoteur qui soit aussi clair et qui trouve des mots aussi justes que toi. Je souhaite remercier Pierre (Schaus) d'avoir accepté de prendre part à mon travail et de l'avoir soutenu. Mon travail a clairement été influencé par tes fréquents passages au bureau. Tu as toujours su trouver la petite pique pour me faire tout donner dans le but te prouver que tu avais tort. Cela m'a beaucoup aidé lorsque j'étais confronté à des difficultés techniques plus difficiles qu'à l'accoutumée. Concrètement, vous avez été le duo idéal : Pierre Schaus pour me pousser à me dépasser et à avancer dans ma thèse et Pierre Dupont pour me faire prendre du recul sur mon travail et m'éviter de m'y perdre.

J'aimerai également remercier les membres du comité d'accompagnement de ma thèse et les membres du jury de thèse qui ont pris le temps d'étudier mon travail et de me fournir des avis et des conseils pour améliorer la qualité générale de celui-ci.

Je souhaite remercier toutes les personnes qui sont passées ou qui sont encore au département. L'ambiance y est toujours particulièrement agréable et ça a été un plaisir de passer mes journées avec vous. Je remercie en particulier l'équipe PAT pour tous les (nombreux) services qui m'ont été rendus, les académiques, qui m'ont encadré dans mes enseignements, et les doctorants, qui ont partagé leur quotidien. Je remercie particulièrement Alexander, Victor et Roberto pour les discussions de bureau, Guillaume pour les sous-matrices de somme maximale, Xavier pour le cours d'introduction à l'algorithmique, Hélène pour la rédaction de thèse, Benoît pour les kickers.

J'ai également une pensée pour tous les étudiants rencontrés. Ils m'ont tant apporté, et j'espère que la réciproque est vraie.

Je remercie ma famille et mes amis. Même si l'apport technique est moindre, ce travail, même en partie, n'aurait jamais vu le jour sans vous. J'ai une pensée particulière pour mes plus grandes influences : Papa, Maman, Anne, Samuel et Pierre. Merci, Pierre, de m'avoir appris à travailler moins, mais mieux !

J'aimerai terminer en beauté avec Pascale. Je te remercie du plus profond du cœur pour tout ce que tu as pu m'apporter. Tu connais mes pensées et je ne les étalerais pas ici, car "les autres, on s'en fout!"

Contents

The maximal sum submatrix problem | 1

The problem of finding a maximal sum submatrix is a core contribution of this thesis because it served as a starting point for other contributions. This chapter, dedicated to that problem, is organized as follows. The problem is formally stated and later interpreted in the context of gene expression analysis. Extensions to the identification of multiple submatrices are subsequently approached, and an overview of the related works is given. Constraint programming, which has a major role in all contributions of this thesis, is finally presented.

1.1 Definition

Consider a matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ with m rows and n columns. The matrix is associated with a set of rows $\mathrm{R} = \{\mathrm{r}_1, \ldots, \mathrm{r}_m\}$ and a set of columns $\mathrm{C} = \{\mathrm{c}_1, \ldots, \mathrm{c}_n\}$. Entry $\boldsymbol{M}_{i,j}$ is the real value in row i and column j of the matrix, with $i \in \mathrm{R}$ and $j \in \mathrm{C}$. FIGURE 1.1a presents an illustrative instance matrix $\boldsymbol{M}$.

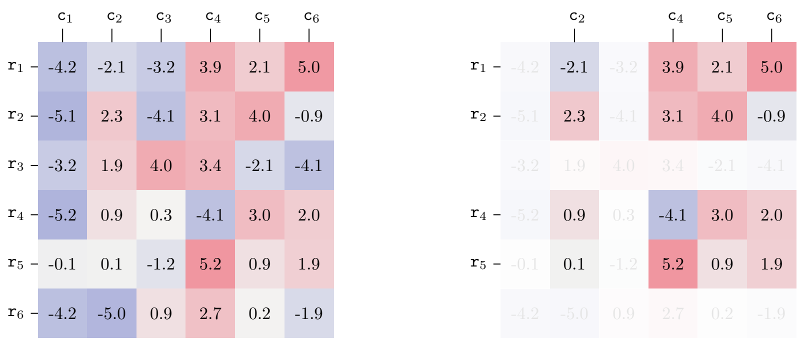
A submatrix is a subset of rows and columns from an input matrix. Formally, the submatrix (I, J) denotes all the elements $\boldsymbol{M}_{i,j}$ of the matrix $\boldsymbol{M}$ such that $i \in \mathrm{I}$ and $j \in \mathrm{J}$ with $\mathrm{I} \subseteq \mathrm{R}$ and $\mathrm{J} \subseteq \mathrm{C}$. We say that (I, J) *covers* all these elements. Alternatively, we say that these elements are covered by (I, J).

The *weight* of a submatrix refers to the sum of entries it covers. The weight of a submatrix (I, J) is maximal if any other subset of rows and columns is of smaller weight. That submatrix (I, J) is of *maximal sum*.

FIGURE 1.1b presents the maximal sum submatrix associated with the input matrix $\boldsymbol{M}$ in FIGURE 1.1a. The weight of $(\{\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_4, \mathrm{r}_5\}, \{\mathrm{c}_2, \mathrm{c}_4, \mathrm{c}_5, \mathrm{c}_6\})$ is 27.3. There is no other combination of rows and columns with larger weight.

The problem of finding such a maximal sum submatrix (MSS) is formally stated in DEFINITION 1.

DEFINITION 1. **The Maximal Sum Submatrix Problem**
Let $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ be a matrix of m rows and n columns. Let θ be a minimum threshold to consider entries as potentially relevant or informative. Let $\mathrm{R} =$



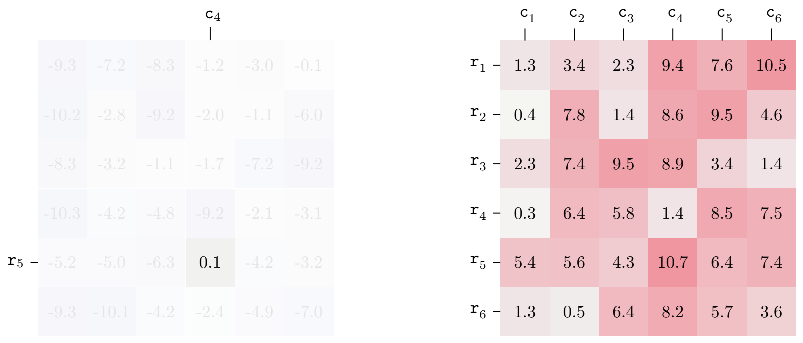
(a) Instance matrix M. **(b) Submatrix of weight $w = 27.3$.**

Figure 1.1: An example matrix and one of the possible submatrices associated. Low and high values are indicated in blue and red, respectively.

$\{r_1, \ldots, r_m\}$ and $C = \{c_1, \ldots, c_m\}$ be the set of rows and columns of $\mathbf{M}$, respectively. The maximal sum submatrix problem consists in finding the submatrix (I^*, J^*), with $I^* \subseteq R$ and $J^* \subseteq C$, such that:

$$(I^*, J^*) = \underset{\substack{I \subseteq R \\ J \subseteq C}}{\operatorname{argmax}} \sum_{\substack{i \in I \\ j \in J}} \left(\mathbf{M}_{i,j} - \theta \right) \quad . \tag{1.1}$$

The maximization term in Equation (1.1) rewards matrix entries with values above the threshold θ. Matrix entries with values below θ are penalized. Changing this threshold allows controlling the size of the optimal submatrix. Figure 1.2 illustrates solutions for two definitions of θ and Figure 1.1b presents the solution with $\theta = 0$. One only considers matrices with values both above and below the threshold. The optimal solution is trivially



(a) Threshold 5.1. **(b) Threshold −5.5.**

Figure 1.2: The submatrices of maximal sum obtained from different thresholds applied on the instance matrix in Figure 1.1a.

defined otherwise.

In the remaining of this work, we consider the simpler formulation of the problem defined below:

$$(\mathrm{I}^*, \mathrm{J}^*) = \underset{\substack{\mathrm{I} \subseteq \mathrm{R} \\ \mathrm{J} \subseteq \mathrm{C}}}{\operatorname{argmax}} \sum_{\substack{i \in \mathrm{I} \\ j \in \mathrm{J}}} \boldsymbol{M}_{i,j} \ . \tag{1.2}$$

The latter formulation implicitly assumes that the threshold θ is equal to zero. A domain expert can set a different threshold by subtracting some constant to all matrix entries. The threshold θ is explicitly subtracted during the maximization in Equation (1.1). Alternatively, that threshold is implicitly subtracted *prior* to the maximization in Equation (1.2).

The maximal sum submatrix problem is $\mathcal{NP}$-hard from a simple reduction of the maximum edge weight biclique problem [DKT96].

Theorem 1.
The maximal sum submatrix problem is $\mathcal{NP}$-hard.

Proof. The $\mathcal{NP}$-hard maximum edge weight biclique problem (MWBP) introduced by Dawande, Keskinocak, and Tayur [DKT96] is defined as follows: given a bipartite graph $\mathrm{B} = (\mathrm{V}_1 \cup \mathrm{V}_2, \mathrm{E})$, find $\mathrm{U}_1 \subseteq \mathrm{V}_1$, $\mathrm{U}_2 \subseteq \mathrm{V}_2$ such that $\forall (u,v) \in \mathrm{U}_1 \times \mathrm{U}_2 : (u,v) \in \mathrm{E}$ (biclique constraint) and such that the sum of the edge weights $\sum_{(u,v) \in \mathrm{U}_1 \times \mathrm{U}_2} w(u,v)$ is maximum. That problem can be reduced to the maximal sum submatrix problem by defining a $|\mathrm{V}_1| \times |\mathrm{V}_2|$ matrix with values $\boldsymbol{M}_{i,j} = w(i,j)$ if $(i,j) \in \mathrm{E}$, $-\infty$ otherwise. Any solution to the maximal sum submatrix problem with an objective value larger than $-\infty$ corresponds to a biclique in B and its objective is the same as in the original maximum edge weight biclique problem. □

This maximal sum submatrix problem is difficult to solve, especially in gene expression analysis, as one is typically interested in solving it for large instances made of thousands of genes and possibly hundreds of samples.

1.2 Interpretation in gene expression analysis

Gene expression data is typically represented as a large matrix of gene expression levels across various samples. Values are generally in a continuous range, for instance on a logarithmic scale, and properly normalized: negative and positive values represent expression values below and above a threshold θ, respectively. That threshold may correspond to the median expression level over the whole matrix, or a row-specific value representing the average expression level of a gene across all samples.

The objective function defined in Equation (1.2) seeks for subsets of rows and columns with globally high values. In biological terms, one looks for a subset of biomarkers that are, after appropriate normalization, relatively highly expressed among a subset of samples.

After proper normalization, positive values are considered the interesting ones for the maximal sum submatrix problem. They correspond, by default, to the high levels of expression one is interested in finding in the instance matrix. One could also look for low expression levels by replacing such a normalized matrix $\boldsymbol{M}$ by its opposite $-\boldsymbol{M}$.

1.3 Identifying k submatrices

A natural extension to the MSS problem is to identify K submatrices that together maximize the sum of the entries covered. Such an extension has many practical data-mining applications where one is interested in discovering K strong relations between two groups of variables (rows and columns) represented or representable as a matrix.

In gene expression analysis, rows correspond to genes and columns to samples, and the value in $\boldsymbol{M}_{i,j}$ is the measurement of the expression of gene i in sample j. One is typically interested in finding a subset of genes that present high expression in a sample subset. It would indicate that a particular biological pathway made of these genes is active in these samples. Finding multiple pathways is a common task in gene expression analysis. Indeed, a gene might participate in multiple pathways specific to different subsets of samples. Similarly, several biological pathways might be active simultaneously in a sample.

In migration data, rows correspond to source location and columns to destinations. The value $\boldsymbol{M}_{i,j}$ represents the number of persons that moved from source location i to destination j. The goal is to find multiple groups of source and destination pairs characterized by large, conjoint migration flow. The data could be a square matrix with identical row and column labels. The matrix content above the diagonal would be redundant with the content below it, and the diagonal does not appear relevant for migration studies. In practice, however, countries could be split into source locations and destinations, without overlaps, based on:

- the continent they belong to,
- the richness of the countries,
- common governments policies,
- …

Such splits allow to answer more specific questions regarding migration flows among continents, poor and rich countries, or between locations with major policies differences. Likewise, locations could be divided between free and infected regions when studying the spread of a pandemic.

A sports journalist could be interested in Olympic games to discover multiple pairs of (countries subsets, sports subsets) of strong performance. The matrix value $\boldsymbol{M}_{i,j}$ then represents the number of medals obtained by the country i in sport j. Similarly to migration data, the aim is to find specific countries with high performance in specific sports and conversely.

A simple adaptation of the problem in Definition 1 to maximize the sum of the weights of the K submatrices is not relevant. Indeed, the optimal solution is trivially defined as selecting K times the maximal sum submatrix. Such a solution does not bring any useful information other than the maximal sum submatrix. One must modify the objective function or impose additional constraints to find pairwise-different submatrices.

We considered two such alternatives. Chapter 3 presents an extension with disjunction constraints preventing any overlap of the K submatrices. Then, an extension with overlap opportunities is presented in Chapter 4. The problem is defined as the sum of entries covered by at least one submatrix. These extensions are defined such that the ordering of the submatrices is irrelevant.

1.3.1 Disjoint submatrices

Chapter 3 presents an extension to the MSS problem to find K submatrices. It relies on a modification of the problem introducing a disjunction constraint. The disjunction constraint enforces that the intersection of entries covered by any pair of submatrices is empty. In other words, each matrix entry cannot be selected more than once. It must be stressed that any submatrix pair may share rows or columns. The disjunction constraint only prevents submatrices from sharing both rows and columns at the same time.

Such a structure of the solution is commonly called *nonoverlapping nonexclusive nonexhaustive* in the context of biclustering:

- each matrix entry can be covered by at most one bicluster,
- every bicluster pair can share some row *or* some columns but not both,
- some rows and columns can be excluded from all biclusters.

1.3.2 Possibly overlapping submatrices

Chapter 4 presents an extension to the identification of K possibly overlapping submatrices with maximal weight. It relies on a modification of the objective function such that covered entries contribute strictly once to the objective. In other words, the coverage is important, not the number of times an entry is covered. The objective is to discover K submatrices that together cover the largest sum of entries of the input matrix.

1.4 Related work

The maximal sum submatrix problem and its extensions have several close and distant variants. This section presents the main similarities and differences with related works.

1.4.1 Biclustering

Biclustering, also known as co-clustering, is one of the most common approaches in gene expression analysis as it identifies specific subsets of rows and columns which jointly form homogeneous entries.

A substantial number of biclustering methods and applications have been described since the application of biclustering, introduced in [Har72], to gene expression data analysis [CC00]. Several biclustering algorithms reviews have been published emphasizing on various characteristics of the biclustering algorithms, applications, or results.

For example, Madeira *et al.* study in [MO04] a collection of sixteen biclustering methods and categorize them according to the structures and patterns of biclusters they can find, the methods used to perform the search, and the approach used to evaluate the solution. In [PGA15], the survey mentioned above is updated and extended to forty-seven biclustering algorithms. Each method is further categorized based on the use, or not, of evaluation metrics within the search.

Padilha *et al.* conduct, in [PC17a], a comparative study on seventeen algorithms on a large collection of synthetic and real datasets. They conclude that algorithms only achieved satisfactory results in a specific context, and that the best results are obtained by selecting an algorithm depending on the specific task at hand. A similar conclusion is presented in [Ere+12], based on results from a comparative study on twelve algorithms on a suite of synthetic datasets and eight real datasets.

A systematic summary of basic and advanced biclustering applications for biological and biomedical data is presented in [Xie+18]. Guidance on the appropriate algorithms and tools to effectively analyze specific data types and to generate valuable biological knowledge is provided.

Biclustering is typically applied on a dataset in the form of a matrix $\boldsymbol{M}$ where the entry $\boldsymbol{M}_{i,j}$ represents the value of a specific row i (*e.g.* a gene) obtained for a specific column j (*e.g.* a sample). A bicluster is a submatrix of $\boldsymbol{M}$ defined by a subset of *selected rows* and a subset of *selected columns*. The selected rows or selected columns need not be contiguous in the original matrix $\boldsymbol{M}$.

Biclustering algorithms tend to produce biclusters sharing similar expression values, such as minimizing the variance across the selected genes and selected samples. However, some relevant biclusters may be missed when, due to the presence of a few outliers, they lack the assumed homogeneity of expression values among a few gene/sample combinations.

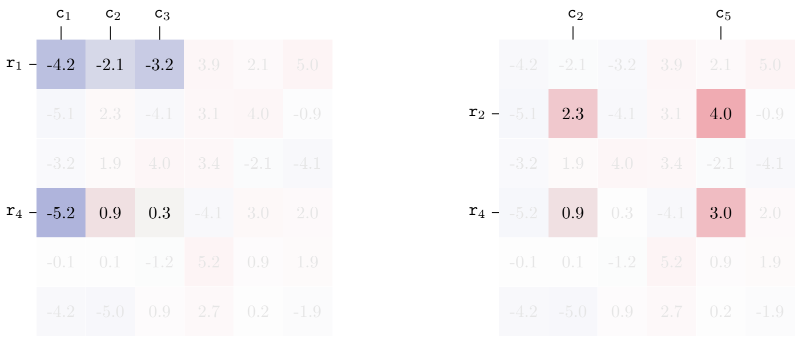
As an alternative, the maximal sum submatrix problem seeks for subsets of rows and of columns with globally high values. They form a rectangular, and not necessarily contiguous, submatrix of the original data matrix exactly like biclusters do. Yet, the mathematical criterion used to find such submatrix differs and is less influenced by the presence of some outliers. There is no assumption of homogeneity in the maximal sum submatrix problem. The difference is illustrated in FIGURE 1.1b where a highly negative entry -4.1 in $(\mathrm{r}_4, \mathrm{c}_4)$ is selected. This results from the sums of the entries in r_4 and in c_4 which contribute positively to the maximal sum. In contrast, as one looks for a *rectangular* submatrix, a positive entry may be excluded from the optimal solution if it is penalized by the presence of negative values along its row and its column. This is the case, for example, for entry 4.0 in row r_3 and column c_3 of this toy example. In the biclustering context, any entry that differs from entries of a bicluster or from entries in its row or its column is not expected to be selected. FIGURE 1.3 represents the results obtained with two different biclustering algorithms, namely CCA and ISA (further described in the section 5.2 Mining approaches), starting from the same toy example (FIGURE 1.1a). Both their solutions strongly differ from the one represented in FIGURE 1.1b. In particular, the CCA solution includes many negative entries as they imply a lower variance along selected rows and selected columns. In contrast, the ISA solution only includes positive entries but is missing several genes and samples that should arguably be selected as in FIGURE 1.1b.

1.4.1.1 Cohesive biclusters

Cohesive biclusters, with high average values, are build by aggregating entries that are higher than a certain threshold such that the average value of the bicluster is higher than a second threshold [Pio+13; Pio+15]. Then all entries must be higher than the first threshold, while in the maximal sum submatrix problem, there is no expected minimum value for an entry to be selected.

1.4.2 Maximum subarray

The maximum (contiguous) subarray problem introduced in [Ben84] identifies a subarray of maximal sum from an array. For a one-dimensional ar-



(a) CCA solution. **(b) ISA solution.**

Figure 1.3: Bicluster found using CCA and ISA on the instance matrix in Figure 1.1a.

ray, this problem can be solved in linear time by Kadane's algorithm [Ben84]. Cubic and sub-cubic time complexity algorithms have been proposed in the two-dimensional case [Ben84; Tak02; TT98]. This problem is however simpler than the maximal sum submatrix problem since the selected submatrix is constrained to be formed of contiguous rows and contiguous columns from the original matrix. Such a restriction is however not justified in the general context of transcriptomics since it would require to know in advance unique and specific orders in which the genes and the samples can be clustered.

1.4.3 Maximal ranked tile mining

The maximal ranked tile mining problem has been introduced in [Le +14]. This is a special case of the maximal sum submatrix problem for which the matrix entries are discrete ranks, corresponding to a permutation of column indices on each row.

Two extensions to the maximal ranked tile mining problem to mine K submatrices have been considered. The *diverse ranked tiling* imposes a disjunction constraint on the rows: no two submatrices can have overlaps on their rows. The *ranked tiling* considers penalties for the possible overlaps of submatrices.

While the discrete ranking is an important characteristic of the maximum ranked tile mining problem, and the associated extensions, the corresponding constraint programming solutions do not require discrete entries nor benefit from it. In this work, we present optimization approaches to solve problems with arbitrary continuous entries.

1.4.4 Subgroup discovery

Subgroup discovery is a data mining technique that extracts classification rules concerning a target variable [Atz15; Her+11]. It departs from the standard learning of a classifier as the extracted rules are not necessarily intended to cover all possible instances. Besides, such a technique focuses on the interpretability of the classification rules rather than the generalization capability to classify new instances. Mining a maximal sum submatrix is somewhat similar to subgroup discovery, but there is no supervision by a specific target class variable. The hotspot detection problem, which can be considered as a particular form of subgroup discovery, aims at identifying subgroups that are unexpected with respect to some baseline information [FG15]. In the maximal sum submatrix problem, one does consider globally high values without any baseline distribution of the data.

1.5 Constraint programming

Constraint programming (CP) is a flexible programming paradigm that is capable of solving optimization problems. It received an increasing interest for solving unsupervised (clustering) data-mining problems [Le +14; SAG17; Kuo+18; AGS16; Bes+16; DV+17; CS17].

As a declarative approach, it only requires to model the problem and, by using existing solvers, let it search and find solutions.

A model is defined as a constraint satisfaction problem $CSP = (V, D, C)$ where V is the set of variables, D is their respective domains, and C is a set of constraints defined over the variables. A *feasible solution* is an assignment of the variables to values of their domains such that all constraints are satisfied.

1.5.1 Constraints

Constraints are exploited to iteratively reduce the domains of variables. Such *constraint propagation* reduces the number of variable assignments to consider. Once all unfeasible values are removed from the domains of variables, the *fix-point* of the propagation is reached. Then the solver selects a variable $X \in V$ that is unbound and recursively calls the solver while assigning a value to this variable.

Through depth-first-search (DFS) exploration of a search tree, the solver either reaches a solution or backtracks when the domain of variables becomes empty. Using a branch & bound (B & B) depth-first-search avoids the exploration of proven suboptimal solutions.

1.5.2 Efficient backtracking

Efficient backtracking is achieved through trailing, a state management strategy that facilitates the restoration of the computation state to an earlier version, effectively *undoing* changes that were imposed since then.

The *trail* exposes two methods: `saveState` and `restoreState` to respectively time-stamp the current state and restore it. Its implementation is captured in terms of two simple stacks. The first stack holds entries to undo, the second one holds the frame sizes stacked between two consecutive calls to `saveState`. Trailing enables the design of reversible objects defined on the trail. We refer to MiniCP [MSV18] for a detailed description of trailed-based solvers.

1.5.2.1 LNS

The search space of some problems can be so large that proving optimality requires excessive computational time. To overcome this limitation, exhaustive CP search can be embedded into a large neighborhood search (LNS) [Sha98]. LNS is a local search approach using CP to discover improvements around the current best solution:

- First the CP exhaustive search is used during a limited time, to discover an initial solution.
- For a given number of iterations, the CP exhaustive search is used again but this time with some variables partially fixed as in the current best solution.

The LNS strategy is commonly implemented as follows: after a given number of backtracking, the constraints on a fraction of the variables of the best solution found so far are removed and the search restarts in another region.

LNS may improve the time to identify a good solution at the cost of losing the possibility to prove optimality since the search is no longer complete.

1.6 Submatrix modeling

Submatrices are modeled in subsequent chapters using either binary decision variables or set variables. Preference is given to the model that best clarifies the description and simplifies the notations. In practice, however, all original CP implementations rely on sparse-sets. A sparse-set is an efficient data structure to store, add and remove elements from a collection [BT93]. Storing the dynamic size of the sparse-set in the *trail* allows restoring the collection to earlier versions in $\mathcal{O}(1)$. A submatrix is modeled using four sparse-sets to encode:

1. the rows that might or might not be in the submatrix,
2. the columns that might or might not be in the submatrix,
3. the rows required in the submatrix,
4. the columns required in the submatrix.

Rows and columns excluded from a submatrix are implicitly defined as the rows and columns absent from the four sparse-sets.

Mining a submatrix of maximal sum 2

The maximal sum submatrix problem requires dedicated algorithms to be applied in gene expression data with hundreds of thousands of entries. This chapter presents such algorithms. A reduction of the search space is exposed first. Then, optimization approaches relying on constraint programming and mixed integer linear programming are presented. Finally, experiments on gene expression and synthetic data are presented. These experiments provide guidance on the best strategies to solve problems with large instance matrices.

2.1 Introduction

The problem of finding a maximal sum submatrix (MSS) is presented in the previous chapter. The particular application of the MSS problem to biological data is presented in Section 1.2. Le Van et al. [Le +14] proposed a constraint programming approach combined with large neighborhood search (LNS) to solve the special case where matrix entries are discrete ranks.

This chapter aims at proving the applicability of the problem in instances matrices with orders of magnitude more rows and columns as is common in biological applications. Gene expression data is typically represented as a large matrix of tens of thousands of gene expression levels across hundreds of samples. A submatrix of maximal sum defines in such data a subset of genes that present a high expression pattern for a specific subset of samples. The study of such data is a valuable tool to improve the understanding of the underlying biological processes.

The main contributions are:

1. the refinement of the search space of the maximal sum submatrix problem;

2. the definition of an upper bound that is easy to compute, speedying up the search for a solution;

3. the design and implementation of two CP and two mixed integer linear programming models;
4. experiments on real gene expression data and synthetic data showing the performance of each implementation;
5. the study of the relative benefits of the proposed methods across various problem instances.

2.2 Search space

For a defined subset of columns J, the objective function can be formulated as $f(\mathrm{I},\mathrm{J}) = \sum_{i\in \mathrm{I}} r_i$ with r_i being the contribution of the row i. In other words, r_i is the sum of entries in row i and in the set of columns J:

$$r_i = \sum_{j\in \mathrm{J}} \boldsymbol{M}_{i,j} \ . \tag{2.1}$$

This is important as the actual search can be limited to one dimension through independent computation of the contributions along the other dimension. Indeed, for any of the two dimensions being fixed, optimization along the other dimension is straightforward since it amounts to select only the subset of entries whose contribution is positive. The subset of rows ${\mathrm{I}_\mathrm{J}}^* \subseteq \mathrm{R}$ that maximizes the weight of a submatrix with a fixed subset of columns $\mathrm{J} \subseteq \mathrm{C}$ is identified as:

$${\mathrm{I}_\mathrm{J}}^* = \underset{\mathrm{I}\subseteq \mathrm{R}}{\operatorname{argmax}} \sum_{\substack{i\in \mathrm{I}\\ j\in \mathrm{J}}} \boldsymbol{M}_{i,j} = \{i \in \mathrm{R} \mid r_i \geq 0\} \ . \tag{2.2}$$

In the gene expression analysis context, with order(s) of magnitude more rows (the genes) than columns (the samples), one typically searches for a subset of columns and selects the associated optimal subset of rows.

The search space of the maximal sum submatrix problem contains $2^m \times 2^n$ *feasible solutions* that are rectangular submatrices $(\mathrm{I} \subseteq \mathrm{R}, \mathrm{J} \subseteq \mathrm{C})$ of the original m by n matrix $\boldsymbol{M}$. Thanks to the independent contribution along one dimension, the number of feasible solutions to be explicitly evaluated is thus reduced to $2^n \times m$.

2.3 Optimization approaches

There is more than a single way to address the maximal sum submatrix problem. Two types of approaches are considered: constraint programming and mixed integer linear programming. A submatrix is modeled using boolean decision variables to insert or exclude rows and columns in a submatrix. That common modeling is first presented. Three CP approaches and two MILP models are subsequently given.

Note that the actual constraint programming implementations rely on sparse-sets (see SECTION 1.6).

2.3.1 Modeling with boolean decision variables

A submatrix from an m by n matrix $\boldsymbol{M}$ can be modeled with two boolean decision vectors $\mathbf{R}$ and $\mathbf{C}$ associated with the rows and columns, respectively.

DEFINITION 2. Boolean decision vectors for the rows and columns
Let $\mathbf{R} = (\mathbf{R}_1, \dots, \mathbf{R}_m)$ *denote a boolean decision vector associated with the rows* $\{\mathtt{r}_1, \dots, \mathtt{r}_m\}$ *of matrix* $\boldsymbol{M}$*. The domain of* $\mathbf{R}_i$ *is* $Dom(\mathbf{R}_i) = \{0, 1\}$ *for any* $i \in \{1, \dots, m\}$*. The boolean decision variable* $\mathbf{R}_i$ *is assigned the value:*

- 1 *if the row* $\mathtt{r}_i$ *is selected, or belongs to the submatrix;*
- 0 *if the row* $\mathtt{r}_i$ *is excluded, or does not belong to the submatrix.*

Similarly, $\mathbf{C} = (\mathbf{C}_1, \dots, \mathbf{C}_n)$ *denotes the columns of matrix* $\boldsymbol{M}$*. The domain of* $\mathbf{C}_j$ *is* $Dom(\mathbf{C}_j) = \{0, 1\}$ *for any* $j \in \{1, \dots, n\}$*. The boolean decision variable* $\mathbf{C}_j$ *is assigned the value:*

- 1 *if the column* $\mathtt{c}_j$ *is selected, or belongs to the submatrix;*
- 0 *if the column* $\mathtt{c}_j$ *is excluded, or does not belong to the submatrix.*

Defining a submatrix $(\mathtt{I}, \mathtt{J})$ consists in assigning a value to each variable of $\mathbf{R}$ and $\mathbf{C}$ such that $\mathtt{I} = \{i \in \mathtt{R} \mid \mathbf{R}_i = 1\}$ and $\mathtt{J} = \{j \in \mathtt{C} \mid \mathbf{C}_i = 1\}$.

DEFINITION 3. Unbound, selected and unselected rows and columns
Let $\mathbf{R}^{\in} = \{i \in \mathtt{R} \mid \mathbf{R}_i = 1\}$ *denote the selected rows. Let* $\mathbf{C}^{\in} = \{j \in \mathtt{C} \mid \mathbf{C}_j = 1\}$ *denote the selected columns.*
Let $\mathbf{R}^{\notin} = \{i \in \mathtt{R} \mid \mathbf{R}_i = 0\}$ *denote the rows that do not belong to the submatrix. Let* $\mathbf{C}^{\notin} = \{j \in \mathtt{C} \mid \mathbf{C}_j = 0\}$ *denote the columns that do not belong to the submatrix.*
Let $\mathbf{R}^{?} = \{i \in \mathtt{R} \mid Dom(\mathbf{R}_i) = \{0, 1\}\}$ *denote the unbound rows, or rows that are not assigned a value yet. Let* $\mathbf{C}^{?} = \{j \in \mathtt{C} \mid Dom(\mathbf{C}_j) = \{0, 1\}\}$ *denote the unbound columns.*

An assignment of the variables $(\mathbf{R}_1, \dots, \mathbf{R}_m)$ and $(\mathbf{C}_1, \dots, \mathbf{C}_n)$ is complete if $\mathbf{R}^{?} = \emptyset$ and $\mathbf{C}^{?} = \emptyset$. Any complete assignment defines a submatrix which is a feasible solution to the MSS problem. Finding a *best solution* requires some way of specifying which feasible solutions are better than others. An *objective function* allows to specify which solutions are better than others. The weight of a submatrix is an objective function to compare submatrices. It maps any feasible solution to a real value that can be maximized.

Finding a maximal sum submatrix consists in finding a complete assignment of the variables in **R** and **C** such that $f(\mathbf{R}^{\in}, \mathbf{C}^{\in})$ is maximal:

$$f(\mathbf{R}^{\in}, \mathbf{C}^{\in}) = \sum_{\substack{i \in \mathbf{R}^{\in} \\ j \in \mathbf{C}^{\in}}} \boldsymbol{M}_{i,j} \ . \tag{2.3}$$

2.3.2 Constraint programming

The performance of constraint programming techniques, introduced in Section 1.5, depends on the model representing a problem. Three different models are shown in this subsection. First, the original model proposed by Le Van et al. [Le +14] is presented. A refinement of that model is subsequently given, exploiting the reduction of the search space presented in Section 2.2. Finally, a new algorithm capturing the whole MSS problem in a global constraint is defined.

2.3.2.1 CP-LNS$_0$

The maximal sum submatrix problem is modeled by Le Van et al. [Le +14] as follows:

$$\text{maximize} \sum_{i \in \mathrm{R}, j \in \mathrm{C}} \mathbf{R}_i \times \mathbf{C}_j \times \boldsymbol{M}_{i,j} \ , \tag{2.4}$$

$$\forall i \in \mathrm{R} : \ \mathbf{R}_i = 1 \Leftrightarrow \sum_{j \in \mathrm{C}} \mathbf{C}_j \times \boldsymbol{M}_{i,j} \geq 0 \ , \tag{2.5}$$

$$\forall j \in \mathrm{C} : \ \mathbf{C}_j = 1 \Leftrightarrow \sum_{i \in \mathrm{R}} \mathbf{R}_i \times \boldsymbol{M}_{i,j} \geq 0 \ . \tag{2.6}$$

Expression (2.4) states the optimization problem. The exploration of the depth-first-search tree leads to a leaf, or *feasible solution*, as soon as all rows and columns variables are bound.

The model relies on the idea that any row should be selected if and only if the sum of entries along the selected columns is positive. These constraints on the rows are reified in Equation (2.5). A constraint c is reified if there is a Boolean variable b associated with c such that c is satisfied if and only if b is true. Reified constraints are similarly defined on the columns in Equation (2.6).

Redundant constraints are constraints that can be omitted without changing the solution space. The constraints in Equation (2.5) and Equation (2.6) are redundant as they are implicitly defined in the maximization term in Expression (2.4). However, these constraints are usefull as they permit a stronger filtering during the search. Assume a complete assignment on the columns, $\mathbf{C}^{?} = \emptyset$. The optimal solution can be found by applying $|\mathbf{R}^{?}|$ redundant constraints from Equation (2.5). These constraints filter $2^{|\mathbf{R}^{?}|} - 1$ nodes of the search tree.

The large neighborhood search strategy, presented in Section 1.5.2.1, is implemented here as follows:

1. Search for a solution until a given number of backtracking happens.
2. Perform a uniform random selection of half of the columns.
3. Constrain these variables to their value in the current best solution.
4. Repeat 1 on the other variables.

LNS may improve the time to identify a good solution at the cost of losing the possibility to prove optimality since the search is no longer complete.

2.3.2.2 CP-LNS

An improved CP model obtaining the same filtering as the original one but resulting in a lighter propagation of the constraints is proposed as follows:

$$\text{maximize} \sum_{i \in \mathrm{R}} \mathbf{R}_i \times r_i \ , \tag{2.7}$$

$$\text{with} \quad \forall i \in \mathrm{R} : \ r_i = \sum_{j \in \mathrm{C}} \mathbf{C}_j \times M_{i,j} \ , \tag{2.8}$$

$$\text{such that} \quad \forall i \in \mathrm{R} : \ \mathbf{R}_i = 1 \Leftrightarrow r_i \geq 0 \ . \tag{2.9}$$

Expression (2.7) states the optimization problem. The objective is to maximize the sum of rows contributions r_i, formalized in Equation (2.8), for the rows that are selected.

First, the search is performed on the column variables until all are bound. By virtue of the search space reduction presented in Section 2.2, the tree reaches a leaf as soon as the column variables are bound. The rows contributions are computed afterwards in Equation (2.8). Finally, each row with positive contribution is constrained in Equation (2.9) to be selected. Rows with negative contributions are excluded.

This improved model avoids the computation of the quite-heavy reified sum constraints in Equation (2.6) and reduces the number of terms in the objective function. As each product between variables in the objective is translated into a small binary constraint, reducing their number from $|\mathbf{R}| \times |\mathbf{C}|$ to $|\mathbf{R}|$ makes a significant difference in the time spent to compute the fix-point in each node.

In the gene expression analysis context, $m = |\mathbf{R}|$ can easily be orders of magnitude larger than $n = |\mathrm{C}|$. Then, the following symmetrical counterpart of CP-LNS would benefit of a smaller number of terms in the objective and a smaller number of reified constraints:

$$\begin{aligned}
&\text{maximize} \sum_{j \in \mathrm{C}} \mathbf{C}_j \times c_j \ , \\
\text{with} \quad &\forall j \in \mathrm{C} : \ c_j = \sum_{i \in \mathrm{R}} \mathbf{R}_i \times \boldsymbol{M}_{i,j} \ , \\
\text{such that} \quad &\forall j \in \mathrm{C} : \ \mathbf{C}_j = 1 \Leftrightarrow c_j \geq 0 \ .
\end{aligned}$$

However, the benefits are balanced by the branching performed on the rows rather than on the columns as in Equations (2.7-2.9). The size of the search space is significantly increased from $2^n \times m$ to $2^m \times n$.

Note that the CP-LNS model and its symetrical counterpart have an identical search space and identical depth-first-search exploration if they are given as input matrix $\boldsymbol{M}$ and its transpose $\boldsymbol{M}^\top$, respectively.

2.3.2.3 CP Global Constraint

A *global constraint* is a constraint that captures a relation between a non-fixed number of variables. An example is the constraint alldifferent$(x_1, \ldots, x_n)$, which specifies that values assigned to the variables $x_1, \ldots, x_n$ must be pairwise distinct. Typically, the relations between variables in a global constraint can be expressed as the conjunction of several simpler constraints. A global constraint facilitate the work of the constraint solver by providing it with a better view of the structure of the problem [HK06].

The CP models CP-LNS$_0$ and CP-LNS employ reified constraints to express the relations between row and column variables $\mathbf{R}_1, \ldots, \mathbf{R}_m, \mathbf{C}_1, \ldots, \mathbf{C}_n$. Theses relations can equivalently be expressed as a global constraint capturing the whole MSS problem. Algorithm 1 presents the pseudo-code of an algorithm encapsulated inside a global constraint. The call to the bounding and filtering procedures has been made explicit. In practice, the lines 10 to 16 would be encapsulated in the global constraint triggered by the fix-point algorithm.

The key characteristics of the global constraint approach described in Algorithm 1 are:

- A feasible solution at each node of the search tree.
- An efficient bounding procedure.
- An efficient procedure to filter the domains.

Algorithm 1: CP Global Constraint

```
Data: $\boldsymbol{M}$, C, R
Output: Columns of the maximal sum submatrix in $\boldsymbol{M}$
1  $best \leftarrow -\infty$                          // best so far objective
2  $bestCol \leftarrow \emptyset$                          // best so far columns
3  $r^{psum}$: array$[m]$                       // partial sums of rows
4  Function dfs():
5      if $C^? \neq \emptyset$ then
6          $j \leftarrow$ selectUnboundVar( $\mathbf{C}^?$)
7          foreach $v \in \{0,1\}$ do
8              saveState()
9              Dom($\mathbf{C}_j$) $\leftarrow v$
10             $currentBest \leftarrow$ evaluate()
11             if $currentBest > best$ then
12                 $best \leftarrow currentBest$
13                 $bestCol \leftarrow \mathbf{C}^{\in}$
14             $ub \leftarrow$ upperBound()
15             if $ub > best$ then
16                 filter()
17                 dfs()
18             restoreState()
19     return $bestCol$
```

The actual implementation of ALGORITHM 1 benefits from the use of reversible objects to allow efficient backtracking (see SECTION 1.5.2 Efficient backtracking) for each call to the `restoreState()` function.

The CPGC name comes from the fact that the implementation captures relations between the rows and columns and provides a bound on the sum of covered entries. However, it also presents dominance rules and filtering procedures to reduce the time spent finding a solution. Consequently, ALGORITHM 1 is rather a dedicated solution to the MSS problem than an actual global constraint.

A feasible solution at each node of the search tree. CP usually updates its feasible solution and best so far lower bound in the leaf-node of the search tree, that is when every variable of the problem is bound. One can observe that for the maximal sum submatrix problem, any partial assignment of the variables can be extended implicitly as a complete solution for which the unbound variables would be set to 0. In other words, all $\mathbf{C}^?$ variables are considered unselected in each node. Consequently, there is no need to wait that every variable is actually bound to evaluate the solution and possibly update the best so far lower bound. The value of the objective function of the feasible solution is computed in the `evaluate()` function, ALGORITHM 2, as the sum of the positive rows contributions of the partial solution:

$$\sum_{i \in \mathrm{R}} \max(0, r_i{}^{psum}) \quad , \tag{2.10}$$

$$\text{with} \quad r_i{}^{psum} = \sum_{j \in \mathbf{C}^{\in}} \boldsymbol{M}_{i,j} \quad . \tag{2.11}$$

That definition is a direct consequence from the search space reduction presented in SECTION 2.2.

An efficient bounding procedure. CP uses a branch & bound depth-first-search to avoid the exploration of proven suboptimal solutions. The B & B performance depends on the strength and efficiency of the procedure to compute the upper bound. A simpler yet efficient bounding procedure for the maximal sum submatrix problem is presented hereafter. Intuitively, one computes an upper bound on the row contribution of each row and sums up all the positive bounds on the rows.

The upper bound on the contribution of a row is the sum of the matrix entries in the selected columns plus the sum of the positive entries in the unbound columns:

$$r_i{}^{ub} = r_i{}^{psum} + \sum_{j \in \mathbf{C}^?} \max\left(0, \boldsymbol{M}_{i,j}\right) \quad . \tag{2.12}$$

Algorithm 2: Evaluating the objective value

```
Data: $M$, C, R, $r^{psum}$
Output: The weight of the best submatrix with columns $C^{\in}$
1  Function evaluate():
2      for $j \in C^?$ do
3          if $Dom(C_j) \neq \{0,1\}$ then
4              $C^? \leftarrow C^? \setminus j$
5              if $Dom(C_j) = \{1\}$ then
6                  foreach $i \in R^?$ do
7                      ${r_i}^{psum} \leftarrow {r_i}^{psum} + M_{i,j}$
8      $obj \leftarrow 0$
9      foreach $i \in R^?$ do
10         $obj \leftarrow obj + \max(0, {r_i}^{psum})$
11     return $obj$
```

The upper bound for any partial assignment is the sum of positive bounds on the rows:

$$\sum_{i \in R} \max\left(0, {r_i}^{psum} + \sum_{j \in C^?} \max\left(0, M_{i,j}\right)\right) \quad . \tag{2.13}$$

The bound is admissible but not tight as it may optimistically evaluate the objective:

$$\underbrace{\sum_{i \in R} \max\left(0, {r_i}^{psum} + \sum_{j \in C^?} \max\left(0, M_{i,j}\right)\right)}_{\text{Upper bound from Equation 2.13}} \geq \underbrace{\sum_{\substack{i \in R^* \\ j \in C^*}} M_{i,j}}_{\text{Optimal solution}} \quad . \tag{2.14}$$

Indeed, it relies on a per-row relaxation of the problem, each row selecting a possibly different set of columns.

The `upperBound()` function, Algorithm 3, is an implementation of the proposed upper bound using reversible double to store the incremental modifications of the partial contribution of the rows. Using a reversible sparse-set for the candidate row variables $\mathbf{R}^?$ allows an efficient exclusion or inclusion of the rows through descent or backtrack [Sai+13]. Indeed, as soon as the bound on the row contribution becomes negative it should not be considered in the descendant nodes of the search tree. The number of rows to consider goes from exactly $|\mathbf{R}|$ to at most $|\mathbf{R}|$.

Algorithm 3: Computing the upper bound

```
Data: $M$, C, R, $r^{psum}$
Output: The upper bound of a partial assignment
1 Function upperBound():
2     $ub \leftarrow 0$
3     for $i \in R^?$ do
4         $r_i{}^{ub} \leftarrow r_i{}^{psum}$
5         foreach $j \in C^?$ do
6             $r_i{}^{ub} \leftarrow r_i{}^{ub} + \max(0, M_{i,j})$    // see equation 2.12
7         if $r_i{}^{ub} > 0$ then
8             $ub \leftarrow ub + r_i{}^{ub}$
9         else
10            $R^? \leftarrow R^? \setminus i$    // $r_i{}^{psum}$ always $\leq 0$
11    return $ub$
```

An efficient filtering procedure. The `filter()` function, ALGORITHM 4, evaluates the upper bound result for two one-step look-ahead scenarios:

1. if column j would be selected,
   - that look-ahead upper bound is denoted $ub_j{}^{\in}$;
   - any column j such that $ub_j{}^{\in} \leq best$ is discarded: its inclusion can only lead to worst solution than the best solution found so far;
2. if j would not be selected,
   - that look-ahead upper bound is denoted $ub_j{}^{\notin}$;
   - any column j such that $ub_j{}^{\notin} \leq best$ is included: its exclusion can only lead to worst solution than the best solution found so far.

The time complexity for computing all the look-ahead upper bounds is in $\mathcal{O}\left(|R^?| \times |C^?|\right)$. Indeed the look-ahead bound update of each line for each column can be obtained in $\mathcal{O}(1)$ from $r_i{}^{ub}$.

2.3.3 Mixed Integer Linear Programming

Mixed integer linear programming (MILP) [NW88] involves the optimization of a linear objective function, subject to linear constraints. Some or all of the variables are required to be integer. A MILP solver explores a branch & bound tree using linear programming (LP) bounds at each node of the search tree.

Algorithm 4: Filtering impossible values

```
Data: $M$, C, R, $r^{ub}$
Function filter():
    for $j \in C^?$ do
        $ub_j^{\in} \leftarrow 0$        // upper bound if $j$ was selected
        $ub_j^{\notin} \leftarrow 0$        // upper bound if $j$ was excluded
        for $i \in R^?$ do
            if $M_{i,j} > 0$ then
                if $r_i^{ub} - M_{i,j} > 0$ then
                    $ub_j^{\notin} \leftarrow ub_j^{\notin} + r_i^{ub} - M_{i,j}$
                $ub_j^{\in} \leftarrow ub_j^{\in} + r_i^{ub}$
            else if $M_{i,j} < 0$ then
                $ub_j^{\notin} \leftarrow ub_j^{\notin} + r_i^{ub}$
                if $r_i^{ub} + M_{i,j} > 0$ then
                    $ub_j^{\in} \leftarrow ub_j^{\in} + r_i^{ub} + M_{i,j}$
            if $ub_j^{\in} \leq best$ then
                $\mathrm{Dom}(C_j) \leftarrow \mathrm{Dom}(C_j) \setminus \{1\}$
            if $ub_j^{\notin} \leq best$ then
                $\mathrm{Dom}(C_j) \leftarrow \mathrm{Dom}(C_j) \setminus \{0\}$
```

It differs from a classical branch and bound as a linear programming (LP) *relaxation*, obtained by removing all the integrality constraints of a node, is solved before branching. The domain of all rows and columns variables changes from $\{0, 1\}$ to $[0, 1]$. That relaxed problem can be solved in polynomial time and the solution is an upper bound on the objective value of the constrained problem. As an upper bound, the LP relaxation solution can be used to prune out suboptimal solutions. If any integer variable is associated with a fractional value in the linear programming relaxation, two subproblems are generated imposing restrictions on the domain of that variable. When all integrality constraints are satisfied in the solution of a node, then it corresponds to a feasible solution and the lower bound is possibly updated.

In an initial formulation, each entry of the matrix is associated with a decision variable that takes the value 1 if and only if both rows and columns are selected. The objective function is computed as the sum of the selected matrix entries.

2.3.3.1 Initial model

The maximal sum submatrix problem can be linearized as:

$$\begin{aligned}
&\text{maximize} \sum_{\substack{i \in \mathrm{R} \\ j \in \mathrm{C}}} \boldsymbol{M}_{i,j} \times x_{i,j} \ , && && (2.15)\\
\text{subject to} \quad & x_{i,j} \leq \mathbf{R}_i \ , && \forall i \in \mathrm{R}, \forall j \in \mathrm{C} \ , && (2.16)\\
& x_{i,j} \leq \mathbf{C}_j \ , && \forall i \in \mathrm{R}, \forall j \in \mathrm{C} \ , && (2.17)\\
& x_{i,j} \geq \mathbf{R}_i + \mathbf{C}_j - 1 \ , && \forall i \in \mathrm{R}, \forall j \in \mathrm{C} \ , && (2.18)\\
& \mathbf{R}_i \in \{0, 1\} \ , && \forall i \in \mathrm{R} \ , && (2.19)\\
& \mathbf{C}_j \in \{0, 1\} \ , && \forall j \in \mathrm{C} \ , && (2.20)\\
& x_{i,j} \in \{0, 1\} \ , && \forall i \in \mathrm{R}, \forall j \in \mathrm{C} \ , && (2.21)
\end{aligned}$$

where:

- each row i is associated with a binary decision variable $\mathbf{R}_i$;
- each column j is associated with a binary decision variable $\mathbf{C}_j$;
- each matrix entry is associated with a binary decision variable $x_{i,j}$.

The objective function is computed as the sum of matrix entries whose decision variable is set to one. Equations (2.16-2.18) enforce that variable $x_{i,j} = 1$ if and only if $\mathbf{R}_i = 1$ and $\mathbf{C}_j = 1$. All these decision variables are relaxed to the interval $[0, 1]$ at each node of the search tree of the MILP solver.

2.3.3.2 Big M model

An improved and more compact MILP formulation of the problem is proposed hereafter. It relies on the search space study in SECTION 2.2: a row is selected if and only if its contribution to the weight of a submatrix is positive.

A variable ${r_i}^+$ is defined for each row $i \in \mathrm{R}$. Variable ${r_i}^+$ takes the value of the contribution of row i if the contribution is positive, 0 otherwise:

$$\forall i \in \mathrm{R} \ : \ {r_i}^+ = \begin{cases} \sum_{j \in \mathrm{C}^\in} \boldsymbol{M}_{i,j} \ , & \text{if} \quad \sum_{j \in \mathrm{C}^\in} \boldsymbol{M}_{i,j} > 0 \ , \\ 0 \ , & \text{otherwise} \ . \end{cases} \tag{2.22}$$

That formulation is equivalent to the following:

$$\forall i \in \mathrm{R} \ : \ {r_i}^+ = \max(0, \sum_{j \in \mathrm{C}^\in} \boldsymbol{M}_{i,j}) \ . \tag{2.23}$$

The objective is to maximize the sum over these variables ${r_i}^+$. This model uses "big M" constants to linearize EQUATION (2.23). The big M method extends the use of the simplex algorithm to problems that contain `smaller-than` constraints. It does so by associating the constraints with large positive constants which would not be part of any optimal solution, if it exists.

The maximal sum submatrix problem using big M constants is stated as follows:

$$\text{maximize} \sum_{i \in \mathrm{R}} {r_i}^+ \ , \tag{2.24}$$

$$\text{subject to} \quad r_i = \sum_{j \in \mathrm{C}} \boldsymbol{M}_{i,j} \times \mathbf{C}_j \ , \qquad \forall i \in \mathrm{R} \ , \tag{2.25}$$

$${r_i}^+ \leq \mathbf{R}_i \times {\mathfrak{M}_i}^+ \ , \qquad \forall i \in \mathrm{R} \ , \tag{2.26}$$

$${r_i}^+ \leq r_i + (1 - \mathbf{R}_i) \times {\mathfrak{M}_i}^- \ , \qquad \forall i \in \mathrm{R} \ , \tag{2.27}$$

$$\mathbf{R}_i \in \{0, 1\} \ , \qquad \forall i \in \mathrm{R} \ , \tag{2.28}$$

$$\mathbf{C}_j \in \{0, 1\} \ , \qquad \forall j \in \mathrm{C} \ , \tag{2.29}$$

where ${\mathfrak{M}_i}^+$ and ${\mathfrak{M}_i}^-$ are the upper bound and lower bound, respectively, on the sum of entries in row i.

The combination of the maximization term in EQUATION (2.24) and EQUATIONS (2.26) and (2.27) linearize ${r_i}^+ = \max(0, r_i)$ with r_i being the sum of the selected entries in row i. Specifically, EQUATIONS (2.26) and (2.27) ensure that ${r_i}^+ \leq \max(0, r_i)$.

Consider the influence of the possible values of the decision variable $\mathbf{R}_i$

on the model:

$$\begin{aligned}
\forall i \in \mathrm{R}: \quad \mathbf{R}_i = 1 &\implies {r_i}^+ \leq r_i + (1 - \mathbf{R}_i) \times {\mathfrak{M}_i}^- \leq \mathbf{R}_i \times {\mathfrak{M}_i}^+ \ , && (2.30)\\
& {r_i}^+ \leq r_i \ , && (2.31)\\
\forall i \in \mathrm{R}: \quad \mathbf{R}_i = 0 &\implies {r_i}^+ \leq \mathbf{R}_i \times {\mathfrak{M}_i}^+ \leq r_i + (1 - \mathbf{R}_i) \times {\mathfrak{M}_i}^- \ , && (2.32)\\
& {r_i}^+ \leq 0 \ . && (2.33)
\end{aligned}$$

From the maximization term in EQUATION (2.24),

$$r_i > 0 \implies {r_i}^+ = r_i \implies \{0\} \notin \mathrm{Dom}(\mathbf{R}_i) \ . \qquad (2.34)$$

Indeed, if $r_i \geq 0$, the best assignment for ${r_i}^+$ is ${r_i}^+ = r_i$ which requires that $\mathbf{R}_i = 1$. Otherwise, the best assignment is ${r_i}^+ = 0$ which requires that $\mathbf{R}_i = 0$. This formulation is valid if and only if ${\mathfrak{M}_i}^+ \geq r_i$ and ${\mathfrak{M}_i}^- + r_i \geq 0$ for any row $i \in \mathrm{R}$.

To avoid rounding errors and ill conditioned matrices, the big M constants can be replaced in EQUATIONS (2.26) and (2.27) as:

$$\begin{aligned}
\forall i \in \mathrm{R}: \quad {\mathfrak{M}_i}^+ &= \sum_{j \in \mathrm{C}} \max(0, \boldsymbol{M}_{i,j}) \ , && (2.35)\\
\forall i \in \mathrm{R}: \quad {\mathfrak{M}_i}^- &= -\sum_{j \in \mathrm{C}} \min(0, \boldsymbol{M}_{i,j}) \ . && (2.36)
\end{aligned}$$

2.4 Experiments

This section describes experiments conducted to assess the relative performance of three algorithms to solve the maximal sum submatrix problem. CP-LNS denotes the improved version of the method CP-LNS$_0$ proposed by Le Van et al. [Le +14]. The other algorithms are original methods proposed in the present work:

1. CPGC: a CP approach with a global constraint;

2. MILP: a mixed integer linear programming approach using "big M" constants.

These algorithms are first compared on data matrices which are generated in a controlled setting. Experiments on the breast cancer gene expression data used in [Le +14] are reported next.

The main criterion to assess the performance of the various methods is the computing time to solve a particular problem instance.

All algorithms have been implemented in the Scala programming language (2.11.4). Each run is executed with a single thread on a MacBook Pro

(OS version 10.10.5) laptop (Intel i7-2720 CPU @ 2.20-3.30GHz, 4GB RAM per run). Constraint programming implementations are based on OscaR [Osc12] (version 4.0.0) and MILP is based on Gurobi (version 7.0.2). The code, datasets and supplementary results are provided in [BS19].

2.4.1 Evaluation

One could assess algorithms performance through runtime or number of feasible solutions. While the former may depend on implementation details, the latter strongly depends on the time spent in each node. As an example, the large number of reified constraints in $\mathtt{CP\text{-}LNS}_0$ has a major impact on the time spent to compute the fix-point in each node while the filtering is as strong as the filtering of the CP-LNS model. While both should perform equally well in terms of the number of feasible solutions, it was observed in preliminary experiments that $\mathtt{CP\text{-}LNS}_0$ is significantly slower than CP-LNS.

We prefer the runtime comparisons as it is a more common approach and we made sure to implement the algorithms in the most comparable fashion. This is technically assessed through an any-time profile defined below.

2.4.1.1 Any-time profile

In practice, an important criterion for the user is the time required to solve an instance and the ability to find the best solution within a given budget of time. Using *any-time profiles*, one can summarize these characteristics.

The idea behind any-time profiles is that an algorithm should produce as high quality solution as possible at any moment of its running time [LS14]. It directly provides a cumulative probability for a method to solve an arbitrary instance after a given budget of time. In the MSS problem, a high solution quality corresponds to a submatrix of large sum. For each instance, runs not completed in a maximum budget of time t^{max} are interrupted.

Definition 4. **Any-time profile**
Let $f(i,t)_a$ be the objective value of the best solution found by an algorithm a, from a set of algorithms A, at a given time t on an instance i from a set of instances S.
Let t^{max} be the maximal time allocated for any algorithm to solve an instance. The any-time profile of an algorithm a is the solution quality $Q(a,t,A,t^{max},S)$ computed on all instances as a function of time:

$$Q(a,t,A,t^{max},S) = \frac{1}{|S|}\sum_{i\in S}\frac{f(i,t)_a}{f(i,t^{max})_{a*}}\ , \tag{2.37}$$

$$\textit{with} \qquad f(i,t^{max})_{a*} = \operatorname*{argmax}_a f(i,t^{max})_a\ . \tag{2.38}$$

2.4.2 Synthetic data

Synthetic data are generated by implanting a submatrix (I, J) of interest in a larger matrix $\boldsymbol{M} = (\mathrm{R}, \mathrm{C})$ made of m rows and n columns, following a similar protocol as proposed by Le Van et al. [Le +14]. The implanted submatrix (I, J) forms a specific selection of rows and columns chosen uniformly at random. For each row index i in $\{1, \dots, m\}$ and each column index j in $\{1, \dots, n\}$ of $\boldsymbol{M}$, a binary variable x_i and y_j, respectively, is sampled from a Bernoulli distribution $\mathrm{B}(p)$. The associated row or column is included in the submatrix (I, J) if $\mathrm{B}(p) = 1$. Hence, $\mathrm{I} = \{i \in \mathrm{R} \mid x_i \sim \mathrm{B}(p) = 1\}$ and $\mathrm{J} = \{j \in \mathrm{C} \mid y_j \sim \mathrm{B}(p) = 1\}$. Next, the full matrix $\boldsymbol{M}$ is generated according to two normal distributions:

- $\mathcal{N}(1, 1)$ whenever the particular entry belongs to the implanted submatrix,
- $\mathcal{N}(-3, 1)$ otherwise.

More formally, an instance is defined from parameters m, n, and p as:

$$\forall i \in \{1, \dots, m\}, j \in \{1, \dots, n\} : \tag{2.39}$$

$$x_i \sim \mathrm{B}(p) \ , \tag{2.40}$$

$$y_j \sim \mathrm{B}(p) \ , \tag{2.41}$$

$$\boldsymbol{M}_{i,j} \sim \begin{cases} \mathcal{N}(1, 1) \ , & \text{if} \quad x_i = 1 \land y_j = 1 \ , \\ \mathcal{N}(-3, 1) \ , & \text{otherwise} \ . \end{cases} \tag{2.42}$$

Such a generation protocol favors the occurrence of higher values in the implanted submatrix and lower values elsewhere. Yet, given the standard deviations chosen equal to 1, both ranges of values may overlap. Therefore, the implanted submatrix is not guaranteed to be an optimal solution to the maximal sum submatrix problem. This generation protocol looks however realistic to define a rectangular (and not necessarily contiguous) submatrix of interest in a larger matrix.

Problem instances are generated for various matrix sizes (m, n) and a varying parameter p. As p increases, the size of the implanted submatrix is expected to increase as well. In the gene expression analysis context, m can easily be two orders of magnitude larger than n and the submatrix of interest is typically small as compared to the full matrix. Such cases are included in the controlled experiments reported below but a larger spectrum of problem instances is also considered.

2.4.3 Gene expression data

The proposed case study concerns biomarker discovery for breast cancer subtypes using heterogeneous molecular data types. For a biological analysis and interpretation of the results, the reader is redirected to the work of Le Van et al. [Le +14]. The preprocessed data consists of a matrix of $m = 2,211$ rows and $n = 94$ columns.

Matrix entries are first transformed to discrete ranks along each row. Ranking allows to easily combine heterogeneous data with different scales. A given threshold $\theta \times n$ is then subtracted from each entry. As θ increases, the proportion of positive entries decreases and, consequently, a smaller submatrix of interest is expected to be found. Hence, the control parameter θ plays a similar role as the parameter p, defined in Section 2.4.2, but in an opposite way.

2.4.4 Results

Figure 2.1 presents the any-time profile on 50 synthetic data with 100 rows and $p = \{0.05, 0.3, 0.7\}$ for 20 columns (column 1) or 100 columns (column 2). The CP-LNS$_0$ method is clearly outperformed by the CP-LNS method when the number of columns increases.

These results clearly illustrate the benefits of CP-LNS over CP-LNS$_0$: while both have identical filtering, the latter uses a lot of reified constraints and more terms in the objective function which imply longer time spent to compute the fix-point in each node of the search tree.

Figure 2.2 presents the any-time profile on 50 synthetic data with 10,000 rows and $p = \{0.05, 0.3, 0.7\}$ for 100 columns (column 1) or 1,000 columns (column 2) and the any-time profile on breast cancer gene expression data with 2,211 rows, 94 columns and variable choices of θ (columns 3 and 4).

Results on synthetic data. The CP-LNS method is clearly outperformed by the two other methods. It can barely produce any solution within the allocated time budget. The best approach is CPGC followed by MILP. The reported curves are stopped whenever the proof of optimality is obtained or else the maximal running time is reached. Hence, CPGC also exhibits best results whenever proving optimality is possible in the allocated running time.

Results on gene expression data. Each curve corresponds here to the performance of an algorithm on a single instance, the one obtained for a specific choice of θ. On the whole spectrum of instances considered, the clear winner is CPGC. The most interesting instances are those for which $\theta \geq 0.9$ since such settings correspond to small submatrices which are more likely to illustrate an interesting biological pattern. In such cases, the best approaches are CPGC and CP-LNS.

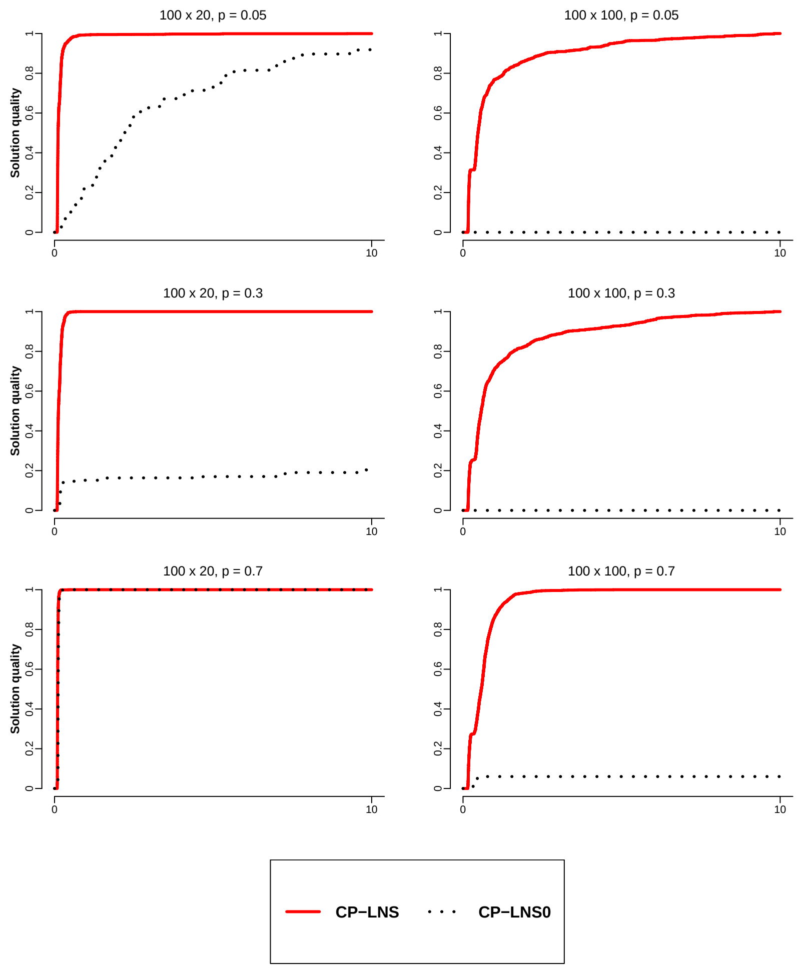


Figure 2.1: Any-time profiles of the constraint programming method with large neighborhood search by Le Van et al. [Le +14], called CP-LNS$_0$, and an adapted version of it, CP-LNS.

Reported curves correspond to the average solution quality over synthetic instances as a function of time, in seconds. Results are computed on 50 synthetic instances with 100 rows, 20 columns (left) or 100 columns (right), a variable p and a maximum budget of time of 10 seconds.

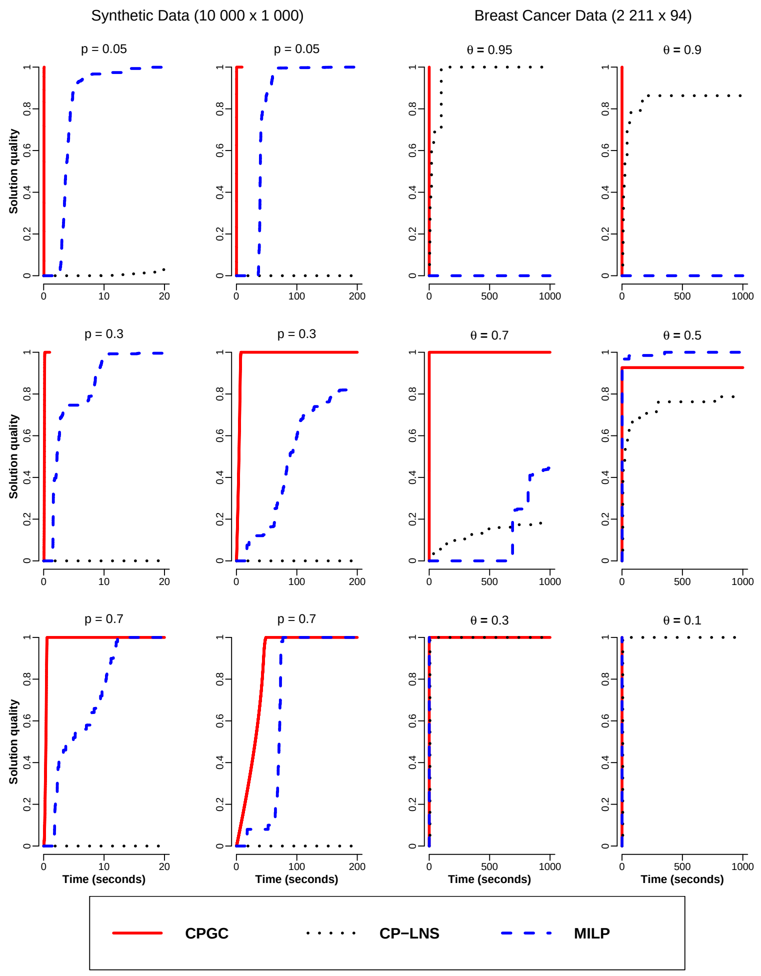


Figure 2.2: Any-time profiles of constraint programming with a global constraint (CPGC), the constraint programming method with large neighborhood search (CP-LNS) and a mixed integer linear programming method (MILP).

***Columns 1 and 2*: reported curves correspond to the average solution quality over all *synthetic instances* as a function of time, in seconds. Results are computed on** 50 ***synthetic instances* with** 10, 000 **rows,** 100 **(column 1) or** 1, 000 **(column 2) columns, a variable** p **and a maximum running time of** 20 **(column 1) or** 200 **(column 2) seconds.**

***Columns 3 and 4*: reported curves correspond to the solution quality over each *gene expression problem instance* obtained for a specific** θ **as a function of the time (in seconds). Results are computed on breast cancer *gene expression data* with** 2, 211 **rows,** 94 **columns and various** θ **values for a maximum CPU time of** 1, 000 **seconds.**

Influence of randomness on LNS. Each restart of the LNS allows exploring the neighborhood of the best solution found so far. However, the restart frequency influences the ability to find improving solutions in the neighborhood. In practice, the LNS restarts after a parametrized number of backtracks. There are two extreme cases:

1. restart on the first backtrack, which is equivalent to generating solutions without exploration of the solution neighborhood,
2. restart after an infinite number of backtracks, or no restart, which is somewhat equivalent to a complete search.

The best parameter value for LNS restart probably lies between these extreme cases and is influenced by the restart strategy and the problem to solve. These elements could be tuned to fit certain contexts or problems. However, the quality of the LNS restarts also depends on the randomness when fixing some of the variables.

We performed additional experiments to evaluate the influence of randomness in the restart strategy. We generated a set of instances, and each instance is solved ten times with LNS and different seeds. This provides ten different exploration LNS for each instance. We also considered 12 different parameter values for the number of backtracks to perform before restarting the LNS. We then compare the solution quality of each run with the best solution found overall for each instance. We considered instances as provided in the synthetic data. All runs have the same solution after the allocated time. We considered more difficult instances for which the difference between the implanted submatrix and background noise is smaller. We considered 10 synthetic $1,000 \times 200$ matrices with $p = 0.5$ and a background noise following the $\mathcal{N}(-1, 1)$ distribution and the implanted submatrix following the $\mathcal{N}(0, 1)$ distribution. FIGURE 2.3 reports the impact of the randomness and the influence of the number of backtracks before restarting the LNS. The randomness has a limited impact on the solution quality in these experiments. The influence of the number of backtracks is minor but exists. While there may be a trend, the optimal parameter value may change with the optimal solution to find. The actual expectation of the user should also influence the parameter. If the goal is to find a good solution within a short time, it would be best to increase the restarts. If the goal is to find the best solution within a long time, it would be worth spending some of the time in a larger exploration of the neighborhood.

2.4.4.1 Summary

As expected by the size of the search tree, CP-LNS is sensible to the size of the instance matrix producing barely no results on the larger synthetic

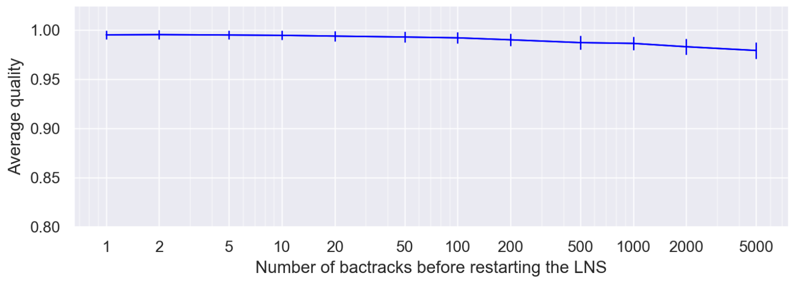


Figure 2.3: Solution quality found by CP-LNS as a function of the number of backtrack before restarting the LNS. The quality corresponds to the objective value of an instance divided by the maximal objective value found for that instance over all runs. The reported values correspond to the average quality of 10 instances and 10 repeats per instance.

instances within the time budget. On the opposite, CPGC achieves the best results. Indeed the model uses a dedicated global constraint with efficient filtering through computation of an upper bound and fast update of the lower bounds. The results of MILP are surprisingly good given its inability to express specialized constraints such as these of constraint programming. This is explained by the benefits of the linear programming relaxation to tighten the gap between the lower and the upper bounds. The current major issue is related to the "big M" approach that fails to guide the search in some settings on the gene expression data. When θ is smaller, the big M constant $\mathfrak{M}^-$ is tighter. As a consequence, the result of the LP relaxation as a higher chance to be a tighter bound. It follows a speed-up of the search as it implies a more efficient pruning of the tree.

2.5 Conclusions

The maximal sum submatrix problem consists in finding a rectangular submatrix in a large matrix whose sum is maximal. This problem is originally motivated, in the context of gene expression analysis, by the search of a subset of highly expressed genes in a specific subset, to be found, of relevant samples exhibiting such a pattern. A close variant of this problem, known as *maximal ranked tile* mining problem, has already been studied and tackled with constraint programming (CP) combined with large neighborhood search (LNS) [Le +14].

We present here key properties of the maximal sum submatrix problem to speed up the search for a solution. This results in an improved CP-LNS implementation. We also propose two new algorithms to solve this problem. Experiments reported both on synthetic data and the original gene expression data used by Le Van et al. [Le +14] illustrate the benefits of our proposed methods. In particular, a CP approach with a global constraint, CPGC, is the most effective one in a large spectrum of problem instances. Overall, the CPGC method is also best at proving optimality when such proof can be obtained within the allocated process time budget.

The second approach proposed here, called MILP, relies on mixed integer linear programming. It is arguably the simplest to formulate and to address with a standard solver for such problems. It is competitive with the other methods and largely outperforms CP-LNS as well in our controlled experiments. It exhibits however some performance degradation on some instances from gene expression data, most likely as a consequence of the specific relaxation it is based on.

The proposed experiments are only concerned with implementation efficiency. Le Van et al. [Le +14] illustrated the relevance to biological data studies of a special case of the MSS problem for which the matrix entries are discrete ranks. An actual evaluation of the maximal sum submatrix problem on biological data is proposed in Chapter 5.

Mining K disjoint submatrices of maximal sum | 3

There are many data-mining applications of the maximal sum submatrix (MSS) problem where one could benefit from finding more than one submatrix of maximal sum. Finding multiple gene subsets specific to subsets of samples is a common task in gene expression analysis. This chapter presents an extension to the MSS problem to find K different submatrices. The specificity of that extension is the definition of disjunction constraints that prevent overlap between submatrices. Optimization approaches relying on column generation, constraint programming and mixed integer linear programming to solve that problem are presented. Then, experiments on synthetic data are conducted. These experiments show the benefits of a hybrid column generation approach using constraint programming to generate columns over the MILP approach.

3.1 Introduction

The objective of the maximum weighted set of disjoint submatrices problem is to discover K disjoint submatrices that together cover the largest sum of entries of an input matrix. It has many practical data-mining applications, as the related biclustering problem, such as gene module discovery in bioinformatics. It relies on an explicit formulation of disjunction constraints: submatrices must not overlap. In other words, all matrix entries must be covered by *at most* one submatrix. The particular case of $K = 1$, called the maximal sum submatrix (MSS) problem, was successfully tackled with constraint programming (CP) in the previous chapter. Unfortunately, the case of $K > 1$ is more challenging to solve as the selection of rows cannot be decided in polynomial time solely from the selection of K sets of columns. It can be proved to be $\mathcal{NP}$-hard.

We introduce in this chapter an hybrid column generation approach using

CP to generate columns. It is compared to a standard mixed integer linear programming (MILP) through experiments on synthetic datasets. Overall, fast and valuable solutions are found by column generation while the MILP approach cannot handle a large number of variables and constraints.

3.1.1 Problem definition

The maximal sum submatrix problem, introduced in the Chapter 1, consists in finding a subset of rows and columns of an input matrix $\boldsymbol{M}$ that maximizes the sum of the covered entries.

In this chapter, we consider an extension to the identification of K submatrices. It relies on:

1. a modification of the objective function computed as the sum of the K submatrices weights, and
2. the explicit addition of disjunction constraints.

The disjunction constraints prevents the identification of K identical submatrices corresponding to the maximal sum submatrix. By allowing overlaps on the rows *or* the columns, but not both simultaneously, we avoid the identification of redundant submatrices. Moreover, the interpretability of the solution by a domain expert is eased. Such a solution is usually called *nonoverlapping nonexclusive nonexhaustive* in the biclustering context [MO04]:

- each matrix entry can be covered by at most one bicluster,
- every bicluster pair can share some row *or* some columns but not both,
- some rows and columns can be excluded from all biclusters.

From a biological viewpoint, the studied problem consists in finding multiple pairs of rows and columns such that:

- a gene should be specific enough to participate in at most one pathway,
- a sample should be only considered if it presents a (single) clear pattern of expression and not a combination of multiple highly expressed pathways.

Definition 5. **The Maximum Weighted Set of Disjoint Submatrices Problem**
Let $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ be a matrix of m rows and n columns. Let $\mathcal{K} = \{1, \ldots, K\}$ be the index-set for a target number K of submatrices to find. The maximum

weighted set of disjoint submatrices problem is to select a set of K submatrices $(\mathrm{I}^{1*}, \mathrm{J}^{1*}), \ldots, (\mathrm{I}^{K*}, \mathrm{J}^{K*})$ *such that:*

$$\left(\mathrm{I}^{1*}, \mathrm{J}^{1*}\right), \ldots, \left(\mathrm{I}^{K*}, \mathrm{J}^{K*}\right) = \underset{(\mathrm{I}^1,\mathrm{J}^1),\ldots,(\mathrm{I}^K,\mathrm{J}^K)}{\operatorname{argmax}} \sum_{k=1}^{K} \sum_{\substack{i \in \mathrm{I}^k \\ j \in \mathrm{J}^k}} \boldsymbol{M}_{i,j} \ , \tag{3.1}$$

$$\text{s.t.:} \quad \mathrm{I}^k \subseteq \mathrm{R} \ , \qquad \forall k \in \mathcal{K} \ , \tag{3.2}$$

$$\mathrm{J}^k \subseteq \mathrm{C} \ , \qquad \forall k \in \mathcal{K} \ , \tag{3.3}$$

$$\left(\mathrm{I}^k \times \mathrm{J}^k\right) \cap \left(\mathrm{I}^{k'} \times \mathrm{J}^{k'}\right) = \emptyset \ , \quad \forall k, k' \in \mathcal{K} : \ k \neq k' \ . \tag{3.4}$$

The maximization term in Equation (3.1) enforces that the sum of the covered entries is maximal. The disjunction constraints in Equation (3.4) enforce that each matrix entry is selected by at most one submatrix. Restricting to G (> 1) overlaps would result in $\lceil K/G \rceil$ groups of G identical submatrices. While any submatrix pair may share rows *or* columns, the constraint prevents any pair from sharing rows *and* columns simultaneously. Note that the specific submatrix ordering is irrelevant given the definition provided.

3.1.2 Applications

The maximum weighted set of disjoint submatrices (MWSDS) problem has many practical data-mining applications where one is interested in discovering K specific relations between two groups of variables. A few examples are provided hereafter.

- *In gene expression analysis.*

  The rows correspond to genes and columns to samples, and the value in $\boldsymbol{M}_{i,j}$ is the measurement of the expression of gene i in sample j. One is typically interested in finding a subset of genes that present high expression in a subset of the samples. It would indicate that a particular biological pathway made of these genes is active in these samples. Finding multiple pathways is a common task in gene expression analysis. Interpreting submatrices overlaps can be challenging in such a context. The disjunction constraint ensures that the role of a matrix entry $\boldsymbol{M}_{i,j}$ in a submatrix is clear: its contribution to the decision of including gene i and sample j in the submatrix is $\boldsymbol{M}_{i,j}$. It follows that the contribution of each gene, and each sample, in the definition of the pathway can be easily computed as the sum of matrix entries for each of the selected samples and genes respectively.

- *In migration data.*

  The value $\boldsymbol{M}_{i,j}$ represents the number of persons that moved from source location i to destination j. The goal is to find multiple groups of sources and destinations pairs characterized by large, conjoint migration flow. Using different source and destination locations may allow finding migration flows between locations based on some features such as richness or government policies. Overlaps on both the rows and columns would not necessarily be desirable. Indeed, the shared set of rows of overlapping submatrices would correspond to source location where migration is not highly specific to some destination. Similarly, the shared set of columns of overlapping submatrices would correspond to destinations that attract many immigrants from various origins.

- *In sports data.*

  A sports journalist could be interested in Olympic games to discover multiple pairs of (countries subsets, sports subsets) of strong performance. The matrix value $\boldsymbol{M}_{i,j}$ then represents the number of medals obtained by the country i in sport j. Similarly to migration data, the aim is to find specific countries with high performance in specific sports and conversely.

3.1.3 Related work

The work presented in this chapter extends the maximal sum submatrix problem to $K > 1$ by adding disjunction constraints and adapting the objective function.

The work related to the maximal sum submatrix problem is introduced in Section 1.4. Additional work is presented and notable differences with the maximum weighted set of disjoint submatrices problem are enlighted.

- *The minimum sum-of-squares clustering problem.*

  It involves the definition of non-overlapping sets of rows, or columns, covering all matrix entries. Although the problem differs, we use a similar approach as in [AHL12]: the combination of an integer linear programming (ILP) and delayed column generation.

- *The biclustering problem.*

There is a variety of biclustering problems. These problem are concerned with the discovery of homogeneous submatrices rather than maximizing the weight of covered entries.

- *The maximum subarray problem.*

  That problem is simpler than the maximum weighted set of disjoint submatrices as a single submatrix is required and it is constrained to be formed of contiguous subsets of rows and columns.

- *The maximal ranked tile mining problem and extensions.*

  It is a special case of the maximal sum submatrix problem for which matrix entries are discrete ranks. The diverse ranked tiling problem impose disjunction on the rows, while in the maximum weighted set of disjoint submatrices problem, disjunction is imposed on the matrix entries. In the ranked tiling problem, overlaps might exists even if they are penalized.

3.1.4 Contributions

The contributions of this chapter are the following.

1. The introduction of the maximum weighted set of disjoint submatrices (MWSDS) problem as a generalization of the maximal sum submatrix (MSS) problem.

2. A mathematical programming approach to solve the MWSDS problem.

3. The formulation of the MWSDS problem as an integer linear programming (ILP) relying on constraint programming to produce relevant variables.

4. An evaluation of the performance of these two alternatives and the benefit of the ILP+CP combination over a greedy approach on synthetic datasets.

3.2 Constraint programming approaches

The general constraint programming paradigm is introduced in Section 1.5.

3.2.1 Search space

The search space of the maximal sum submatrix problem can be limited to searching on a single dimension, as explained in SECTION 2.2 SEARCH SPACE. Indeed, the optimal subset of rows I^* can be found in polynomial time from a fixed selection of columns $\mathrm{J} \subseteq \mathrm{C}$:

$$\forall i \in \mathrm{R} : \sum_{j \in \mathrm{J}} \boldsymbol{M}_{i,j} > 0 \implies i \in \mathrm{I}^* \ .$$

Let us consider the MWSDS problem when all columns are fixed. This is formally stated in DEFINITION 6 as the MWSDS problem with fixed column selections.

DEFINITION 6. **The MWSDSP with fixed column selections**
The problem is a particular case of the maximum weighted set of disjoint submatrices problem presented in DEFINITION *(5). In this case, the selections of columns for each submatrices, $\mathrm{J}^1, \dots, \mathrm{J}^K$, are given. The problem is to define the rows of each submatrix:*

$$\mathrm{I}^{1*}, \dots, \mathrm{I}^{K*} = \underset{\mathrm{I}^1,\dots,\mathrm{I}^K}{\operatorname{argmax}} \ , \sum_{k=1}^{K} \sum_{\substack{r \in \mathrm{I}^k \\ c \in \mathrm{J}^k}} \boldsymbol{M}_{i,j} \tag{3.5}$$

$$\text{s.t.:} \quad \mathrm{I}^k \subseteq \mathrm{R} \ , \qquad \forall k \in \mathcal{K} \ , \tag{3.6}$$

$$\mathrm{J}^k \subseteq \mathrm{C} \ , \qquad \forall k \in \mathcal{K} \ , \tag{3.7}$$

$$\left(\mathrm{I}^k \times \mathrm{J}^k\right) \cap \left(\mathrm{I}^{k'} \times \mathrm{J}^{k'}\right) = \emptyset \ , \quad \forall k, k' \in \mathcal{K} : \ k \neq k' \ . \tag{3.8}$$

For $K > 1$, once the columns J^k of all the K submatrices are fixed, it remains to decide for each row i and each submatrix k whether i is to be selected ($i \in \mathrm{I}^k$) or not. These K decisions per row cannot be optimally taken in polynomial time, as stated in THEOREM (2). As a consequence, the search will have to assign both the row and column set variables, as opposed to the simpler $K = 1$ problem.

THEOREM 2.
The maximum weighted set of disjoint submatrices problem with fixed column selections is $\mathcal{NP}$-hard

Proof. We reduce the Maximum Weighted Independent Set (MWIS) problem to our problem. MWIS is $\mathcal{NP}$-hard (by immediate reduction from the Independent Set problem [GJ90]), and aims at finding, in a graph $G =< V, E >$ with weights w_v on each vertex $v \in V$, the set of vertices with the maximum sum such that they do not share edges in G. For simplicity, we represent edges and vertices as numbers: $V = \{1, \ldots, |V|\}$ and $E = \{1, \ldots, |E|\}$. We reduce an instance of the MWIS to an instance of the MWSDS problem with fixed column selections. We create a 1 by $(|V| + |E|)$ matrix M: $M_{1,i} = w_i$ if $i \in \{1, \ldots, |V|\}$, and $M_{1,i} = 0$ otherwise. The columns sets $C^1, \ldots, C^{|V|}$ are constructed as follows: $C^v = \{v\} \cup \{|V| + e \mid e \in E \wedge \text{edge } e \text{ has } v \text{ as origin or destination}\}$. Each vertex in the graph G is transformed in a submatrix. If the single row of matrix M is selected by a submatrix, then the vertex is included in the MWIS. The non-overlapping constraint of MWSDS problem forbids two adjacent vertices (i.e., submatrices) to both be included in the solution (constructing an independent set), due to the way the column selections $C^1, \ldots, C^{|V|}$ are constructed. Resolving the MWSDS problem then leads to the same optimal objective result as the original MWIS problem, and the selected rows R^v, $\forall v \in [1, \ldots, |V|]$, indicates, for each node v, if the node is inside the MWIS ($R^v = \{1\}$) or not ($R^v = \emptyset$). As computing the MWIS in general graphs is $\mathcal{NP}$-hard, and as the MWSDS problem with fixed column selections can encode the MWIS problem, we conclude that the MWSDS problem with fixed column selections is $\mathcal{NP}$-hard. □

3.2.2 Greedy baseline

A simple approach to solving the maximum weighted set of disjoint submatrices problem is to solve the maximal sum submatrix repeatedly using the CPGC implementation presented in Chapter 2. For each new maximal sum submatrix found, the corresponding values are replaced by $-\infty$, forbidding subsequent iterations from selecting these entries again.

Each iteration is performed until the first of these scenarios occurs:

1. optimality is proved;
2. absence of solution is proved;
3. at least one solution is found and a given time-out is reached.

3.2.3 Column generation

We propose a column generation (CG) approach [DDS06] to find solutions to the MWSDS problem. It relies on CP, introduced in Section 1.5, in an integer linear programming (ILP) setting. The CP part identifies candidate submatrices. The ILP efficiently combines submatrices and guides the CP part. ILP is equivalent to mixed integer linear programming, introduced in Section 2.3.3, except that all variables are required to be integer in the ILP.

Let us represent the given matrix $\boldsymbol{M}$ of $m \times n$ entries as the vector $\mathbf{V}$ of size v $(= m \times n)$ obtained by stacking the columns of the matrix $\boldsymbol{M}$. The maximum weighted set of disjoint submatrices problem is formulated using a $v \times 2^{m+n}$

binary matrix $\mathbf{B}$ representing all 2^{m+n} possible submatrices. Each column l of $\mathbf{B}$ corresponds to a submatrix l such that $\mathbf{B}_{i,l} = 1$ if and only if entry $\mathbf{V}_i$ is covered by the submatrix $l \in \mathcal{L}$ where $\mathcal{L} = \{1, \ldots, 2^{m+n}\}$ denotes all the possible submatrices. The weight w_l of submatrix l is the sum of entries it covers: $w_l = \sum_{i=1}^{v} \mathbf{V}_i \times \mathbf{B}_{i,l}$. Equations (3.1) and (3.4) can be formulated as an ILP:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{l \in \mathcal{L}} w_l \times x_l \ , && && (3.9a)\\
\text{s.t.} \quad & \sum_{l \in \mathcal{L}} \mathbf{B}_{i,l} \times x_l \leq 1 \ , && \forall i \in \{1, \ldots, v\} \ , && (3.9b)\\
& \sum_{l \in \mathcal{L}} x_l \leq K \ , && && (3.9c)\\
& x_l \in \{0, 1\} \ , && \forall l \in \mathcal{L} \ , && (3.9d)\\
& \mathbf{B}_{i,l} \in \{0, 1\} \ , && \forall i \in \{1, \ldots, v\}, \forall l \in \mathcal{L} \ . && (3.9e)
\end{aligned}
$$

The binary decision variable x_l encodes the selection of submatrix l. Equation (3.9b) ensures submatrices disjunction and Equation (3.9c) enforces the selection of at most K submatrices. Equation (3.9d) defines the integrality constraints on the variables x_l.

Defining the matrix $\mathbf{B}$ before solving the ILP is computationally not feasible, even for small input matrices $\boldsymbol{M}$. In subproblem solving, the master problem (or ILP), in Equations (3.9a)-(3.9d), is restricted to a subset $\mathcal{L}' \subseteq \mathcal{L}$ of submatrices effectively defining a restricted master problem (RMP). Iteratively, an RMP is solved, and one or multiple new submatrices (columns) are inserted in $\mathcal{L}'$, defining a new RMP. Submatrices (columns) are candidates for insertion to an RMP if its insertion can improve the objective function of the RMP.

To find such candidate submatrices, we define a linear programming relaxation of the restricted master problem (LP-RMP) which comes along the integrality constraints relaxation of the ILP in an LP, and the subsetting of $\mathcal{L}$. We use the dual of the LP-RMP to find submatrices with a positive reduced cost, given that the problem is a maximization problem. Such submatrix can improve the LP-RMP. If no such submatrix exists, the optimal solution to the LP-RMP is an optimal solution to the LP [Bar+98]. The dual of the LP-RMP is:

$$\text{minimize} \qquad \theta \times K + \sum_{i=1}^{v} \lambda_i \quad , \tag{3.10a}$$

$$\text{s.t.} \quad \theta + \sum_{i=1}^{v} \mathbf{B}_{i,l} \times \lambda_i \geq w_l \ , \quad \forall l \in \mathcal{L}' \ , \tag{3.10b}$$

$$\lambda_i \geq 0 \ , \quad \forall i \in \{1, \ldots, v\} \ , \tag{3.10c}$$

$$\theta \geq 0 \ . \tag{3.10d}$$

The dual values λ_i and θ corresponding to the primal constraints defined in Equations (3.9b) and (3.9c), respectively, are obtained by solving an LP-RMP. Each column x_l of the RMP is associated with a constraint, in Equation (3.10b), in the dual.

Finding a submatrix with a positive reduced cost is called pricing. Such a submatrix is defined as any submatrix $l \in \mathcal{L}$ for which

$$-\theta - \sum_{i=1}^{v} \mathbf{B}_{i,l} \times \lambda_i + w_l < 0 \ . \tag{3.11}$$

The LP-RMP is optimal if the pricing problem has no solution. Moreover, if the LP-RMP (being optimal) and the RMP have the same objective value, then the solution to the ILP is optimal.

The pricing problem can be reformulated as:

$$\sum_{i=1}^{v} \left[\mathbf{B}_{i,l} \times (\mathbf{V}_i - \lambda_i)\right] > \theta \ . \tag{3.12}$$

This problem amounts to identifying a submatrix ($\mathbf{B}_l$) in the input matrix ($\mathbf{V}$) modified by the λ_i values such that its weight is larger than some θ. It is equivalently reformulated as a maximal sum submatrix problem:

$$(\mathrm{I}^*, \mathrm{J}^*) = \underset{\substack{\mathrm{I} \subseteq \mathrm{R} \\ \mathrm{J} \subseteq \mathrm{C}}}{\operatorname{argmax}} \sum_{i \in \mathrm{I}, j \in \mathrm{J}} \boldsymbol{M}_{i,j} \ , \tag{3.13}$$

$$\text{with} \quad \boldsymbol{M}_{i,j} = \mathbf{V}_{i \times m + j} - \lambda_{i \times m + j} - \theta \ , \tag{3.14}$$

$$\mathrm{R} = \{1, \ldots, m\} \ , \tag{3.15}$$

$$\mathrm{C} = \{1, \ldots, n\} \ , \tag{3.16}$$

and submatrix $(\mathrm{I}^*, \mathrm{J}^*)$ is transformed into $\mathbf{B}_l$.

Such pricing problem can then be solved using an exact CP or mixed integer linear programming implementation. An optimal solution is associated with maximal reduced cost. Solving to optimality is not a trivial task, however, as the problem is $\mathcal{NP}$-hard (see Theorem 1). Alternatively, any submatrix with positive reduced cost, or positive weight, is a (possibly suboptimal) solution to the pricing problem. The pricing can then be solved using the CPGC implementation presented in CHAPTER 2 or even with the greedy approach presented in the previous section.

The greedy approach produces up to K solutions to the pricing problem, whereas the CPGC approach produces a single solution. Preliminary experiments showed that using the greedy approach to find solutions to the pricing problem slightly improves the column generation approach as compared to CPGC. In practice, we use the greedy approach to solve each pricing problem.

3.2.3.1 Important considerations

Implementation details may have an important role in the effectiveness of the approach. Such details are presented next.

Redundant constraints. To maximize the information given by the dual values, we avoid having redundant constraints, notably the constraints in EQUATION (3.9b). For example, if two submatrices overlap on more than one cell, we enforce only one constraint representing all the overlapping cells. Precisely, constraint in EQUATION (3.9b) is replaced by the following:

$$\sum_{l \in S} x_l \leq 1 \ , \qquad \forall S \in \Big\{ \{ l \in \mathcal{L} \mid \mathbf{B}_{i,l} = 1 \} \ \Big| \ i \in \{1, \ldots, v\} \Big\}. \tag{3.17}$$

That is, we enforce one non-overlap constraint per group of entries sharing the same intersecting submatrices, or an *overlapping group*. EQUATION (3.17) uses the set notation to implicitly remove duplicates. We then redistribute the dual value of the constraint equally (we divide it by the number of entries) over all the entries in this overlapping group. This allows the method to avoid a pitfall of most solvers: when facing multiple equivalent constraint, only one will be *tight*, i.e. having a non-zero dual value. Redistributing the duals on all the entries in an overlapping group allows the subproblem solver to find more interesting submatrices.

Linear relaxation. The LP-RMP does not necessarily provide a binary decision on the submatrix selection. To effectively identify a solution to the original MWSDS problem, the RMP is solved for any solution to the LP-RMP. Observe that the objective value of the LP-RMP is an upper bound to the objective value of the RMP. All experiments present the results of the RMP solution.

A greedy hot-start. The subset L' defining the first RMP to solve is obtained using the greedy approach searching for K submatrices. This serves as a `greedy hot-start` for the column generation approach.

Given the non-trivial pricing problem, there is no guarantee that the greedy subroutine identifies an optimal solution to the pricing problem. While it would be possible to use a branch-and-price algorithm [Sav97], it would be non-trivial to solve the pricing problem to optimality. The running time needed to solve the LP-RMP to optimality (i.e. to the point where no new submatrix with positive reduced cost exists) is already quite high, as shown in the experiment section below.

Number and weight of the submatrices. Guidance on the search for better submatrices requires many submatrices in the RMP with large weight. Moreover, the greedy subroutine may identify many solutions (i.e. submatrices) to the pricing problem. As the number of submatrices to find increases, the weight of these submatrices likely decreases. It is then more useful to seek multiple submatrices later in the column generation process. As a consequence, at iteration p of the column generation, up to p solutions, or submatrices, to the pricing problem are identified and are inserted in the RMP.

3.2.4 Mixed Integer Linear Programming

We propose a mixed integer linear programming (MILP) model with analogies to the "big M" formulation of the maximal sum submatrix problem presented in Section 2.3.3.2. The model uses the binary variables $\mathbf{R}_i{}^k$ and $\mathbf{C}_j{}^k$ to represent the selection of row i and column j in the submatrix k. These decision variables are used to compute the contribution $r_i{}^{k+}$ of the row i to the submatrix k. The sum of the row contributions is the objective function to be maximized. The model presented below is based on a "big M" formulation of the MWSDS where:

$$\forall i \in \mathrm{R} : \ \mathfrak{M}_i{}^+ = \sum_{j \in \mathrm{C}} \max(0, \boldsymbol{M}_{i,j}) \ , \tag{3.18}$$

$$\forall i \in \mathrm{R} : \ \mathfrak{M}_i{}^- = \sum_{j \in \mathrm{C}} \min(0, \boldsymbol{M}_{i,j}) \ , \tag{3.19}$$

and $\mathfrak{M}_i{}^+$ and $\mathfrak{M}_i{}^-$ are the upper bound and lower bound, respectively, on the sum of entries covered by row i. The MILP model is formulated as follows:

$$\begin{aligned}
\text{maximize} \quad & \sum_{\substack{i\in\mathrm{R}\\ k\in\mathcal{K}}} r_i{}^{k+} \ , && && \text{(3.20a)}\\
\text{s.t.:} \quad r_i{}^{k+} \ \leq\ & \sum_{j\in\mathrm{C}} \left(\boldsymbol{M}_{i,j} \times \mathrm{C}_j{}^k\right) && && \\
& +(\mathrm{R}_i{}^k - 1) \times \mathfrak{M}_i{}^- \ , && \forall i \in \mathrm{R}, k \in \mathcal{K} \ , && \text{(3.20b)}\\
r_i{}^{k+} \ \leq\ & \mathfrak{M}_i{}^+ \times \mathrm{R}_i{}^k \ , && \forall i \in \mathrm{R}, k \in \mathcal{K} \ , && \text{(3.20c)}\\
2 \times v_{i,j}^k \ \leq\ & \mathrm{R}_i{}^k + \mathrm{C}_j{}^k \ , && \forall i \in \mathrm{R}, j \in \mathrm{C}, k \in \mathcal{K} \ , && \text{(3.20d)}\\
\mathrm{R}_i{}^k + \mathrm{C}_j{}^k \ \leq\ & 1 + v_{i,j}^k \ , && \forall i \in \mathrm{R}, j \in \mathrm{C}, k \in \mathcal{K} \ , && \text{(3.20e)}\\
\sum_{k\in\mathcal{K}} v_{i,j}^k \ \leq\ & 1 \ , && \forall i \in \mathrm{R}, j \in \mathrm{C} \ . && \text{(3.20f)}
\end{aligned}$$

Constraints in Equations (3.20b) and (3.20c) ensure that the row contribution $r_i{}^{k+}$ is computed correctly. If $\mathrm{R}_i{}^k = 0$, constraint in Equation (3.20c) ensures the row contribution is zero, with the right hand side of constraint in Equation (3.20b) being always positive. Otherwise ($\mathrm{R}_i{}^k = 1$), constraints in Equations (3.20b) and (3.20c) ensure

$$r_i{}^{k+} = \sum_{j\in\mathrm{C}} \left(\boldsymbol{M}_{i,j} \times \mathrm{C}_j{}^k\right) \ ,$$

thus computing the effective value of the contribution.

Equations (3.20d) and (3.20e) linearize

$$v_{i,j}^k = \mathrm{R}_i{}^k \times \mathrm{C}_j{}^k \ .$$

The binary variable $v_{i,j}^k$ indicates if cell (i,j) is selected by submatrix k and ensures submatrices disjunction through constraint in Equation (3.20f).

This model is plagued by the number of variables and constraints which are both in $O(m \times n \times K)$, mainly due to the non-overlap constraints.

3.3 Experiments

This section describes experiments conducted to assess the performance of the proposed algorithms and to provide guidance on the selection of the appropriate solution.

3.3.1 Evaluation

Given enough time and memory, both the column generation (CG) approach and the MILP approach converge to the optimal solution. Therefore comparing performance solely on the objective value of an approach is irrelevant. As a consequence, CG and MILP approaches are evaluated and compared given a budget of time, the time-out $t^{\max}$, on synthetic datasets with implanted submatrices using any-time profiles. The definition of an any-time profile is provided in Section 2.4.1.1, Definition 4.

All experiments are performed using Java 1.8.0 on an AMD Bulldozer clocked at 2.1 GHz; one core and 6 GB of RAM per instance and a time-out $t^{\max}$ of 2 hours. MILP and CG approaches rely on Gurobi 8.1.0 [Gur18]. The greedy hot-start of the CG process is given 5 minutes evenly split between each of its K iterations of solving a maximal sum submatrix problem. Solutions to the MSS problem are carried out on OscaR [Osc12] using a constraint programming approach relying on a global constraint, called CPGC. The algorithm CPGC is provided in Section 2.3.2.3. It is a depth-first-search approach composed of major CP ingredients:

- filtering rules,
- bounding procedure,
- dominance rules,
- variable-value heuristic.

3.3.2 Datasets

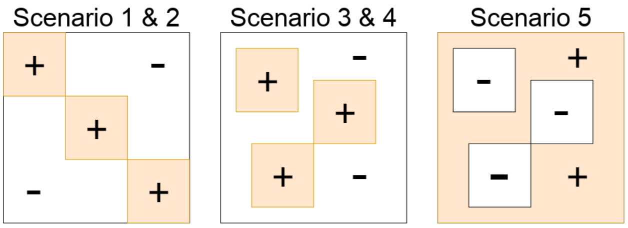
Datasets are generated by implanting K submatrices, called + entries, on a background noise, called - entries. In a first dataset, we consider alternative dispositions of + and - entries drawn from different distributions. Each combination defines a scenario presented in Figures 3.1a to 3.1b. For each scenario, 14 different matrices are generated according to different input matrix size and number of implanted submatrices, as presented in Figure 3.1c.

A total of 70 instances is generated such that the hot-start is bound to find suboptimal solutions, giving very little information to the CG method. The benefit of CG is evaluated relatively to the suboptimal hot-start solution through the objective value improvement.

3.3.3 Results

Figure 3.2a presents the any-time profile of each method for the first dataset. It clearly illustrates that column generation can escape the suboptimal regions of the search space trapping the hot-start. Given roughly 25 times larger time-out than the suboptimal hot-start, MILP is outperformed by both the greedy hot-start and the column generation approach.

Local optimums (trapping the hot-start) are provided as starting solutions for CG. Such local optimum can be found before the given time-out. The



(a) Layout.

Scenario	+ entries	- entries
1 and 3	$K+1$	-1
2 and 4	$\sim \mathcal{N}(K+1,\ 1)$	$\sim \mathcal{N}(-1,\ 0.8)$
5	$\sim \mathcal{N}(2,\ 2)$	$\sim \mathcal{N}(-2,\ 1)$

(b) Generative distributions.

$m \times n$	$K=2$	$K=5$	$K=8$	$K=10$	$K=20$
50×50	$s=s_1$	$s=s_1$			
100×100	$s=s_1$	$s=s_1$		$s=s_1$	
200×200	$s=s_1$	$s=s_1$	$s=s_1$	$s=s_1$	$s=s_1$
500×500	$s=s_1$	$s=s_1$		$s=s_1$	$s=s_1$

(c) Parameters for the first dataset.

$m \times n$	$K=2$	$K=5$	$K=10$
400×100	$s=s_2$	$s=s_2$	$s=s_2$
320×125	$s=s_2$	$s=s_2$	$s=s_2$
200×200	$s=s_2$	$s=s_2$	$s=s_2$

(d) Parameters for the second dataset.

Figure 3.1: Dataset construction. 3.1a and 3.1b: layout and generative distributions of the implanted + and - entries, respectively. 3.1c and 3.1d: Parameters considered in the generation of the first dataset, with $s_1 = \{1.0\}$, and the second dataset, with $s_2 = \{0.05, 0.01, 0.2, 0.5\}$, respectively. Implanted submatrices are of size $(\frac{m \times s}{K}; \frac{n \times s}{K})$.

shift between hot-start and CG curves in the first 300 seconds is explained by the fact that CG can refine solutions as soon as the hot-start subroutine is completed.

In the second dataset, 720 instances are generated according to the layout of scenarios 3 and 4 from FIGURE 3.1a. It differs, however, by the size of the input matrix, the number, and size of implanted submatrices. More importantly, values are drawn from different distributions:

- - entries $\sim \mathcal{N}(-1, 1)$;
- + entries $\sim \mathcal{N}(1, 0.5)$.

The definition of closer distribution averages is considered as more realistic, for gene expression analysis, than the very different distributions used in the scenarios of the first dataset. The generation script is available on Zenodo [Bra+19a].

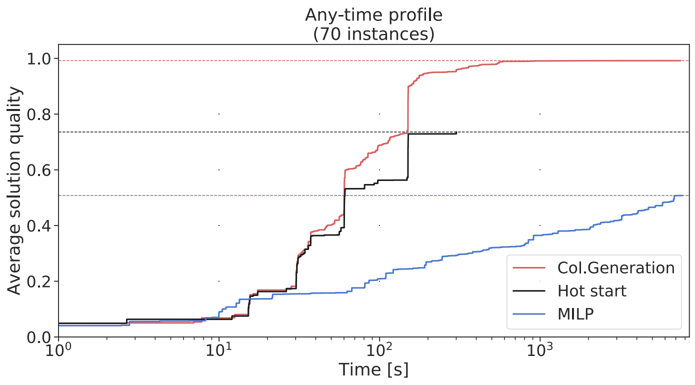
FIGURE 3.2b presents the any-time profile of CG and MILP on the second dataset. Whereas the average solution quality of CG and MILP should rise to 1, given enough time, it is clear that CG is significantly faster than MILP, with higher solution quality within reasonable time limits, even for small instance matrices. The poor performance of MILP are explained by the number of variables and constraints required to model the problem: MILP obtains satisfactory results for the smaller problems, with $K = 2$, only. FIGURE 3.2c presents the any-time profile of CG and MILP on the second dataset while only considering the instances with two implanted submatrices. The hot-start rarely ends before the allocated 5 minutes, explaining the near-perfect overlap between hot-start and CG curves.

3.4 Conclusions and perspectives

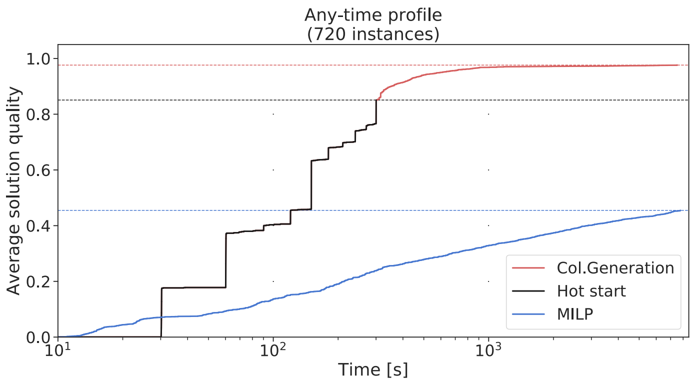
We present a new optimization problem, called the maximum weighted set of disjoint submatrices (MWSDS) problem along with two methods to solve it. One is based on mathematical programming, the other on constraint programming.

Our main contribution, the column generation (CG) method for the MWSDS problem, finds new candidate submatrices using dual variables of a linear relaxation of the submatrix selection problem. Experiments on synthetic datasets indicate that CG finds better solutions than the mixed integer linear programming (MILP) approach.

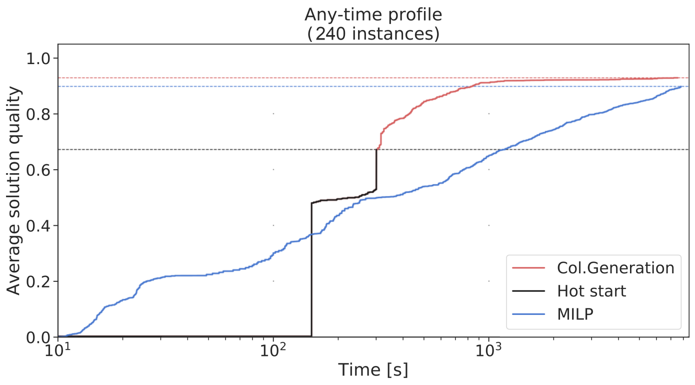
The performance of the CG can be further improved by complementing the exploration with a branch-and-price algorithm [Sav97]. Such improvement is non-trivial, however: the time taken to solve the underlying LP problem is already quite long but is nonetheless an attractive direction for future work.



(a) Averaged results on the first dataset.



(b) Averageg results on the second dataset.



(c) Subset of the second dataset with $K = 2$.

Figure 3.2: Comparison of the different methods proposed to solve the maximum weighted set of disjoint submatrices problem. Each graph presents any-time profiles as described in Definition 4, Section 2.4.1.1. For each instance, the time-out is fixed at 2 hours. The hot-start time-out is set to 5 minutes. Col.Generation starts as soon as the hot-starts is completed.

Experiments proposed here are only concerned with implementation efficiency. Chapter 5 consists of an evaluation of the problem of finding K submatrices of maximal sum with overlap. Biological evaluations of the proposed implementations should be performed. Moreover, results should be compared to standard biclustering algorithms to evaluate the relevance of the maximum weighted set of disjoint submatrices problem as compared to the maximal sum submatrix problem and common biclustering problems.

Mining k overlapping submatrices of maximal sum

4

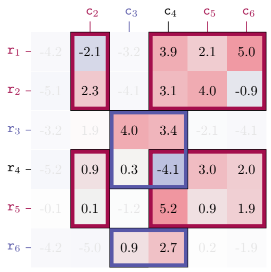
The previous chapter presents an extension to the maximal sum submatrix problem to find K submatrices that do not overlap. In many applications, overlaps might be desirable. This chapter presents an extension to the maximal sum submatrix problem to find K different submatrices that might overlap. The specificity ot that extension is the objective function designed to avoid finding identical submatrices. A complete constraint programming approach to solve that problem is proposed. Then, experiments on synthetic data and real datasets are conducted. These experiments provide evidences of the practicality of the approach both in terms of computational time and quality of the solutions discovered.

4.1 Introduction

A natural extension to the maximal sum submatrix (MSS) problem is to identify K submatrices. In the previous chapter, disjunction constraints are stated to prevent a matrix entry from being selected more than once in the final solution. An alternative approach to seek for multiple non-repetitive submatrices is to modify the objective function.

4.1.1 Problem definition

An example matrix and two submatrices are presented in FIGURE 4.1. The maximal sum submatrix, of weight 27.3, is illustrated with a dark red frame. Let us assume that one seeks for $K = 2$ submatrices that together maximize the sum of covered entries. It is clear from that example that the best solution is to select the maximal sum submatrix twice: $27.3 \times 2 = 54.6$. Such a solution does not provide any additional information. We therefore need to define a new objective function to prevent the identification of redundant submatrices.


c2 c3 c4 c5 c6
r1 -4.2 -2.1 -3.2 3.9 2.1 5.0
r2 -5.1 2.3 -4.1 3.1 4.0 -0.9
r3 -3.2 1.9 4.0 3.4 -2.1 -4.1
r4 -5.2 0.9 0.3 -4.1 3.0 2.0
r5 -0.1 0.1 -1.2 5.2 0.9 1.9
r6 -4.2 -5.0 0.9 2.7 0.2 -1.9

The submatrix framed in dark red, with rows $\{r_1, r_2, r_4, r_5\}$ and columns $\{c_2, c_4, c_5, c_6\}$, is of maximal sum, with weight equal to 27.3. The submatrix framed in dark blue, with rows $\{r_3, r_4\}$ and columns $\{c_3, c_4, c_6\}$, weights 7.2. The combination of these two submatrices defines an optimal solution to the MWSC problem with an objective value of 38.6.

Figure 4.1: Example of matrix and associated submatrices of maximal sum.

The objective of the maximum weighted submatrix coverage (MWSC) problem is to discover K submatrices that together cover the largest sum of entries of the input matrix. The problem is formally stated in DEFINITION 7 and an example is provided in FIGURE 4.1.

DEFINITION 7. **The Maximum Weighted Submatrix Coverage Problem**
Let $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ be a matrix of m rows and n columns. Let $\mathcal{K} = \{1, \ldots, K\}$ be the index-set for a target number K of submatrices to find. The maximum weighted submatrix coverage problem is to select a set of K submatrices $\left(\mathrm{I}^{1*}, \mathrm{J}^{1*}\right), \ldots, \left(\mathrm{I}^{K*}, \mathrm{J}^{K*}\right)$ such that the sum of the matrix entries covered by at least one submatrix is maximal:

$$\left(\mathrm{I}^{1*}, \mathrm{J}^{1*}\right), \ldots, \left(\mathrm{I}^{K*}, \mathrm{J}^{K*}\right) = \underset{\left(\mathrm{I}^{1},\mathrm{J}^{1}\right),\ldots,\left(\mathrm{I}^{K},\mathrm{J}^{K}\right)}{\operatorname{argmax}} \sum_{\substack{i \in \mathrm{I} \\ j \in \mathrm{J}}} \boldsymbol{M}_{i,j} \times \mathbb{1}_{cover}\left((i, j)\right) \quad , \tag{4.1}$$

where cover is the set of all entries covered by a submatrix,

$$cover = \bigcup_{k \in \mathcal{K}} \mathbf{R}^k \times \mathbf{C}^k \quad , \quad with \;\; \mathbf{R}^k \times \mathbf{C}^k = \{(i, j) \mid i \in \mathbf{R}^k \;\wedge\; j \in \mathbf{C}^k\} \quad , \tag{4.2}$$

and $\mathbb{1}_{cover}$ is the indicator function

$$\mathbb{1}_{cover}\left((i, j)\right) = \begin{cases} 1 & \textit{if } (i, j) \in cover \;, \\ 0 & \textit{if } (i, j) \notin cover \;. \end{cases} \tag{4.3}$$

4.1.2 Applications

The maximum weighted submatrix coverage problem has similar applications to the maximum weighted set of disjoint submatrices problem presented in the previous chapter, in SECTION 3.1.2. Submatrices overlap appear as a reasonnable assumption in many cases. Consider gene expression analysis. It is likely that, in real data, some rows or columns do not belong to any submatrix at all and that the submatrices overlap in some places. However, overlap can be challenging from the point of view of interpretability:

- For a gene participating in multiple pathways (submatrices), what is the actual influence of the gene on each pathway?
- For a matrix entry covered by multiple submatrices, what is its influence on each submatrix?
- For a particular submatrix overlapping with others, would it still be relevant if other submatrices were discarded?
- ...

4.1.3 Related work

The work related to the maximal sum submatrix problem is introduced in Section 1.4. Notable differences with the maximum weighted set of disjoint submatrices problem are enlighted.

- *The biclustering problem [Har72; MO04; PGA15; PC17a].*

  There is a variety of biclustering problems. These problem are concerned with the discovery of homogeneous submatrices rather than maximizing the sum of the covered entries. Common approaches are heuristic based and greedily selects the next bicluster after randomization of entries covered by the previously discovered submatrices.

- *The maximum subarray problem [Ben84].*

  That problem is simpler than the maximum weighted set of disjoint submatrices as a single submatrix is required and it is constrained to be formed of contiguous subsets of rows and columns.

- *The ranked tiling [Le +14].*

  It is a special case of the maximum weighted set of disjoint submatrices problem for which matrix entries are discrete ranks. In the ranked tiling problem, overlaps might exists even if they are penalized. In the maximum weighted set of disjoint submatrices problem, there is an implicit penalty to enforce that each matrix entry contributes only once to the objective value.

4.1.4 Contributions

The contributions of this chapter are the following.

- The introduction of the maximum weighted submatrix coverage (MWSC) problem as a generalization of the maximal sum submatrix problem.
- A constraint programming (CP) approach for solving the maximum weighted submatrix coverage problem including
  - filtering,
  - bounding,
  - dominance rules,
  - a variable-value heuristic,
  - a large neighborhood search.
- A comparative evaluation, on synthetic and real datasets, of the performance of
  - the CP approach,
  - a greedy baseline approach, using the maximal sum submatrix problem as subroutine, and
  - two mathematical programming models.

4.2 Constraint programming approach

The general constraint programming paradigm is introduced in Section 1.5.

4.2.1 Notations

We model a submatrix using the set variables $\mathcal{R}$ and $\mathcal{C}$ for the rows and columns, respectively. The domain of a set variable $\mathcal{S}$ is the set of all the (sub)sets of $\mathcal{S}$. That domain is approximated by a closed interval denoted $\left[\mathcal{S}^{\in}, \mathcal{S}^{\in} \cup \mathcal{S}^{?}\right]$, in CP, by means of the set domain bounds representation [Ger97]. $\mathcal{S}^{\in}$ are the mandatory elements and $\mathcal{S}^{?}$ are the possible additional ones, with $\mathcal{S}^{\in} \cap \mathcal{S}^{?} = \emptyset$. Such an interval represents all the sets in between those two bounds according to the inclusion relation $\{\mathcal{S} \mid \mathcal{S}^{\in} \subseteq \mathcal{S} \subseteq \left(\mathcal{S}^{\in} \cup \mathcal{S}^{?}\right)\}$. A set variable is bound whenever it contains a single set in its domain. This situation happens when set interval bounds are equal, or equivalently, the possible set is empty: $\mathcal{S}^{?} = \emptyset$.

For a set variable $\mathcal{S}$, the update operations of the domain are:

- The *inclusion* of an item j in the mandatory set $\mathcal{S}^{\in}$:

- denoted require$(j, \mathcal{S})$,
- implies that $\mathcal{S}^{\in} \leftarrow \mathcal{S}^{\in} \cup \{j\}$ and $\mathcal{S}^{?} \leftarrow \mathcal{S}^{?} \setminus \{j\}$.

■ The *exclusion* of an item j from the possible set $\mathcal{S}^{?}$:

- denoted exclude$(j, \mathcal{S})$,
- implies that $\mathcal{S}^{?} \leftarrow \mathcal{S}^{?} \setminus \{j\}$, and $j \notin \mathcal{S}^{\in}$.

For each submatrix $k \in \mathcal{K} = \{1, \dots, K\}$, a set variable $\mathcal{R}^k$ and a set variable $\mathcal{C}^k$ are introduced to represent the rows and columns of a submatrix k. Then, $\mathcal{R}^{k,\in}$ denotes the set of mandatory rows in submatrix k.

We define $\mathcal{R}^{k,\in}_{+j}$ and $\mathcal{R}^{k,\in}_{-j}$ as the subsets of $\mathcal{R}^{k,\in}$ whose matrix value in column j is positive and strictly negative, respectively. Similar notations holds for the columns:

$$\mathcal{R}^{k,\in}_{+j} = \{i \in \mathcal{R}^{k,\in} \mid \boldsymbol{M}_{i,j} \geq 0\} \ , \qquad \mathcal{C}^{k,\in}_{+i} = \{j \in \mathcal{C}^{k,\in} \mid \boldsymbol{M}_{i,j} \geq 0\} \ , \tag{4.4}$$
$$\mathcal{R}^{k,\in}_{-j} = \{i \in \mathcal{R}^{k,\in} \mid \boldsymbol{M}_{i,j} < 0\} \ , \qquad \mathcal{C}^{k,\in}_{-i} = \{j \in \mathcal{C}^{k,\in} \mid \boldsymbol{M}_{i,j} < 0\} \ . \tag{4.5}$$

Similar notations holds for $\mathcal{R}^{k,?}$ and for $\mathcal{C}^{k,?}$:

$$\mathcal{R}^{k,?}_{+j} = \{i \in \mathcal{R}^{k,?} \mid \boldsymbol{M}_{i,j} \geq 0\} \ , \qquad \mathcal{C}^{k,?}_{+i} = \{j \in \mathcal{C}^{k,?} \mid \boldsymbol{M}_{i,j} \geq 0\} \ , \tag{4.6}$$
$$\mathcal{R}^{k,?}_{-j} = \{i \in \mathcal{R}^{k,?} \mid \boldsymbol{M}_{i,j} < 0\} \ , \qquad \mathcal{C}^{k,?}_{-i} = \{j \in \mathcal{C}^{k,?} \mid \boldsymbol{M}_{i,j} < 0\} \ . \tag{4.7}$$

The sum of the elements in a given row i and in a column set $\mathcal{S}$ is noted as $\underset{\text{row } i}{\text{sum}}(\mathcal{S})$. Similarly, the sum of elements in column j and row set $\mathcal{S}$ is:

$$\underset{\text{row } i}{\text{sum}}(\mathcal{S}) = \sum_{j \in \mathcal{S}} \boldsymbol{M}_{i,j} \ , \qquad \underset{\text{col } j}{\text{sum}}(\mathcal{S}) = \sum_{i \in \mathcal{S}} \boldsymbol{M}_{i,j} \ . \tag{4.8}$$

The set of matrix entries selected by at least one submatrix is denoted $cover^{\in}$. The set of matrix entries excluded by all submatrices is denoted $cover^{\notin}$:

$$cover^{\in} = \{(i,j) \mid \exists k : i \in \mathcal{R}^{k,\in} \wedge j \in \mathcal{C}^{k,\in}\} \tag{4.9}$$
$$cover^{\notin} = \{(i,j) \mid \forall k : i \notin (\mathcal{R}^{k,\in} \cup \mathcal{R}^{k,?}) \vee j \notin (\mathcal{C}^{k,\in} \cup \mathcal{C}^{k,?})\} \tag{4.10}$$

The CP resolution is made via a depth-first-search exploration. The following subsections discuss the search space, sketch the algorithm and its key components.

4.2.2 Search Space

As explained in the Section 3.2.1 of the previous chapter, the search space of the MWSC problem with $K = 1$ can be limited to searching on a single dimension, for instance C^1. Indeed, the variable $\mathcal{R}^1$ can be fixed optimally in polynomial time by a simple inspection argument:

$$\forall i \in \mathcal{R}^{1,?} : \underset{\text{row } i}{\text{sum}}(C^1) > 0 \implies i \in \mathcal{R}^{1,\in} \quad . \tag{4.11}$$

For $K > 1$, once all the columns set variables are fixed ($C^{k,?} = \emptyset$ for all $k \in \mathcal{K}$) it remains to decide for each row i and each submatrix k whether i should be part of $\mathcal{R}^k$ or not. Those K decisions per row does not enjoy the monotonicity or the anti-monotonicity properties as illustrated on the next example.

Example 1.
Let us consider a 1×3 *input matrix* $\mathbf{M}$*:*

$$\mathbf{M} = \begin{bmatrix} 2 & 2 & -3 \end{bmatrix} \quad . \tag{4.12}$$

Let us consider $K = 2$ *with column selection* $C^1 = \{1, 3\}$, $C^2 = \{2, 3\}$*. The sum of entries covered by selecting the row in both* $\mathcal{R}^1$ *and* $\mathcal{R}^2$ *is negative* (-1) *for each submatrix individually. But since weights of covered elements count only once, the value* -3 *is added only once and the objective value obtained is* 1*. Now consider the matrix* $\mathbf{M}$*:*

$$\mathbf{M} = \begin{bmatrix} -2 & -2 & 3 \end{bmatrix} \quad . \tag{4.13}$$

Individually for each submatrix, the sum of entries covered by selecting the row in both $\mathcal{R}^1$ *and* $\mathcal{R}^2$ *is positive* (1)*. But since weights of covered elements count only once, the value* 3 *is added only once and the final objective value is* -1*.*

Actually, those K decisions per row cannot be optimally taken in polynomial time anymore as stated in Theorem 3. As a consequence, the CP search will have to branch both on the rows and columns variables rather than branching on the columns only.

Theorem 3.
Fixing optimally the row set variables $\mathcal{R}^k$ *for all* $k \in \mathcal{K}$*, given fixed column set variables* C^k *for all* $k \in \mathcal{K}$*, is* $\mathcal{NP}$*-hard.*

Proof. We reduce the $\mathcal{NP}$-hard Set Cover Problem [Kar72] to our problem: Given a universe $U = \{1, \cdots, n\}$ and a set $\{S_1, \cdots, S_K\}$ of K subsets of U, the Set Cover Problem is to find the minimum number of sets such that their union covers the universe. We construct a matrix with a single row and $n + K$ columns. The unique row values of this matrix are given by the regular expression $[K+1]\{n\}[-1]\{K\}$ (value $K+1$ repeated n times followed by -1 repeated K times). The column variables are fixed to $C^k = S_k \cup \{n+k\}$. In this reduction, S_k is selected if and only if $\mathcal{R}^k = \{1\}$ for every set k. A first observation is that any optimal solution covers the universe otherwise it could be improved by K by selecting any additional set that contains an uncovered element. The optimal objective function can thus be written as $N \cdot (K+1) - |\{k \mid \mathcal{R}^k = \{1\}\}|$. As $N \cdot (K+1)$ is fixed, maximizing this expression amounts at minimizing $|\{k \mid \mathcal{R}^k = \{1\}\}|$ which is exactly the set cover objective. □

4.2.3 Resolution via Depth-First-Search

The CP resolution through DFS exploration is sketched in Algorithm 5. All the procedures are assumed to take the decision variables $\{\mathcal{R}^1, \cdots, \mathcal{R}^K, C^1, \cdots, C^K\}$ and the input matrix $\boldsymbol{M}$ as parameters.

Algorithm 5: Sketch of the DFS resolution

```
Function solveDFS():
    if !allVariablesBound() then
        $\mathcal{S}$ ← selectUnboundSetVar()
        $i$ ← selectValue( $\mathcal{S}^?$ )
        foreach $action \in$ [require($i, \mathcal{S}$), exclude($i, \mathcal{S}$)] do
            saveState()
            post($action$)
            propagateDominanceRule()
            ($lb, cb, ub$) ← updateBounds()
            $best$ ← max($best, cb$)
            if $ub > best$ then
                solveDFS()
            restoreState()
```

The procedure `selectUnboundSetVar` chooses a not yet bound set variable among $\{\mathcal{R}^{1,?}, \cdots, \mathcal{R}^{K,?}, C^{1,?}, \cdots, C^{K,?}\}$. The subsequent line chooses for the selected row or column set of some submatrix k, the specific row or column i (among the possible ones) to be included on the left branch and to be excluded on the right branch. The explored search tree is binary. The combination of the two lines and the recursion corresponds to the insertion and exclusion of all rows and all columns in each submatrix. Once the constraint is posted, and the previous state saved for later backtracking, the procedure `propagateDominanceRule` can include (exclude) rows or columns in every submatrix that can be proven to participate (or not) in any optimal solution. The `updateBounds` function updates and returns the lower, current and upper bounds for the state. The current bound is obtained by transforming the partial assignment into two complete feasible solutions,

1. one that excludes all rows in $\mathcal{R}^{\cdot,?}$, and
2. one that excludes all columns in $\mathcal{C}^{\cdot,?}$.

If the current bound *cb* is better than the best value found so far, stored in variable *best*, the current state $(\mathcal{R}^{1,\in}, \cdots, \mathcal{R}^{K,\in}, \mathcal{C}^{1,\in}, \cdots, \mathcal{C}^{K,\in})$ is a better solution, then:

- the value of variable *best* is updated,
- the submatrices of the solution are stored.

Then, a check is made to ensure that better solution exists down the tree node. That is done by verifying that the upper bound is greater than the best objective value found so far; if that is the case, the DFS continues recursively. Once these steps are done, the state is backtracked and the next state visited.

Efficient backtracking is achieved through trailing, a state management strategy that facilitates the restoration of the computation state to an earlier version. Trailing enables the design of reversible objects. We refer to MiniCP [MSV18] for a detailed description of trailed-based solvers and to [Sai+13] for a trailed based implementation of set domains with sparse-sets.

Only four functions are specific to our method to solve the maximum weighted submatrix coverage problem:

- `propagateDominanceRule` that prunes the domains of the set variables,
- `updateBounds` that prunes the bounds on the objective, helping the branch & bound algorithm to prune impossible values,
- `selectUnboundSetVar` and `selectValue` that define the strategy of the depth-first-search.

The following subsections are dedicated to the main functions of the algorithm.

The other functions are method available in most constraint programming solvers:

- `saveState` and `restoreState` save and restore, respectively, the state of the reversible variables to allow efficient backtracking,
- `post` adds a new constraint to the solver.

4.2.4 Functions selectUnboundSetVar and selectValue

`selectUnboundSetVar` chooses, at each step of the DFS, the next (unbounded) row or column interval set $\mathcal{S}$ to branch on, while `selectValue` selects the value $l \in \mathcal{S}^?$ to include or exclude from this set when branching. That is, when a pair $(\mathcal{S}, l)$ has been chosen, the DFS branches on the left, by setting require$(l, \mathcal{S})$, and on the right, by setting exclude$(l, \mathcal{S})$. The decision of the interval set and of the value are not done independently. To choose the next (set, value) pair to branch on, our algorithm maintains two (reversible) counters per row or column and per submatrix:

- $t^{\text{row}}_{k,i}$ contains the sum of matrix entry values that will be immediately added to the objective value if row i is included in $\mathcal{R}^k$. Similarly, $t^{\text{col}}_{k,j}$ contains the sum of matrix entry values that will be immediately added to the objective value if col j is included in C^k:

$$t^{\text{row}}_{k,i} = \underset{\text{row } i}{\text{sum}}\left(\{j \mid j \in C^{k,\in} \ \wedge (i,j) \notin cover^{\in}\}\right) \ , \tag{4.14}$$

$$t^{\text{col}}_{k,j} = \underset{\text{col } j}{\text{sum}}\left(\{i \mid i \in \mathcal{R}^{k,\in} \ \wedge (i,j) \notin cover^{\in}\}\right) \ . \tag{4.15}$$

- $p^{\text{row}}_{k,i}$ contains the sum of positive values in the line i that *could* be taken by submatrix k, *i.e.* whose columns have not been excluded:

$$p^{\text{row}}_{k,i} = \underset{\text{row } i}{\text{sum}}\left(\{j \mid j \in (C^{k,\in} \cup C^{k,?}) \ \wedge (i,j) \notin cover^{\in}\}\right) \ , \tag{4.16}$$

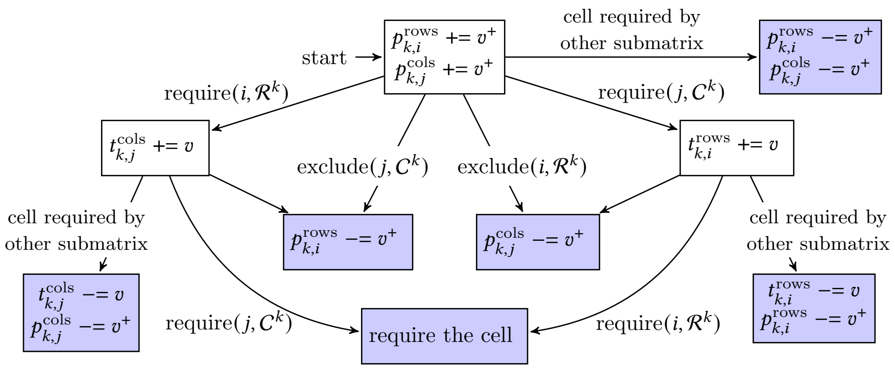
$$p^{\text{col}}_{k,j} = \underset{\text{col } j}{\text{sum}}\left(\{i \mid i \in (\mathcal{R}^{k,\in} \cup \mathcal{R}^{k,?}) \ \wedge (i,j) \notin cover^{\in}\}\right) \ . \tag{4.17}$$

The algorithm then selects the (submatrix, row), or (submatrix, column), pair (k, i), or (k, j), that maximizes $t^{\text{row}}_{k,i}$, or $t^{\text{col}}_{k,j}$. Ties are broken by maximizing $p^{\text{row}}_{k,i}$, or $p^{\text{col}}_{k,j}$. The selected interval set and value are then $\mathcal{R}^k$ and i, or C^k and j.

Recomputing these counters at each iteration is costly as this operation is in $O(Kmn + K(m+n))$ for the MWSC problem with an $m \times n$ matrix and K submatrices. We propose here to maintain these counters using the finite state machine (FSM) shown in Figure 4.2. The algorithm we propose virtually maintains an FSM for each (row, column, submatrix) triplet. The FSMs are updated each time a row/column is added to/excluded from a submatrix:

- When a row i is included in, or removed from, the submatrix k, at most n FSMs must be updated: one for each matrix entry in the row.
- When a column j is included in, orremoved from, the submatrix k, at most m FSMs must be updated: one for each matrix entry in the column.

- Updating a matrix entry is in $O(1)$, if it does not become *selected* by a submatrix (i.e. the row and column of the matrix entry are both in the mandatory sets of the submatrix).
- If a matrix entry becomes *selected*, $K-1$ other matrix entries must be updated.



start
$p^{\text{rows}}_{k,i}$ += v^+ $p^{\text{cols}}_{k,j}$ += v^+
cell required by other submatrix
$p^{\text{rows}}_{k,i}$ −= v^+ $p^{\text{cols}}_{k,j}$ −= v^+
require$(i,\mathcal{R}^k)$
require$(j,\mathcal{C}^k)$
$t^{\text{cols}}_{k,j}$ += v
$t^{\text{rows}}_{k,i}$ += v
exclude$(j,\mathcal{C}^k)$
exclude$(i,\mathcal{R}^k)$
cell required by other submatrix
$p^{\text{rows}}_{k,i}$ −= v^+
$p^{\text{cols}}_{k,j}$ −= v^+
cell required by other submatrix
$t^{\text{cols}}_{k,j}$ −= v $p^{\text{cols}}_{k,j}$ −= v^+
require$(j,\mathcal{C}^k)$
require the cell
require$(i,\mathcal{R}^k)$
$t^{\text{rows}}_{k,i}$ −= v $p^{\text{rows}}_{k,i}$ −= v^+

Figure 4.2: Finite state machine maintained for each (row, column, submatrix) i,j,k in the variable and value selection algorithms. For simplicity, $v = M_{i,j}$, $v^+ = \max(v,0)$ and $v^- = \min(v,0)$. FSMs states in blue are terminal states.

Let Δ_{rows} denote the number of added or excluded (submatrix, row) tables between two calls of the algorithm. Let Δ_{cols} denote the number of added or excluded (submatrix, column) tables between two calls of the algorithm. Let Δ_{selected} denote the number of selected matrix entries between two calls of the algorithm.

The update of the finite state machine between two calls of the algorithm runs in $O(\Delta_{\text{rows}}n + \Delta_{\text{cols}}m + \Delta_{\text{selected}}K)$. To that update process must be added the verification of the counters to select the best set/value pair, which is in $O(K(m+n))$.

Over a complete branch of the DFS tree, which has a maximum depth of $K(m+n)$, we have that:

$$\sum_{\text{branch}} \Delta_{\text{rows}} \leq K \cdot m \ , \tag{4.18}$$

$$\sum_{\text{branch}} \Delta_{\text{cols}} \leq K \cdot n \ , \tag{4.19}$$

$$\sum_{\text{branch}} \Delta_{\text{selected}} \leq n \cdot m \ . \tag{4.20}$$

Over a complete branch, the FSM-based algorithm maintains the states and returns the best set/value pair in $O(K^2(m+n)^2)$, which is a significant improvement over the recomputation-based algorithm which runs in $O(K^2(n^2m + nm^2))$ over a complete branch.

4.2.5 Function propagateDominanceRule

In some cases, given a partial assignment with some rows and columns already included in the set variables C^k and $\mathcal{R}^k$, dominance rules permit to detect additional rows or columns that must be included in any optimal solution extending this partial assignment, or rows or columns that never participate in an optimal solution. The current state is defined by $(\mathcal{R}^{k,\in}, \mathcal{R}^{k,?}, C^{k,\in}, C^{k,?})$, and we denote the optimal solution extending this state as $(\mathcal{R}^{k,*\in}, \emptyset, C^{k,*\in}, \emptyset)$ with $\mathcal{R}^{k,\in} \subseteq \mathcal{R}^{k,*\in}$, $\mathcal{R}^{k,*\in} \subseteq (\mathcal{R}^{k,\in} \cup \mathcal{R}^{k,?})$, $C^{k,\in} \subseteq C^{k,*\in}$, $C^{k,*\in} \subseteq (C^{k,\in} \cup C^{k,?})$.

Theorem 4 gives the condition to be satisfied to detect that a row i should be included in submatrix l in any optimal solution extending the current state.

Theorem 4.

$$\forall i \in \mathcal{R}^{l,?} \ : \ \underset{row\ i}{\text{sum}}\left((C^{l,\in} \cup C^{l,?}_{-i}) \setminus (\bigcup_{k|k \neq l} C^{k,\in}_{+i} \cup C^{k,?}_{+i}) \right) > 0 \Rightarrow i \in \mathcal{R}^{l,*\in} \quad (4.21)$$

Proof. (sketch) Let us assume the worst-case scenario: despite selecting all the columns with negative values in this row i, while other submatrices would take the columns with positive values, the submatrix still has a positive sum contribution for this row i. Therefore this row must be included in submatrix l in any optimal solution extending the current state. □

Theorem 5 gives the condition to be satisfied to detect that a row i will never be included submatrix l in any optimal solution extending the current state, using the best-case scenario.

Theorem 5.

$$\forall i \in \mathcal{R}^{l,?} \ : \ \underset{row\ i}{\text{sum}}\left((C^{l,\in} \cup C^{l,?}_{+i}) \setminus (\bigcup_{k|k \neq j} C^{k,\in}_{-i} \cup C^{k,?}_{-i}) \right) < 0 \Rightarrow j \notin \mathcal{R}^{l,*\in} \quad (4.22)$$

These two properties, and their symmetric counterparts for the columns, can be used in any node of the search tree to reduce the search space.

Recomputing the rules at each call to `propagateDominanceRule` is expensive:

- $O(Kmn)$ at each call,

- $O(K^2(m^2n + mn^2))$ over a complete branch of the DFS.

We describe below how to maintain the rules on rows. The method is symmetric for columns.

We maintain virtual FSMs for each triplet (row, column, submatrix) as shown in Figure 4.3. The FSMs collectively maintain two reversible values, shared between FSMs, for each (submatrix k, row i) table:

- $\text{lb}_{k,i}$ is the value of the worst-case scenario for submatrix k and row i. It corresponds to the left part of Equation (4.21):

$$\text{lb}_{k,i} = \underset{\text{row } i}{\text{sum}}\left((C^{l,\in} \cup C^{l,?}_{-i}) \setminus (\bigcup_{k|k\neq l} C^{k,\in}_{+i} \cup C^{k,?}_{+i}) \right) . \tag{4.23}$$

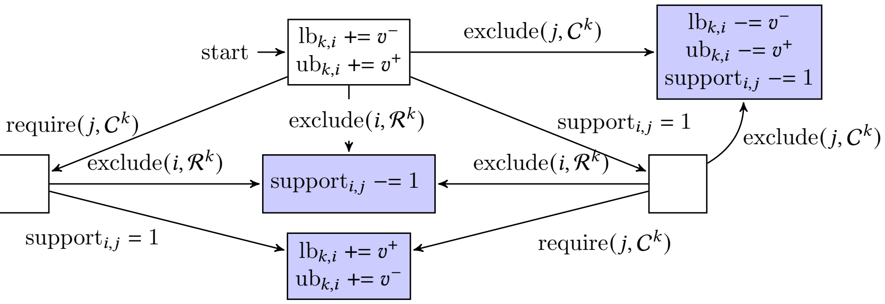
- $\text{ub}_{k,i}$ is the value of the best-case scenario for submatrix k and row i. It corresponds to the left part of Equation (4.22):

$$\text{ub}_{k,i} = \underset{\text{row } i}{\text{sum}}\left((C^{l,\in} \cup C^{l,?}_{+i}) \setminus (\bigcup_{k|k\neq j} C^{k,\in}_{-i} \cup C^{k,?}_{-i}) \right) . \tag{4.24}$$

The FSMs also maintain the number of *supports* of each matrix entry (i, j), *i.e.* the number of submatrices that could still select the matrix entry:

$$\text{support}_{i,j} = \left| \{k \mid i \in (\mathcal{R}^{k,\in} \cup \mathcal{R}^{k,?}) \land j \in (C^{k,\in} \cup C^{k,?})\} \right| . \tag{4.25}$$

Each $\text{support}_{i,j}$, shared across all FSMs, is maintained as reversible integer by the solver. Its state can then be backtracked.



start
$\text{lb}_{k,i}$ += v^-
$\text{ub}_{k,i}$ += v^+
exclude(j, C^k)
$\text{lb}_{k,i}$ −= v^-
$\text{ub}_{k,i}$ −= v^+
$\text{support}_{i,j}$ −= 1
require(j, C^k)
exclude$(i, \mathcal{R}^k)$
$\text{support}_{i,j} = 1$
exclude(j, C^k)
exclude$(i, \mathcal{R}^k)$
$\text{support}_{i,j}$ −= 1
exclude$(i, \mathcal{R}^k)$
$\text{support}_{i,j} = 1$
$\text{lb}_{k,i}$ += v^+
$\text{ub}_{k,i}$ += v^-
require(j, C^k)

Figure 4.3: Finite state machine maintained for each (row, column, submatrix) i, j, k in propagateDominanceRule method. For simplicity, $v = M_{i,j}$, $v^+ = \max(v, 0)$ and $v^- = \min(v, 0)$. FSMs states in blue are terminal states.

The transition and update operations of our finite state machines are the following:

- When a row (resp. column) is excluded from a submatrix, the FSMs of at most n (resp. m) matrix entries must be updated.

  The contribution of the matrix entry to ub and lb is removed, and the support of the matrix entry is decremented. Each of these operations are in constant time, and overall takes $O(n)$ (resp. $O(m)$).

- When a matrix entry (i, j) becomes supported by only one remaining submatrix k, $\text{support}_{i,j} = 1$, and column j is included in that submatrix k, $j \in C^{k,\in}$, the value of $\text{ub}_{k,i}$ and $\text{lb}_{k,i}$ are updated.

  Since $j \in C^{k,\in}$ and $\text{support}_{i,j} = 1$, it implies that $i \in (\mathcal{R}^{k,\in} \cup \mathcal{R}^{k,?})$. The contribution of the matrix entry to ub and lb is added. That operation is also in constant time, and in $O(K)$ for all submatrices.

- When a row i (resp. column j) is included in a submatrix k, a check on all columns j (resp. rows i) must be performed.

  If a matrix entry (i, j) with $\text{support}_{i,j} = 1$ and $i \in \mathcal{R}^{k,\in}$ and $j \in C^{k,\in}$ exists, $lb_{k,i}$ and $ub_{k,i}$ are updated to include the value of the matrix entry. Overall, this operation is in $O(n)$ (resp. $O(m)$).

Once the updates of the FSMs are done, each (row, submatrix) pair is verified with respect to the rules in $O(Km)$. A call to `propagateDominanceRule` is in $O(Km + \Delta_{\text{rows}} n + \Delta_{\text{cols}} m + \Delta_{\text{required}} K + \Delta_{\text{support}=1} K)$. Over a complete branch, the number of operations required is in $O(Km^2 + Kmn)$. If the rules are applied symmetrically on columns, the overall running time is in $O(K \max(m, n)^2)$.

4.2.6 Function UPDATEBOUNDS

Upper bounds on the objective for the current tree node must be computed efficiently, in order to run the branch & bound. The chosen method also provides a lower bound, with no additional (marginal) computational cost.

The upper bound ub is the sum of every matrix entry that is either selected in a submatrix or that is positive and could still be selected. The lower bound lb is similarly defined, but keeping negative-valued matrix entries. Formally, they are computed as follows:

$$\text{ub} = \sum \{\boldsymbol{M}_{i,j} \mid (i, j) \in \mathit{cover}^{\in} \lor (\boldsymbol{M}_{i,j} > 0 \land (i, j) \notin \mathit{cover}^{\notin})\} \ , \tag{4.26}$$

$$\text{lb} = \sum \{\boldsymbol{M}_{i,j} \mid (i, j) \in \mathit{cover}^{\in} \lor (\boldsymbol{M}_{i,j} < 0 \land (i, j) \notin \mathit{cover}^{\notin})\} \ . \tag{4.27}$$

Recomputing these bounds from scratch in each node is costly: $O(Knm)$. The running time can be improved by incrementally maintaining the number of submatrices supporting each matrix entry, in the same way as done for the `propagateDominanceRule` function. These bounds, stored as reversible floating point numbers, can be maintained easily:

- When a row i is included in a submatrix k, check if any column j is already in $C^{k,\in}$, and that $(i,j) \notin cover^{\in}$ yet. If that is the case and $\boldsymbol{M}_{i,j} > 0$ (resp. < 0), increase ub (resp. lb) by $\boldsymbol{M}_{i,j}$. This operation runs in $\mathcal{O}(n)$.
- The similar operation must be performed when a column is included in a submatrix. Each of these operations runs in $\mathcal{O}(m)$.
- When a row i is excluded from a submatrix k, check if any column j is not already excluded ($j \notin (C^{k,\in} \cup C^{k,?})$). If that is the case, decrease support$_{i,j}$ by one. This operation runs in $\mathcal{O}(n)$.
- The same operation goes for excluded columns in $\mathcal{O}(m)$.
- When the support$_{i,j}$ is reduced to zero, if $\boldsymbol{M}_{i,j} > 0$ (resp. < 0), then decrease ub (resp. lb) by $\boldsymbol{M}_{i,j}$. This operation runs in $\mathcal{O}(1)$.

The whole maintenance process for the bounds behaves in $\mathcal{O}(\Delta_{rows}n + \Delta_{cols}m)$. Over a complete branch, the incremental method is in $\mathcal{O}(Knm)$, while the one based on recomputations is in $\mathcal{O}(K^2(n^2m + nm^2))$.

4.2.7 Large neighborhood search

The exhaustive approach presented above eventually finds and proves the optimum value provided enough time is given. Unfortunately, the search space is so large that even for small matrices and a limited number of submatrices, it tends to quickly find a good solution but is not able to improve it. To overcome this limitation, we propose to embed the exhaustive CP search into a large neighborhood search (LNS)[Sha98]. LNS is a local search approach using CP to discover improvements around the current best solution:

- First the CP exhaustive search is used during a limited time, to discover an initial solution.
- For a given number of iterations, the CP exhaustive search is used again while fixing some of the variables (fragment), as detailed below, to their value in the current best solution.

In addition, to limit the risk of having an iteration stuck for too long, we limit the DFS to 1000 *failures*.

The current best solution at iteration t has the form $((\mathcal{R}^{1,t,*\in},\cdots,\mathcal{R}^{K,t,*\in});(C^{1,t,*\in},\cdots,C^{K,t,*\in}))$. We propose three different fragment selection heuristics that define parts of the solution to constrain when restarting the LNS for next iteration:

1. Select uniformly at random a subset of rows and columns in the set of rows and columns used by some submatrix: $R^p \subseteq (\bigcup_{k\in M^p}\mathcal{R}^{k,t,*\in})$, $C^p \subseteq (\bigcup_{k\in M^p} C^{k,t,*\in})$, then for each submatrix, include the set of rows and columns intersecting with those sets:

$$\mathcal{R}^{k,t+1,\in} = \mathcal{R}^{k,t,\in} \cap R^p \ , \qquad \text{and} \quad \mathcal{R}^{k,t+1,?} = R \setminus \mathcal{R}^{k,t+1,\in} \ , \qquad (4.28)$$

   and similarly for the columns.

2. A similar operator is defined with rows and columns selected inside the whole matrix: $R^p \subseteq R$, $C^p \subseteq C$. This allows for greater diversification, notably by allowing discovery of previously unselected rows/columns.

3. Selecting uniformly at random a subset of submatrices $M^p \subseteq \{1,\cdots,K\}$. For each of these submatrices, select at random different subsets of rows and columns $R^p_k \subseteq \mathcal{R}^{k,t,*\in}$, $C^p_k \subseteq C^{k,t,*\in}$ that is constrained: $\mathcal{R}^{k,t+1,\in} = \mathcal{R}^{k,t,\in} \cap R^p_k$, $\mathcal{R}^{k,t+1,?} = R \setminus \mathcal{R}^{k,t+1,\in}$ and similarly for columns.

Experiments show that these three operators are complementary.

4.3 Experiments

This section describes experiments conducted to assess the performance of the proposed algorithms and to provide guidance on the selection of the appropriate solution. We first evaluate the methods on synthetic datasets, where the optimum is known, then on real datasets.

We compare our exhaustive CP and LNS methods against a greedy baseline approach, CP-Greedy, that solves at each step the maximal sum submatrix ($K = 1$) problem using the constraint programming approach presented in Chapter 2. That approach iteratively selects the next best submatrix, on a modified matrix in which the previously selected entries are set to 0 such that there is no incentive to select several times the same positive entries. Each iteration is performed within $\frac{t^{\max}}{K}$ with $t^{\max}$ the allocated budget of time.

The implementation has been carried out on OscaR [Osc12], using Java 1.8.0 on an AMD Bulldozer clocked at 2.1 GHz; one core and 3 Go of RAM per instance. The source code is available here: `https://github.com/GuillaumeDerval/MWSCP`.

4.3.1 Evaluation

Approaches are compared using any-time profiles as described in DEFINITION 4 in SECTION 2.4.1.1.

4.3.2 Synthetic dataset

A synthetic dataset composed of 1,617 instances has been generated using a Python script which is available on Zenodo [Der+18]. For those, the optimal solution is known as they were all generated by implanting randomly K submatrices before adding some noise. Note that the optimal solution may be slightly different than the implanted submatrices because of the noise addition. TABLE 4.1 describes parameter values considered in the generation.

Table 4.1: Parameters for the synthetic dataset generation

Parameter	Values used	Description
(m, n)	$(800, 200), (640, 250), (400, 400)$	Size of the matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$
K	2, 4, 8	Number of submatrices
o	0, 0.3, 0.6	Minimum overlap between submatrices (in % of matrix entries)
σ	0, 0.5, 1.0	Background noise variance (mean is 0)
(r, s)	(35, 70), (50, 50)	Size of submatrices (noisy, Gaussian with $\sigma = \frac{r \text{ or } s}{20}$)
seed	$[0, 9]$	Seed for matrix generation

FIGURE 4.4 gives the any-time profiles of the CP-Greedy baseline method, along with CP-Exhaustive (the exhaustive process presented above) and CP-LNS. The results clearly illustrates the overall better performance of the CP-LNS whenever the computation time roughly exceeds 20 seconds.

TABLE 4.2a presents the performance of the algorithms for each parameter value considered in the synthetic data generation. Reported performance is computed as the average performance of each algorithm obtained before a certain limit of computation time.

4.3.3 Real dataset

We also experiment with non-synthetic datasets of several types described in SECTION 4.1.2:

1. *olympic*,
2. *migration*, and

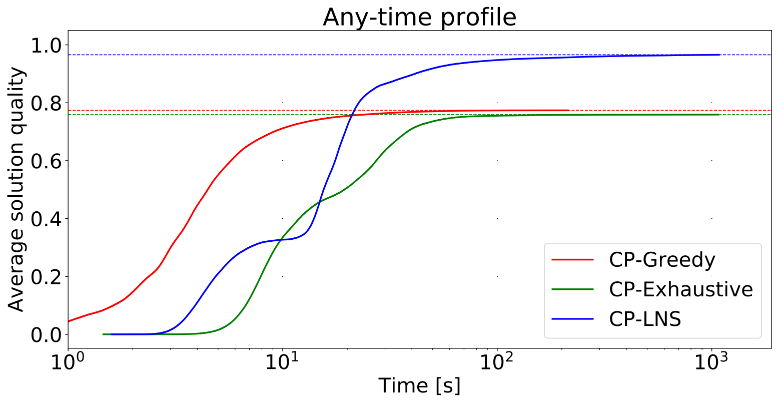


Figure 4.4: Comparison between CP-Greedy, CP-Exhaustive and CP-LNS on 1,617 matrices generated as described in Section 4.3.2. The graph presents the any-time profiles as described in Definition 4 (Section 2.4.1.1). For each instance, 18 minutes were allocated for computations.

3. *genes.*

The results presented in Table 4.2b are similar to those obtained for synthetic datasets. CP-LNS is the best method on most datasets given 10 seconds of computation time. There are two notable exceptions (*alizadeh* and *garber* datasets), in which case LNS did not find the optimum in the 20 minutes allowed for each dataset.

4.3.4 Comparison against Mixed Integer Linearly and Quadratically Constrained Programming

We tested our methods against MIP (linear) and MIQCP (quadratic terms in the constraints) methods. As these two methods do not perform well on bigger instances, we do not integrate them in our experiments presented above on large matrices.

$$\begin{array}{rll|rll}
 & \text{MIP model} & & & \text{MIQCP model} & \\
\max & \sum_{i,j} \boldsymbol{M}_{i,j} \cdot s_{i,j} & & \max & \sum_{i,j} \boldsymbol{M}_{i,j} \cdot s_{i,j} & \\
s_{i,j} & \geq e_{i,j,k} & \forall i,j,k & K \cdot s_{i,j} & \geq \sum_k r_{k,i} \cdot c_{k,j} & \forall i,j \\
s_{i,j} & \leq \sum_k e_{i,j,k} & \forall i,j & s_{i,j} & \leq \sum_k r_{k,i} \cdot c_{k,j} & \forall i,j \\
e_{i,j,k} + 1 & \geq r_{k,i} + c_{k,j} & \forall i,j,k & & & \\
2 \cdot e_{i,j,k} & \leq r_{k,i} + c_{k,j} & \forall i,j,k & & & \\
\end{array}$$
$$\text{All variables} \in \{0,1\}$$

MIP and MIQCP methods are plagued by the number of variables, that is in $\mathcal{O}(Knm)$ for MIP and $\mathcal{O}(K(n+m))$ for MIQCP, and by the number of

Table 4.2: Comparison between CP-Greedy (GRE), CP-Exhaustive (EX) and CP-LNS (LNS). The table shows the $Q(a, t)$ for each algorithm a given a certain amount of time t (see DEFINITION 4 in SECTION 2.4.1.1). Migration, olympic and gene expression data are described by Dao et al. [Dao+18], IOC Research and Reference Service, The Guardian [IOC], and de Souto et al. [de +08], respectively.

(a) Synthetic dataset

	10s			20s			100s			1080s		
Parameters	GRE	EX	LNS	GRE	EX	LNS	GRE	EX	LNS	GRE	EX	LNS
$\{m = 400, n = 400\}$	**0.70**	0.33	0.37	0.74	0.57	**0.76**	0.76	0.75	**0.95**	0.77	0.75	**0.97**
$\{m = 640, n = 250\}$	**0.71**	0.34	0.32	0.75	0.48	**0.79**	0.77	0.74	**0.95**	0.77	0.75	**0.97**
$\{m = 800, n = 200\}$	**0.73**	0.34	0.29	**0.77**	0.48	0.61	0.79	0.77	**0.94**	0.79	0.78	**0.96**
$K = 2$	**0.85**	0.78	0.32	0.85	**0.88**	0.83	0.85	0.90	**0.96**	0.85	0.91	**0.97**
$K = 4$	**0.72**	0.20	0.30	**0.77**	0.51	0.72	0.78	0.74	**0.94**	0.78	0.75	**0.96**
$K = 8$	**0.57**	0.03	0.36	**0.64**	0.13	0.61	0.68	0.62	**0.94**	0.68	0.62	**0.97**
$o = 0\%$	**0.58**	0.27	0.34	0.67	0.45	**0.71**	0.71	0.66	**0.97**	0.71	0.66	**0.98**
$o = 30\%$	**0.71**	0.34	0.31	**0.73**	0.50	0.69	0.75	0.75	**0.93**	0.75	0.76	**0.95**
$o = 60\%$	**0.85**	0.40	0.34	**0.86**	0.57	0.77	0.86	0.86	**0.94**	0.86	0.86	**0.97**
$\sigma = 0.0$	0.73	0.34	**0.78**	0.78	0.63	**0.80**	0.81	0.77	**0.98**	0.81	0.78	**1.00**
$\sigma = 0.5$	**0.72**	0.33	0.04	**0.75**	0.44	0.67	0.78	0.74	**0.94**	0.78	0.74	**0.97**
$\sigma = 1.0$	**0.69**	0.33	0.16	**0.73**	0.44	0.68	0.73	0.75	**0.93**	0.73	0.75	**0.94**
$\{r = 50, s = 50\}$	**0.71**	0.34	0.34	**0.75**	0.52	0.73	0.77	0.76	**0.94**	0.77	0.77	**0.96**
$\{r = 35, s = 70\}$	**0.71**	0.32	0.32	**0.76**	0.50	0.71	0.78	0.75	**0.95**	0.78	0.75	**0.97**

(b) Real datasets

$K = 4$		1s			5s			20s		
Type	Dataset	GRE	EX	LNS	GRE	EX	LNS	GRE	EX	LNS
migration	migration_0.001	**0.96**	0.92	**0.96**	0.96	0.92	**0.99**	0.96	0.92	**1.00**
migration	migration_0.003	0.87	0.89	**0.93**	0.87	0.89	**0.99**	0.87	0.89	**1.00**
migration	migration_0.005	0.83	0.79	**0.96**	0.83	0.79	**1.00**	0.83	0.79	**1.00**
olympic	olympic_0.01	0.88	0.69	**0.92**	0.88	0.91	**0.97**	0.91	0.91	**1.00**
olympic	olympic_0.02	0.79	0.69	**0.87**	0.84	0.84	**0.97**	0.84	0.84	**1.00**
olympic	olympic_0.04	0.62	0.81	**0.91**	0.76	0.82	**0.96**	0.93	0.82	**1.00**
olympic	olympic_0.06	0.80	0.92	**0.93**	0.97	0.92	**0.98**	0.97	0.92	**0.99**
$K = 4$		10s			20s			100s		
Type	Dataset	GRE	EX	LNS	GRE	EX	LNS	GRE	EX	LNS
gene	alizadeh-2000-v1_095	**1.00**	0.48	0.82	**1.00**	0.48	0.82	**1.00**	0.48	0.92
gene	armstrong-2002-v1_095	0.73	0.60	**0.92**	0.73	0.60	**0.99**	0.73	0.60	**1.00**
gene	bhattacharjee-2001_095	0.82	0.31	**0.98**	0.91	0.86	**0.99**	0.91	0.96	**1.00**
gene	bittner-2000_095	**0.96**	0.53	0.86	0.96	0.53	**0.98**	0.96	0.53	**0.98**
gene	bredel-2005_095	0.98	0.86	**1.00**	0.98	0.86	**1.00**	0.98	0.86	**1.00**
gene	chen-2002_095	0.74	0.80	**1.00**	0.89	0.80	**1.00**	0.89	0.80	**1.00**
gene	chowdary-2006_095	0.82	0.83	**1.00**	0.82	0.83	**1.00**	0.87	0.83	**1.00**
gene	dyrskjot-2003_095	0.97	0.94	**0.99**	0.97	0.94	**1.00**	0.97	0.94	**1.00**
gene	garber-2001_095	**0.59**	0.24	0.58	**0.82**	0.32	0.58	**1.00**	0.50	0.86
gene	golub-1999-v1_095	0.86	0.88	**0.92**	0.86	0.88	**0.95**	0.86	0.88	**0.96**

constraints, which is $O(Knm)$ for MIP and $O(nm)$ for MIQCP. Tables 4.3a and 4.3b show that both models are slow compared to our LNS method, and are heavily affected by matrix size, number of submatrices to find and noise. For bigger submatrices, such as the synthetic and real ones presented in the previous section, both methods timeout either without returning solutions or with comparatively poor solutions.

4.4 Conclusions

We presented a generalization of the maximal sum submatrix problem to multiple submatrices, called the maximum weighted submatrix coverage problem, along with a method to solve that problem based on constraint programming and large neighborhood search. Experiments on both synthetic and real datasets show that our CP-LNS method finds consistently better solutions (when more than 10 seconds are allocated) than both MIP/MIQCP, an exhaustive CP method and a greedy approach using the method presented in Chapter 2 to solve the MSS problem.

The proposed experiments are only concerned with implementation efficiency. An evaluation of the maximum weighted submatrix coverage problem on biological data is proposed in Chapter 5.

Table 4.3: Comparison between CP-LNS, MIP and MIQCP, on a synthetic dataset, generated as described in Section 4.3.2. All methods were given a fixed time limit of 300 seconds. The metric used is the any-time profile (see Definition 4 in Section 2.4.1.1). CP-LNS finds the optimum on each dataset. The time when the best found solution was found is indicated inside parentheses. Experiments made on Gurobi 8.1.0.

(a) Varying number of submatrices and noise, with matrices of size 50×50 **and submatrices of size** 16×16

K	σ	CP-LNS		MIP		MIQCP	
2	0.0	**1.00**	(1s)	**1.00**	**(0s)**	**1.00**	(1s)
2	0.5	**1.00**	**(1s)**	**1.00**	(7s)	**1.00**	(7s)
2	1.0	**1.00**	**(1s)**	0.89	(233s)	0.79	(57s)
3	0.0	**1.00**	(2s)	**1.00**	**(1s)**	**1.00**	(2s)
3	0.5	**1.00**	**(3s)**	**1.00**	(140s)	**1.00**	(138s)
3	1.0	**1.00**	**(3s)**	0.74	(254s)	0.48	(256s)
4	0.0	**1.00**	(2s)	**1.00**	**(1s)**	**1.00**	(62s)
4	0.5	**1.00**	**(3s)**	**1.00**	(252s)	0.88	(290s)
4	1.0	**1.00**	**(6s)**	0.64	(260s)	0.69	(225s)
5	0.0	**1.00**	**(4s)**	**1.00**	(79s)	**1.00**	(275s)
5	0.5	**1.00**	**(5s)**	0.82	(257s)	0.69	(237s)
5	1.0	**1.00**	**(6s)**	0.77	(24s)	0.36	(38s)

(b) Varying size of the matrix and noise, with matrices of size $m \times m$ **and** $K = 2$ **submatrices of size** $\lfloor \frac{m}{3} \rfloor \times \lfloor \frac{m}{3} \rfloor$

m	σ	CP-LNS		MIP		MIQCP	
50	0.0	**1.00**	**(0s)**	**1.00**	(1s)	**1.00**	(3s)
50	0.5	**1.00**	**(1s)**	**1.00**	(5s)	**1.00**	(7s)
50	1.0	**1.00**	**(1s)**	0.95	(207s)	0.82	(204s)
100	0.0	**1.00**	(4s)	**1.00**	**(1s)**	1.00	(33s)
100	0.5	**1.00**	**(1s)**	0.86	(293s)	1.00	(45s)
100	1.0	**1.00**	**(3s)**	0.65	(269s)	0.82	(191s)
200	0.0	**1.00**	(17s)	**1.00**	**(8s)**	**1.00**	(135s)
200	0.5	**1.00**	**(21s)**	0.37	(191s)	3%	(81s)
200	1.0	**1.00**	**(6s)**	0%	(0s)	5%	(134s)
400	0.0	**1.00**	**(1s)**	**1.00**	(31s)	**1.00**	(54s)
400	0.5	**1.00**	**(1s)**	0%	(1s)	0%	(0s)
400	1.0	**1.00**	**(1s)**	0%	(1s)	4%	(301s)

Mining submatrices of maximal sum in gene expression data

5

The previous chapters addressed the maximal sum submatrix problem and extensions to the identification of K submatrices. This chapter explores the relevance of maximal sum submatrices to discover **genes subsets** *associated with* **subgroups of samples** *to be identified. The* **K-CPGC** *method is introduced first. It is a computationally efficient algorithm to identify K submatrices of maximal sum in a large gene expression matrix. Comparisons show that it identifies more significantly enriched subsets of genes and specific subgroups of samples which are easily interpretable by biologists. Experiments also show its ability to identify more reliable Gene Ontology terms. These results illustrate the benefits of the proposed approach in terms of interpretability and of biological enrichment quality.*

5.1 Introduction

Gene expression data is typically represented as a large matrix of gene expression levels across various samples. The study of such data is a valuable tool to improve the understanding of the underlying biological processes. A frequent objective of gene expression analysis is to group genes according to their expression under certain conditions or to group conditions based on the expression of a number of genes. Biclustering, also known as co-clustering, is one of the most common approaches for such a task as it identifies specific subsets of rows and of columns which jointly form homogeneous entries [MO04; Xie+18].

Biclustering algorithms tend to produce biclusters sharing similar expression values, for example by minimizing the variance across the selected genes and selected samples. However, some relevant biclusters may be missed when, due to the presence of a few outliers, they lack the assumed homogeneity of expression values among a few gene/sample combinations.

As an alternative, the maximal sum submatrix (MSS) problem seeks for subsets of rows and of columns with globally high values. In biological terms, one looks for a subset of biomarkers which are, after appropriate normalization, relatively highly expressed among a subset of samples. One could also look for patterns of low expression simply by considering the opposite values of a normalized version of the original matrix. By default, we will look for high expression patterns. Both subsets of selected genes and of selected samples are *a priori* unknown and must be identified. They form a rectangular, and not necessarily contiguous, submatrix of the original data matrix exactly like biclusters do. Yet, the mathematical criterion used to find such submatrix differs and is less influenced by the presence of some outliers. In the sequel, we use the terms *submatrix* and *bicluster* interchangeably and, depending on the context, they refer to the solution of existing biclustering algorithms or of our own method.

From a biological viewpoint, there might be several biclusters to be identified from the same original data matrix. Indeed, a single gene may participate in multiple pathways which may or may not be co-active under several conditions [MO04]. Specific genes may also be representative of expression patterns among some samples, while other genes would be more informative for other subsets of samples. In other words, one typically looks at several biclusters which might partially overlap in terms of genes or of samples they contain.

5.1.1 Biclustering and maximal sum submatrix

The maximal sum submatrix problem, introduced in Definition 1, Section 1, consists in finding a submatrix with maximal sum of the selected entries.

The data matrix typically represents gene expression values in a continuous range, for instance on a logarithmic scale and properly normalized: negative values, respectively positive values, represent expression values below, respectively above, a threshold θ. For example, θ may correspond to the median expression level over the whole data matrix, or a row-specific value representing the average expression level of a gene across all samples. After such normalization, positive values are considered as the interesting ones. By default, they correspond to the high levels of expression one is interested in finding in the data matrix. One could also look for low levels of expression by replacing such a normalized matrix $\boldsymbol{M}$ by its opposite $-\boldsymbol{M}$.

Figure 5.1a depicts a toy example of such a normalized data matrix. Positive values, in red, are considered to have high expression levels and negative values, in blue, correspond to low expression levels. Figure 5.1b represents the optimal solution to the maximal sum objective. It defines a specific rectangular submatrix, or bicluster, of genes and samples, maximizing the sum of its entries. It can include a few outliers in terms of high expression levels. For example, the −4.1 entry (row r_4, column c_4) is included in the optimal solution because such a low value is compensated along its row and its column by other positive values, hence all selected rows and selected columns contribute positively to the objective function. In contrast, as one looks for a *rectangular* submatrix, a positive entry may be excluded from the optimal solution if it is penalized by the presence of negative values along its row and its column. This is the case, for example, for the entry 4.0 in row r_3 and column c_3 of this toy example.

Figures 5.1c and 5.1d represent the results obtained with two different biclustering algorithms, namely CCA and ISA (further described in the Section 5.2 Mining approaches), starting from the same toy example Figure 5.1a. Both their solutions strongly differ from the one represented in Figure 5.1b. In particular, the CCA solution includes many negative entries as they imply a lower variance along selected rows and selected columns. In contrast, the ISA solution only includes positive entries but is missing several genes and samples that should arguably be selected as in Figure 5.1b. Experimental results reported in Section 5.4 Results illustrate the benefits of the proposed approach to extract biologically relevant gene subsets.

The maximum weighted submatrix coverage (MWSC) problem is an extension to the maximal sum submatrix problem that consists in finding K submatrices such that the sum of all entries covered by the submatrices is maximum. In this chapter, we study the applicability of that maximum weighted submatrix coverage problem to discover several, and possibly overlapping, biclusters from gene expression data and we show its benefits compared to existing biclustering algorithms.

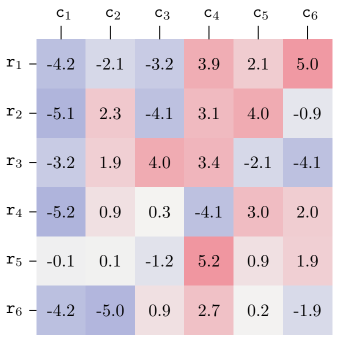
5.1.2 Contributions

The main contributions of this chapter are:

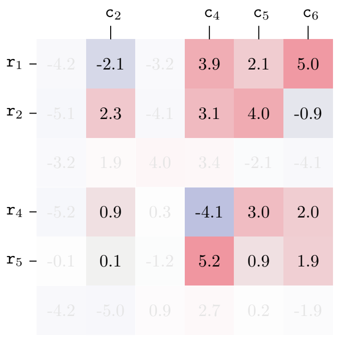
1. K-CPGC, a greedy extension to the CPGC method provided in Section 2.3.2.3 to produce several, possibly overlapping, biclusters of maximal sums in gene expression data,

2. a rigorous statistical validation protocol to assess the performance of six well-known biclustering methods compared between them and with K-CPGC,

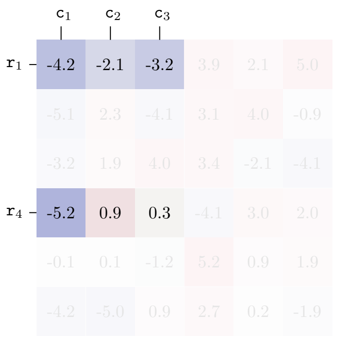
3. practical experiments on 17 gene expression cDNA microarray datasets from *Saccharomyces cerevisiae* samples under various controlled conditions,



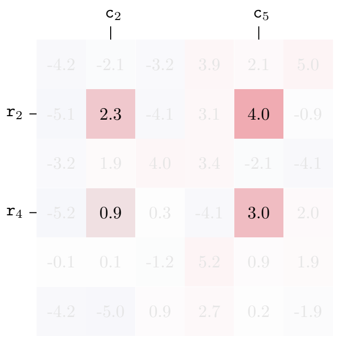
(a) Instance matrix *M*.



(b) Maximal sum submatrix.



(c) CCA solution.



(d) ISA solution.

Figure 5.1: An instance matrix, the submatrix of maximal sum, the bicluster found with CCA and with ISA algorithms. Low and high values are indicated in blue and red, respectively.

4. practical experiments on 18 single-channel Affymetrix chip datasets from various *Human* tissue samples,

5. a gene enrichment analysis showing that the proposed K-CPGC method outperforms biclustering algorithms to find biologically relevant biclusters,

6. a freely available R package implementing K-CPGC.

5.2 Mining approaches

This section briefly presents six biclustering algorithms frequently cited in the literature and for which software implementations are publicly available [PC17a; Li+09; Pre+06; BKC10; Ere+12; Yu+12]. Next, we present the constraint programming approach, CPGC, to identify a submatrix of maximal sum and its extension to extract K submatrices. Our evaluation protocol, including the data collection and experimental setup, is also detailed.

5.2.1 Biclustering algorithms

5.2.1.1 Cheng and Church's algorithm

Cheng and Church's Algorithm (CCA) is based on iteratively adding or removing rows and columns to a current bicluster in order to minimize the variance within it [CC00]. The variance in a bicluster $(\mathtt{I}, \mathtt{J})$ is evaluated as a mean squared residue MSR:

$$\mathrm{MSR}(\mathtt{I}, \mathtt{J}) = \frac{1}{|\mathtt{I}||\mathtt{J}|} \sum_{\substack{i \in \mathtt{I} \\ j \in \mathtt{J}}} (\boldsymbol{M}_{i,j} - \boldsymbol{M}_{i,\mathtt{J}} - \boldsymbol{M}_{\mathtt{I},j} + \boldsymbol{M}_{\mathtt{I},\mathtt{J}})^2 \ , \tag{5.1}$$

where $\mathcal{M}_{i,J}$ is the average of the ith row in the bicluster,

$$\boldsymbol{M}_{i,\mathtt{J}} = \frac{1}{|\mathtt{J}|} \sum_{j \in \mathtt{J}} \boldsymbol{M}_{i,j} \ , \tag{5.2}$$

$\mathcal{M}_{I,j}$ the average of the jth column,

$$\boldsymbol{M}_{\mathtt{I},j} = \frac{1}{|\mathtt{I}|} \sum_{i \in \mathtt{I}} \boldsymbol{M}_{i,j} \ , \tag{5.3}$$

and $\mathcal{M}_{I,J}$ the average of all elements in the bicluster,

$$\boldsymbol{M}_{\mathtt{I},\mathtt{J}} = \frac{1}{|\mathtt{I}||\mathtt{J}|} \sum_{\substack{i \in \mathtt{I} \\ j \in \mathtt{J}}} \boldsymbol{M}_{i,j} \ . \tag{5.4}$$

A parameter δ defines a threshold of maximum MSR for a bicluster to be accepted. The identification of multiple biclusters is achieved iteratively by replacing all entries of the previously identified bicluster(s) by random values within the range of the original data matrix.

5.2.1.2 Conserved gene expression motifs

Conserved Gene Expression Motifs (xMOTIFs) finds biclusters with simultaneously conserved genes in subsets of samples in a discretized data matrix [MK02]. Each discretized entry corresponds to a continuous range of expression values from the original matrix. Genes are considered conserved across a subset of samples if the discretized expression values are identical. This approach greedily searches for a largest xMOTIF starting from various random seeds. When such an xMOTIF is found, the corresponding samples are removed from the original matrix and the whole process is iterated. This approach is thus constrained to return biclusters without overlap between the respective samples they contain.

5.2.1.3 Iterative signature algorithm

Iterative Signature Algorithm (ISA) starts from a randomly selected bicluster and greedily adds or removes columns and rows till reaching some prescribed minimal average value T_C (T_R) across the selected columns (rows) [BIB03]. Several biclusters can be found by restarting from another randomly selected bicluster.

5.2.1.4 Qualitative biclustering

QUalitative BIClustering (QUBIC) discretizes the original matrix and builds a graph where each node corresponds to a gene, and each edge weight is the number of samples for which two genes have the same nonzero discretized value. It then searches for biclusters corresponding to heavy subgraphs [Li+09].

5.2.1.5 Plaid

Plaid fits a generative statistical model with K components from which each entry $\boldsymbol{M}_{i,j}$ of the original matrix is assumed to have been generated [LO02].

$$\boldsymbol{M}_{i,j} = B + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} \kappa_{jk} + \varepsilon_{ij}$$

where B is a background level, μ_k is a specific bicluster effect, α_{ik} and β_{jk} are row and column effects, ρ and κ are cluster memberships respectively along the rows and the columns, ε is a random noise. The `Plaid` algorithm fits such an additive model by minimizing a mean square error between the modeled and observed data [TBK05]. This algorithm may actually return less than K biclusters because a specific bicluster is returned only if it offers a better fit (= a lower residue) than biclusters found from random permutations of the original matrix.

5.2.1.6 Spectral

Spectral relies on singular value decomposition to cluster genes and samples simultaneously after a specific normalization of rows and columns [Klu+03]. It looks for distinctive checkerboard patterns which form biclusters including contiguous rows and contiguous columns. The net result is a set of biclusters of low variance such that each gene and each sample exactly belong to a single bicluster.

5.2.1.7 CPGC

The CPGC algorithm to solve the maximal sum submatrix problem is provided in SECTION 2.3.2.3. It is a depth-first-search (DFS) approach composed of major CP ingredients:

- filtering rules,
- bounding procedure,
- dominance rules,
- variable-value heuristic.

A solution to the maximal sum submatrix problem is represented by two vectors of boolean decision variables $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_m)$ for the rows and $\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_n)$ for the columns, with $\mathbf{R}_i \in \{0, 1\}$ and $\mathbf{C}_j \in \{0, 1\}$. When a decision variable is equal to 1, its corresponding row or column is selected in the solution. When it is equal to 0, its corresponding row or column is not part of the selected submatrix. The algorithm searches through the space of possible variable assignments in the form of a tree as depicted in FIGURE 5.2. Initially, at the root, all decision variables are unbound and the algorithm explores such a tree in a depth-first-search fashion. Any configuration with no unbound variable defines a specific submatrix and is called a feasible solution. The goal is to find an optimal solution, *i.e.* a solution of maximal sum, among the feasible solutions.

The complexity of this approach is defined by the number of nodes explored and the complexity of the methods executed at each node. The CPGC approach explores $O(2^n)$ nodes, or possible assignments of column variables. The time complexity of the methods performed at each node of the search

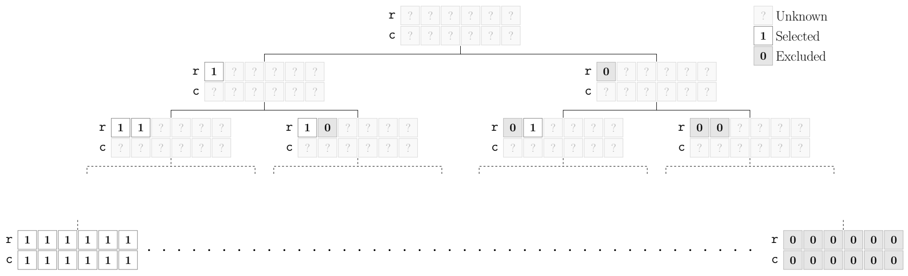


Figure 5.2: Search tree. This figure illustrates the search tree defined on the set of possible submatrices. A question mark refers to an unbound variable that can be equal to 0 or 1.

tree is in $O(m \times n)$. The global time complexity of CPGC is therefore in $O(2^n \times m \times n)$.

The space complexity of the nodes is in $O(m + n)$. The number of nodes to maintain effectively is in $O(n)$, by virtue of the DFS exploration strategy. The global space complexity of CPGC is therefore in $O\left(n \times (m + n)\right)$.

These bounds on the space and time complexities do not consider the substantial reduction of the search space induced by the filtering procedures. In experiments with instance matrices of 10, 000 rows by 1, 000 columns, the best solutions are found within short periods of time, usually less than a few seconds. Moreover, providing more time (up to 1, 000 seconds) never improves the objective value. Further technical details, experiment and result descriptions are provided in Chapter 2 Mining a submatrix of maximal sum. These results suggest that CPGC is scalable to tackle reasonably large problems from biological to biomedical domains.

5.2.2 The greedy K-CPGC

The CPGC algorithm looks for a single submatrix of maximal sum from an original data matrix while there might be several biclusters to be identified. In gene expression analysis, the same gene may indeed participate in multiple pathways. Hence one would like to identify K biclusters with possible overlaps between them. The control parameter K must be chosen by the data analyst (*e.g.* $K = 10$) but, as illustrated in the Section 5.4 Results, a biological interpretation of the biclusters found may help in this regard. Formally, any row and any column of the original data matrix may belong to zero, one or up to K biclusters. Hence, each decision variable can now take 2^K values. The extension to the MSS problem to identify K solutions would thus lead to a search space containing $O(2^{K^m} \times 2^{K^n})$ feasible solutions. A complet assignment of the column variables does not help in this regard. Indeed, each decision for row i in submatrix k depends on the decisions on row i for the

$K-1$ other submatrices, as stated in THEOREM 3, SECTION 4.2.2.

Consequently, one can no longer hope to find optimal solutions in a reasonable time from gene expression datasets. Instead, we propose to follow a greedy strategy as commonly adopted in several biclustering algorithms [MO04; CC00; PGA15].

A first submatrix is found by solving the maximal sum submatrix optimization problem with CPGC. Next, the values of the selected entries in this solution are replaced in the original matrix by zeros. A zero value is indeed neutral with respect to the maximal sum objective. In other words, any particular entry that has already been selected can again be selected but without any benefit nor loss in the objective value. Such a strategy allows for a possible overlap between several biclusters, neither forcing such overlap nor discarding it *a priori.* This process can be iterated till producing K biclusters.

The time complexity of the method is computed as K times the complexity of the CPGC subroutine. The greedy procedure does not alter the space complexity. Identifying K submatrices with a large total sum is performed within a reasonable time (in the order of a minute with the computational power described in the experiments), which is unsurprising given the performance of the CPGC subroutine.

An implementation of this greedy algorithm, called K-CPGC, is freely available as an R package [Bra20].

5.3 Experiments on human tissues and on *Saccharomyces cerevisiae*

5.3.1 Datasets

In this study, we look for biologically relevant biclusters computed from 35 publicly available gene expression microarray datasets. The first 18 datasets were obtained from human tissues using single-channel Affymetrix chips (Affy), proposed and preprocessed by de Souto et al. [de +08]. Similarly to the latter work, expression values are transformed prior to further analyzes:

$$\boldsymbol{M}_{i,j}{}^{*} \leftarrow \log_2\left(\frac{\boldsymbol{M}_{i,j}}{m_i}\right) ,$$

where m_i is the median of row i and $\boldsymbol{M}_{i,j}{}^{*}$ is the value in row i and column j after transformation. The subsequent 17 datasets, proposed and preprocessed by Jaskowiak, Campello, and Costa Filho [JCC13], were obtained from *Saccharomyces cerevisiae* samples under various controlled conditions using double-channel cDNA (cDNA) technology. These expression values are left unaltered. TABLE 5.1 summarizes this collection by reporting the number of genes and samples measurements in each dataset.

The datasets analyzed during the current study are available in the biclustlib repository [PC17a; PC17c; PC17b].

Table 5.1: Data collection summary.

	Name	Chip	Genes	Samples	Organism	Tissue/Condition
1	armstrong-v1	Affy	1081	72	Human	Blood
2	armstrong-v2	Affy	2194	72	Human	Blood
3	bhattacharjee	Affy	1543	203	Human	Lung
4	chowdary	Affy	182	104	Human	Breast, Colon
5	dyrskjot	Affy	1203	40	Human	Bladder
6	gordon	Affy	1626	181	Human	Lung
7	laiho	Affy	2202	37	Human	Colon
8	nutt-v1	Affy	1377	50	Human	Brain
9	nutt-v2	Affy	1070	28	Human	Brain
10	nutt-v3	Affy	1152	22	Human	Brain
11	pomeroy-v1	Affy	857	34	Human	Brain
12	pomeroy-v2	Affy	1379	42	Human	Brain
13	ramaswamy	Affy	1363	190	Human	Multi-tissue
14	shipp	Affy	798	77	Human	Blood
15	singh	Affy	339	102	Human	Prostate
16	su	Affy	1571	174	Human	Multi-tissue
17	west	Affy	1198	49	Human	Breast
18	yeoh-v1	Affy	2526	248	Human	Bone marrow
19	alpha factor	cDNA	1099	18	Yeast	Cell cycle synchronisation
20	cdc 15	cDNA	1086	24	Yeast	Cell cycle synchronisation
21	cdc 28	cDNA	1044	17	Yeast	Cell cycle synchronisation
22	elutriation	cDNA	935	14	Yeast	Cell cycle synchronisation
23	1mM menadione	cDNA	1050	9	Yeast	Environmental modifications
24	1M sorbitol	cDNA	1030	7	Yeast	Environmental modifications
25	15mM diamide	cDNA	1038	8	Yeast	Environmental modifications
26	25mM DTT	cDNA	991	8	Yeast	Environmental modifications
27	constant 32nM H_2O_2	cDNA	976	10	Yeast	Environmental modifications
28	diauxic shift	cDNA	1016	7	Yeast	Environmental modifications
29	complete DTT	cDNA	962	7	Yeast	Environmental modifications
30	heat shock 1	cDNA	988	8	Yeast	Environmental modifications
31	heat shock 2	cDNA	999	7	Yeast	Environmental modifications
32	nitrogen depletion	cDNA	1011	10	Yeast	Environmental modifications
33	YPD 1	cDNA	1011	12	Yeast	Environmental modifications
34	YPD 2	cDNA	1022	10	Yeast	Environmental modifications
35	Yeast sporulation	cDNA	1006	7	Yeast	Sporulation

5.3.2 Experimental setup

Our objective is to assess to which extent biclustering algorithms and our own K-CPGC approach are able to find biclusters representative of the controlled conditions in our evaluation study. To do so, we analyze the gene subsets found by each approach and we check which of them are significantly enriched.

To compare all approaches on a fair basis, we look for (up to) $K = 10$ biclusters for each controlled experiment. As detailed below, some algorithms do not produce so many solutions while others, including K-CPGC, could be tuned to produce more solutions. Ten biclusters from each data matrix are also considered as reasonable for the subsequent biological interpretation of the results.

All algorithms used in this work are available through R packages: *biclust* [Kai+18], *isa2* [CKB10] and *mssm* [Bra20] for K-CPGC. By default, the control parameters of each biclustering algorithm are those recommended by their original authors. For example, as proposed by the authors of CCA, the original data matrices are initially multiplied by 100 to match the range of data values their control parameters are assuming. The discretization step of xMOTIFs is performed with 10 equally spaced intervals from minimum to maximum. The K-CPGC threshold θ (seeSection 1.2 Interpretation in gene expression analysis) is set to the 75th percentile of expression values, specifically to each dataset. We consider such a threshold as representative of the objective of capturing high expression patterns. Given the performance of the CPGC approach on larger datasets, the K-CPGC method waits for convergence of the CPGC method. In other words, each call to the CPGC method is interrupted whenever the solution is proved optimal, or the best solution has not been improved for 10 seconds. We additionally compare the performance of the CPGC subroutine to the other approaches.

5.3.3 Evaluation

In order to evaluate the biological relevance of the biclusters returned by the various algorithms in this study, a gene enrichment analysis is performed from the selected genes in each bicluster. Specifically, we perform an enrichment step for the selected genes through the Gene Ontology (GO; considering the Biological Process Ontology) [Ash+00] using the *clusterProfiler* R package [Yu+12].

For each of the 35 datasets, each of the 8 algorithms produces up to 10 biclusters. For each bicluster, the enrichment step provides a list of GO terms and false discovery rate (FDR) corrected p-values [BH95]. This p-value refers to the probability of selecting at random n genes out of the N genes from the original expression matrix, with c out of n being associated with the same functional class C. Let s be the true proportion of the N genes associated with the functional class C, the p-value associated with a GO term, or functional class C, is computed as:

$$\Pr(c \mid N, s, n) = \frac{\binom{sN}{c}\binom{(1-s)N}{n-c}}{\binom{N}{n}} \ . \tag{5.5}$$

For each GO term, or functional class C, we calculate the p-value of the current submatrix enrichment as the probability of selecting at random at least c genes of this functional class C in the submatrix, where c is the actual number of genes from this class present in the current submatrix [Li+09]. The smaller the p-values of the terms associated with a submatrix, the more likely the selected genes come from the same biological process.

According to the methodology proposed in [PC17a; Li+09; Pre+06; Ere+12], a specific *bicluster* is considered *enriched* if there is at least one GO term with a FDR corrected p-value below 5%. An algorithm is considered better if it produces more enriched biclusters.

A refined analysis has also been proposed in [PC17a; JCC14] through pairwise comparison of the smallest p-value among the GO terms found from the selected genes returned by each algorithm. Such a comparison could be criticized as it is limited to a single p-value for each algorithm, not necessarily computed for comparable GO terms. Instead, when comparing two algorithms A_1 and A_2, for any GO term considered significantly enriched (FDR corrected p-value $< 5\%$) by both algorithms, one computes a performance difference as:

$$\text{diff}(A_1, A_2) = -\log(\frac{p_{A_1}}{p_{A_2}}) \ . \tag{5.6}$$

The larger $\text{diff}(\text{A}_1, \text{A}_2)$, the smaller the corrected p-value of p_{A_1} compared to p_{A_2} with a positive difference whenever A_1 outperforms A_2.

5.4 Results

While this work focuses on the biological relevance of identified submatrices, it must be stressed that K-CPGC usually finds the best solutions in less than a minute. On average, K-CPGC requires 14.7 seconds, the median being equal to 1.7 seconds. The longest run is performed within 195.7 seconds on dataset 18 (Yeoh-v1).

All reports of time in the present work are computed from experiments on a MacBook Pro (OS version 10.10.5) laptop (Intel i7-2720 CPU 2.20-3.30GHz, 1GB RAM per run) with a single thread using the CPGC implementation presented in Chapter 2 combined with a python script to identify K submatrices. That implementation is equivalent to that of the `mssm` R package developed in this work. Time performances have not been thoroughly studied but are similar with both implementations.

5.4.1 Significantly enriched biclusters

TABLE 5.2 gives a global overview of the ability of the various algorithms to find significantly enriched biclusters among the 35 gene expression datasets from human tissues and *Saccharomyces cerevisae.* The K-CPGC algorithm clearly outperforms the other approaches in this global overview: it is the best in terms of the number of enriched biclusters found. Some algorithms are only able to produce a limited number of distinct biclusters, even less enriched ones. This is due to the specifics of each algorithm. For instance, several random initializations used by ISA do not guarantee to find distinct solutions. Plaid only returns biclusters that offers a better fit to their underlying statistical model than those obtained through random permutations of the original matrix. As for K-CPGC, a slight increase of the threshold θ would lead to producing more biclusters while constraining further the objective of finding high expression patterns. The reported setting looks sound anyway as the prescribed number of 10 biclusters for each dataset is very close to being found with this approach.

Table 5.2: Total number of identified and enriched biclusters.

Algorithm	Biclusters	Enriched biclusters
CCA	349	108
ISA	163	90
K-CPGC	342	**177**
Plaid	102	57
QUBIC	269	107
Spectral	147	44
xMOTIFs	309	60
CPGC	35	35

Results reported for each algorithm on the 35 gene expression datasets from human tissues and *Saccharomyces cerevisae.* The defined target is $K = 10$ biclusters for each dataset, for a maximum of 350 biclusters overall. A bicluster is considered significantly enriched if the subset of genes it contains is associated with at least one GO term with an FDR corrected p-value below 5%.

5.4.2 Statistical assessment

A non-parametric Friedman test [Fri37] is routinely used in the machine learning literature to assess the relative performance of various classification algorithms across several datasets [Dem06]. We adopt here the same methodology to compare biclustering algorithms and our own K-CPGC method. For each dataset, the algorithms under study are ranked according to the number of enriched biclusters they return. Table 5.3 reports the number of enriched biclusters identified per dataset by each algorithm and its associated rank. The last row reports the average rank R_A of algorithm A over all datasets. The Friedman statistic has a χ^2_F distribution with $v-1$ degrees of freedom where N is the number of datasets and v the number of algorithms being tested:

$$\chi^2_F = \frac{12N}{v(v+1)}\left[\sum_{A=1}^{v} R^2_A - \frac{v(v+1)^2}{4}\right] . \tag{5.7}$$

The results presented in Table 5.3 lead to reject the null hypothesis of no difference between the $v = 8$ algorithms over $N = 35$ datasets with an associated p-value equal to 1.33×10^{-11}. It should be highlighted that some algorithms present performance discrepancies regarding the data collections. Namely, ISA, QUBIC and Spectral present higher enrichment performance on the Human tissues collection than on the Yeast collection. Each of the other approaches provides comparable performance on both collections of datasets.

We proceed with a post hoc test, the Hochberg's sequential procedure [Hoc88], to determine whether K-CPGC significantly outperforms the other algorithms. Figure 5.3 reports a diagram of critical differences between the ranks of the various algorithms. The horizontal lines in **bold** represent the differences between ranks that are required for statistical significance. Such intervals increase as more approaches are included in the comparison following the Hochberg's correction for multiple testing. In conclusion, K-CPGC has a significantly better rank compared to all other approaches.

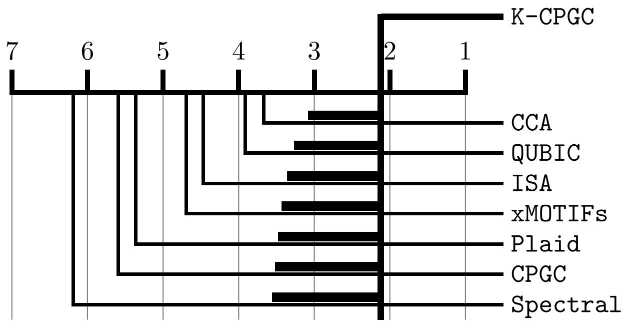


Figure 5.3: Critical difference of ranks. Comparison between the average rank of each algorithm over $N = 35$ datasets, with a 5% level of significance and Hochberg's correction for multiple testing.

Table 5.3: Number of enriched biclusters found by each algorithm on each dataset.

dataset	CCA	ISA	K-CPGC	Plaid	QUBIC	Spectral	xMOTIFs	CPGC
1	1 (5.0)	**7** (1.0)	6 (2.0)	1 (5.0)	0 (7.5)	0 (7.5)	2 (3.0)	1 (5.0)
2	0 (7.5)	**8** (1.0)	6 (2.0)	1 (5.0)	1 (5.0)	0 (7.5)	2 (3.0)	1 (5.0)
3	1 (6.0)	**8** (1.0)	7 (2.0)	1 (6.0)	5 (3.0)	0 (8.0)	3 (4.0)	1 (6.0)
4	1 (5.5)	2 (2.5)	1 (5.5)	2 (2.5)	0 (8.0)	**5** (1.0)	1 (5.5)	1 (5.5)
5	2 (4.0)	**6** (1.5)	2 (4.0)	1 (6.5)	**6** (1.5)	0 (8.0)	2 (4.0)	1 (6.5)
6	1 (5.5)	**7** (1.0)	5 (2.0)	2 (4.0)	0 (7.5)	0 (7.5)	3 (3.0)	1 (5.5)
7	2 (5.0)	3 (4.0)	**8** (1.0)	1 (7.0)	7 (2.5)	7 (2.5)	1 (7.0)	1 (7.0)
8	3 (5.0)	**8** (1.5)	**8** (1.5)	1 (7.5)	6 (3.0)	5 (4.0)	2 (6.0)	1 (7.5)
9	1 (7.0)	2 (4.5)	**7** (1.0)	1 (7.0)	6 (2.5)	6 (2.5)	2 (4.5)	1 (7.0)
10	1 (5.0)	1 (5.0)	**5** (1.0)	1 (5.0)	0 (8.0)	2 (2.0)	1 (5.0)	1 (5.0)
11	0 (7.5)	1 (4.5)	**4** (1.5)	0 (7.5)	**4** (1.5)	1 (4.5)	1 (4.5)	1 (4.5)
12	2 (3.5)	**8** (1.0)	7 (2.0)	0 (7.5)	2 (3.5)	0 (7.5)	1 (5.5)	1 (5.5)
13	0 (7.0)	**3** (1.5)	**3** (1.5)	0 (7.0)	2 (3.5)	0 (7.0)	2 (3.5)	1 (5.0)
14	2 (4.5)	3 (2.5)	1 (6.5)	0 (8.0)	2 (4.5)	**10** (1.0)	3 (2.5)	1 (6.5)
15	**3** (2.0)	**3** (2.0)	2 (4.5)	0 (8.0)	**3** (2.0)	2 (4.5)	1 (6.5)	1 (6.5)
16	1 (5.5)	**8** (1.5)	**8** (1.5)	0 (7.5)	4 (3.0)	0 (7.5)	2 (4.0)	1 (5.5)
17	0 (7.0)	**3** (2.0)	1 (4.5)	0 (7.0)	**3** (2.0)	0 (7.0)	**3** (2.0)	1 (4.5)
18	2 (2.0)	1 (4.0)	**3** (1.0)	0 (7.0)	1 (4.0)	0 (7.0)	0 (7.0)	1 (4.0)
19	8 (2.5)	0 (7.0)	**9** (1.0)	6 (4.0)	8 (2.5)	0 (7.0)	0 (7.0)	1 (5.0)
20	6 (2.0)	3 (4.5)	**10** (1.0)	3 (4.5)	4 (3.0)	0 (8.0)	2 (6.0)	1 (7.0)
21	2 (4.0)	1 (6.5)	**8** (1.0)	4 (2.0)	2 (4.0)	0 (8.0)	2 (4.0)	1 (6.5)
22	6 (2.0)	1 (6.5)	**8** (1.0)	0 (8.0)	3 (4.5)	5 (3.0)	3 (4.5)	1 (6.5)
23	2 (2.5)	0 (7.0)	**4** (1.0)	0 (7.0)	2 (2.5)	0 (7.0)	1 (4.5)	1 (4.5)
24	**4** (1.5)	0 (6.5)	**4** (1.5)	0 (6.5)	0 (6.5)	0 (6.5)	1 (3.5)	1 (3.5)
25	**5** (1.5)	0 (7.0)	**5** (1.5)	3 (3.0)	0 (7.0)	0 (7.0)	1 (4.5)	1 (4.5)
26	**4** (1.5)	0 (7.5)	**4** (1.5)	2 (4.5)	2 (4.5)	0 (7.5)	3 (3.0)	1 (6.0)
27	**4** (1.0)	1 (6.5)	3 (2.0)	2 (4.0)	2 (4.0)	0 (8.0)	2 (4.0)	1 (6.5)
28	**5** (1.0)	0 (7.0)	4 (2.0)	2 (3.5)	2 (3.5)	0 (7.0)	0 (7.0)	1 (5.0)
29	3 (3.5)	1 (6.0)	**6** (1.0)	3 (3.5)	4 (2.0)	0 (8.0)	1 (6.0)	1 (6.0)
30	**5** (1.0)	0 (7.5)	2 (2.5)	1 (5.0)	1 (5.0)	0 (7.5)	2 (2.5)	1 (5.0)
31	4 (2.5)	1 (5.5)	**5** (1.0)	4 (2.5)	1 (5.5)	0 (8.0)	1 (5.5)	1 (5.5)
32	**8** (1.5)	0 (7.5)	5 (4.5)	7 (3.0)	**8** (1.5)	0 (7.5)	5 (4.5)	1 (6.0)
33	6 (3.0)	0 (8.0)	7 (2.0)	3 (4.0)	**9** (1.0)	1 (6.0)	1 (6.0)	1 (6.0)
34	6 (2.0)	0 (7.5)	4 (3.0)	3 (4.5)	**7** (1.0)	0 (7.5)	3 (4.5)	1 (6.0)
35	**7** (1.0)	0 (6.5)	5 (2.0)	2 (3.0)	0 (6.5)	0 (6.5)	0 (6.5)	1 (4.0)
avg. rank	3.7	4.5	**2.1**	5.4	3.9	6.2	4.7	5.6

Numbers in parentheses are the associated ranks. In case of ties, average ranks are assigned. The last row corresponds to the algorithm ranks averaged over the 35 datasets. Best performances are highlighted in bold. It is observed that all enriched biclusters have different GO enrichment. Note that CPGC is the original algorithm identifying a single submatrix of maximal sum per dataset.

The analysis so far has been focusing on the number of biclusters for which the subset of genes they contain is associated with *at least one significant GO term.* Since K-CPGC and CCA are the best methods according to this analysis, one looks now at *all significant GO terms* identified by both algorithms. FIGURE 5.4 reports the difference metric between p-values of these GO terms according to EQUATION (5.6). It shows that K-CPGC outperforms CCA in this regard since it exhibits a positive difference in 638 out of 1,054 cases. In other words, K-CPGC identifies gene subsets which are generally estimated more significant as they correspond more often to lower p-values.

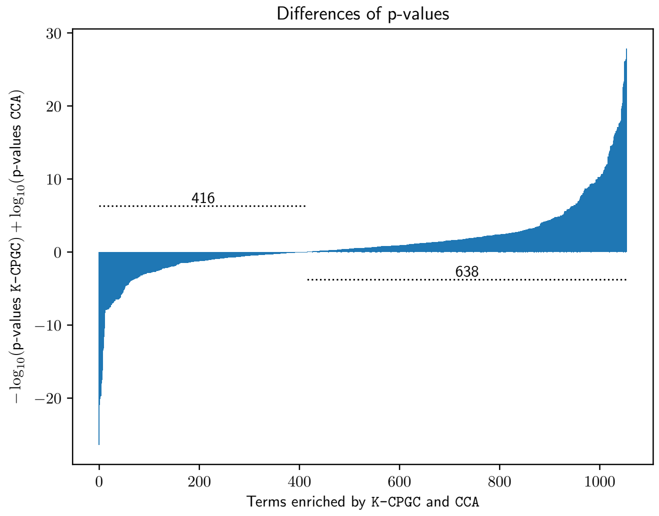


Figure 5.4: Comparison of K-CPGC and CCA p-values for enriched GO terms. This figure presents the (logarithmic) ratio of corrected p-values associated with each GO term identified by both K-CPGC and CCA. Positive values (638 GO terms) are in favor of K-CPGC.

5.4.3 Biological relevance

We further analyze the actual gene subsets identified by K-CPGC from *Saccharomyces cerevisae* samples to check whether the 20 most significantly enriched GO terms it identifies in each dataset are consistent with the controlled conditions under which these experiments were conducted. K-CPGC produces up to 10 submatrices of maximal sum per dataset. Each submatrix is associated with a subset of genes and a subset of samples. Up to 10 gene subsets are identified by K-CPGC per dataset. An enrichment step provides a list of GO terms and FDR corrected p-values [BH95] for each gene subset.

The 20 most significantly enriched GO terms are reported in TABLES 2 to 18 (in Additional tables). Each table presents the results of one dataset. One GO term is listed per row and the associated rank is provided in the first column. The second column gives the unique GO identifier of a term. The third column contains a short description for each term. Finally, the corrected p-value of the enrichment step is given in the last column. These tables confirm that identified GO terms are consistent with the controlled conditions under which these experiments were conducted.

The GO terms identified in the first four datasets, representative of cell cycles, are associated with some form of biogenesis, including ribosome, RNA, peptide and macromolecules synthesis. The GO terms identified in the next 12 datasets are indeed associated with various forms of response to stress-induced environments, including many representatives of the response to the stimulus, oxidation-reduction processes, and cellular responses to stress. Some GO terms also refer to generic responses to stress which are less specific to the controlled condition. For example, in the *complete DTT* dataset, many GO terms relate to alteration in the general patterns of protein biosyntheses as reported by Miller *et al.* [MXG79]. The last experiment related to yeast sporulation includes GO terms referring to cell cycle, sporulation, and reproductive processes.

5.4.4 Influence of the parameters

The proposed K-CPGC method could be considered as including two control parameters: the number K of biclusters one looks for and the threshold θ defining the level of expression above which interesting patterns are searched.

We fix θ to the 75th percentile of expression values and we argue that this is a reasonable choice to find high expression patterns. Yet, the user may be interested in playing with this parameter as it influences indirectly the sizes of the biclusters found. In the limit, if θ is set below the minimal expression value, all entries of the normalized matrix will be positive and the solution to the maximal sum problem is trivially identified as the full matrix. Similarly, if θ is set above the maximal expression value, all entries of the normalized matrix become negative and the optimal solution is the empty matrix. An intermediate θ value between these extreme cases is typically chosen. For a fixed data matrix *increasing* θ tends to produce *smaller* biclusters. The actual bicluster sizes found is difficult to predict exactly, as it also depends on the actual distribution of expression value in the matrix, but the analyst may easily play with θ to find biclusters of interest.

For a fair comparison between all algorithms, we fix the maximal number of biclusters to be found to $K = 10$. In practice, however, this is not a critical choice since the analyst can start with $K = 1$ and use the proposed gene enrichment analysis to check whether the successive biclusters returned by increasing K are still significantly enriched. Let us consider the evolution of the GO enrichment as a function of K.

We report here the evolution of the number of enriched biclusters and the evolution of the number of enriched GO terms as a function of K. Value of parameter K ranges from 1 up to no significant improvement.

Each dataset is separately normalized by subtracting a threshold θ to all matrix entries before using K-CPGC. The threshold θ is set to the 75th percentile of expression values, specifically to each dataset, as default value. We consider such a threshold as representative of the objective of capturing high expression patterns. We also examine the 65th and 85th percentiles of expression values to complement analyzes. We report as K-CPGC_0_**xx** the results of our approach after normalization through subtraction of the **xx**th percentile.

FIGURE 5.5 presents the evolution of the number of enriched biclusters as a function of K. We observe a rapidly growing number of enriched biclusters up to the chosen value in the main manuscript ($K = 10$). Our approach, with its three different θ values, essentially identifies more enriched biclusters than other approaches in the first part of the graph. Nevertheless, CCA produces more enriched biclusters in the long run. We explain the absence of important improvements in our approach by the size of the discovered gene subsets. Indeed, biclusters are identified only when there is some signal remaining in the matrix. Therefore, identifying larger subsets of genes is penalized, as the remaining signal depends on the previously identified (and masked) biclusters. TABLE 5.4 presents the average size of gene subsets for $K = 10$ and $K = 40$. We observe that the difference between CCA and the three variants of K-CPGC increases with K. A data analyst would consider smaller values of K given the size of the datasets, the size of the identified subsets of genes and the results from Figure 5.5.

FIGURE 5.6 presents the number of different enriched GO terms identified by each approach for increasing values of K. We observe a rapidly growing number of enriched GO terms up to the chosen value in the main manuscript ($K = 10$), for most approaches, as in FIGURE 5.5. It confirms the ability of K-CPGC to quickly identify a large part of the relevant signal. It is furthermore noticeable that K-CPGC_0_65 and K-CPGC_0_75 identify more GO terms than all others approaches for any value of the parameter K.

Interpretations of both graphs can be reunited by observing that:

1. K should be large enough to observe differences regarding the enrichment,
2. K should, however, be small enough to ensure that there still is a signal to be found, regarding the number of biclusters and the number of GO terms.

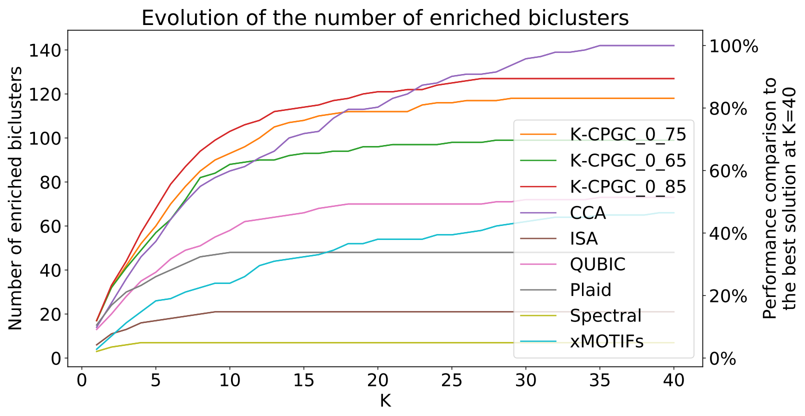


Figure 5.5: Evolution of the number of enriched biclusters as a function of K. The left axis presents the cumulated number of biclusters identified as K, the maximal number of biclusters to be found, increases. The right axis presents a performance computed as the number of enriched biclusters divided by the number of enriched biclusters identified by the best algorithm at $K = 40$. The 100% performance corresponds to identifying 142 enriched biclusters.

Table 5.4: Size of gene subsets averaged on 17 *Saccharomyces cerevisae* datasets.

Name	$K = 10$	$K = 40$
K-CPGC_0_65	129	98
K-CPGC_0_75	110	93
K-CPGC_0_85	81	69
CCA	84	40
ISA	82	82
QUBIC	170	122
Plaid	130	126
Spectral	6	6
xMOTIFs	38	17

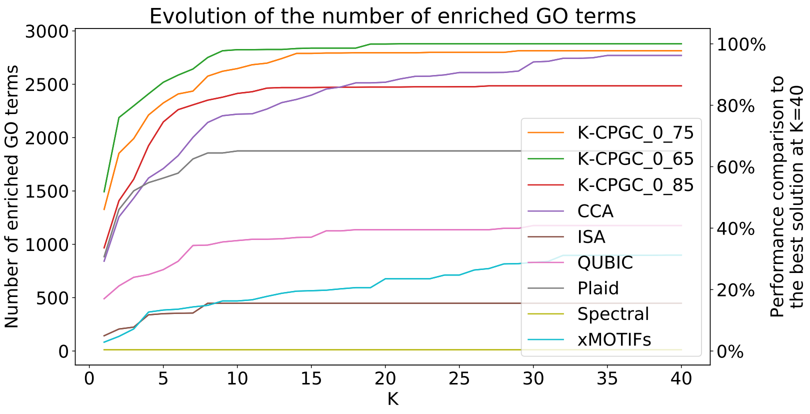


Figure 5.6: Evolution of the number of enriched GO terms as a function of K. The left axis presents the cumulated number of different GO terms identified as K, the maximal number of biclusters to be found, increases. The right axis presents a performance computed as the number of different enriched GO terms identified divided by the number of different enriched GO terms identified by the best algorithm at $K = 40$. The 100% performance corresponds to identifying 2879 enriched GO terms.

5.5 Discussion

The experiments and results reported in this work show that the K-CPGC method outperforms six well-known biclustering algorithms to identify biologically relevant gene subsets among subgroups of samples. The various algorithms are compared essentially based on their ability to return gene subsets which are associated with significantly enriched GO terms. One could consider that such a performance assessment validates only part of the results as it focuses on the genes (rows) and does not validate *a posteriori* the identified subgroups of samples (columns). This is actually a common limitation of the assessment of biclustering methods from gene expression data [PC17a]. For the *Saccharomyces cerevisae* experiments reported here, there is no gold standard in terms of subgroups of samples to be identified. Yet, these subgroups of samples are, at least indirectly, validated because their components directly influence the subsets of genes which are returned. This is particularly clear for the CPGC approach as one looks for a rectangular submatrix of maximal sum and the returned genes are directly constrained by the selected samples in such a submatrix. This is also true for biclustering algorithms since, for instance, they look for homogeneous expression patterns both across rows and columns. Notwithstanding, in a medical context, for example, the actual samples are typically associated with specific patients. In such a case, direct validation of the identified subgroups of samples could be performed by comparing these subgroups with actual clinical annotations. Interpreting the evaluation of unsupervised method on their ability to recover an expected structure is difficult, however. As an illustrative example, Padilha *et al.* [PC17a] evaluated the ability of several biclustering algorithms to recover the predefined sample classes. They showed that the best methods are biased towards methods that force every row and every column to be biclustered.

The K-CPGC approach can also be used to find low expression patterns instead of high-level ones simply by considering the opposite of the normalized data matrix. These are two obvious possibilities but it is straightforward to generalize this approach. For instance, if one would be interested in finding patterns of average expression values (neither over-expressed nor under-expressed), one can easily transform the original matrix to a new one, *e.g.* according to a Gaussian or RBF kernel, in which a higher value would represent an original entry closer to the average expression value. This average (or median) value can be computed overall, row-wise or column-wise. Countless variants are easy to define and illustrate the flexibility of this approach.

5.6 Conclusions

We propose a novel algorithm, K-CPGC, to find K non-redundant and possibly overlapping submatrices of maximal sum from a large gene expression matrix. The returned solutions have the same bi-dimensional structure as biclusters produced by existing biclustering algorithms. Yet, the mathematical objective is different and more explicitly optimized with the proposed methodology. Indeed, the role of a matrix entry (i, j) in a submatrix is clear: its contribution to the decision of including gene i and sample j in the submatrix is $\boldsymbol{M}_{i,j}$. It follows that the contribution of each gene in the definition of a gene subset, respectively each sample, can be easily computed as the sum of matrix entries for each of the selected samples, respectively genes.

Through enrichment analysis performed on 35 gene expression datasets from human tissues and *Saccharomyces cerevisae* samples, we show that the K-CPGC method outperforms biclustering algorithms when looking for biologically relevant gene subsets. Not only is our approach efficient, but it also identifies more enriched biclusters than other biclustering methods. The K-CPGC approach provides stronger results (lower p-values of gene subsets or GO terms) than these alternative algorithms. These results illustrate the benefits of the proposed approach in terms of biological enrichments and biological relevance.

The K-CPGC is, however, not limited to gene expression analysis. For example, Liu and Wang [LW03] use a drug activity dataset consisting of a matrix of 10,000 compounds with 30 features for each compound. The K-CPGC algorithm could be used to identify subsets of compounds presenting highly valued entries in subsets of features.

This method has the potential to find relevant gene subsets across various -omics technologies since, unlike biclustering algorithms, it does not look for homogeneous gene expression values. The specific search order it follows could also be easily adapted to discover small relevant submatrices rather than large biclusters, hence focusing on rare but relevant expression patterns.

The K-CPGC method and the biclustering algorithms it is compared to are *unsupervised* methods since they do not require any particular annotation of the analyzed samples. A different and interesting setting arises when the samples, or at least a fraction of them, are labeled according to various conditions or clinical variables. In such a context, a new objective would be to identify subsets of genes that are maximally relevant to discriminate between subsets of samples from different conditions.

Mining submatrices of maximal sum quicker

6

Constraint programming approaches presented in previous chapters are dedicated to solving a particular task. Modifications of the objective function and addition of constraint may not always be trivial. It may require changing some or all functions of the presented algorithms. In this chapter, we design a constraint that can be used in any constraint programming (CP) model. A tighter bound and new filtering rules are also provided to define a new model for the maximal sum submatrix problem. A CP implementation using that model requires less computational time than an implementation using a model with an upper bound and filtering rules as presented in Chapter 2, particularly for the difficult instances. The improved upper bound and stronger filtering reduce the size of the search space by removing infeasible solutions. Evaluation on synthetic data suggests that the new model is globally preferable in terms of computational time and size of the search space than the original approach.

6.1 Introduction

cpgc, a constraint programming (CP) approach relying on an upper bound to the weight of the maximal sum submatrix is introduced in Chapter 2, Section 2.3.2.3. That approach has two limitations that this chapter aims at overcoming:

1. cpgc has shorter time requirements to solve instances than competing approaches. Such better performance is particularly clear with matrices containing only a few positive entries or when the optimal solution covers a small fraction of the input matrix. The upper bound struggles to solve matrix instances containing many positive entries, however.

2. cpgc is a dedicated algorithm. Efficient propagation of additional constraint might require a thorough modification of the algorithm.

6.1.1 Contributions

The main contributions of the chapter are:

- The formal definition of the `SubmatrixWeight` constraint to model the sum of entries covered by a submatrix.
- The definition and implementation of an upper bound and filtering rules specifying a competitive constraint programming approach.
- An evaluation of the benefits of the new approach on synthetic data as compared to CPGC, the CP approach with global constraint, defined in Chapter 2.

6.2 Constraint programming model

CP is a flexible programming paradigm for solving (discrete) optimization problems. A constraint programming model is a triplet (V, D, C) where V is the set of variables, D their domains, and C is a set of constraints. A feasible solution is an assignment of the variables to values of their domains such that all constraints are satisfied. Constraints are exploited to reduce iteratively the number of variable assignments to consider. Once all unfeasible values are removed from the domains of the variables, the solver selects a variable $X \in V$ that is nonfixed and recursively calls itself while assigning a value to this variable. By exploring a depth-first-search (DFS) tree, the solver either reaches a solution or backtracks when the domain of a variable becomes empty.

6.2.1 Model

We model a submatrix using the set variables $\mathcal{R}$ and $\mathcal{C}$ for the rows and columns, respectively. A variable w denotes the weight of the submatrix $(\mathcal{R}; \mathcal{C})$. The maximal sum submatrix can be modeled by maximizing the variable w under the constraint that

$$w = \sum_{i \in \mathcal{R}, j \in \mathcal{C}} M_{i,j} \quad . \tag{6.1}$$

The domain of a set variable $\mathcal{S}$ is the set of all the (sub)sets of $\mathcal{S}$. That domain is approximated by a closed interval denoted $\left[\mathcal{S}^{\in}, \mathcal{S}^{\in} \cup \mathcal{S}^{?}\right]$, in CP, by means of the set domain bounds representation [Ger97].

- $\mathcal{S}^{\in}$ are the mandatory elements,
- $\mathcal{S}^{?}$ are the possible additional ones ($\mathcal{S}^{\in} \cap \mathcal{S}^{?} = \emptyset$).

Such an interval represents all the sets in between those two bounds according to the inclusion relation $\{\mathcal{S} \mid \mathcal{S}^{\in} \subseteq \mathcal{S} \subseteq (\mathcal{S}^{\in} \cup \mathcal{S}^{?})\}$. A set variable is fixed whenever it contains a single set in its domain. This situation happens when set interval bounds are equal, or equivalently, the possible set is empty: $\mathcal{S}^{?} = \emptyset$.

6.2.2 Notations

Definition 8. **Mandatory and possible rows and columns**
Let $\mathcal{R}^{\in}$ (resp. $\mathcal{C}^{\in}$) denote the mandatory *rows (resp. columns) and $\mathcal{R}^{?}$ (resp. $\mathcal{C}^{?}$) the* possible *rows (resp. columns) that might be added to the mandatory rows (resp. columns).*

Definition 9. **Partial assignment and optimal extension**
Let $(\mathcal{R}^{\in}, \mathcal{R}^{?}, \mathcal{C}^{\in}, \mathcal{C}^{?})$ and $(\mathcal{R}^{\in *}, \emptyset, \mathcal{C}^{\in *}, \emptyset)$ denote a partial assignment and the optimal solution extending it, respectively, with $\mathcal{R}^{\in} \subseteq \mathcal{R}^{\in *}$, $\mathcal{R}^{\in *} \subseteq (\mathcal{R}^{\in} \cup \mathcal{R}^{?})$, $\mathcal{C}^{\in} \subseteq \mathcal{C}^{\in *}$, $\mathcal{C}^{\in *} \subseteq (\mathcal{C}^{\in} \cup \mathcal{C}^{?})$. Let $(\mathcal{R}^{\in *}; \mathcal{C}^{\in *})$ denote the submatrix corresponding to the optimal assignment extending the partial assignment.

We refer to $(\mathcal{R}^{\in *}; \mathcal{C}^{\in *})$ as the *best extension* since the optimal solution extending a partial assignment might be suboptimal for the pure maximal sum submatrix problem, which has no variable fixed. We refer to the sum of entries covered by the best extension as the *best value*, or simply *best*, with $best = \sum\limits_{i \in \mathcal{R}^{\in *}, j \in \mathcal{C}^{\in *}} \boldsymbol{M}_{i,j}$.

Example 2.
Let us consider the following matrix with the partial assignment $(\{r_1\}, \{r_2, r_3, r_4, r_5, r_6\}, \{c_1\}, \{c_2, c_3, c_4, c_5\})$. *Mandatory variables are depicted in grey. The best extension is defined by variables* $(\{r_1, r_4, r_5\}; \{c_1, c_2\})$, *depicted in light-grey, and best* $= 12$.

	c1	c2	c3	c4	c5
r1	*2*	*-3*	*-2*	*-8*	*-1*
r2	*-2*	*-7*	*-7*	*-5*	*3*
r3	*-10*	*1*	*6*	*1*	*3*
r4	*-1*	*7*	*-2*	*-3*	*3*
r5	*4*	*3*	*-1*	*5*	*-4*
r6	*-4*	*-1*	*-3*	*3*	*-5*

Definition 10. **Partial sum of a row or column**
Let ${r_i}^{psum}$ denote the partial sum *of a row i with ${r_i}^{psum} = \sum_{j \in \mathcal{C}^{\in}} \boldsymbol{M}_{i,j}$. Let the partial sum of column j be symmetrically defined: ${c_j}^{psum} = \sum_{i \in \mathcal{R}^{\in}} \boldsymbol{M}_{i,j}$.*

Definition 11. **Contribution of an element**
Let $\mathbb{C}(\mathbf{a} \longrightarrow \mathbf{b})$ denote the contribution *(or impact) of* $\mathbf{a}$ *on the computation of* $\mathbf{b}$.

Definition 11 let us clarify concepts as follows:

- $\mathbb{C}(i \in \mathcal{R}^{\in *} \longrightarrow best)$ is the contribution of row $i \in \mathcal{R}^{\in *}$ to the weight (or sum of entries) of the best extension $(\mathcal{R}^{\in *}; \mathcal{C}^{\in *})$:

$$\mathbb{C}(i \in \mathcal{R}^{\in *} \longrightarrow best) = \sum_{j \in \mathcal{C}^{\in *}} \boldsymbol{M}_{i,j} \ . \tag{6.2}$$

- $\mathbb{C}(i \notin \mathcal{R}^{\in *} \longrightarrow best)$ is the contribution of $i \notin \mathcal{R}^{\in *}$ to the best value:

$$\mathbb{C}(i \notin \mathcal{R}^{\in *} \longrightarrow best) = 0 \ . \tag{6.3}$$

The variable w of our model of the maximal sum submatrix corresponds to the weight of submatrix $(\mathcal{R}; \mathcal{C})$. The `SubmatrixWeight` constraint enforcing $w = \sum_{i \in \mathcal{R}, j \in \mathcal{C}} \boldsymbol{M}_{i,j}$ is introduced in the next section. Filtering algorithms tightening the domain of the variables and dominance rules are subsequently presented, achieving the same level of filtering as the CPGC approach introduced in Chapter 2.

6.3 The SubmatrixWeight constraint

The maximal sum submatrix can be modeled by maximizing the variable w under the constraint that

$$w = \sum_{i \in \mathcal{R}, j \in \mathcal{C}} \boldsymbol{M}_{i,j} \ . \tag{6.4}$$

The latter constraint is equivalent to the `SubmatrixWeight` constraint defined below:

$$\texttt{SubmatrixWeight}(\mathcal{R}, \mathcal{C}, \boldsymbol{M}, w) \ . \tag{6.5}$$

That constraint enforces variable w to be equal to the weight of submatrix $(\mathcal{R}; \mathcal{C})$ given a matrix $\boldsymbol{M}$. The domain of $\mathcal{R}$ (resp. $\mathcal{C}$) is the set of all the (sub)sets of rows (resp. columns) of the matrix $\boldsymbol{M}$ and the domain of w is $\mathbb{R}$. A submatrix of maximal sum is an assignment of $\mathcal{R}$, $\mathcal{C}$, and w, such that the value of w is maximal and constraint `SubmatrixWeight` holds.

6.3.1 Filtering the domain of the weight variable

The objective of a filtering algorithm is to remove values that do not participate in any solution of the constraint. Effective filtering algorithms remove as many inconsistent values as possible. We are interested in achieving bound-consistency for the `SubmatrixWeight` constraint. In a bound-consistent constraint, every variable bound (maximum or minimum) occurs in a solution of the constraint.

Definition 12. **Values $w^{\min}$ and $w^{\max}$**
Let w^{min} (resp. w^{max}) denote the minimal (resp. maximal) value in the domain of variable w: $\left[w^{min}, w^{max}\right]$.

Two opposite optimization problems must be solved to filter the domain of w and achieve bound-consistency: minimizing and maximizing the sum of entries covered by sets of rows and columns.

Definition 13. **Values $\underline{w}$ and $\bar{w}$**
Let $\underline{w}$ and $\bar{w}$ denote the optimal values to the following optimization problems:

$$\underline{w} = \min \sum_{i \in \mathcal{R}, j \in C} \boldsymbol{M}_{i,j} \ , \qquad \bar{w} = \max \sum_{i \in \mathcal{R}, j \in C} \boldsymbol{M}_{i,j} \ . \tag{6.6}$$

Values $\underline{w}$ and $\bar{w}$ can be used to filter the domain of w:

$$\left[w^{\min}, w^{\max}\right] \leftarrow \left[\max(w^{\min}, \underline{w}), \min(w^{\max}, \bar{w})\right] \ . \tag{6.7}$$

Unfortunately, finding $\bar{w}$ and $\underline{w}$ are $\mathcal{NP}$-hard problems, as stated in Theorem 1. However, we can design good bounds for them. An upper bound to $\bar{w}$, *baseUB*, which can be computed in $O(m+n)$, has been introduced in Chapter 2:

$$\begin{aligned} baseUB = &\sum_{i \in \mathcal{R}^{\in}} \left[\sum_{j \in C^{\in}} \boldsymbol{M}_{i,j} + \sum_{j \in C^{?}} \max\left(0, \boldsymbol{M}_{i,j}\right) \right] \\ &+ \sum_{i \in \mathcal{R}^{?}} \max \left[0, \sum_{j \in C^{\in}} \boldsymbol{M}_{i,j} + \sum_{j \in C^{?}} \max\left(0, \boldsymbol{M}_{i,j}\right) \right] \ . \end{aligned} \tag{6.8}$$

A lower bound to $\underline{w}$, *baseLB*, can be defined from *baseUB* by replacing all maximization terms by minimization terms. The filtering on the domain of w becomes:

$$\left[w^{\min}, w^{\max}\right] \leftarrow \left[\max\left(w^{\min}, baseLB\right), \min\left(w^{\max}, baseUB\right)\right] \ . \tag{6.9}$$

We define ${r_i}^{pub}$ and ${c_j}^{pub}$ (see Definition 14) and rewrite *baseUB* for the sake of clarity: $baseUB = \sum_{i \in \mathcal{R}^{\in}} {r_i}^{pub} + \sum_{i \in \mathcal{R}^{?}} \max\left(0, {r_i}^{pub}\right)$.

Definition 14. **Bound to a row or a column**
Let ${r_i}^{pub}$, denoted the bound to the row *i, be the partial sum of i plus all the positive entries in the possible columns:*

$${r_i}^{pub} = {r_i}^{psum} + \sum_{j \in C^{?}} \max\left(0, \boldsymbol{M}_{i,j}\right) \ . \tag{6.10}$$

Let ${c_j}^{pub}$ denote the bound to the column *j:*

$${c_j}^{pub} = {c_j}^{psum} + \sum_{i \in \mathcal{R}^{?}} \max\left(0, \boldsymbol{M}_{i,j}\right) \ . \tag{6.11}$$

6.3.2 Filtering the domain of the set variables

It happens that decisions leading to unfeasible solutions can be detected before taking them. Filtering such decisions from the domains of the variables reduces the size of the solution space.

Definition 15. **Inclusion and exclusion operations**
Let $\mathcal{S} \leftarrow e$ denote a situation where element e is included in the mandatory set $\mathcal{S}^{\in}$:

$$\mathcal{S}^{\in} \leftarrow \mathcal{S}^{\in} \cup \{e\} \ , \quad \textit{and} \quad \mathcal{S}^{?} \leftarrow \mathcal{S}^{?} \setminus \{e\} \ . \tag{6.12}$$

Let $\mathcal{S} \setminus e$ denote a situation where element e is excluded from the possible set:

$$\mathcal{S}^{?} \leftarrow \mathcal{S}^{?} \setminus \{e\} \ , \quad \textit{and} \quad e \notin \mathcal{S}^{\in} \ . \tag{6.13}$$

Let $v_{\mathcal{S}\leftarrow e}$ and $v_{\mathcal{S}\setminus e}$ denote the value v under the hypothesis that $\mathcal{S} \leftarrow e$ and $\mathcal{S} \setminus e$, respectively.

Filtering rule in Equation (6.14) defines a sufficient condition to keep only assignments of $\mathcal{R}$ including a row i, therefore forcing $\mathcal{R} \leftarrow i$. Excluding i from the set variable $\mathcal{R}$ would lead to unfeasible solutions: $baseUB_{\mathcal{R}\setminus i} < w^{\min} \implies w^{\max}_{\mathcal{R}\setminus i} < w^{\min}$ and the domain of w would be empty. Similarly, Equation (6.15) defines a sufficient condition to exclude row i from $\mathcal{R}$ ($\mathcal{R} \setminus i$).

$$\forall(\mathcal{R}^{\in}, \mathcal{R}^{?}, \mathcal{C}^{\in}, \mathcal{C}^{?}), i \in \mathcal{R}^{?} : baseUB_{\mathcal{R}\setminus i} < w^{\min} \implies \mathcal{R} \leftarrow i \ , \tag{6.14}$$
$$\forall(\mathcal{R}^{\in}, \mathcal{R}^{?}, \mathcal{C}^{\in}, \mathcal{C}^{?}), i \in \mathcal{R}^{?} : baseUB_{\mathcal{R}\leftarrow i} < w^{\min} \implies \mathcal{R} \setminus i \ . \tag{6.15}$$

Those two filtering rules are evaluated for all possible rows in $O(m)$ at each node of the search tree from incremental updates of ${r_i}^{pub}$:

$$baseUB_{\mathcal{R}\leftarrow i} = baseUB + \min\left(0, {r_i}^{pub}\right) \ , \qquad \text{and} \tag{6.16}$$
$$baseUB_{\mathcal{R}\setminus i} = baseUB - \max\left(0, {r_i}^{pub}\right) \ . \tag{6.17}$$

6.3.3 Dominance rules

Dominance rules define decisions leading to suboptimal solutions and possible alternative decisions leading to better solutions. Considering only the alternative decisions benefits to the search by reducing the size of the search tree.

A partial assignment may satisfy the maximization constraint on w only if its best extension may improve upon the best solution found so far. Invalid assignments are filtered out by fixing the maximal weight so far as the lower bound of the domain of w. THEOREM 6 reduces the domain of w by defining a new, possibly better, submatrix in each node of the search tree. Fortunately, that lower bound comes at no additional computational cost as it is already computed in *baseUB*.

THEOREM 6.
For any partial assignment $\left(\mathcal{R}^{\in}, \mathcal{R}^{?}, \mathcal{C}^{\in}, \mathcal{C}^{?}\right)$,

$$\left[w^{min}, w^{max}\right] \leftarrow \left[\max(w^{min}, \sum_{i\in\mathcal{R}^{\in}, j\in\mathcal{C}^{\in}} \boldsymbol{M}_{i,j}), \min\left(w^{max}, baseUB\right)\right] \ . \quad (6.18)$$

Proof. Let $\sum_{i\in\mathcal{R}^{\in}, j\in\mathcal{C}^{\in}} \boldsymbol{M}_{i,j}$ be the weight of the extension $(\mathcal{R}^{\in}, \emptyset, \mathcal{C}^{\in}, \emptyset)$ of a partial assignment. Let us define any other extension $(\mathcal{X}, \emptyset, \mathcal{Y}, \emptyset)$, with $\mathcal{R}^{\in} \subseteq \mathcal{X} \subseteq \left(\mathcal{R}^{\in} \cup \mathcal{R}^{?}\right)$ and $\mathcal{C}^{\in} \subseteq \mathcal{Y} \subseteq \left(\mathcal{C}^{\in} \cup \mathcal{C}^{?}\right)$. Such $(\mathcal{X}, \emptyset, \mathcal{Y}, \emptyset)$ is not a solution to the maximal sum submatrix problem if

$$\sum_{i\in\mathcal{X}, j\in\mathcal{Y}} \boldsymbol{M}_{i,j} < \sum_{i\in\mathcal{R}^{\in}, j\in\mathcal{C}^{\in}} \boldsymbol{M}_{i,j} \quad (6.19)$$

as its weight is not maximum. The partial assignment $\left(\mathcal{R}^{\in}, \mathcal{R}^{?}, \mathcal{C}^{\in}, \mathcal{C}^{?}\right)$ and its extensions can be filtered out of the search space. □

THEOREM 7 defines a rule to detect a row that never belongs to the best extension to an assignment.

THEOREM 7.
Given a partial assignment, any row $i \in \mathcal{R}^{?}$ *with* $r_i^{pub} < 0$ *never belongs to the best extension:*

$$\forall i \in \mathcal{R}^{?} \ : \ r_i^{pub} < 0 \implies i \notin \mathcal{R}^{\in *} \ . \quad (6.20)$$

Proof. Let us define $\mathcal{X}$, $\mathcal{Y}$ and i such that:

$$\mathcal{X} \in \left\{\mathcal{X} \mid \mathcal{R}^{\in} \subseteq \mathcal{X} \subseteq \left(\mathcal{R}^{\in} \cup \mathcal{R}^{?} \setminus \{i\}\right)\right\} \ , \quad (6.21)$$

$$i \in \mathcal{R}^{?} \quad \wedge \quad r_i^{pub} < 0 \ , \quad (6.22)$$

$$\mathcal{Y} \in \left\{\mathcal{Y} \mid \mathcal{C}^{\in} \subseteq \mathcal{Y} \subseteq \left(\mathcal{C}^{\in} \cup \mathcal{C}^{?}\right)\right\} \ . \quad (6.23)$$

For any partial assignment $(\mathcal{X}, \{i\}, \mathcal{Y}, \emptyset)$:

$$w_{\mathcal{R}\leftarrow i} = w_{\mathcal{R}\setminus i} + \sum_{j\in\mathcal{Y}} \boldsymbol{M}_{i,j} \ , \quad (6.24)$$

$$\text{and} \quad r_i^{pub} = \sum_{j\in\mathcal{C}^{\in}} \boldsymbol{M}_{i,j} + \sum_{j\in\mathcal{C}^{?}} \max\left(0, \boldsymbol{M}_{i,j}\right) \geq \sum_{j\in\mathcal{Y}} \boldsymbol{M}_{i,j} \ , \quad (6.25)$$

$$\text{then} \quad r_i^{pub} < 0 \implies \sum_{j\in\mathcal{Y}} \boldsymbol{M}_{i,j} < 0 \implies w_{\mathcal{R}\leftarrow i} < w_{\mathcal{R}\setminus i} \ . \quad (6.26)$$

Any extension $(\mathcal{X}, \emptyset, \mathcal{Y}, \emptyset)$ is a better solution than $(\mathcal{X} \cup \{i\}, \emptyset, \mathcal{Y}, \emptyset)$. Consequently, i never belongs to the best extension:

$$i \notin \left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right) \implies i \notin \mathcal{R}^{\in *} \ . \tag{6.27}$$

□

The dominance rules in Theorem 7 are evaluated for all the possible rows in $O(m)$ at each node of the search tree. The time complexity is explained by incremental updates of r^{psum}, r^{pub}: all ${r_i}^{psum}$ and ${r_i}^{pub}$ for all possible rows are updated in $O(m)$ at each node of the search tree.

Note that Equations (6.14) and (6.15) and Theorem 7 have symmetrical counterparts for the columns, which can also be used to reduce the solution space at each node of the search tree.

6.4 A tighter bound and new dominance rules

The `SubmatrixWeight` constraint and the dominance rules proposed in Section 6.3 achieve the same level of filtering as the global constraint implemented in Chapter 2. This section introduces a tighter upper bound to $\bar{w}$ and new dominance rules to reduce further the domains of the variables. Many positive entries contributing to *baseUB* do not contribute to the tighter bound, as suggested by theorems and definitions presented in this section. Note that most of the theorems and definitions have a symmetrical counterpart on the columns or the rows. All dominance rules are presented in a different subsection from the bound subsection, even if some allow to define the bound parts, to distinguish them from the bound.

6.4.1 Tighter upper bound

Only columns $\{j \in C^{?} \mid {c_j}^{pub} \geq 0\}$ should contribute to $\bar{w}$ as none of the columns in $\{j \in C^{?} \mid {c_j}^{pub} < 0\}$ belong to the best extension (from the symmetrical counterpart of Theorem 7). By extension, only columns $\{j \in C^{?} \mid {c_j}^{pub} \geq 0\}$ should contribute to *baseUB* as well as its constituents r^{pub}.

Definition 16. **Modified-bound of a row or a column**
The modified-bound ${r_i}^{pub+}$ *is an adjustment of* ${r_i}^{pub}$ *to only consider columns* $\{j \in C^{?} \mid {c_j}^{pub} \geq 0\}$*:*

$$ {r_i}^{pub+} = {r_i}^{psum} + \sum_{j \in C^{?} \mid {c_j}^{pub} \geq 0} \max\left(0, \boldsymbol{M}_{i,j}\right) \ . \tag{6.28}$$

Let ${c_j}^{pub+}$ *be symmetrically defined:*

$$ {c_j}^{pub+} = {c_j}^{psum} + \sum_{i \in \mathcal{R}^{?} \mid {r_i}^{pub} \geq 0} \max\left(0, \boldsymbol{M}_{i,j}\right) \ . \tag{6.29}$$

Only columns $\{j \in C^? \mid {c_j}^{pub+} \geq 0\}$ should contribute to $\bar{w}$ as none of the columns in $\{j \in C^? \mid {c_j}^{pub+} < 0\}$ belong to the best extension (from the symmetrical counterpart of the dominance rule presented in Theorem 9). Consider c_4 of Example 2: the maximal contribution of c_4 is ${c_{\mathrm{c}_4}}^{pub+} = -2$ as r_6 is never to be selected (given that ${r_{\mathrm{r}_6}}^{pub} = -1$). Column c_4 cannot be in the best extension as it may only decrease the objective value (${c_{\mathrm{c}_4}}^{pub+} < 0$).

Entry $\boldsymbol{M}_{i,j}$ should contribute to ${r_i}^{pub+}$ only if $j \in C^?$ can belong to the best extension (${c_j}^{pub+} \geq 0$). Then, $\mathbb{C}(\boldsymbol{M}_{i,j} \longrightarrow {r_i}^{pub+})$ *should* be diminished by the part of $\boldsymbol{M}_{i,j}$ that ensures that ${c_j}^{pub+} \geq 0$. Let x be the maximal contribution of $\boldsymbol{M}_{i,j}$ ($0 \leq x \leq \boldsymbol{M}_{i,j}$) to the row and $\boldsymbol{M}_{i,j} - x$ be the rest for ${c_j}^{pub+}$: ${c_j}^{pub+} \longleftarrow {c_j}^{pub+} - x$. The largest value x such that ${c_j}^{pub+} - x \geq 0$ is $\min(\boldsymbol{M}_{i,j}, {c_j}^{pub+})$. The best scenario for row i is to consider only positive x. This defines ${r_i}^{eub}$, computed in $\mathcal{O}(m \times n)$.

Definition 17. **Explicit-bound of a row or a column**
The explicit-bound ${r_i}^{eub}$ *is an adjustment of* ${r_i}^{pub+}$ *to explicitly account for the contribution of* $\boldsymbol{M}_{i,j}$ *to* ${c_j}^{pub+}$ *(to ensure* ${c_j}^{pub+} \geq 0$*):*

$$ {r_i}^{eub} = {r_i}^{psum} + \sum_{j \in C^?} \max\left(0, \min\left(\boldsymbol{M}_{i,j}, {c_j}^{pub+}\right)\right) \quad , \quad \textit{or equivalently} \tag{6.30} $$

$$ {r_i}^{eub} = {r_i}^{pub+} + \sum_{j \in C^? \mid \boldsymbol{M}_{i,j} > {c_j}^{pub+} \geq 0} \left({c_j}^{pub+} - \boldsymbol{M}_{i,j}\right) \quad . \tag{6.31} $$

The explicit-bound ${c_j}^{eub}$ *is symmetrically defined.*

The remainder of this section introduces *newUB*, computed in $\mathcal{O}(m \times n)$, and tunes a component of it, specifically the term ${c_j}^{s}$, to ensure that $\bar{w} \leq$ *newUB*. The dominance rule showing that ${r_i}^{eub} < 0 \implies i \notin \mathcal{R}^{\in *}$ helps defining the bound but is presented in Theorem 11, Section 6.4.2 Dominance Rules.

Definition 18. **Tighter upper bound to the MSS problem**
For any partial assignment $(\mathcal{R}^{\in}, \mathcal{R}^?, C^{\in}, C^?)$,

$$ \begin{aligned} \textit{newUB} = & \sum_{i \in \mathcal{R}^{\in}} {r_i}^{psum} \\ & + \sum_{i \in \mathcal{R}^? \mid {r_i}^{eub} \geq 0} {r_i}^{psum} \\ & + \sum_{j \in C^? \mid {c_j}^{s} \geq 0} {c_j}^{psum} \\ & + \sum_{\substack{i \in \mathcal{R}^? \mid {r_i}^{eub} \geq 0 \\ j \in C^? \mid {c_j}^{pub+} \geq 0}} \max\left(0, \boldsymbol{M}_{i,j}\right) \quad . \end{aligned} \tag{6.32} $$

The weight of the best extension, *best*, can be rewritten as sums of contributions from mandatory and possible rows and columns, similarly to Equation (6.32):

$$\begin{aligned} best = &\sum_{i \in \mathcal{R}^{\in}} {r_i}^{psum} \\ &+ \sum_{i \in \left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right)} {r_i}^{psum} \\ &+ \sum_{j \in \left(C^{?} \cap C^{\in *}\right)} {c_j}^{psum} \\ &+ \sum_{\substack{i \in \left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right) \\ j \in \left(C^{?} \cap C^{\in *}\right)}} \boldsymbol{M}_{i,j} \ . \end{aligned} \tag{6.33}$$

Theorem 8 proves that the sum of the second and fourth term of Equation (6.32) is higher than that of Equation (6.33). More formally, it proves that the possible rows, together, contribute more to *newUB* than to *best*.

Theorem 8.
For any partial assignment $(\mathcal{R}^{\in}, \mathcal{R}^{?}, C^{\in}, C^{?})$,

$$\begin{aligned} &\sum_{i \in \mathcal{R}^{?} | {r_i}^{eub} \geq 0} {r_i}^{psum} + \sum_{\substack{i \in \mathcal{R}^{?} | {r_i}^{eub} \geq 0 \\ j \in C^{?} | {c_j}^{pub+} \geq 0}} \max\left(0, \boldsymbol{M}_{i,j}\right) \\ &\qquad \geq \sum_{i \in \left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right)} {r_i}^{psum} + \sum_{\substack{i \in \left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right) \\ j \in \left(C^{?} \cap C^{\in *}\right)}} \boldsymbol{M}_{i,j} \ . \end{aligned} \tag{6.34}$$

Proof. The symmetrical counterpart of dominance rule in Theorem 9 states that for any $j \in C^{?}$:

$$ {c_j}^{pub+} < 0 \implies j \notin C^{\in *} \ . \tag{6.35}$$

Then, $\left(C^{?} \cap C^{\in *}\right) \subseteq \{j \in C^{?} \mid {c_j}^{pub+} \geq 0\}$ and for any $i \in \mathcal{R}^{?}$:

$$ {r_i}^{psum} + \sum_{j \in C^{?} | {c_j}^{pub+} \geq 0} \max\left(0, \boldsymbol{M}_{i,j}\right) \geq {r_i}^{psum} + \sum_{j \in \left(C^{?} \cap C^{\in *}\right)} \boldsymbol{M}_{i,j} \ . \tag{6.36}$$

Moreover, dominance rule in Theorem 11 states that for any $i \in \mathcal{R}^{?}$:

$$ {r_i}^{eub} < 0 \implies i \notin \mathcal{R}^{\in *} \ , \tag{6.37}$$

which implies that $\left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right) \subseteq \{i \in \mathcal{R}^{?} \mid {r_i}^{eub} \geq 0\}$. Then, Equation (6.34) holds if there is no row $i \in \mathcal{R}^{?}$ such that:

$$ {r_i}^{eub} \geq 0 \quad \wedge \quad {r_i}^{psum} + \sum_{j \in C^{?} | {c_j}^{pub+} \geq 0} \max\left(0, \boldsymbol{M}_{i,j}\right) < 0 \ . \tag{6.38}$$

Such a row cannot exist as, for any possible row i,

$$r_i^{psum} + \sum_{j \in C^? \mid c_j^{pub+} \geq 0} \max\left(0, \boldsymbol{M}_{i,j}\right) \geq r_i^{eub} \quad . \tag{6.39}$$

□

Consider any column $j \in \left(C^? \setminus C^{\in *}\right)$ with $c_j^{pub+} \geq 0$. Entries in j and rows $\{i \in \mathcal{R}^? \mid r_i^{eub} \geq 0\}$ with $\boldsymbol{M}_{i,j} \geq 0$ should not contribute to the left-hand side of Equation (6.34) as they do not contribute to the right-hand side. However, such entries do contribute to the left-hand side to compensate for the (possibly negative) r_i^{psum}.

Let us compute the part of $\boldsymbol{M}_{i,j}$ required to compensate for r_i^{psum} and the part that makes the left-hand side exceed the right-hand side of Equation (6.34). From definition of r_i^{eub}, $\max(0, \min(\boldsymbol{M}_{i,j}, c_j^{pub+}))$ of $\boldsymbol{M}_{i,j}$ is dedicated to r_i^{eub} while the rest, $\boldsymbol{M}_{i,j} - \max(0, \min(\boldsymbol{M}_{i,j}, c_j^{pub+}))$, only increases the left-hand side of Equation (6.34). Similarly, if $\boldsymbol{M}_{i,j} \geq -\min(0, r_i^{psum})$, value $-\min(0, r_i^{psum})$ of $\boldsymbol{M}_{i,j}$ compensates for r_i^{psum} while the rest, $\boldsymbol{M}_{i,j} + \min(0, r_i^{psum})$, only increases the left-hand side.

Definition 19. **Value c_j^s**
Let us define, for any $j \in C^?$,

$$\begin{aligned} c_j^s &= c_j^{psum} \\ &+ \sum_{i \in \mathcal{R}^? \mid r_i^{eub} \geq 0} \max\Big[\max\left(0, \boldsymbol{M}_{i,j}\right) - \max\left(0, \min\left(\boldsymbol{M}_{i,j}, c_j^{pub+}\right)\right), \\ &\qquad\qquad \max\left(0, \min\left(0, r_i^{psum}\right) + \boldsymbol{M}_{i,j}\right)\Big] \end{aligned} \tag{6.40}$$

as the left-hand side of Equation (6.34) exceeds the right-hand side by at least
$$\max(0, \boldsymbol{M}_{i,j}) - \max(0, \min(\boldsymbol{M}_{i,j}, c_j^{pub+})) \quad ,$$
and at least
$$\max(0, \min(0, r_i^{psum}) + \boldsymbol{M}_{i,j}) \quad ,$$
for any $i \in \mathcal{R}^?$ and any $j \in \left(C^? \setminus C^{\in *}\right)$ such that
$$r_i^{eub} \geq 0 \quad \wedge \quad c_j^{pub+} \geq 0 \quad .$$

The inequality of Equation (6.34) is not modified by the addition of

$$\sum_{j \in \left(C^? \setminus C^{\in *}\right) \mid c_j^s \geq 0} c_j^{psum} \quad , \tag{6.41}$$

to its left-hand side (see Definition 19). Moreover, $c_j^{psum} \geq 0 \implies c_j^s \geq 0 \implies \sum_{j \in \left(C^? \cap C^{\in *}\right) \mid c_j^s \geq 0} c_j^{psum} \geq \sum_{j \in \left(C^? \cap C^{\in *}\right)} c_j^{psum}$. Consequently, adding

$$\sum_{i \in \mathcal{R}^{\in}} r_i^{psum} + \sum_{j \in C^? \mid c_j^s \geq 0} c_j^{psum} \tag{6.42}$$

to the left-hand side, which becomes equal to *newUB*, and adding

$$\sum_{i \in \mathcal{R}^{\in}} {r_i}^{psum} + \sum_{j \in \left(\mathcal{C}^{?} \cap \mathcal{C}^{\in *}\right)} {c_j}^{psum} \tag{6.43}$$

to the right-hand side, which becomes equal to *best*, of EQUATION (6.34), preserves the inequality. Consequently, *newUB* is an upper bound to *best* as the bound is larger or equal to the best value.

The definition of *newUB* allows stronger filtering on the domain of w:

$$\left[w^{\min}, w^{\max}\right] \leftarrow \left[w^{\min}, \min\left(w^{\max}, newUB\right)\right] \ . \tag{6.44}$$

The upper bound *newUB* to *best* is equal to 15 in EXAMPLE 2. The upper bound *baseUB* to *best* is equal to 25 is EXAMPLE 2.

6.4.2 New dominance rules

Let us define rules to filter out solutions from the solution space or values from the domain of the variables. THEOREMS 9, 10, and 11 define rules to detect rows (or columns) that never belong to the best extension to any partial assignment.

THEOREM 9.
Given a partial assignment, any row $i \in \mathcal{R}^{?}$ with ${r_i}^{pub+} < 0$ never belongs to the best extension:

$$\forall i \in \mathcal{R}^{?} \ : \ {r_i}^{pub+} < 0 \implies i \notin \mathcal{R}^{\in *} \ . \tag{6.45}$$

Proof. Excluding row i is a better decision in all extensions to any partial assignment when maximizing w, similarly to proof given in THEOREM 7. Consequently, i never belongs to the best extension: $i \notin \left(\mathcal{R}^{?} \cap \mathcal{R}^{\in *}\right) \implies i \notin \mathcal{R}^{\in *}$. □

LEMMA 1.
Given a partial assignment, any row $i \in \mathcal{R}^{?}$ with ${r_i}^{pub+} \geq 0$ never belongs to the best extension if $\exists j \in \left(\mathcal{C}^{?} \setminus \mathcal{C}^{\in *}\right)$ such that $\boldsymbol{M}_{i,j} > {r_i}^{pub+}$:

$$\forall i \in \mathcal{R}^{?}, j \in \mathcal{C}^{?} \mid \boldsymbol{M}_{i,j} > {r_i}^{pub+} \geq 0 \ : \ j \notin \mathcal{C}^{\in *} \implies i \notin \mathcal{R}^{\in *} \ . \tag{6.46}$$

Proof. For any $i \in \mathcal{R}^{?}$, any $j \in \mathcal{C}^{?}$ such that $\boldsymbol{M}_{i,j} > {r_i}^{pub+} \geq 0$, the value of ${r_i}^{pub+}$ after exclusion of j is

$${r_i}^{pub+}{}_{\mathcal{C} \setminus j} = {r_i}^{pub+} - \max\left(0, \boldsymbol{M}_{i,j}\right) \ . \tag{6.47}$$

Then, $\mathcal{C} \setminus j \implies {r_i}^{pub+}{}_{\mathcal{C} \setminus j} < 0 \implies i \notin \mathcal{R}^{\in *}$. □

EXAMPLE 2 illustrates LEMMA 1: c_3 ($c_{\mathrm{c}_3}{}^{pub+} = 4$) only decreases the objective if r_3 is removed from the partial assignment ($c_{\mathrm{c}_3}{}^{pub+}{}_{\mathcal{R}\setminus \mathrm{r}_3} = -2 \implies \mathrm{c}_3 \notin \mathcal{C}^{\in *}$). Also, excluding c_3 implies the exclusion of r_3: $r_{\mathrm{r}_3}{}^{pub+}{}_{\mathcal{C}\setminus \mathrm{c}_3} = 1 - 6$.

THEOREM 10.
Given a partial assignment, row i ($\in \mathcal{R}^? \mid r_i{}^{eub} \geq 0$) and column j ($\in \mathcal{C}^? \mid c_j{}^{eub} \geq 0$) never belong to the best extension if $r_i{}^{pub+} + c_j{}^{pub+} - \max\left(0, \boldsymbol{M}_{i,j}\right) < 0$:

$$\begin{aligned} &\forall i \in \mathcal{R}^?, j \in \mathcal{C}^? \mid r_i{}^{pub+} \geq 0 \ \wedge \ c_j{}^{pub+} \geq 0 \ : \\ &\qquad r_i{}^{pub+} + c_j{}^{pub+} - \max\left(0, \boldsymbol{M}_{i,j}\right) \implies i \notin \mathcal{R}^{\in *} \wedge j \notin \mathcal{C}^{\in *} \end{aligned} \tag{6.48}$$

Proof. Let us define $\mathcal{X}$, $\mathcal{Y}$, i and j such that:

$$\mathcal{X} \in \left\{\mathcal{X} \mid \mathcal{R}^{\in} \subseteq \mathcal{X} \subseteq \left(\mathcal{R}^{\in} \cup \mathcal{R}^? \setminus \{i\}\right)\right\} \ , \tag{6.49}$$

$$i \in \mathcal{R}^? \quad \wedge \quad r_i{}^{pub+} \geq 0 \ , \tag{6.50}$$

$$\mathcal{Y} \in \{\mathcal{Y} \mid \mathcal{C}^{\in} \subseteq \mathcal{Y} \subseteq \left(\mathcal{C}^{\in} \cup \mathcal{C}^? \setminus \{j\}\right)\} \ , \tag{6.51}$$

$$j \in \mathcal{C}^? \quad \wedge \quad c_j{}^{pub+} \geq 0 \ . \tag{6.52}$$

For any partial assignment $(\mathcal{X}, \{i\}, \mathcal{Y}, \{j\})$:

$$\begin{aligned} w_{\substack{\mathcal{R}\leftarrow i \\ \mathcal{C}\leftarrow j}} = {} & w_{\substack{\mathcal{R}\setminus\{i\} \\ \mathcal{C}\setminus\{j\}}} \\ & + r_i{}^{psum} + \sum_{j' \in \left(\mathcal{C}^? \cap \mathcal{Y}\right)} m_{i,j'} + c_{j'}{}^{psum} + \sum_{i' \in \left(\mathcal{R}^? \cap \mathcal{X}\right)} m_{i',j} + \boldsymbol{M}_{i,j} \ , \end{aligned} \tag{6.53}$$

and

$$\begin{aligned} & r_i{}^{pub+} + c_j{}^{pub+} - \max\left(0, \boldsymbol{M}_{i,j}\right) \\ & \quad \geq r_i{}^{psum} + \sum_{j' \in \left(\mathcal{C}^? \cap \mathcal{Y}\right)} m_{i,j'} + c_j{}^{psum} + \sum_{i' \in \left(\mathcal{R}^? \cap \mathcal{X}\right)} m_{i',j} + \boldsymbol{M}_{i,j} \ . \end{aligned} \tag{6.54}$$

Then,

$$r_i{}^{pub+} + c_j{}^{pub+} - \max\left(0, \boldsymbol{M}_{i,j}\right) < 0 \implies w_{\substack{\mathcal{R}\leftarrow i \\ \mathcal{C}\leftarrow j}} < w_{\substack{\mathcal{R}\setminus\{i\} \\ \mathcal{C}\setminus\{j\}}} \ , \tag{6.55}$$

and $(\mathcal{X}, \emptyset, \mathcal{Y}, \emptyset)$ is a better solution than $(\mathcal{X} \cup \{i\}, \emptyset, \mathcal{Y} \cup \{j\}, \emptyset)$. Consequently, i or j never belongs to the best extension: $i \notin \mathcal{R}^{\in *} \vee j \notin \mathcal{C}^{\in *}$. However, $i \notin \mathcal{R}^{\in *} \implies j \notin \mathcal{C}^{\in *}$ and symmetrically from LEMMA 1 and its symmetrical counterpart. Consequently, neither i nor j belong to the best extension: $i \notin \mathcal{R}^{\in *} \wedge j \notin \mathcal{C}^{\in *}$. □

Example 2 illustrates Theorem 10: r_3 and c_3 are mainly supported by value 6 in $(\mathrm{r}_3, \mathrm{c}_3)$ as ${r_{\mathrm{r}_3}}^{pub+} = -5+6$ and ${c_{\mathrm{c}_3}}^{pub+} = -2+6$. Yet, $(\mathrm{r}_3, \mathrm{c}_3)$ cannot support them simultaneously as the maximal benefit of taking both r_3 and c_3 is $-5 - 2 + 6 = {r_{\mathrm{r}_3}}^{pub+} + {c_{\mathrm{c}_3}}^{pub+} - 6 \leq 0$.

Lemma 2.
For any row $i \in \mathcal{R}^?$,

$$ {r_i}^{pub+} \neq {r_i}^{eub} \iff \left\{ j \in C^? \mid \boldsymbol{M}_{i,j} > {c_j}^{pub+} \geq 0 \right\} \neq \emptyset \ . \tag{6.56} $$

Theorem 11.
Given a partial assignment, any possible row $i \in \mathcal{R}^?$ with ${r_i}^{eub} < 0$ never belongs to the best extension:

$$ \forall i \in \mathcal{R}^? \ : \ {r_i}^{eub} < 0 \implies i \notin \mathcal{R}^{\in *} \ . \tag{6.57} $$

Proof. Let us define a set $D_i = \left\{ j \in C^? \mid \boldsymbol{M}_{i,j} > {c_j}^{pub+} \geq 0 \right\}$.

$$ \begin{aligned} \forall i \in \mathcal{R}^? \mid D_i = \emptyset : \quad {r_i}^{eub} < 0 &\implies {r_i}^{pub+} < 0 \ , \quad \text{Lemma 2} \ , \\ {r_i}^{pub+} < 0 &\implies i \notin \mathcal{R}^{\in *} \ , \quad \text{Theorem 9} \ . \end{aligned} \tag{6.58} $$

For any partial assignment $(\mathcal{X}, \{i\}, \mathcal{Y}, D_i)$, if $|D_i| > 0$, Lemma 1 states that excluding row i implies the exclusion of all columns in D_i. Let $(\mathcal{X}, \emptyset, \mathcal{Y}, \emptyset)$ and $(\mathcal{X} \cup \{i\}, \emptyset, \mathcal{Y} \cup D_i^*, \emptyset)$ be the best extensions when excluding and selecting row i, respectively, with $D_i^* = D_i \cap C^{\in *}$. Let us compute the difference between the weight of $(\mathcal{X} \cup \{i\}, \emptyset, \mathcal{Y} \cup D_i^*, \emptyset)$ and $(\mathcal{X}, \emptyset, \mathcal{Y}, \emptyset)$:

$$ \begin{aligned} & {r_i}^{psum} + \sum_{j \in \left(C^? \cap C^{\in *}\right) \setminus D_i^*} \boldsymbol{M}_{i,j} + \sum_{j \in D_i^*} \boldsymbol{M}_{i,j} + \sum_{j \in D_i^*} \left({c_j}^{psum} + \sum_{i' \in \left(\mathcal{R}^? \cap \mathcal{R}^{\in *}\right) \setminus \{i\}} m_{i',j} \right) \\ & \leq {r_i}^{psum} + \sum_{j \in \left(C^? \cap C^{\in *}\right) \setminus D_i^*} \boldsymbol{M}_{i,j} + \sum_{j \in D_i^*} {c_j}^{pub+} && (6.59) \\ & \leq {r_i}^{psum} + \sum_{j \in \left(C^? \cap C^{\in *}\right) \setminus D_i^*} \max\left(0, \boldsymbol{M}_{i,j}\right) + \sum_{j \in D_i^*} {c_j}^{pub+} \ . && (6.60) \end{aligned} $$

The latter (right-hand side of EQUATION 6.60) is smaller than ${r_i}^{eub}$:

$$ {r_i}^{psum} + \sum_{j \in (C^? \cap C^{\in *}) \setminus D_i^*} \max(0, \boldsymbol{M}_{i,j}) + \sum_{j \in D_i^*} {c_j}^{pub+} \tag{6.61}$$

$$\begin{aligned} &= {r_i}^{psum} + \sum_{j \in (C^? \cap C^{\in *}) | {c_j}^{pub+} \geq \boldsymbol{M}_{i,j}} \max(0, \boldsymbol{M}_{i,j}) \\ &+ \sum_{j \in (C^? \cap C^{\in *}) | \boldsymbol{M}_{i,j} > {c_j}^{pub+} \geq 0} {c_j}^{pub+} \end{aligned} \tag{6.62}$$

$$ = {r_i}^{psum} + \sum_{j \in (C^? \cap C^{\in *})} \max\left(0, \min\left(\boldsymbol{M}_{i,j}, {c_j}^{pub+}\right)\right) \tag{6.63}$$

$$ \leq {r_i}^{psum} + \sum_{j \in C^?} \max\left(0, \min\left(\boldsymbol{M}_{i,j}, {c_j}^{pub+}\right)\right) = {r_i}^{eub} \ . \tag{6.64}$$

This illustrates that if ${r_i}^{eub}$ is negative, selecting i and its dependent columns D_i^* can only alter the objective function: i must be excluded. □

Consider EXAMPLE 2: c_3 depends on r_3 and reciprocally. Selecting r_3 comes with a cost of 10, to be balanced by $(\mathrm{r}_3, \mathrm{c}_3)$: $\mathbb{C}((\mathrm{r}_3, \mathrm{c}_3) \rightarrow \mathrm{r}_3) = 5 \implies \mathbb{C}((\mathrm{r}_3, \mathrm{c}_3) \rightarrow c_{\mathrm{c}_3})^{eub} = 1$ and ${c_{\mathrm{c}_3}}^{eub} = -1$. Similarly, c_3 comes with a cost of 2 to be balanced by $(\mathrm{r}_3, \mathrm{c}_3)$: $\mathbb{C}((\mathrm{r}_3, \mathrm{c}_3) \rightarrow \mathrm{c}_3) = 2 \implies \mathbb{C}((\mathrm{r}_3, \mathrm{c}_3) \rightarrow r_{\mathrm{r}_3})^{eub} = 4$ and ${r_{\mathrm{r}_3}}^{eub} = -2$. It is clear that $(\mathrm{r}_3, \mathrm{c}_3)$ cannot support both r_3 and c_3.

Let us define ${c_j}^{lb} = {c_j}^{psum} + \sum_{i \in \mathcal{R}^? | {r_i}^{eub} \geq 0} \min(0, \boldsymbol{M}_{i,j})$. THEOREM 12 define rules to detect columns (or rows) that always belong to the best extension to any partial assignment.

THEOREM 12.
Given a partial assignment, any column $j \in C^?$ with ${c_j}^{lb} > 0$ always belongs to the best extension:

$$\forall i \in \mathcal{R}^? \ : \ {c_j}^{lb} > 0 \implies j \in C^{\in *} \ . \tag{6.65}$$

Proof. Inserting column j is a better decision in all extensions to any partial assignment when maximizing w, similarly to proof given in THEOREM 7. Therefore, j always belongs to the best extension: $j \in \left(\mathcal{R}^? \cap \mathcal{R}^{\in *}\right) \implies j \in C^{\in *}$. □

Observe that if ${r_i}^{eub} + \boldsymbol{M}_{i,j} < 0$, one would likely exclude the row rather than adding $\boldsymbol{M}_{i,j}$, with costs ${r_i}^{eub}$ and $-\boldsymbol{M}_{i,j}$, respectively. Let us define ${c_j}^{elb} = {c_j}^{psum} + \sum_{i \in \mathcal{R}^? | {r_i}^{eub} \geq 0} \max\left(\min(0, \boldsymbol{M}_{i,j}), -{r_i}^{eub}\right)$ as the smallest contribution of j if it was selected. It accounts for the influence of j on the set of possible rows upon (hypothetical) selection of j. Then, ${c_j}^{elb} > 0 \implies j \in C^{\in *}$ as j always belongs to the best extensions.

6.5 Experiments

This section describes experiments conducted to assess the performance of two implementations using:

1. the *baseUB* upper bound and associated filtering and dominance rules, and

2. the *newUB* upper bound and the corresponding new filtering and new dominance rules.

The two implementations, respectively called after the name of the upper bound they rely on, are compared using data matrices generated in a controlled setting. By comparing these two implementations, we can evaluate the increased filtering provided with the new upper bound, filtering rules, and dominance rules.

We consider small instances matrices to allow the search to solve to optimality. This allows comparing implementations on the time or number of nodes to find the optimal solution and the time or number of nodes required to prove optimality. Additional experiments on large instance matrices using large neighborhood search is left as future work.

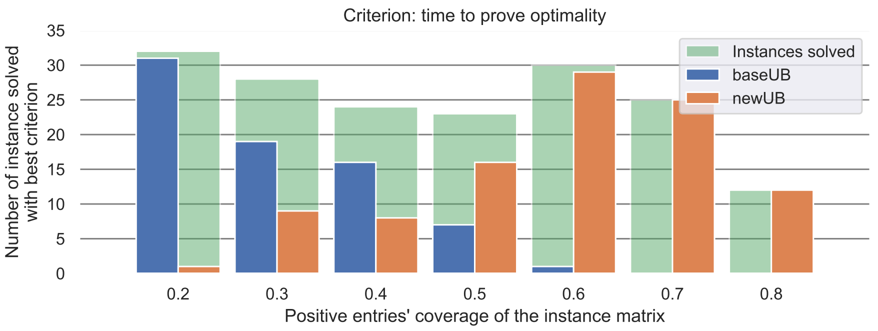
Algorithms have been implemented in Scala (2.13.1). Each run is executed with a single thread on a MacBook Pro (OS version 10.10.5, `Intel i7-2720 CPU @ 2.20-3.30GHz`, 4GB RAM per run). CP implementations are based on `OscaR` [Osc12] (version 4.1.0). The code and datasets are available at [Bra21]. Only instances solved within 2 minutes by both methods are considered to compare the time to find and prove the optimal solution.

6.5.1 Synthetic dataset

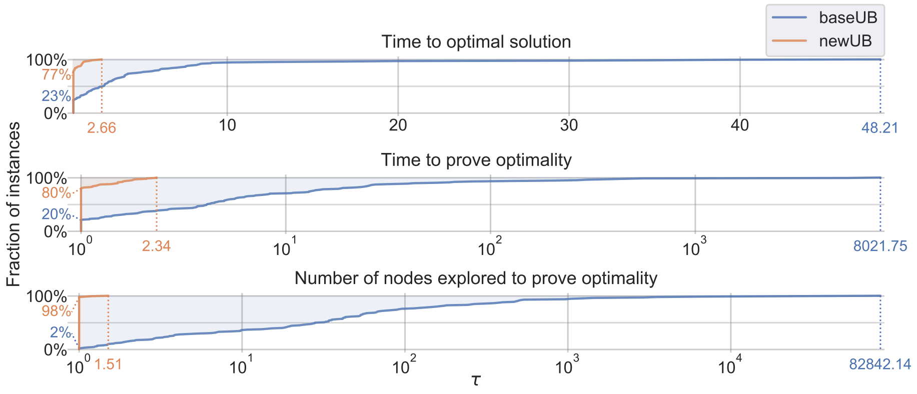
We generated 224 instances using a similar approach to [BSD17] while considering harder to solve instances for which positive entries do not belong to the optimal solution. Each instance is actually generated as a matrix of size $(x, y) \in \{(40, 40), (50, 32), (64, 25), (80, 20)\}$ with values drawn from a normal distribution $\mathcal{N}(-3, 1)$, a number $K \in \{2, 5\}$ of submatrices is defined with values drawn from $\mathcal{N}(1, 1)$. Each submatrix is defined as uniformly selected subsets of rows and columns of sizes $(\sqrt{p} \times x, \sqrt{p} \times y)$ with $p \in \{0.2, 0.5\}$. A value is added or subtracted to all matrix entries such that a fraction $d \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ of the matrix entries are positive.

6.5.2 Results

FIGURE 6.1a presents the number of instances that each algorithm solves more efficiently. An instance is considered more efficiently solved by an algorithm



(a) Time to prove optimality as a function of the rate of positive entries



(b) Performance profiles

Figure 6.1: Results of the performance of *newUB* and *baseUB*, and the associated filtering and dominance rules, computed on 224 generated instances. (a): each bar represents the fraction of instances solved more efficiently by each approach. The green bars correspond to the sum of the orange and blue bars. (b): each point (x,y) on a curve with label c indicates that the performance of c is within a factor x from the performance of the other algorithm in a fraction y of the instances. Results are reported only for instances solved by both approaches within 2 minutes.

if the time required to prove the optimality of its solution is smaller than that of other algorithms. Counts are reported as a function of the rate of positive entries in the instances matrices. It is clear from the graph that the performance are influenced by the fraction of positive entries in the instance matrix. The new upper bound appears less sensitive to the influence of positive entries.

While it is interesting to quantify the performance of each algorithm, performance profiles [DM02] may provide more general insights and conclusions. A performance profile is a cumulative distribution function $F(\tau)$ of a given performance metric τ. In this paper, the τ value is the ratio between the solving metric (typically, time or number of nodes) of a target approach and that of the best approach for the particular instance. Let us consider the third graph in Figure 6.1b as an example. We can read that $F_{baseUB}(\tau = 15) = 0.50$. This means that the performance, computed as the number of nodes explored to prove optimality, of *baseUB* is within a factor 15 of the performance of *newUB* in 50% of the problems. Alternatively, reading that $F_{newUB}(\tau = 1) = .98$ means that *newUB* requires less nodes to prove optimality in 98% of the problems. This illustrates the benefit of *newUB* in terms of nodes explored, even in instances containing only a small fraction of positive entries. Nevertheless, one might be more interested in computational time.

The first graph in Figure 6.1b presents the performance profiles using as the performance metric the time required to get the optimal solution, but not considering the time required to prove that it is optimal. Note that the x-axis is linear while it is semi-log in other graphs. The performance of *newUB* is within a factor of 2.7 from the performance of *baseUB* in all instances. The performance of *baseUB* is within a factor of 2.7 from *newUB* in half of the instances. Performance of *baseUB* is within a factor 48.2 of the performance of *newUB* in all instances when considering the time required to find the best solution.

The second graph in Figure 6.1b presents the performance profiles using the time required to prove optimality as the performance metric. We observe that *baseUB* is efficient on many instances but fails miserably in some instances. If given 10 times the time required by *newUB* to solve an instance, *baseUB* solves roughly 75% of the instances. This clearly illustrates that the new upper bound, while being more computationally intensive, is a better option if there is no knowledge on which approach would be best.

6.6 Conclusions

A new model for the maximal sum submatrix problem was proposed, which achieves the same filtering as the constraint programming approach defined in Chapter 2. That new CP model and the approach defined in an earlier chapter prove to be quite competitive due to reduced computational-time complexity. We illustrated limit cases for which the upper bound is not tight and is consequently inefficient to filter out suboptimal parts of the search tree. Those cases have been addressed by defining a new bound and new dominance rules. Experiments on synthetic data expose the limits of the *baseUB*, which may largely fail on unfavorable instances. The new upper bound *newUB* appears as a relevant alternative to *baseUB* as it does not require many additional resources to solve all instances. It actually requires fewer resources in some instances.

Most extensions to the maximal sum submatrix problem rely on the identification of a submatrix of maximal sum. Those extensions might benefit from *newUB* and the dominance rules introduced in this paper just as the bound and rules benefit to implementations solving the maximal sum submatrix problem. The benefits of *newUB* and dominance rules to those extensions should be evaluated in future works.

Conclusions and perspectives | 7

7.1 Few problems and many solutions

This thesis presents the maximal sum submatrix problem. It consists in finding subsets of rows and of columns of a matrix. The submatrix has to be of maximal weight: the sum of entries in the submatrix must be as high as possible. Two extensions are also presented. They concern the identification of multiple submatrices. The multiple submatrices cannot overlap in the maximum weighted set of disjoint submatrices problem, presented in Chapter 3 Mining K disjoint submatrices of maximal sum. Submatrices may overlap in the maximum weighted submatrix coverage problem, presented in Chapter 4 Mining k overlapping submatrices of maximal sum.

The three optimization problems considered in this thesis are $\mathcal{NP}$-hard. It is hopeless to find optimal solutions on large instance matrices such as gene expression matrices with hundreds of thousands of entries. An optimal solution is easy to find in some circumstances, however. This is related to the implicit threshold and the maximization term that rewards positive entries and penalizes negative entries. Changing the threshold can drastically change the solution space and the search space. The optimal solution is easier to find but less relevant from a biological point of view for extreme threshold values.

The thesis focuses on the more difficult instances with intermediate threshold values. This is motivated by the biological applications where small but non-empty submatrices are to be expected. It is quite common for $\mathcal{NP}$-hard problems to fix a trade-off between the time spend to find a solution and the expected quality of that solution. The trade-off is set here by fixing a maximal amount of time to find the best solution possible. There is an underlying assumption that one is interested in high solution quality in short amounts of time. In all experiments, the maximal time allocated ranges from few minutes to few hours.

Many algorithmic contributions of this document are related to the reduction of the computational time to explore the search space. It usually comes from the design of efficient bounds and efficient filtering procedures or dominance rules. Still, incomplete searches, such as greedy search or large neighborhood search, are usually required to provide the best performances.

This can be explained by our assumption that one is interested in high solution quality in short amounts of time. One could argue that time might be less critical than the solution quality in a biological context. The results presented throughout the thesis could be vastly impacted in such context. It would be worth testing how solution quality advances may improve the biological results obtained.

This document presents many algorithms and implementations. They differ on the solver they rely on, the search strategy considered, and the problems they are designed to address. Other differences are explained by the fact that they have been designed to produce the highest performance profile possible in a particular (set of) experiments. Still, there are many similitudes between the problems to solve and the experiments performed. Table 7.1 presents all implementations used during the thesis and illustrates some of the similarities and differences between them.

7.2 Guidance and perspectives for the biologist

Results in Chapter 5 are supporting the relevance of the maximal sum submatrix problem as compared to standard biclustering approaches. This suggests that searching for relevant bicluster should be done through the maximization of the sum of the covered entries. Still, it may not be advisable to use one of the implementations provided in this work without some prior considerations. Some important elements are presented hereafter:

1. What is the actual problem?
   - (a) Are there multiple or just one submatrix to find?
   - (b) Should overlaps be allowed?
   - (c) Are there additional constraints that are not considered in the provided implementation(s)?
2. What is the actual data?
   - (a) Does it require preprocessing?
   - (b) Does it require normalization?
3. Should quality or efficiency be favored?

Experiments performed in Chapter 5 are restricted to the search of overlapping submatrices with a focus on efficiency. It is important to compare these results with the performances of the implementations provided in Chapter 4 and the approaches for finding disjoint submatrices proposed in Chapter 3. Comparisons should take into account both the quality and efficiency of each implementation.

Table 7.1: Implementations presented throughout the thesis

	Problem	Subm. model	Solver	Optim. Sol.	Features
CP-LNS$_0$	MSS	Booleans	OscarCP	✗	
CP-LNS	MSS	Booleans	OscarCP	✗	· Reduced search space
CPGC	MSS	Sparse-Sets	OscarCP	✗	· Reduced search space · Upper bound · Dominance rules · Variable value heuristic
MILP	MSS	Booleans	Gurobi	✓	· $\theta(m \times n)$ variables
MILP Big M	MSS	Booleans	Gurobi	✓	· $\theta(m + n)$ variables
baseUB	MSS	Sparse-Sets	OscarCP	✓	· Dominance rules · Light upper bound
newUB	MSS	Sparse-Sets	OscarCP	✓	· Tight but costly rules and bounds
Greedy baseline /Hot start	MWSDS	Sparse-Sets	OscarCP	✗	· Subroutine: CPGC Best solution in a budget of time · Overlaps penalized with $-\infty$
Column Generation	MWSDS	Sparse-Sets	Gurobi + OscarCP	✓	· Suboptimal solving of the pricing problem with the Greedy baseline
MILP	MWSDSP	Booleans	Gurobi	✓	· $\theta(m + n)$ variables
CP greedy	MWSC	Sparse-Sets	OscarCP	✗	· Subroutine: CPGC for a given amount of time · Overlaps neutralized
CP exhaust.	MWSC	Sparse-Sets	OscarCP	✓	· Finite state machines · Bounds and dominance rules · Variable value heuristic
CP LNS	MWSC	Sparse-Sets	OscarCP	✗	· Finite state machines · Bounds and dominance rules · Variable value heuristic · Three LNS restarts
MIP model	MWSC	Booleans	Gurobi	✓	· $\theta(kmn)$ variables
MIQCP	MWSC	Booleans	Gurobi	✓	· $\theta(k(m + n))$ variables
K-CPGC	MWSC	Sparse-Sets	OscarCP	✗	· Subroutine: CPGC until convergence · Overlaps neutralized

One of the current challenges in cancer research is the integration of various molecular data to define clinically or biologically meaningful subtypes. Combining the increasingly available genomes, transcriptomes and epigenomes provides a broader view of the underlying biology.

A motivation for the application of the maximal sum submatrix problem to biological data is the ability to deal with heterogeneity, as obtained by combining data types. Mutation matrices are sparse binary matrices with ones corresponding to a particular mutation for a particular sample; zeros corresponds to no occurrence. There are biological connexions between mutations and gene expression, and integrative methods are promising [Sub+20]. It is not clear yet how gene expression matrices and mutation matrices could be combined in the context of the MSS problem. Moreover, actual experiments on the maximal amount of heterogeneity that the proposed optimization problems can be dealt with should be carried out.

The maximal sum submatrix mining problem could be extended to a supervised classification setting. For example, in gene expression analysis, one typically wants to find genes that discriminate between two conditions. In other words, the columns could be *a priori* labeled according to two conditions. The objective can then be to identify a subset of maximally relevant rows to discriminate between subsets of samples from different conditions. This could be encoded in a larger matrix for which columns represent pairs of columns in either conditions from the original matrix and the value stored is interpreted as a distance value for a particular gene across both conditions.

The maximal sum submatrix problem could also be applied to outlier detection and biclustering. For example, using an appropriate data transformation, entries that are close to the mean or to the median could be mapped to relatively large positive entries. Similarly, entries far away from the mean would be mapped to low values. Consequently, a submatrix of maximal sum after such transformation would correspond to subsets of rows and of columns exhibiting similar entries. Explicit comparisons to existing biclustering algorithms could be considered in such a setting.

7.3 Guidance and perspectives for the computer scientist

Throughout the thesis, it has been shown that CP can be used to efficiently address the maximal sum submatrix problem and its extensions.

Two major advantages of constraint programming and mixed integer linear programming approaches over the design of ad-hoc softwares are:

1. that the researcher can rely on the solvers with all the tools and search strategies that are provided with them,
2. and existing solutions, as proposed in this document, can be combined

with additional constraints of objective function changes, given some additional work, to fit the particular problems that one might encounter.

However, additional work might be required to efficiently benefit from the solvers.

Some examples of constraints addition or objective function modifications include:

1. constraining the number of samples, such that submatrices are associated with more than a few samples,

2. constraining the number of genes, such that very generic pathways including too many genes are excluded,

3. modifying the objective function to maximize the median value, to reduce the sensitivity to extreme values.

Such changes could have a large influence on the search space, the performances, and the effectiveness of the bounds and dominance rules designed for the maximal sum submatrix problem. The actual influence may vary a lot and should be evaluated for the particular modifications considered. The specific performances of the implementations provided for the MSS should also be evaluated.

From a computational point of view, it would be interesting to see how CP performs on some of the standard biclustering problems. Indeed, many of the biclustering variants can be expressed as optimization problems to be solved. It would be ideal to start with the biclustering problem defined in [CC00]. The objective is to minimize the variance within a bicluster by minimizing its minimum-square residue (see Cheng and Church's algorithm in CHAPTER 5). Simple upper bounds and variable value heuristics could be easily defined. Efficient bounds updates and look-aheads could be designed as the minimum-square residue is computed as a sum of averages, similarly to the weight of submatrices for the MSS problem. Moreover, this Cheng and Church's algorithm provided the best performances with our algorithm in the biological evaluations in CHAPTER 5 MINING SUBMATRICES OF MAXIMAL SUM IN GENE EXPRESSION DATA. Finally, it is the first definition of a biclustering problem and is still among the most cited techniques in biclustering.

Future work should refine the maximum weighted submatrix coverage problem presented in CHAPTER 4 as it presents a bias: the objective function favors overlaps of submatrices on negative entries. Indeed, once a negative entry is covered, adding it to another submatrix is cost-free. On the other hand, once a positive entry is covered, the objective value does not benefit from the matrix entry coverage by additional submatrices. Intuitively, one would like to prevent the selection of negative entries in one submatrix, or worst, in multiple submatrices. One would like to favor the selection of positive entries. In particular, one would like to favor the overlaps on the positive entries rather than on the negative entries. A possible modification of the maximum weighted submatrix coverage problem could consist in applying a penalty for selecting matrix entries covered by multiple submatrices. In any case, modifications of the objective function would require dedicated modifications of the algorithms provided in this work.

The search space limits the performance of all methods proposed to solve the maximal sum submatrix problem or one of its extensions. The m by n entries of the input matrix determine the search space. Unsurprisingly, the decision to include or not a row r_1 in any node of the search space should be completely identical to the decision for the row r_2 if both rows are identical, or if:

$$\boldsymbol{M}_{r_1,j} = \boldsymbol{M}_{r_2,j} \quad , \quad \forall j \in \mathrm{C} \quad .$$

Merging all sets of identical rows would help to reduce the size of the search space. Similarly, merging all sets of identical columns would improve performance. Solvers for the maximal sum submatrix problem or one of its extensions should work on a modified matrix $\boldsymbol{M}'$ with all sets of identical rows and all sets of identical columns being merged. One might not expect to find many identical rows or columns in large and real instances matrices. It is then to determine the rules to merge rows or columns close enough to share their domain in all possible extensions to a partial assignment.

Acronyms

B & B branch & bound. 9, 20, 22, 60, 65

CG column generation. 35, 41, 44, 46, 47, 49

CP constraint programming. 8–10, 13–18, 20, 26, 30, 33–36, 39, 41, 43, 47, 49, 53, 56–60, 66–72, 77, 79, 95, 96, 112, 118, 119

DFS depth-first-search. 9, 16, 18, 20, 47, 57, 59–62, 64, 66, 79, 80, 96

FDR false discovery rate. 83–85, 88

FSM finite state machine. 61, 62, 64, 65

ILP integer linear programming. 38, 39, 41–43

LNS large neighborhood search. 10, 13, 17, 30, 32, 33, 56, 66, 67, 69, 71, 110, 115

LP linear programming. 22, 24, 33, 42, 49

LP-RMP linear programming relaxation of the restricted master problem. 42–45

MILP mixed integer linear programming. 13–15, 22, 24–26, 34–36, 41, 43, 45–47, 49, 118

MSS maximal sum submatrix. 1–9, 13–16, 18, 20, 24–28, 33–36, 38–41, 43, 45, 47, 51, 53, 55, 56, 67, 71, 73–75, 79–81, 95–98, 101, 112, 113, 115, 116, 118–120

MWSC maximum weighted submatrix coverage. 54, 56, 58, 60, 61, 71, 75, 115, 119, 120

MWSDS maximum weighted set of disjoint submatrices. 35–41, 44, 45, 49–51, 54, 55, 115

RMP restricted master problem. 42–45

Additional tables

Table 2: 01_alpha_factor.

	term	description	p-adjust.
1	GO:0042254	ribosome biogenesis	1.191613e-17
2	GO:0022613	ribonucleoprotein complex biogenesis	2.715624e-16
3	GO:0006364	rRNA processing	3.099617e-13
4	GO:0016072	rRNA metabolic process	3.099617e-13
5	GO:0034470	ncRNA processing	2.532992e-12
6	GO:0042274	ribosomal small subunit biogenesis	6.232817e-12
7	GO:0034660	ncRNA metabolic process	9.263555e-11
8	GO:0006396	RNA processing	4.683446e-10
9	GO:0000462	maturation of SSU-rRNA from tricistronic rRNA transcript	5.519431e-10
10	GO:0030490	maturation of SSU-rRNA	1.055367e-09
11	GO:0044085	cellular component biogenesis	6.271420e-08
12	GO:0000460	maturation of 5.8S rRNA	9.602590e-08
13	GO:0000466	maturation of 5.8S rRNA from tricistronic rRNA transcript	9.602590e-08
14	GO:0000478	endonucleolytic cleavage involved in rRNA processing	3.796748e-07
15	GO:0000479	endonucleolytic cleavage of tricistronic rRNA transcript	3.796748e-07
16	GO:0050801	ion homeostasis	4.712789e-07
17	GO:0055080	cation homeostasis	4.712789e-07
18	GO:0000469	cleavage involved in rRNA processing	8.120976e-07
19	GO:0044764	multi-organism cellular process	1.170023e-06
20	GO:0000746	conjugation	1.170023e-06

Table 3: 02_cdc_15.

	term	description	p-adjust.
1	GO:0042254	ribosome biogenesis	1.171585e-25
2	GO:0022613	ribonucleoprotein complex biogenesis	8.445391e-23
3	GO:0034660	ncRNA metabolic process	8.445391e-23
4	GO:0034470	ncRNA processing	1.209137e-22
5	GO:0006364	rRNA processing	2.809275e-22
6	GO:0016072	rRNA metabolic process	2.809275e-22
7	GO:0006396	RNA processing	4.656052e-19
8	GO:0016070	RNA metabolic process	4.518930e-17
9	GO:0090304	nucleic acid metabolic process	1.573745e-15
10	GO:0044085	cellular component biogenesis	8.007472e-15
11	GO:0010467	gene expression	6.087563e-12
12	GO:0042274	ribosomal small subunit biogenesis	1.510783e-11
13	GO:0006139	nucleobase-containing compound metabolic process	1.371075e-09
14	GO:0042273	ribosomal large subunit biogenesis	2.956637e-09
15	GO:0046483	heterocycle metabolic process	9.716562e-09
16	GO:0006725	cellular aromatic compound metabolic process	1.529971e-08
17	GO:0000460	maturation of 5.8S rRNA	1.955398e-08
18	GO:0000466	maturation of 5.8S rRNA from tricistronic rRNA transcript	1.955398e-08
19	GO:0030490	maturation of SSU-rRNA	1.955398e-08
20	GO:0000462	maturation of SSU-rRNA from tricistronic rRNA transcript	3.623493e-08

Table 4: 03_cdc_28.

	term	description	p-adjust.
1	GO:0002181	cytoplasmic translation	2.553447e-29
2	GO:0006412	translation	3.901098e-27
3	GO:0043043	peptide biosynthetic process	3.901098e-27
4	GO:0043604	amide biosynthetic process	3.901098e-27
5	GO:0006518	peptide metabolic process	8.745269e-25
6	GO:0043603	cellular amide metabolic process	2.993049e-23
7	GO:1901566	organonitrogen compound biosynthetic process	4.118333e-20
8	GO:0042254	ribosome biogenesis	1.682898e-16
9	GO:1901564	organonitrogen compound metabolic process	1.951916e-16
10	GO:0019538	protein metabolic process	1.482145e-13
11	GO:0022613	ribonucleoprotein complex biogenesis	2.570928e-12
12	GO:0044267	cellular protein metabolic process	2.730851e-12
13	GO:0042274	ribosomal small subunit biogenesis	5.359186e-10
14	GO:0006364	rRNA processing	2.515863e-09
15	GO:0042273	ribosomal large subunit biogenesis	5.502242e-08
16	GO:0016072	rRNA metabolic process	5.567943e-08
17	GO:0044271	cellular nitrogen compound biosynthetic process	3.490743e-07
18	GO:0034645	cellular macromolecule biosynthetic process	4.093896e-07
19	GO:0009059	macromolecule biosynthetic process	5.144026e-07
20	GO:0000462	maturation of SSU-rRNA from tricistronic rRNA transcript	9.260947e-07

Table 5: 04_elutriation.

	term	description	p-adjust.
1	GO:0042254	ribosome biogenesis	5.720403e-34
2	GO:0022613	ribonucleoprotein complex biogenesis	2.835895e-31
3	GO:0034660	ncRNA metabolic process	6.707776e-28
4	GO:0034470	ncRNA processing	1.025721e-25
5	GO:0006364	rRNA processing	1.627219e-24
6	GO:0016072	rRNA metabolic process	2.300636e-24
7	GO:0010467	gene expression	4.712617e-23
8	GO:0006807	nitrogen compound metabolic process	1.210243e-21
9	GO:0034641	cellular nitrogen compound metabolic process	3.630761e-20
10	GO:0006396	RNA processing	7.580284e-20
11	GO:0042274	ribosomal small subunit biogenesis	5.847937e-17
12	GO:0044085	cellular component biogenesis	9.716219e-17
13	GO:0016070	RNA metabolic process	1.620486e-16
14	GO:0006139	nucleobase-containing compound metabolic process	3.841940e-16
15	GO:0046483	heterocycle metabolic process	1.615821e-15
16	GO:0006725	cellular aromatic compound metabolic process	2.396934e-15
17	GO:0090304	nucleic acid metabolic process	2.418670e-15
18	GO:1901360	organic cyclic compound metabolic process	2.720483e-15
19	GO:0030490	maturation of SSU-rRNA	4.493421e-15
20	GO:0000462	maturation of SSU-rRNA from tricistronic rRNA transcript	1.435419e-14

Table 6: 05_1mM_menadione.

	term	description	p-adjust.
1	GO:0032196	transposition	4.575689e-13
2	GO:0032197	transposition, RNA-mediated	4.575689e-13
3	GO:0006979	response to oxidative stress	5.485371e-09
4	GO:0034599	cellular response to oxidative stress	2.435800e-08
5	GO:0055114	oxidation-reduction process	4.793730e-08
6	GO:0044710	single-organism metabolic process	3.606023e-07
7	GO:0042221	response to chemical	1.670056e-04
8	GO:0002181	cytoplasmic translation	0.001291574
9	GO:0019725	cellular homeostasis	1.889669e-03
10	GO:0070887	cellular response to chemical stimulus	2.766072e-03
11	GO:0006623	protein targeting to vacuole	2.766072e-03
12	GO:0072665	protein localization to vacuole	2.766072e-03
13	GO:0072666	establishment of protein localization to vacuole	2.766072e-03
14	GO:0006575	cellular modified amino acid metabolic process	3.421278e-03
15	GO:0008652	cellular amino acid biosynthetic process	3.421278e-03
16	GO:0045454	cell redox homeostasis	1.143711e-02
17	GO:0046394	carboxylic acid biosynthetic process	1.296318e-02
18	GO:0006520	cellular amino acid metabolic process	1.296318e-02
19	GO:0016053	organic acid biosynthetic process	1.296318e-02
20	GO:0006820	anion transport	1.296318e-02

Table 7: 06_1M_sorbitol.

	term	description	p-adjust.
1	GO:0009056	catabolic process	7.574141e-08
2	GO:0006508	proteolysis	7.574141e-08
3	GO:0044710	single-organism metabolic process	7.676728e-08
4	GO:1901575	organic substance catabolic process	9.198719e-08
5	GO:0044723	single-organism carbohydrate metabolic process	8.168864e-07
6	GO:0044763	single-organism cellular process	6.135262e-06
7	GO:0006091	generation of precursor metabolites and energy	6.135262e-06
8	GO:0030163	protein catabolic process	6.135262e-06
9	GO:0050896	response to stimulus	6.135262e-06
10	GO:0044257	cellular protein catabolic process	7.282433e-06
11	GO:0055114	oxidation-reduction process	9.709549e-06
12	GO:0044248	cellular catabolic process	1.761001e-05
13	GO:0006796	phosphate-containing compound metabolic process	2.396838e-05
14	GO:0006793	phosphorus metabolic process	2.636866e-05
15	GO:0005975	carbohydrate metabolic process	3.061083e-05
16	GO:0051603	proteolysis involved in cellular protein catabolic process	3.065663e-05
17	GO:0006950	response to stress	3.753313e-05
18	GO:0051716	cellular response to stimulus	3.755634e-05
19	GO:0044262	cellular carbohydrate metabolic process	4.816775e-05
20	GO:0015980	energy derivation by oxidation of organic compounds	6.838967e-05

Table 8: 07_15mM_diamide.

	term	description	p-adjust.
1	GO:0006457	protein folding	1.846434e-07
2	GO:0006081	cellular aldehyde metabolic process	2.444289e-07
3	GO:0055114	oxidation-reduction process	1.143481e-06
4	GO:0033554	cellular response to stress	1.746171e-06
5	GO:0044710	single-organism metabolic process	1.746171e-06
6	GO:0044712	single-organism catabolic process	1.746171e-06
7	GO:0070887	cellular response to chemical stimulus	2.990616e-06
8	GO:0006950	response to stress	2.990616e-06
9	GO:0050896	response to stimulus	5.417786e-06
10	GO:0042221	response to chemical	7.577489e-06
11	GO:0006979	response to oxidative stress	7.577489e-06
12	GO:0010035	response to inorganic substance	7.577489e-06
13	GO:0051716	cellular response to stimulus	7.938822e-06
14	GO:1901575	organic substance catabolic process	1.074632e-05
15	GO:0009056	catabolic process	1.074632e-05
16	GO:0005975	carbohydrate metabolic process	1.144059e-05
17	GO:0044723	single-organism carbohydrate metabolic process	4.072123e-05
18	GO:0034599	cellular response to oxidative stress	4.072123e-05
19	GO:0044282	small molecule catabolic process	9.929611e-05
20	GO:1901700	response to oxygen-containing compound	1.384823e-04

Table 9: 08_25mM_DTT.

	term	description	p-adjust.
1	GO:0002181	cytoplasmic translation	2.571265e-26
2	GO:0006412	translation	1.262146e-20
3	GO:0043043	peptide biosynthetic process	1.814313e-20
4	GO:0043604	amide biosynthetic process	1.270097e-19
5	GO:0006518	peptide metabolic process	4.110698e-19
6	GO:0043603	cellular amide metabolic process	3.413801e-18
7	GO:1901566	organonitrogen compound biosynthetic process	7.571918e-16
8	GO:0010467	gene expression	1.406438e-15
9	GO:0044260	cellular macromolecule metabolic process	5.610111e-14
10	GO:0022613	ribonucleoprotein complex biogenesis	2.348392e-13
11	GO:0044267	cellular protein metabolic process	2.376363e-13
12	GO:0034641	cellular nitrogen compound metabolic process	1.072584e-12
13	GO:0042254	ribosome biogenesis	1.369765e-12
14	GO:0043170	macromolecule metabolic process	1.769892e-12
15	GO:0019538	protein metabolic process	2.764025e-12
16	GO:0006807	nitrogen compound metabolic process	6.721745e-12
17	GO:1901564	organonitrogen compound metabolic process	1.907148e-11
18	GO:0009059	macromolecule biosynthetic process	1.565513e-10
19	GO:0006364	rRNA processing	1.763186e-10
20	GO:0016072	rRNA metabolic process	1.763186e-10

Table 10: 09_constant_32nM_H2O2.

	term	description	p-adjust.
1	GO:0055114	oxidation-reduction process	6.602138e-14
2	GO:0044710	single-organism metabolic process	2.692915e-09
3	GO:0006979	response to oxidative stress	8.271084e-08
4	GO:0034599	cellular response to oxidative stress	1.049329e-06
5	GO:0042221	response to chemical	1.246732e-05
6	GO:0019725	cellular homeostasis	1.246732e-05
7	GO:0044699	single-organism process	1.246732e-05
8	GO:0048878	chemical homeostasis	1.918285e-05
9	GO:0055072	iron ion homeostasis	2.170300e-05
10	GO:0070887	cellular response to chemical stimulus	2.460483e-05
11	GO:0042592	homeostatic process	2.804128e-05
12	GO:0098771	inorganic ion homeostasis	5.681460e-05
13	GO:0050801	ion homeostasis	7.432589e-05
14	GO:0055080	cation homeostasis	1.652456e-04
15	GO:0044723	single-organism carbohydrate metabolic process	1.716979e-04
16	GO:0044262	cellular carbohydrate metabolic process	2.412582e-04
17	GO:0055065	metal ion homeostasis	2.412582e-04
18	GO:0005975	carbohydrate metabolic process	2.862163e-04
19	GO:0010035	response to inorganic substance	2.862163e-04
20	GO:0055076	transition metal ion homeostasis	3.468919e-04

Table 11: 10_diauxic_shift.

	term	description	p-adjust.
1	GO:0015980	energy derivation by oxidation of organic compounds	7.076662e-17
2	GO:0006091	generation of precursor metabolites and energy	7.076662e-17
3	GO:0055114	oxidation-reduction process	3.610217e-16
4	GO:0045333	cellular respiration	1.631480e-13
5	GO:0044710	single-organism metabolic process	4.596238e-12
6	GO:0009060	aerobic respiration	1.173179e-11
7	GO:0044723	single-organism carbohydrate metabolic process	8.367914e-11
8	GO:0044262	cellular carbohydrate metabolic process	5.735060e-10
9	GO:0005975	carbohydrate metabolic process	2.338722e-09
10	GO:0044763	single-organism cellular process	1.133426e-08
11	GO:1901566	organonitrogen compound biosynthetic process	1.229324e-07
12	GO:0006099	tricarboxylic acid cycle	2.278109e-07
13	GO:0006412	translation	2.830024e-07
14	GO:0043043	peptide biosynthetic process	2.830024e-07
15	GO:0006518	peptide metabolic process	3.950002e-07
16	GO:0043604	amide biosynthetic process	3.950002e-07
17	GO:0006119	oxidative phosphorylation	5.075348e-07
18	GO:0043603	cellular amide metabolic process	1.018864e-06
19	GO:1901564	organonitrogen compound metabolic process	1.018864e-06
20	GO:0022900	electron transport chain	1.136910e-06

Table 12: 11_complete_DTT.

	term	description	p-adjust.
1	GO:0002181	cytoplasmic translation	6.924363e-59
2	GO:0006412	translation	2.772677e-55
3	GO:0043043	peptide biosynthetic process	5.084914e-54
4	GO:0010467	gene expression	1.800433e-52
5	GO:0043604	amide biosynthetic process	1.567400e-51
6	GO:0006518	peptide metabolic process	1.045339e-49
7	GO:0043603	cellular amide metabolic process	3.758523e-45
8	GO:1901566	organonitrogen compound biosynthetic process	2.466839e-40
9	GO:0022613	ribonucleoprotein complex biogenesis	9.607187e-40
10	GO:0042254	ribosome biogenesis	1.661256e-39
11	GO:0044271	cellular nitrogen compound biosynthetic process	6.496315e-38
12	GO:0034645	cellular macromolecule biosynthetic process	8.025738e-37
13	GO:0009059	macromolecule biosynthetic process	3.615328e-36
14	GO:0034641	cellular nitrogen compound metabolic process	6.877468e-34
15	GO:0006807	nitrogen compound metabolic process	1.482335e-31
16	GO:0044260	cellular macromolecule metabolic process	2.894952e-31
17	GO:0043170	macromolecule metabolic process	2.081095e-30
18	GO:0044267	cellular protein metabolic process	5.374936e-30
19	GO:0044085	cellular component biogenesis	2.198896e-28
20	GO:0044249	cellular biosynthetic process	2.496072e-28

Table 13: 12_heat_shock_1.

	term	description	p-adjust.
1	GO:0044699	single-organism process	1.561494e-10
2	GO:0005975	carbohydrate metabolic process	1.905360e-10
3	GO:0044723	single-organism carbohydrate metabolic process	3.756275e-10
4	GO:0044262	cellular carbohydrate metabolic process	9.092422e-10
5	GO:0006091	generation of precursor metabolites and energy	1.459404e-09
6	GO:0044710	single-organism metabolic process	6.348805e-09
7	GO:0044712	single-organism catabolic process	1.075406e-07
8	GO:0015980	energy derivation by oxidation of organic compounds	1.075406e-07
9	GO:0055114	oxidation-reduction process	3.361147e-07
10	GO:0044763	single-organism cellular process	3.455244e-07
11	GO:0006950	response to stress	5.776457e-07
12	GO:0006508	proteolysis	5.776457e-07
13	GO:0009056	catabolic process	6.113607e-07
14	GO:0016052	carbohydrate catabolic process	6.113607e-07
15	GO:0044724	single-organism carbohydrate catabolic process	6.113607e-07
16	GO:0050896	response to stimulus	1.274111e-06
17	GO:1901575	organic substance catabolic process	2.070463e-06
18	GO:0051716	cellular response to stimulus	5.109371e-06
19	GO:0005996	monosaccharide metabolic process	5.109371e-06
20	GO:0072524	pyridine-containing compound metabolic process	5.109371e-06

Table 14: 13_heat_shock_2.

	term	description	p-adjust.
1	GO:0022613	ribonucleoprotein complex biogenesis	6.398150e-58
2	GO:0042254	ribosome biogenesis	1.965485e-54
3	GO:0010467	gene expression	1.192324e-52
4	GO:0034660	ncRNA metabolic process	1.155424e-41
5	GO:0034470	ncRNA processing	3.906884e-40
6	GO:0006396	RNA processing	1.528704e-39
7	GO:0006364	rRNA processing	1.848234e-38
8	GO:0044085	cellular component biogenesis	3.906063e-38
9	GO:0016072	rRNA metabolic process	9.596845e-38
10	GO:0034641	cellular nitrogen compound metabolic process	7.892055e-37
11	GO:0006807	nitrogen compound metabolic process	5.818113e-35
12	GO:0044260	cellular macromolecule metabolic process	2.021910e-28
13	GO:0006412	translation	3.239351e-28
14	GO:0043043	peptide biosynthetic process	1.131198e-27
15	GO:0043170	macromolecule metabolic process	1.257546e-27
16	GO:0016070	RNA metabolic process	1.834100e-26
17	GO:0002181	cytoplasmic translation	7.073856e-26
18	GO:0006518	peptide metabolic process	1.291624e-25
19	GO:0043604	amide biosynthetic process	3.967591e-25
20	GO:0090304	nucleic acid metabolic process	3.270826e-22

Table 15: 14_nitrogen_depletion.

	term	description	p-adjust.
1	GO:1901605	alpha-amino acid metabolic process	1.029080e-19
2	GO:0016053	organic acid biosynthetic process	3.325122e-19
3	GO:0046394	carboxylic acid biosynthetic process	3.325122e-19
4	GO:0008652	cellular amino acid biosynthetic process	8.232890e-19
5	GO:1901607	alpha-amino acid biosynthetic process	1.747650e-18
6	GO:0019752	carboxylic acid metabolic process	5.286085e-18
7	GO:0043436	oxoacid metabolic process	7.086621e-18
8	GO:0006082	organic acid metabolic process	7.887305e-18
9	GO:0044283	small molecule biosynthetic process	7.887305e-18
10	GO:0006520	cellular amino acid metabolic process	8.789373e-18
11	GO:0044281	small molecule metabolic process	7.013921e-17
12	GO:0044711	single-organism biosynthetic process	6.173953e-13
13	GO:0032197	transposition, RNA-mediated	1.140563e-12
14	GO:0032196	transposition	1.140563e-12
15	GO:0044763	single-organism cellular process	2.378619e-12
16	GO:0009066	aspartate family amino acid metabolic process	6.690499e-12
17	GO:0044710	single-organism metabolic process	8.359101e-12
18	GO:0009067	aspartate family amino acid biosynthetic process	4.292215e-10
19	GO:0009084	glutamine family amino acid biosynthetic process	1.397522e-07
20	GO:0009064	glutamine family amino acid metabolic process	2.663769e-07

Table 16: 15_YPD_1.

	term	description	p-adjust.
1	GO:0055114	oxidation-reduction process	4.824315e-08
2	GO:0015980	energy derivation by oxidation of organic compounds	6.532605e-08
3	GO:0045333	cellular respiration	1.978998e-07
4	GO:0006091	generation of precursor metabolites and energy	7.468324e-07
5	GO:0044710	single-organism metabolic process	1.152118e-06
6	GO:0044763	single-organism cellular process	1.929617e-06
7	GO:0009060	aerobic respiration	2.720481e-06
8	GO:0006119	oxidative phosphorylation	2.220305e-05
9	GO:0022904	respiratory electron transport chain	4.184641e-05
10	GO:0042773	ATP synthesis coupled electron transport	4.184641e-05
11	GO:0042775	mitochondrial ATP synthesis coupled electron transport	4.184641e-05
12	GO:0022900	electron transport chain	8.983699e-05
13	GO:0044723	single-organism carbohydrate metabolic process	1.763532e-04
14	GO:0019236	response to pheromone	0.0004331063
15	GO:0019953	sexual reproduction	0.0004331063
16	GO:0044703	multi-organism reproductive process	0.0004331063
17	GO:0051704	multi-organism process	0.0004737045
18	GO:0033554	cellular response to stress	4.773348e-04
19	GO:0000746	conjugation	0.0005125271
20	GO:0000747	conjugation with cellular fusion	0.0005125271

Table 17: 16_YPD_2.

	term	description	p-adjust.
1	GO:0015980	energy derivation by oxidation of organic compounds	2.818721e-12
2	GO:0006091	generation of precursor metabolites and energy	1.958345e-11
3	GO:0055114	oxidation-reduction process	3.189884e-11
4	GO:0045333	cellular respiration	9.617200e-11
5	GO:0009060	aerobic respiration	2.215004e-09
6	GO:0044723	single-organism carbohydrate metabolic process	2.096364e-07
7	GO:0044262	cellular carbohydrate metabolic process	3.787796e-07
8	GO:0044710	single-organism metabolic process	1.647733e-06
9	GO:0006119	oxidative phosphorylation	5.683543e-06
10	GO:0022900	electron transport chain	1.406935e-05
11	GO:0005975	carbohydrate metabolic process	1.446949e-05
12	GO:0006099	tricarboxylic acid cycle	3.202010e-05
13	GO:0022904	respiratory electron transport chain	6.942742e-05
14	GO:0042773	ATP synthesis coupled electron transport	6.942742e-05
15	GO:0042775	mitochondrial ATP synthesis coupled electron transport	6.942742e-05
16	GO:0016310	phosphorylation	9.188584e-05
17	GO:0072329	monocarboxylic acid catabolic process	6.187705e-04
18	GO:0032787	monocarboxylic acid metabolic process	8.357263e-04
19	GO:0070887	cellular response to chemical stimulus	1.097774e-03
20	GO:0007005	mitochondrion organization	1.384831e-03

Table 18: 17_yeast_sporulation.

	term	description	p-adjust.
1	GO:0051321	meiotic cell cycle	2.791587e-30
2	GO:0022402	cell cycle process	8.461740e-30
3	GO:0007049	cell cycle	1.126408e-29
4	GO:1903046	meiotic cell cycle process	1.089221e-28
5	GO:0044702	single organism reproductive process	1.089221e-28
6	GO:0000003	reproduction	8.439478e-26
7	GO:0022414	reproductive process	1.521625e-25
8	GO:0048646	anatomical structure formation involved in morphogenesis	5.579015e-23
9	GO:0030435	sporulation resulting in formation of a cellular spore	1.380838e-22
10	GO:0043934	sporulation	2.989946e-22
11	GO:0030154	cell differentiation	2.040089e-20
12	GO:0009653	anatomical structure morphogenesis	4.514985e-20
13	GO:0048856	anatomical structure development	4.514985e-20
14	GO:0000280	nuclear division	2.341920e-19
15	GO:0048285	organelle fission	2.341920e-19
16	GO:0032502	developmental process	8.174620e-17
17	GO:0044767	single-organism developmental process	8.174620e-17
18	GO:0048869	cellular developmental process	1.516129e-16
19	GO:0034293	sexual sporulation	1.053153e-14
20	GO:0043935	sexual sporulation resulting in formation of a cellular spore	1.053153e-14

Bibliography

[AGS16] J. O. R. Aoga, T. Guns, and P. Schaus. “An Efficient Algorithm for Mining Frequent Sequence with Constraint Programming”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2016, pp. 315–330. DOI: 10.1007/978-3-319-46227-1_20.

[AHL12] D. Aloise, P. Hansen, and L. Liberti. “An improved column generation algorithm for minimum sum-of-squares clustering”. In: *Mathematical Programming* 131.1-2 (Feb. 1, 2012), pp. 195–220. ISSN: 1436-4646. DOI: 10.1007/s10107-010-0349-7.

[Ash+00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. “Gene Ontology: tool for the unification of biology”. In: *Nature genetics* 25.1 (2000), p. 25.

[Atz15] M. Atzmueller. “Subgroup discovery”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.1 (2015), pp. 35–49.

[Bar+98] 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”. In: *Operations research* 46.3 (1998), pp. 316–329.

[Ben84] J. Bentley. “Programming pearls: algorithm design techniques”. In: *Communications of the ACM* 27.9 (1984), pp. 865–873.

[Bes+16] C. Bessiere, L. De Raedt, L. Kotthoff, S. Nijssen, B. O’Sullivan, and D. Pedreschi. *Data mining and constraint programming*. 2016.

[BH95] Y. Benjamini and Y. Hochberg. “Controlling the false discovery rate: a practical and powerful approach to multiple testing”. In: *Journal of the royal statistical society. Series B (Methodological)* (1995), pp. 289–300.

[BIB03] S. Bergmann, J. Ihmels, and N. Barkai. “Iterative signature algorithm for the analysis of large-scale gene expression data”. In: *Physical review E* 67.3 (2003), p. 031902.

[BKC10] D. Bozdağ, A. S. Kumar, and U. V. Catalyurek. “Comparative analysis of biclustering algorithms”. In: *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*. ACM. 2010, pp. 265–274.

[Bra+19a] V. Branders, G. Derval, P. Schaus, and P. Dupont. *Dataset generator for "Mining a maximum weighted set of disjoint submatrices"*. Aug. 2019. DOI: 10.5281/zenodo.3372282.

[Bra21] V. Branders. *Implementation of a constraint for the maximal sum submatrix problem*. Jan. 2021. DOI: 10.5281/zenodo.4608026.

[BS19] V. Branders and P. Schaus. *Code as used in "Mining a sub-matrix of maximal sum"*. Feb. 2019. DOI: 10.5281/zenodo.4607869.

[BT93] P. Briggs and L. Torczon. “An efficient representation for sparse sets”. In: *ACM Letters on Programming Languages and Systems (LOPLAS)* 2.1-4 (1993), pp. 59–69.

[CC00] Y. Cheng and G. M. Church. “Biclustering of expression data.” In: *Ismb*. Vol. 8. 2000. 2000, pp. 93–103.

[CKB10] G. Csárdi, Z. Kutalik, and S. Bergmann. “Modular analysis of gene expression data with R”. In: *Bioinformatics* 26.10 (2010), pp. 1376–1377.

[CS17] M. Chabert and C. Solnon. “Constraint programming for multi-criteria conceptual clustering”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2017, pp. 460–476.

[Dao+18] T. Dao, F. Docquier, M. Maurel, and P. Schaus. “Global migration in the 20th and 21st centuries: the unstoppable force of demography”. In: (2018).

[DDS06] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*. Vol. 5. Springer Science & Business Media, 2006.

[de +08] M. C. de Souto, I. G. Costa, D. S. de Araujo, T. B. Ludermir, and A. Schliep. “Clustering cancer gene expression data: a comparative study”. In: *BMC bioinformatics* 9.1 (2008), p. 497.

[Dem06] J. Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *Journal of Machine learning research* 7.Jan (2006), pp. 1–30.

[Der+18] G. Derval, V. Branders, P. Dupont, and P. Schaus. *Synthetic dataset used in "The maximum weighted submatrix coverage problem: A CP approach"*. Zenodo, Nov. 2018. DOI: 10 . 5281 / zenodo . 1688740.

[DKT96] M. Dawande, P. Keskinocak, and S. Tayur. *On the biclique problem in bipartite graphs.* 1996.

[DM02] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical programming* 91.2 (2002), pp. 201–213.

[DV+17] K.-C. Duong, C. Vrain, et al. "Constrained clustering by constraint programming". In: *Artificial Intelligence* 244 (2017), pp. 70–94.

[Ere+12] K. Eren, M. Deveci, O. Küçüktunç, and Ü. V. Çatalyürek. "A comparative analysis of biclustering algorithms for gene expression data". In: *Briefings in bioinformatics* 14.3 (2012), pp. 279–292.

[FG15] H. Fanaee-T and J. Gama. "Eigenspace method for spatiotemporal hotspot detection". In: *Expert Systems* 32.3 (2015). EXSY-Nov-13-198.R1, pp. 454–464. ISSN: 1468-0394. DOI: `10.1111/exsy.12088`.

[Fri37] M. Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[Ger97] C. Gervet. "Interval propagation to reason about sets: Definition and implementation of a practical language". In: *Constraints* 1.3 (1997), pp. 191–244.

[GJ90] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co., 1990. ISBN: 0716710455.

[Gur18] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual.* `http://www.gurobi.com`. 2018.

[Har72] J. A. Hartigan. "Direct clustering of a data matrix". In: *Journal of the american statistical association* 67.337 (1972), pp. 123–129.

[Her+11] F. Herrera, C. J. Carmona, P. González, and M. J. del Jesus. "An overview on subgroup discovery: foundations and applications". In: *Knowledge and Information Systems* 29.3 (Dec. 2011), pp. 495–525. ISSN: 0219-3116. DOI: `10.1007/s10115-010-0356-2`.

[HK06] W.-J. van Hoeve and I. Katriel. "Global constraints". In: *Foundations of Artificial Intelligence.* Vol. 2. Elsevier, 2006, pp. 169–208.

[Hoc88] Y. Hochberg. "A sharper Bonferroni procedure for multiple tests of significance". In: *Biometrika* 75.4 (1988), pp. 800–802.

[IOC] IOC Research and Reference Service, The Guardian. *Olympic Sports and Medals, 1896-2014.* `https://www.kaggle.com/the-guardian/olympic-games`.

[JCC13] P. A. Jaskowiak, R. J. Campello, and I. G. Costa Filho. "Proximity measures for clustering gene expression microarray data: a validation methodology and a comparative analysis". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 10.4 (2013), pp. 845–857.

[JCC14] P. A. Jaskowiak, R. J. Campello, and I. G. Costa. "On the selection of appropriate distances for gene expression data clustering". In: *BMC bioinformatics.* Vol. 15. 2. BioMed Central. 2014, S2.

[Kai+18] S. Kaiser, R. Santamaria, T. Khamiakova, M. Sill, R. Theron, L. Quintales, F. Leisch, and E. De Troyer. *biclust: BiCluster Algorithms.* R package version 2.0.1. 2018.

[Kar72] R. M. Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations.* Springer, 1972, pp. 85–103.

[Klu+03] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. "Spectral biclustering of microarray data: coclustering genes and conditions". In: *Genome research* 13.4 (2003), pp. 703–716.

[Kuo+18] C.-T. Kuo, S. Ravi, C. Vrain, I. Davidson, et al. "Descriptive Clustering: ILP and CP Formulations with Applications". In: *IJCAI-ECAI 2018, the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence.* 2018.

[Le +14] T. Le Van, M. Van Leeuwen, S. Nijssen, A. C. Fierro, K. Marchal, and L. De Raedt. "Ranked tiling". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer. 2014, pp. 98–113.

[Li+09] G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu. "QUBIC: a qualitative biclustering algorithm for analyses of gene expression data". In: *Nucleic acids research* 37.15 (2009), e101–e101.

[LO02] L. Lazzeroni and A. Owen. "Plaid models for gene expression data". In: *Statistica sinica* (2002), pp. 61–86.

[LS14] M. López-Ibánez and T. Stützle. "Automatically improving the anytime behaviour of optimisation algorithms". In: *European Journal of Operational Research* 235.3 (2014), pp. 569–582.

[LW03] J. Liu and W. Wang. "Op-cluster: Clustering by tendency in high dimensional space". In: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on.* IEEE. 2003, pp. 187–194.

[MK02] T. Murali and S. Kasif. "Extracting conserved gene expression motifs from gene expression data". In: *Biocomputing 2003.* Singapore: World Scientific, 2002, pp. 77–88.

[MO04] S. C. Madeira and A. L. Oliveira. “Biclustering algorithms for biological data analysis: a survey”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 1.1 (2004), pp. 24–45.

[MSV18] L. Michel, P. Schaus, and P. Van Hentenryck. *MiniCP: A Lightweight Solver for Constraint Programming.* Available from `https://minicp.bitbucket.io`. 2018.

[MXG79] M. J. Miller, N.-H. Xuong, and E. P. Geiduschek. “A response of protein synthesis to temperature shift in the yeast Saccharomyces cerevisiae”. In: *Proceedings of the National Academy of Sciences* 76.10 (1979), pp. 5222–5225.

[NW88] G. L. Nemhauser and L. A. Wolsey. “Integer programming and combinatorial optimization”. In: *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin* 20 (1988), pp. 8–12.

[Osc12] OscaR Team. *OscaR: Scala in OR.* `https://bitbucket.org/oscarlib/oscar`. 2012.

[PC17a] V. A. Padilha and R. J. Campello. “A systematic comparative evaluation of biclustering techniques”. In: *BMC bioinformatics* 18.1 (2017), p. 55.

[PC17b] V. A. Padilha and R. J. G. B. Campello. *Cancer benchmark used in "A systematic comparative evaluation of biclustering techniques".* Github, 2017.

[PC17c] V. A. Padilha and R. J. G. B. Campello. *Yeast benchmark used in "A systematic comparative evaluation of biclustering techniques".* Github, 2017.

[PGA15] B. Pontes, R. Giráldez, and J. S. Aguilar-Ruiz. “Biclustering on expression data: A review”. In: *Journal of biomedical informatics* 57 (2015), pp. 163–180.

[Pio+13] G. Pio, M. Ceci, D. D’Elia, C. Loglisci, and D. Malerba. “A Novel Biclustering Algorithm for the Discovery of Meaningful Biological Correlations between microRNAs and their Target Genes”. In: *BMC Bioinformatics* 14.7 (Apr. 2013), S8. ISSN: 1471-2105. DOI: `10.1186/1471-2105-14-S7-S8`.

[Pio+15] G. Pio, M. Ceci, D. Malerba, and D. D’Elia. “ComiRNet: a web-based system for the analysis of miRNA-gene regulatory networks”. In: *BMC bioinformatics* 16.9 (2015), S7.

[Pre+06] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. "A systematic comparison and evaluation of biclustering methods for gene expression data". In: *Bioinformatics* 22.9 (2006), pp. 1122–1129.

[SAG17] P. Schaus, J. O. Aoga, and T. Guns. "CoverSize: a global constraint for frequency-based itemset mining". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2017, pp. 529–546.

[Sai+13] V. l. C. de Saint-Marcq, P. Schaus, C. Solnon, and C. Lecoutre. "Sparse-sets for domain implementation". In: *CP workshop on Techniques foR Implementing Constraint programming Systems (TRICS)*. 2013, pp. 1–10.

[Sav97] M. Savelsbergh. "A Branch-and-Price Algorithm for the Generalized Assignment Problem". In: *Operations Research* 45.6 (1997), pp. 831–841.

[Sha98] P. Shaw. "Using constraint programming and local search methods to solve vehicle routing problems". In: *International conference on principles and practice of constraint programming*. Springer. 1998, pp. 417–431.

[Sub+20] I. Subramanian, S. Verma, S. Kumar, A. Jere, and K. Anamika. "Multi-omics data integration, interpretation, and its application". In: *Bioinformatics and biology insights* 14 (2020), p. 1177932219899051.

[Tak02] T. Takaoka. "Efficient algorithms for the maximum subarray problem by distance matrix multiplication". In: *Electronic Notes in Theoretical Computer Science* 61 (2002), pp. 191–200.

[TBK05] H. Turner, T. Bailey, and W. Krzanowski. "Improved biclustering of microarray data demonstrated through systematic performance tests". In: *Computational statistics & data analysis* 48.2 (2005), pp. 235–254.

[TT98] H. Tamaki and T. Tokuyama. "Algorithms for the Maximum Subarray Problem Based on Matrix Multiplication." In: *SODA*. Vol. 1998. 1998, pp. 446–452.

[Xie+18] J. Xie, A. Ma, A. Fennell, Q. Ma, and J. Zhao. "It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data". In: *Briefings in bioinformatics* (2018).

[Yu+12] G. Yu, L.-G. Wang, Y. Han, and Q.-Y. He. “clusterProfiler: an R package for comparing biological themes among gene clusters”. In: *OMICS: A Journal of Integrative Biology* 16.5 (2012), pp. 284–287. DOI: `10.1089/omi.2011.0118`.