Mining a maximum weighted set of disjoint submatrices

Vincent Branders[0000−0001−8688−7498], Guillaume Derval[0000−0002−6700−3519], Pierre Schaus[0000−0002−3153−8941], Pierre Dupont[0000−0003−4835−6519]

UCLouvain - ICTEAM/INGI
{firstname.lastname}@uclouvain.be

Abstract. 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 differs from the maximum-weighted submatrix coverage problem introduced in [6] by the 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 problem, was successfully tackled with constraint programming in [5]. 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 a hybrid column generation approach using constraint programming 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.

Keywords: Constraint Programming · Maximum Weighted Submatrix · Column Generation · Maximum Weighted Set of Disjoint Submatrices Problem · Bi-cliques · Data-mining.

1 Introduction

1.1 Problem Definition

We are interested in the mining of a numerical matrix to discover submatrices capturing a high total value. Precisely, we consider an input matrix $\mathcal{M}$ with m rows and n columns where element $\mathcal{M}_{i,j}$ is a given real value. The matrix is associated with a set of rows $R = \{r_1, \dots, r_m\}$ and a set of columns $C = \{c_1, \dots, n\}$. We use $(R; C)$ to denote the matrix $\mathcal{M}$.

If $I \subseteq R$ and $J \subseteq C$ are subsets of the rows and of the columns, respectively, the submatrix $(I; J)$ denotes all the elements $\mathcal{M}_{i,j}$ of $\mathcal{M}$ such that $i \in I \wedge j \in J$.

The max-sum submatrix problem (MSSP), introduced in [5], consists in identifying a subset of rows and of columns of an input matrix that maximizes the sum of the covered entries, which is the submatrix weight. The problem is formally stated below.

The Max-Sum Submatrix Problem (MSSP) : Given a matrix $\mathcal{M} \in \mathbb{R}^{m \times n}$, $R = \{1, \dots, m\}$ and $C = \{1, \dots, n\}$ the associated sets of rows and columns, respectively. The submatrix $(I^* \subseteq R, J^* \subseteq C)$ is of max-sum iff:

$$(I^*; J^*) = \underset{I,J}{\operatorname{argmax}} \sum_{i \in I, j \in J} \mathcal{M}_{i,j} \tag{1}$$

In this paper, we consider only the non-trivial problem matrices containing both positive and negative entries. Such a problem is both compelling and challenging to solve. A constraint programming (CP) implementation successfully tackled this difficult problem for matrices of thousands of rows and hundreds of columns, as is typical in several biological applications [5].

A natural extension of the MSSP is to identify K submatrices. The maximum weighted submatrix coverage problem (MWSCP) proposed in [6] is 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. However, it favors overlaps on negative entries: penalties are distributed among overlaps. Moreover, overlaps on positive entries will not improve the objective value.

In the present work, we consider an alternative extension to the identification of K submatrices, relying on an objective function computed as the sum of submatrix weights, and the explicit addition of disjunction constraints. By allowing overlaps on the rows *or* the columns (but not both simultaneously due to the disjunction constraint) we avoid the unexpected behavior of the MWSCP. Moreover, the solution's interpretability by a domain expert is eased. Such a solution is usually called *nonoverlapping nonexclusive nonexhaustive* in the biclustering context [10].

Definition 1. ***The Maximum Weighted Set of Disjoint Submatrices Problem (MWSDSP)****: Given a matrix $\mathcal{M} \in \mathbb{R}^{m \times n}$, $R = \{1, \dots, m\}$ and $C = \{1, \dots, n\}$ be the associated sets of rows and of columns, respectively, and K be a target number of submatrices. The maximum weighted set of disjoint submatrices problem is to select a set of K submatrices $(I^{k*}; J^{k*})$, with $I^{k*} \subseteq R$ and $J^{k*} \subseteq C$ for all $k \in \{1, \dots, K\}$, such that each matrix entry is covered by at most one submatrix and the weight of the covered entries is maximal:*

$$(I^{1*}; J^{1*}), \cdots, (I^{K*}; J^{K*}) = \underset{(I^1;J^1),\cdots,(I^K;J^K)}{\operatorname{argmax}} \sum_{k=1}^{K} w_k \tag{2}$$

$$s.t. \quad (I^k \times J^k) \cap (I^{k'} \times J^{k'}) = \emptyset \quad \forall k, k' \in \{1, \dots, K\}, k \neq k' \tag{3}$$

where $w_k = \sum_{r \in I^k, c \in J^k} \mathcal{M}_{r,c}$ is the weight of submatrix k.

Disjunction constraints (3) 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.

1.2 Contributions

Our contributions are: (1) The introduction of the maximum weighted set of disjoint submatrices problem (MWSDSP) as a generalization of the max-sum submatrix (MSSP) problem; (2) A mathematical programming approach to solve the MWSDSP; (3) The formulation of the MWSDSP as an integer linear program (ILP) relying on constraint programming (CP) to produce relevant variables; (4) An evaluation of the performances of these two alternatives and the benefit of the ILP+CP over a greedy approach on synthetic datasets.

1.3 Motivation

The MWSDSP has many practical data-mining applications where one is interested in discovering K specific relations between two groups of variables.

As an example, in gene expression analysis, $\mathcal{M}_{i,j}$ corresponds to the expression value of gene i in sample j. One is typically interested in finding a subset of genes that present high expression value, i.e., an active biological pathway, in a subset of the samples. Finding multiple pathways specific to some samples is a common task in gene expression analysis. Submatrices overlaps would correspond to non-specific signal. In contrast, shared rows only would correspond to gene simultaneously active in multiple pathways, and shared columns only to subpopulations of samples exhibiting the same pathway activity.

1.4 Related Work

The max-sum submatrix problem (MSSP) and the maximum weighted submatrix coverage problem (MWSCP), presented in section 1.1, are $\mathcal{NP}$-hard [6]. The present work and the MWSCP extend the MSSP to $K > 1$ by adding disjunction constraint and by adapting the objective function, respectively.

In the maximum subarray problem, introduced in [3], the aim is to find a subset of contiguous columns with maximal weight from an array. Polynomial-time complexity algorithms have been proposed for matrices [14]. This problem is simpler than the MWSDSP, however, as a single submatrix is required and it is constrained to be formed of contiguous subsets of rows and columns.

The biclustering problems are concerned with the discovery of homogeneous submatrices rather than maximizing the weight of covered entries. Madeira *at al.* provided a comprehensive review of biclustering problems [10].

The minimum sum-of-squares clustering problem 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 [2]: the combination of an ILP and delayed column generation.

In the ranked tile mining problem, introduced in [9], entries are discrete ranks, corresponding to a permutation of column indices on each row. Moreover, the definition of a parametrized penalty for overlapping coverage discourages but allows identification of repetitive solutions.

2 Constraint Programming Approaches

2.1 Search Space

Let us define a set variable T^k (resp. U^k) to represent the rows (resp. columns) included in submatrix k. The search space of the MSSP can be limited to searching on a single dimension, for instance the column set variable U^1. Indeed, optimal T^1 can be found in polynomial time: $\forall i \in R$: $\sum_{j \in U^1} \mathcal{M}_{i,j} > 0 \implies i \in T^1$.

Let us define the MWSDSP with fixed column selections formally.

Definition 2. ***The MWSDSP with fixed column selections***. *The notations are the same as in Definition (1), but in this case the selections of columns for each submatrices (the C^k sets) are given.*

$$R^{1*}, \cdots, R^{K*} = \underset{R^{1*}, \cdots, R^{K*}}{\operatorname{argmax}} \sum_{k=1}^{K} \sum_{r \in R^k, c \in C^k} \mathcal{M}_{r,c} \tag{4}$$

$$s.t. \quad (R^k \times C^k) \cap (R^{k'} \times C^{k'}) = \emptyset \quad \forall k, k' \in \{1, \ldots, K\}, k \neq k' \tag{5}$$

For $K > 1$, once all the column set variables U^k are fixed, it remains to decide for each row i and each submatrix k whether i is to be selected ($i \in T^k$) or not. These K decisions per row cannot be optimally taken in polynomial time, as stated in Theorem (1). 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 1. *The MWSDSP 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 [8]), 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 MWSDSP 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 MWSDSP 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 MWSDSP 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 MWSDSP with fixed column selections can encode the MWIS problem, we conclude that the MWSDSP with fixed column selections is $\mathcal{NP}$-Hard. □

2.2 Greedy Approach

A simple approach to solving the MWSDSP is to solve the MSSP repeatedly. For each new max-sum submatrix found, the corresponding values are replaced by $-\infty$, forbidding subsequent iterations from selecting these entries again.

Each iteration is performed until optimality or absence of solution are proved; or at least one solution has been found.

2.3 Column Generation

We propose a column generation (CG) approach [7] to find solutions to the MWSDSP. It relies on CP[1] in an ILP setting. The CP part identifie candidates submatrices. The ILP efficiently combines submatrices and guides the CP part.

Let us represent the given matrix $\mathcal{M}$ of $m \times n$ entries as the vector $\mathcal{V}$ of $v = m \times n$ entries obtained by stacking the columns of the matrix $\mathcal{M}$ on top of one another. The MWSDSP is formulated using a $v \times 2^{m+n}$ binary matrix $\mathcal{B}$ representing all 2^{m+n} possible submatrices. Each column l of $\mathcal{B}$ corresponds to a submatrix l such that $\mathcal{B}_{i,l} = 1$ if and only if entry $\mathcal{V}_i$ is covered by the submatrix l. The weight w_l of submatrix l is the sum of its covered entries: $w_l = \sum_{i=1}^{v} \mathcal{V}_i \times \mathcal{B}_{i,l}$. Equations (2) and (3) can be formulated as an ILP:

$$\text{maximize} \quad \sum_{l \in L} w_l \times x_l \tag{6a}$$

$$\text{s.t.} \quad \sum_{l \in L} \mathcal{B}_{i,l} \times x_l \leq 1 \qquad \forall i \in \{1, \ldots, v\} \tag{6b}$$

$$\sum_{l \in L} x_l \leq K \tag{6c}$$

$$x_l \in \{0, 1\} \quad \forall l \in L \tag{6d}$$

where $L = \{1, \ldots, 2^{m+n}\}$ denotes all possible submatrices. The decision variable x_l encodes the selection of submatrix l. Equation (6b) ensures submatrices disjunction and eq. (6c) enforces the selection of at most K submatrices.

Defining the matrix $\mathcal{B}$ before solving the ILP is computationally not feasible, even for small input matrices $\mathcal{M}$. In subproblem solving, the master problem (or ILP), in eq. (6a)-(6d), is restricted to a subset $L' \subseteq L$ of submatrices effectively defining a restricted master problem (RMP). Iteratively, an RMP is solved, and one or multiple new submatrices (columns) are inserted in 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 RMP (LP-RMP) which comes along the integrality constraints (6d) relaxation of the ILP (in an LP) and the subsetting of L. We use the dual of the LP-RMP to find submatrices with a positive reduced cost[2]. 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. The dual of the LP-RMP is:

$$\text{minimize} \quad \theta \times K + \sum_{i=1}^{v} \lambda_i \tag{7a}$$

[1] See [11] for an introduction to CP.

[2] Given that the problem is a maximization problem.

$$\text{s.t.} \quad \theta + \sum_{i=1}^{v} \mathcal{B}_{i,l} \times \lambda_i \geq w_l \quad \forall l \in L' \tag{7b}$$

$$\lambda_i \geq 0 \quad \forall i \in \{1, \dots, v\} \tag{7c}$$

$$\theta \geq 0 \tag{7d}$$

The dual values λ_i and θ corresponding to the primal constraints defined in eq. (6b) and (6c), respectively, are obtained by solving an LP-RMP. Each column x_l of the RMP is associated with a constraint in the dual (eq. 7b).

Finding a submatrix with a positive reduced cost is called pricing. Such a submatrix is defined as any submatrix $l \in L$ for which $-\theta - \sum_{i=1}^{v} \mathcal{B}_{i,l} \times \lambda_i + w_l < 0$. 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: $\sum_{i=1}^{v} [\mathcal{B}_{i,l} \times (\mathcal{V}_i - \lambda_i)] > \theta$.

Solving this pricing problem is not trivial: it amounts to identifying a submatrix in the input matrix modified by the λ_i values such that its weight is larger than some θ. While the pricing routine usually tries to identify a solution with maximum reduced cost, it can return any submatrix with positive reduced cost.

In practice, we use the greedy approach described earlier to find submatrices of weight larger than θ from an input matrix modified according to the λ_i values. This provides solutions to the pricing problem.

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

To maximize the information given by the dual values, we avoid having redundant constraints, notably the constraints (6b). For example, if two submatrices overlap on more than one cell, we enforce only one constraint representing all the overlapping cells. Precisely, constraint (6b) is replaced by the following:

$$\sum_{l \in S} x_l \leq 1 \quad \forall S \in \Big\{ \{l \mid \mathcal{B}_{i,l} = 1\} \;\Big|\; i \in \{1, \dots, v\} \Big\}. \tag{8}$$

That is, we enforce one non-overlap constraint per group of entries sharing the same intersecting submatrices (an *overlapping group*)[3]. 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.

The LP-RMP does not necessarily provide a binary decision on the submatrix selection. To effectively identify a solution to the original MWSDSP, 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.

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.

[3] Equation (8) uses the set notation to implicitly remove duplicates.

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 [13], 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. The authors consider that the use of a branch-and-price algorithm is outside of the paper's scope.

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.

2.4 Mixed Integer Linear Programming

We propose a Mixed Integer Linear Programming model using the binary variables T_i^k and U_j^k to represent the selection of row i and column j for submatrix k. These decision variables are used to compute the contribution of the row i for the submatrix k (r_i^{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 MWSDSP where, $\forall i \in R$, constants $M_i^- = \sum_{j \in C} \min(0, \mathcal{M}_{i,j})$ and $M_i^+ = \sum_{j \in C} \max(0, \mathcal{M}_{i,j})$ are respectively the lower bound and upper bound on the sum of row i's entries. The MILP model is formulated as follows:

$$\begin{aligned}
\text{maximize} \quad & & & \sum_{i \in R, k \in \mathcal{K}} r_i^{k+} & & \text{(9a)}\\
\text{s.t.} \quad & r_i^{k+} & \leq & \sum_{j \in C} \left(\mathcal{M}_{i,j} \times U_j^k \right) + (T_i^k - 1) \times M_i^- & \forall i, k \quad & \text{(9b)}\\
& r_i^{k+} & \leq & M_i^+ \times T_i^k & \forall i, k \quad & \text{(9c)}\\
& 2 \times v_{i,j}^k & \leq & T_i^k + U_j^k & \forall i, j, k \quad & \text{(9d)}\\
& T_i^k + U_j^k & \leq & 1 + v_{i,j}^k & \forall i, j, k \quad & \text{(9e)}\\
& \sum_{k \in \mathcal{K}} v_{i,j}^k & \leq & 1 & \forall i, j \quad & \text{(9f)}
\end{aligned}$$

Constraints (9b) and (9c) ensure that the row contribution r_i^{k+} is computed correctly. If $T_i^k = 0$, constraint (9c) ensures the row contribution is zero, with the right hand side of constraint (9b) being always positive. Otherwise ($T_i^k = 1$), constraints (9b) and (9c) ensure $r_i^{k+} = \sum_{j \in C} \left(\mathcal{M}_{i,j} \times U_j^k \right)$, thus computing the effective value of the contribution.

Equations (9d) and (9e) linearize $v_{i,j}^k = T_i^k \times U_j^k$. The binary variable $v_{i,j}^k$ indicates if cell (i, j) is selected by submatrix k and ensures submatrices disjunction through constraint (9f).

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

3 Experiments

This section describes experiments conducted to assess the performances of the proposed algorithms and to provide guidance on the selection of the appropriate solution. Given enough time and memory, both the column generation (CG) approach and the MILP approach converge to the optimal solution. Therefore comparing performances 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 TO, on synthetic datasets with implanted submatrices using any-time profiles:

Definition 3. ***Any-Time Profile.*** *Let* $\mathrm{f(a,\ i,\ t)}$ *be the objective value of the best solution found so far by an algorithm* a *for an instance* i *at time* t. *Let* $\mathrm{t^{max}}$ *be the provided budget of time before breaking a run. The any-time profile of* a *is the solution quality* $\mathrm{Q_a(t)}$ *of* a *on all instances as a function of time:*

$$\mathrm{Q_a(t)} = \frac{1}{|\mathrm{i}|}\sum_{\mathrm{i}} \frac{\mathrm{f(a,i,t)}}{\mathrm{f(a_i^*,i,t^{max})}} \ \textit{with}\ \mathrm{a_i^*} = \underset{\mathrm{a}}{\mathrm{argmax}}\ \mathrm{f(a,i,t^{max})}\ . \quad (10)$$

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 TO of 2 hours. MILP and CG approaches rely on Gurobi 8.1.0 [1]. The greedy hot-start of the CG process is given 5 minutes evenly split between each of its K iterations of solving an MSSP. Solutions to the MSSP are carried out on OscaR [12] using a constraint programming approach relying on a global constraint (CPGC) provided in [5]. It is a depth-first search approach composed of major CP ingredients: 1) filtering rules, 2) bounding procedure, 3) dominance rules and 4) variable-value heuristic.

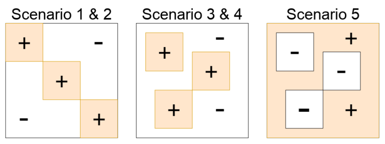
3.1 Datasets and Performances

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 Fig. (1a)-(1b). For each scenario, 14 different matrices are generated according to different input matrix size and number of implanted submatrices, as presented in Fig. (1c). These 70 instances are 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 relative to the suboptimal hot-start solution through the objective value improvement.

Figure (2a) presents the any-time profile of each method for the first dataset. It clearly illustrates that CG 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 the greedy and the CG.

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 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 Fig. (1a). It differs, however, by the size of the input matrix, the number, and size of implanted submatrices. More importantly,



(a)

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)

$m \times n$	$K=2$	$K=5$	$K=8$	$K=10$	$K=20$
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)

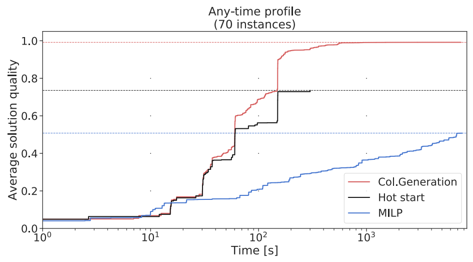
$m \times n$	$K=2$	$K=5$	$K=10$
400×100	$s=s_2$	$s=s_2$	$s=s_2$
320×125	$s=s_2$	$s=s_2$	$s=s_2$
200×200	$s=s_2$	$s=s_2$	$s=s_2$

(d)

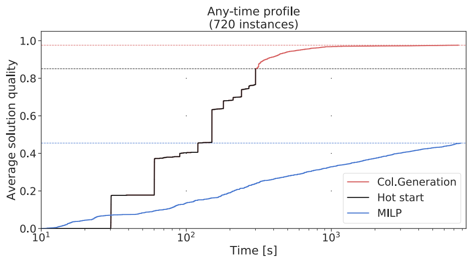
Fig. 1: Dataset construction. (a) Layout and (b) generative distribution of implanted + and - entries. (c) Parameters considered in the first dataset with $s_1 = \{1.0\}$. (d) Parameters considered in the second dataset with $s_2 = \{0.05, 0.01, 0.2, 0.5\}$. Implanted submatrices are of size $\left(\frac{m\times s}{K};\ \frac{n\times s}{K}\right)$.

values are drawn from different distributions: - entries $\sim \mathcal{N}(-1,\ 1)$ and + entries $\sim \mathcal{N}(1,\ 0.5)$. Such matrices, generated following a similar protocol as in [6], are considered better representatives of gene expression matrices. Our script is available on Zenodo [4].

Figure (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. The poor performances 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 (results not shown). In this experiment, the hot-start rarely ends before the allocated 5 minutes, explaining the near-perfect overlap between hot-start and CG curves.



(a) Averaged results on the first dataset.



(b) Averageg results on the second dataset.

Fig. 2: Comparison of the different methods proposed to solve the MWSDSP. The graph presents the any-time profile described in eq. (3). For each instance, the time-out is fixed at 2 hours. The hot-start time-out equals 5 minutes. Col.Generation starts as soon as the hot-starts is completed.

4 Conclusions

We present a new optimization problem, called the Maximum Weighted Set of Disjoint Submatrix Problem (MWSDSP) 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 MWSDSP, 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 MILP approach.

The performances of the CG can be further improved by complementing the exploration with a branch-and-price algorithm [13]. Such improvement is nontrivial, however: the time taken to solve the underlying LP problem is already quite long but is nonetheless an attractive direction for future work.

References

1. Gurobi Optimization, LLC (2018), http://www.gurobi.com
2. Aloise, D., Hansen, P., Liberti, L.: An improved column generation algorithm for minimum sum-of-squares clustering. Mathematical Programming **131**(1), 195–220 (Feb 2012)
3. Bentley, J.: Programming pearls: algorithm design techniques. Communications of the ACM **27**(9), 865–873 (1984)
4. Branders, V., Derval, G., Schaus, P., Dupont, P.: Dataset generator for ”Mining a maximum weighted set of disjoint submatrices” (Aug 2019). https://doi.org/10.5281/zenodo.3372282
5. Branders, V., Schaus, P., Dupont, P.: Combinatorial optimization algorithms to mine a sub-matrix of maximal sum. In: International Workshop on New Frontiers in Mining Complex Patterns. pp. 65–79. Springer (2017)
6. Derval, G., Branders, V., Dupont, P., Schaus, P.: The maximum weighted submatrix coverage problem: A CP approach. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer International Publishing (2019)
7. Desaulniers, G., Desrosiers, J., Solomon, M.M.: Column generation, vol. 5. Springer Science & Business Media (2006)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
9. Le Van, T., Van Leeuwen, M., Nijssen, S., Fierro, A.C., Marchal, K., De Raedt, L.: Ranked tiling. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 98–113. Springer (2014)
10. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **1**(1), 24–45 (2004)
11. Michel, L., Schaus, P., Van Hentenryck, P.: MiniCP: A lightweight solver for constraint programming (2018), available from `https://minicp.bitbucket.io`
12. OscaR Team: OscaR: Scala in OR (2012), available from `https://bitbucket.org/oscarlib/oscar`
13. Savelsbergh, M.: A branch-and-price algorithm for the generalized assignment problem. Operations Research **45**(6), 831–841 (1997)
14. Takaoka, T.: Efficient algorithms for the maximum subarray problem by distance matrix multiplication. Electronic Notes in Theoretical Computer Science **61**, 191–200 (2002)