Integer Programming Based Algorithms for Overlapping Correlation Clustering

Barel I. Mashiach and Roded Sharan(✉)

Blavatnik School of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel
roded@tauex.tau.ac.il

Abstract. Clustering is a fundamental problem in data science with diverse applications in biology. The problem has many combinatorial and statistical variants, yet few allow clusters to overlap which is common in the biological domain. Recently, Bonchi et al. defined a new variant of the clustering problem, termed overlapping correlation clustering, which calls for multi-label cluster assignments that correlate with an input similarity between elements as much as possible. This variant is NP-hard and was solved by Bonchi et al. using a local search heuristic. We revisit this heuristic and develop exact integer-programming based variants for it. We show that these variants perform well across several datasets and evaluation measures.

1 Introduction

Clustering is a fundamental problem in data science. While most clustering methods look for disjoint clusters [2,11], in many real-world application, and specifically in biology, an element might belong to more than one cluster. For example, a gene may have multiple functions, and a protein may belong to multiple protein complexes. One framework for dealing with such multi-label scenarios is *Overlapping Correlation Clustering (OCC)*, which was introduced by Bonchi et al. [3] and generalized a non-overlapping variant called *Correlation Clustering (CC)* [2] or *Cluster Editing* [14]. In *OCC*, one seeks an assignment of clusters to elements that maximizes the correlation with a given pairwise similarity.

Bonchi et al. [3] tackled this problem by iteratively finding the optimal labeling for one element given the labels of all other elements, using efficient, yet heuristic techniques (see Sect. 3.2). Another iterative algorithm for the problem was developed by Andrade et al. [1] which uses *Biased Random-Key Genetic Algorithm (BRKGA)*. A third method used a weighted Lovász theta function for node embedding and subsequently clustering [10]. Finally, Gartzman et al. [8] developed an exact solution by defining an integer-linear-programming formulation for the Jaccard variant of the problem called *ECLIP* (see Sect. 3.1).

Here we revisit previous local methods and develop exact integer-programming based algorithms for them that borrow ideas from both [3,8]. We show a simple algorithm solving each iterative step of Bonchi's algorithm by checking all possible labelings, assuming the number of labels is bounded. Our main contribution is a new iterative method for *OCC* that optimally assigns a

D. Cantone and A. Pulvirenti (Eds.): *From Computational Logic to Computational Biology*,
LNCS 14070, pp. 115–127, 2024.
https://doi.org/10.1007/978-3-031-55248-9_6

specific label given the assignments to all other labels using an integer linear program (ILP) formulation.

2 Preliminaries and Problem Definition

Consider a set of n elements V over which we define pairwise similarity function $s(u, v) \in [0, 1]$. Moreover let K be a set of k clusters $K = \{1, \dots k\}$. In CC the goal is to construct a mapping $\mathcal{L} : V \to K$ that minimizes the cost:

$$C_{CC}(V, \mathcal{L}) = \sum_{\substack{(u,v)\in V\times V \\ \mathcal{L}(u)=\mathcal{L}(v)}} (1 - s(u, v)) + \sum_{\substack{(u,v)\in V\times V \\ \mathcal{L}(u)\neq\mathcal{L}(v)}} s(u, v) \tag{1}$$

In OCC, a multi-labeling mapping $\mathcal{L} : V \to 2^K$ is sought. Let $H(\mathcal{L}(u), \mathcal{L}(v))$ denote a similarity between two cluster assignments of elements u and v according to the mapping $\mathcal{L}$. Following Bonchi et al., we consider two different variants of H: the Jaccard coefficient J and the set-intersection indicator I, defined as follows:

$$J(\mathcal{L}(u), \mathcal{L}(v)) = \frac{|\mathcal{L}(u) \cap \mathcal{L}(v)|}{|\mathcal{L}(u) \cup \mathcal{L}(v)|} \tag{2}$$

$$I(\mathcal{L}(u), \mathcal{L}(v)) = \begin{cases} 1 & \text{if } |\mathcal{L}(u) \cap \mathcal{L}(v)| \neq \emptyset \\ 0 & \text{o.w} \end{cases} \tag{3}$$

Using these definitions the OCC objective is the entry-wise cost function:

$$C_{OCC}(V, \mathcal{L}) = \frac{1}{2} \sum_{u\in V} \sum_{v\in V\setminus\{u\}} |H(\mathcal{L}(u), \mathcal{L}(v)) - s(u, v)| \tag{4}$$

The original problem also addresses an additional constraint p on the mapping $\mathcal{L}$ which limits the number of clusters a single element can be assigned to, and in some cases we also require $p \geq 1$. One should notice that the OCC generalizes CC, which corresponds to the case of $p = 1$, and hence its hardness follows from the fact that CC is NP-hard [6].

3 Methods

3.1 ILP-Based Algorithms

As mentioned above, Gartzman et al. [8] gave a formulation of OCC as an integer-linear program (ILP) for the Jaccard case. The main drawback of this method is that it is not scalable for large datasets and limited due to significant long running time on very small subsets of up to 40 elements (see Fig. 1 below). For consistency with the methods before, we also formulate an ILP for the set-intersection case. The formulation described by Algorithm 1 is simple - we want to find a binary matrix $L \in \{0, 1\}^{n\times k}$ which represents assignments of each of

the n elements to the k labels, denoting $L(i,l)$ as whether element i is assigned to label l. We further define $w^l_{i,j}$ as $L(i,l) \cdot L(j,l)$ representing whether both i and j share the label l. Using these variables we can compute the total number of shared labels between i and j as $y_{i,j}$ by summing $w^l_{i,j}$ over all possible labels and use $b_{i,j}$ as an indicator telling whether i and j share at least one label by setting $b_{i,j}$ to 1 if $y_{i,j} > 0$ and to 0 otherwise. From these definitions we can create the following minimization problem, claiming that $b_{i,j} + s(v_i, v_j) - 2b_{i,j} \cdot s(v_i, v_j)$ is exactly the expression $|H(\mathcal{L}(u), \mathcal{L}(v)) - s(u,v)|$, therefore achieving the objective of the original OCC problem.

Algorithm 1. ILP for OCC with set-intersection (ISEC-ILP)

$\min_L \sum_{i=1}^{n} \sum_{j=i+1}^{n} b_{i,j} + s(v_i, v_j) - 2b_{i,j} \cdot s(v_i, v_j)$ ▷ Equivalent to Eq. 4 (isec)

s.t. $\forall 1 \leq i < j \leq n, 1 \leq l \leq k$
$w^l_{i,j}, b_{i,j} \in \{0,1\}$, $y_{i,j} \in \mathbb{Z}^+$

$w^l_{i,j} \leq L(i,l)$ ▷ Setting $w^l_{i,j} = L(i,l) \cdot L(j,l)$
$w^l_{i,j} \leq L(j,l)$
$L(i,l) + L(j,l) - 1 \leq w^l_{i,j}$

$y_{i,j} = \sum_{l'=1}^{k} w^{l'}_{i,j}$ ▷ Setting $y_{i,j}$ as total number of shared labels

$y_{i,j} - (k+1)b_{i,j} \geq -k$ ▷ Setting $b_{i,j} = 1$ if $y_{i,j} > 0$ and to 0 o.w
$y_{i,j} - (k+1)b_{i,j} \leq 0$

$1 \leq \sum_{l'=1}^{k} L(i,l') \leq p$ ▷ Setting total assignments of an element up to p labels

return L

3.2 Row-Based Clustering

The method of Bonchi et al. [3] for solving the OCC objective iteratively finds the optimal labeling vector for an element v given fixed labeling vectors of all other elements. In each step we aim to find labeling $\mathcal{L}^{t+1}(v)$ which minimizes the cost produced by v with all other elements:

$$\min_{\mathcal{L}^{t+1}(v)} C_{v,p}(\mathcal{L}^{t+1}(v) | \mathcal{L}^t) = \min_{\mathcal{L}^{t+1}(v)} \sum_{u \in V \setminus \{v\}} |H(\mathcal{L}^t(u), \mathcal{L}^t(v)) - s(u,v)| \tag{5}$$

This iterative step requires solving non-trivial optimization subproblems:

- For $H = J$, the problem is related to the *Jaccard-Triangulation* problem [3] which generalizes the NP-hard *Jaccard-Median* problem [5].
- For $H = I$, the problem is related to the *Hit-N-Miss* problem [3] which is isomorphic to the NP-hard *positive-negative partial set-cover* problem [12].

Therefore, Bonchi et al. employ heuristic approaches to tackle the resulting problems. Here we show that one can derive exact yet practical solutions for these problems using fixed parameter and integer programming techniques.

If we denote the number of clusters by a parameter k, then we can use a naive fixed parameter algorithm for precisely solving the local step optimization for every element in $O(2^k)$ time by enumerating all possible labeling vectors for v. Moreover when we know a bound $p \leq k$ on how many clusters a single element can be assigned to, then the number of possible options reduces to $\binom{k}{p} = O(k^p)$.

3.3 Column-Based Clustering

Our main contribution is a new optimization approach which looks in every iteration on each column (label) separately, aiming to assign the label's elements in a way that will minimize the overall clustering objective. Let L^t be a binary matrix of size $n \times k$ which represents the labeling assignment by $\mathcal{L}^t$. We denote $L^t(\cdot, l)$ as the l column representing the assignment for label l at iteration t. Algorithm 2 shows the main flow of our new local search method which iteratively updates the elements assignments L^{t+1} to every label according to previous assignments $\bar{L}^t$ (which for simplicity of notations would be referred just as L^t) to all other labels.

Algorithm 2. Label-by-Label Local Search (COL-ILP)

```
Initialize L^0 to a valid labeling
t ← 1
while C_OCC(V, L^t) decreases do
    Set L̄^t ← L^t
    for l ∈ [k] do
        Find the changes x in label l which maximizes the cost difference (ILP)
        Set L̄^t(·, l) ← L̄^t(·, l) · (1 − x) + (1 − L̄^t(·, l)) · x
        Set L^{t+1} ← L̄^t
    end for
    t ← t + 1
end while
return L^t
```

In order to improve the assignment of a label l we need to calculate the difference in cost of every pair of elements v_i, v_j with respect to the proposed changes in the assignment. There are four cases to consider regarding a specific element v_i:

1. Element v_i is added to label l, i.e., $L^t(i,l) = 0$ and $L^{t+1}(i,l) = 1$.
2. Element v_i remains unlabeled with l, i.e., $L^{t+1}(i,l) = L^t(i,l) = 0$.
3. Element v_i is removed from label l, i.e., $L^t(i,l) = 1$ and $L^{t+1}(i,l) = 0$.
4. Element v_i remains labeled with l, i.e., $L^{t+1}(i,l) = L^t(i,l) = 1$.

These four cases induce 10 types of element pair combinations. To enumerate them, we define $e_i = L^t(i,l)$, and let the variable x_i be 1 if there was a change in v_i with respect to label l and 0 otherwise. By calculating cost differences for each of the 10 pair types we can define an optimization criterion for maximizing the total difference in cost over all pairs:

$$\max_x \sum_i \sum_{j>i} \begin{bmatrix} x_i \cdot x_j \cdot \left[(1-e_i)\cdot(1-e_j)\cdot \hat{C}^{1,1}_{i,j} + e_i \cdot e_j \cdot \hat{C}^{3,3}_{i,j} + (1-e_i)\cdot e_j \cdot \hat{C}^{1,3}_{i,j} + e_i \cdot (1-e_j)\cdot \hat{C}^{3,1}_{i,j}\right] \\ + x_i \cdot (1-x_j)\cdot \left[(1-e_i)\cdot(1-e_j)\cdot \hat{C}^{1,2}_{i,j} + e_i \cdot (1-e_j)\cdot \hat{C}^{3,2}_{i,j} + (1-e_i)\cdot e_j \cdot \hat{C}^{1,4}_{i,j} + e_i \cdot e_j \cdot \hat{C}^{3,4}_{i,j}\right] \\ + (1-x_i)\cdot x_j \cdot \left[(1-e_i)\cdot(1-e_j)\cdot \hat{C}^{2,1}_{i,j} + (1-e_i)\cdot e_j \cdot \hat{C}^{2,3}_{i,j} + e_i \cdot (1-e_j)\cdot \hat{C}^{4,1}_{i,j} + e_i \cdot e_j \cdot \hat{C}^{4,3}_{i,j}\right] \end{bmatrix} \tag{6}$$

where $\hat{C}^{r,q}_{i,j}$ is the cost difference, for element pairs involving one element i with a change of type r and the other element j with a change of type q, for $r, q \in \{1,2,3,4\}$. By definition, $\hat{C}^{r,q}_{i,j} = \hat{C}^{q,r}_{j,i}$, hence from now on we assume that $r \leq q$.

Note that this objective defines a quadratically constrained quadratic program (QCQP) which we can transform into an integer linear program by defining the variables $w_{i,j}$ for every $i < j \in [n]$ to reflect $x_i \cdot x_j$ and adding the constrains: $x_i + x_j - 1 \leq w_{i,j} \leq x_i, x_j$.

Jaccard Label Fix. Let $J^t_{i,j}$ be the Jaccard coefficient of elements v_i and v_j at iteration t, with $|\mathcal{L}^t(v_i) \cap \mathcal{L}^t(v_j)| = n_{i,j}$ and $|\mathcal{L}^t(v_i) \cup \mathcal{L}^t(v_j)| = d_{i,j}$ (i.e., $J^t_{i,j} = \frac{n_{i,j}}{d_{i,j}}$).

We define $J^{r,q}_{i,j}$ as the value of $J^{t+1}_{i,j}$ between two elements i and j, the first with a change of type r and the second with a change of type q, for $r, q \in \{1,2,3,4\}$, and $\hat{J}^{r,q}_{i,j}$ would be the difference between previous Jaccard value and the new one according to these changes. Moreover we let $z^{r,q}$ be an indicator for whether the Jaccard value was decreased or increased.

$$z^{r,q} = \begin{cases} 1 & \text{if } J^t_{i,j} > J^{t+1}_{i,j} \\ 0 & \text{o.w} \end{cases} \tag{7}$$

The change in coefficient $\hat{J}^{r,q}_{i,j}$ as a function of all possible changes r, q appears in Table 1, omitting cases where there is no change. These values are applied to the cost difference formula in (9).

In order to compute the cost change $\hat{C}^{r,q}_{i,j}$ according to $\hat{J}^{r,q}_{i,j}$, we need to take into consideration whether $J^t_{i,j} \geq s(i,j)$ and whether $C^t_{i,j} \geq |\hat{J}^{r,q}_{i,j}|$, thus we define the following indicators for every $i, j \in [n]$ and $r, q \in \{1,2,3,4\}$:

$$g_{i,j} = \begin{cases} 1 & \text{if } J^t_{i,j} \geq s(i,j) \\ 0 & \text{o.w} \end{cases} \qquad k^{r,q}_{i,j} = \begin{cases} 1 & \text{if } C^t_{i,j} \geq |\hat{J}^{r,q}_{i,j}| \\ 0 & \text{o.w} \end{cases} \tag{8}$$

Claim. The cost difference $\hat{C}^{r,q}_{i,j}$ for changes of type r and q in elements v_i and v_j, respectively under the use of Jaccard similarity is:

$$\begin{aligned} \hat{C}^{r,q}_{i,j} = &(g_{i,j} \cdot z^{r,q} + (1-g_{i,j})\cdot(1-z^{r,q}))\cdot\left(|\hat{J}^{r,q}_{i,j}| \cdot k^{r,q}_{i,j} + (2C^t_{i,j} - |\hat{J}^{r,q}_{i,j}|)\cdot(1-k^{r,q}_{i,j})\right) \\ &+((1-g_{i,j})\cdot z^{r,q} + g_{i,j}\cdot(1-z^{r,q}))\cdot\left(-|\hat{J}^{r,q}_{i,j}|\right) \end{aligned} \tag{9}$$

Table 1. Jaccard difference $\hat{J}^{r,q}_{i,j}$ according to type of changes. On time t the Jaccard coefficient is $J^t_{i,j} = \frac{n_{i,j}}{d_{i,j}}$, and on time $t+1$ it has the value $J^{r,q}_{i,j}$.

r	q	$J^{r,q}_{i,j}$	$\hat{J}^{r,q}_{i,j}$	$z^{r,q}$
1	1	$\frac{n_{i,j}+1}{d_{i,j}+1}$	$\begin{cases} \frac{n_{i,j}-d_{i,j}}{d_{i,j}\cdot(d_{i,j}+1)} & \text{if } d_{i,j}>0 \\ -1 & \text{if } d_{i,j}=0 \end{cases}$	0
1	2	$\frac{n_{i,j}}{d_{i,j}+1}$	$\begin{cases} \frac{n_{i,j}}{d_{i,j}\cdot(d_{i,j}+1)} & \text{if } d_{i,j}>0 \\ 1 & \text{if } d_{i,j}=0 \end{cases}$	1
1	4	$\frac{n_{i,j}+1}{d_{i,j}}$	$\frac{-1}{d_{i,j}}$ $(d_{i,j}>0)$	0
2	3	$\frac{n_{i,j}}{d_{i,j}-1}$	$\begin{cases} \frac{-n_{i,j}}{d_{i,j}\cdot(d_{i,j}-1)} & \text{if } d_{i,j}>1 \\ -1 & \text{if } d_{i,j}=1 \end{cases}$	0
3	3	$\frac{n_{i,j}-1}{d_{i,j}-1}$	$\begin{cases} \frac{d_{i,j}-n_{i,j}}{d_{i,j}\cdot(d_{i,j}-1)} & \text{if } d_{i,j}>1 \\ 1 & \text{if } d_{i,j}=1 \end{cases}$	1
3	4	$\frac{n_{i,j}-1}{d_{i,j}}$	$\frac{1}{d_{i,j}}$ $(d_{i,j}>0)$	1

Proof. – If $g_{i,j}=1$ it means that at iteration t the Jaccard between the two elements exceeded their similarity, therefore: $C^t_{i,j} = J^t_{i,j} - s(i,j)$ and $C^{t+1}_{i,j} = |J^t_{i,j} - \hat{J}^{r,q}_{i,j} - s(i,j)| = |C^t_{i,j} - \hat{J}^{r,q}_{i,j}|$.

- If $z^{r,q}=1$ then $\hat{J}^{r,q}_{i,j} = |\hat{J}^{r,q}_{i,j}|$,
  - If $k^{r,q}_{i,j}=1$ then $C^t_{i,j} \geq |\hat{J}^{r,q}_{i,j}| = \hat{J}^{r,q}_{i,j}$, so: $\hat{C}^{r,q}_{i,j} = C^t_{i,j} - (C^t_{i,j} - \hat{J}^{r,q}_{i,j}) = \hat{J}^{r,q}_{i,j} = |\hat{J}^{r,q}_{i,j}|$.
  - If $k^{r,q}_{i,j}=0$ then $C^t_{i,j} < |\hat{J}^{r,q}_{i,j}| = \hat{J}^{r,q}_{i,j}$, so: $\hat{C}^{r,q}_{i,j} = C^t_{i,j} - (\hat{J}^{r,q}_{i,j} - C^t_{i,j}) = 2C^t_{i,j} - \hat{J}^{r,q}_{i,j} = 2C^t_{i,j} - |\hat{J}^{r,q}_{i,j}|$.
- If $z^{r,q}=0$ then $\hat{J}^{r,q}_{i,j} = -|\hat{J}^{r,q}_{i,j}|$, and we get that the cost at time $t+1$ is $C^{t+1}_{i,j} = |C^t_{i,j} + |\hat{J}^{r,q}_{i,j}|| = C^t_{i,j} + |\hat{J}^{r,q}_{i,j}|$, which means that $\hat{C}^{r,q}_{i,j} = -|\hat{J}^{r,q}_{i,j}|$.

– Conversely, if $g_{i,j}=0$ then $C^t_{i,j} = s(i,j) - J^t_{i,j}$ and $C^{t+1}_{i,j} = |J^t_{i,j} - \hat{J}^{r,q}_{i,j} - s(i,j)| = |-C^t_{i,j} - \hat{J}^{r,q}_{i,j}|$.

- If $z^{r,q}=1$ then $\hat{J}^{r,q}_{i,j} = |\hat{J}^{r,q}_{i,j}|$, and we get that the cost at time $t+1$ is $C^{t+1}_{i,j} = |-C^t_{i,j} - |\hat{J}^{r,q}_{i,j}|| = C^t_{i,j} + |\hat{J}^{r,q}_{i,j}|$, which means that $\hat{C}^{r,q}_{i,j} = -|\hat{J}^{r,q}_{i,j}|$.
- If $z^{r,q}=0$ then $\hat{J}^{r,q}_{i,j} = -|\hat{J}^{r,q}_{i,j}|$, and we get that the cost at time $t+1$ is $C^{t+1}_{i,j} = ||\hat{J}^{r,q}_{i,j}| - C^t_{i,j}|$,
  - If $k^{r,q}_{i,j}=1$ then $C^t_{i,j} \geq |\hat{J}^{r,q}_{i,j}|$, so: $\hat{C}^{r,q}_{i,j} = C^t_{i,j} - (C^t_{i,j} - |\hat{J}^{r,q}_{i,j}|) = |\hat{J}^{r,q}_{i,j}|$
  - If $k^{r,q}_{i,j}=0$ then $C^t_{i,j} < |\hat{J}^{r,q}_{i,j}|$, so: $\hat{C}^{r,q}_{i,j} = C^t_{i,j} - (|\hat{J}^{r,q}_{i,j}| - C^t_{i,j}) = 2C^t_{i,j} - |\hat{J}^{r,q}_{i,j}|$

Combining together all these claims we can get the mentioned formula for $\hat{C}^{r,q}_{i,j}$. □

Set-Intersection Label Fix. Similarly to the Jaccard case, we develop a formulation for the set-intersection case. If two elements v_i and v_j share at least

one common label then their cost is $1 - s(v_i, v_j)$, and $s(v_i, v_j)$ otherwise. Thus the cost change for every pair of elements should be $\pm(1 - 2s(v_i, v_j))$ or zero, while the only interesting cases are when elements which share a single label do not share it anymore or when elements which do not have any common label now share the label l, which means when $|\mathcal{L}^t(v_i) \cap \mathcal{L}^t(v_j)|$ changes from one to zero or the other way around. To capture the first case we define the indicators $b_{i,j}$ for every $i, j \in [n]$ as follows:

$$b_{i,j} = \begin{cases} 1 & \text{if } |\mathcal{L}^t(v_i) \cap \mathcal{L}^t(v_j)| = 1 \\ 0 & \text{o.w} \end{cases} \tag{10}$$

Claim. The cost difference $\hat{C}^{r,q}_{i,j}$ for changes of type r and q in elements v_i and v_j respectively under the use of set-intersection similarity is:

$$\hat{C}^{r,q}_{i,j} = \begin{cases} (1 - I^t_{i,j}) \cdot (2\, s(v_i, v_j) - 1) & \text{if } r = 1, q \in \{1, 4\} \\ b_{i,j} \cdot (1 - 2\, s(v_i, v_j)) & \text{if } r = 3, q \in \{3, 4\} \\ 0 & \text{o.w} \end{cases} \tag{11}$$

Proof. As mentioned above we have only two major cases in which the cost by a single pair of elements may change:

- Two elements do not share any label at time t (means $I^t_{i,j} = 0$), but at time $t+1$ start to share the label l only, either by both being newly assigned to it ($r = q = 1$) or when one of them joined it at time $t+1$ ($r = 1$) and the other one was already assigned to it ($q = 4$). In this case their mutual cost changes from $s(v_i, v_j)$ to $1 - s(v_i, v_j)$ and so the difference is $2s(v_i, v_j) - 1$.
- Two elements are both assigned to label l at time t ($q \in \{3, 4\}$) and do not share any other label (means $b_{i,j} = 1$), but one of them is omitted from label l at time $t+1$ ($r = 3$). In this case their mutual cost changes from $1 - s(v_i, v_j)$ to $s(v_i, v_j)$ and so the difference is $1 - 2s(v_i, v_j)$.

□

In order to support the bound p on the number of clusters an element can be assigned to, we should let $x_i = 0$ if $e_i = 0$ and $|\mathcal{L}^t(v_i)| = p$, which means that no change should be performed regarding label l if this element is not currently part of this label and is already assigned to other p labels. Moreover we may validate if needed that every element is assigned to at least one label by letting $x_i = 0$ if $e_i = 1$ and $|\mathcal{L}^t(v_i)| = 1$.

We initialize the labeling using a randomized assignment which satisfies the condition of maximum p labels for each node (and $p \geq 1$ if required). We also test a variant of our algorithm which uses Bonchi's solution for initialization.

3.4 Performance Evaluation

The algorithms are foremost evaluated based on the total cost achieved, normalized by the size of the similarity matrix (number of elements pairs). In addition

we use measures from [9] that account for overlaps between the computed clusters. The first measures are Sensitivity (*Sen*) and Positive predictive value (*PPV*) whose geometric average is the accuracy (*Acc*) measure. In order to define *Sen* and *PPV* we let $T_{i,j}$ be the number of elements which are present both in a true cluster N_i (out of the n clusters induced by the true labeling mapping $\mathcal{L}^*$) and in a suggested cluster M_j (out of m clusters induced by the computed labeling mapping $\mathcal{L}$). Then *Sen* and *PPV* are defined as follows:

$$Sen = \frac{\sum_{i=1}^{n} \max_{j=1}^{m} T_{i,j}}{\sum_{i=1}^{n} N_i} \quad PPV = \frac{\sum_{j=1}^{m} \max_{i=1}^{n} T_{i,j}}{\sum_{j=1}^{m} \sum_{i=1}^{n} T_{i,j}} \tag{12}$$

Sen reflects the weighted average coverage of the predicted labels by their best-matching true labels, and *PPV* reflects the weighted average reliability for the true labels to predict that an element belongs to their best-matching predicted labels [4]. The accuracy is balancing these two measures [17] and therefore explicitly penalizes predicted labels that do not match any of the true ones [13]. It is important to note that the value of *PPV* is relatively low in overlapping labels settings, and so would not be a good enough measure alone for checking the resulting clustering quality. Therefore we use also the maximum matching ratio (*MMR*) and *Fraction* measures which together may overcome this difficulty.

$$MMR = \frac{\sum_{i=1}^{n} \max_{j=1}^{m} O(N_i, M_j)}{n} \tag{13}$$

$$Fraction = \frac{|\{i | i \in [n], \exists j \in [m], O(N_i, M_j) \geq \omega\}|}{n} \tag{14}$$

We may think of the *MMR* as finding the maximum weighted matching in a bipartite graph between the labels of $\mathcal{L}$ and the labels of $\mathcal{L}^*$ according the the overlap-score defined between two labels as $O(N_i, M_j) = \frac{|N_i \cap M_j|^2}{|N_i| \cdot |M_j|}$. As for the *Fraction*, it allows us to count the number of true labels which are highly overlaps with the predicted labels, given an overlap threshold ω which we set to 0.25 [17]. The overall performance is measured by summing the three measures of *MMR*, *Acc* and *Fraction*, which will be referred as the *composite score*.

4 Results

We examine the following algorithms:

- *ROW-BON* - vertex-by-vertex method with Bonchi's local steps.
- *ROW-FPT* - vertex-by-vertex method with naive FPT local steps.
- *FULL-ILP* - ILP for the entire *OCC* problem (either *ECLIP* or *ISEC-ILP*)
- *COL-ILP* - our proposed algorithm using a label-by-label method with Bonchi or random initialization.

For the assessment of our methods we use two different multi-label datasets from MULAN [16] which are most commonly used for training multi-labeling

classifiers: EMOTIONS ($n = 593, k = 6, p = 3$) [15], and YEAST ($n = 2417, k = 14, p = 11$) [7]. In order to create the similarity matrices for a dataset we used Jaccard coefficient and set-intersection indicator between the given true labels.

All four algorithms where implemented in Python3.9 and run on a 3.2 GHz CPU with their relevant parameters k and p over 100 iterations. We apply all algorithms to subsets of different sizes from 10 to 100. The subsets were randomly chosen so as to preserve the relative sizes of the different clusters - we first select a cluster l with probability p_l which is the fraction of nodes assigned to the cluster from the total number of node assignments to clusters, and then choose uniformly an element l, repeating those steps until the desired number of elements is obtained. We calculated the mean value of every measure across all 100 iterations, omitting some of the results which ran for more than 1000 s.

Figure 1 shows that our new methods balance between the fast heuristic of *ROW-BON* to the slow fully exact method of *ECLIP* or *ISEC-ILP*. Interestingly, using Bonchi's labeling as an initialization for *COL-ILP* leads to a decrease in runtime. On the larger YEAST dataset, ROW-FPT and ECLIP are infeasible already for small subsets.

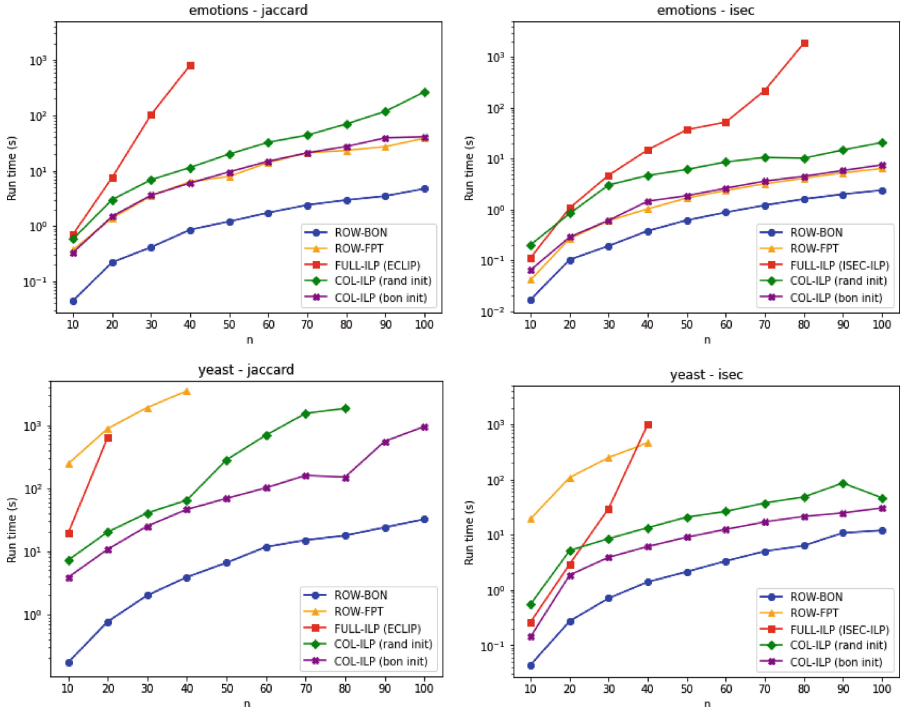

Fig. 1. The mean running time on EMOTIONS (top) and YEAST (bottom) subsets of different sizes. **left**: the Jaccard variant, **right**: the set-intersection variant.

Next, we assess the quality of the different solutions by computing the average edge loss which is defined as the total cost divided by n^2. As evident from Fig. 2, the exact methods of *ECLIP* and *ISEC-ILP* achieve zero cost, yet are too expensive to compute in part of the range. In contrast, *COL-ILP* is feasible across the range and outperforms the other heuristic searches in the vast majority of the cases, with the Bonchi initialization yielding smaller loss compared to the random one.

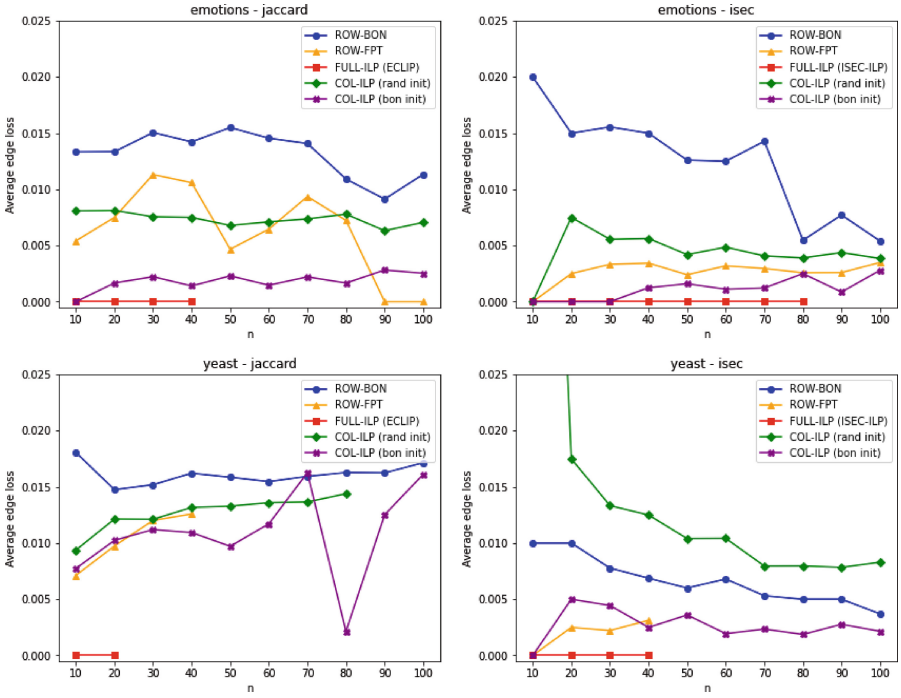

Fig. 2. The mean average edge loss on EMOTIONS (top) and YEAST (bottom) subsets of different sizes. **left**: the Jaccard variant, **right**: the set-intersection variant.

Last, we compare the composite score of the different solutions (Fig. 3). Again we observe that *COL-ILP* outperforms the other heuristic methods in most applications.

To get more insights into the composite score comparison, we present in Fig. 4 how the detailed parts combine together to yield the composite score for subsets of 20 elements. For EMOTIONS, the advantage of the *COL-ILP* methods is derived from larger *MMR* values, while for YEAST, the *Fraction* is a dominant factor.

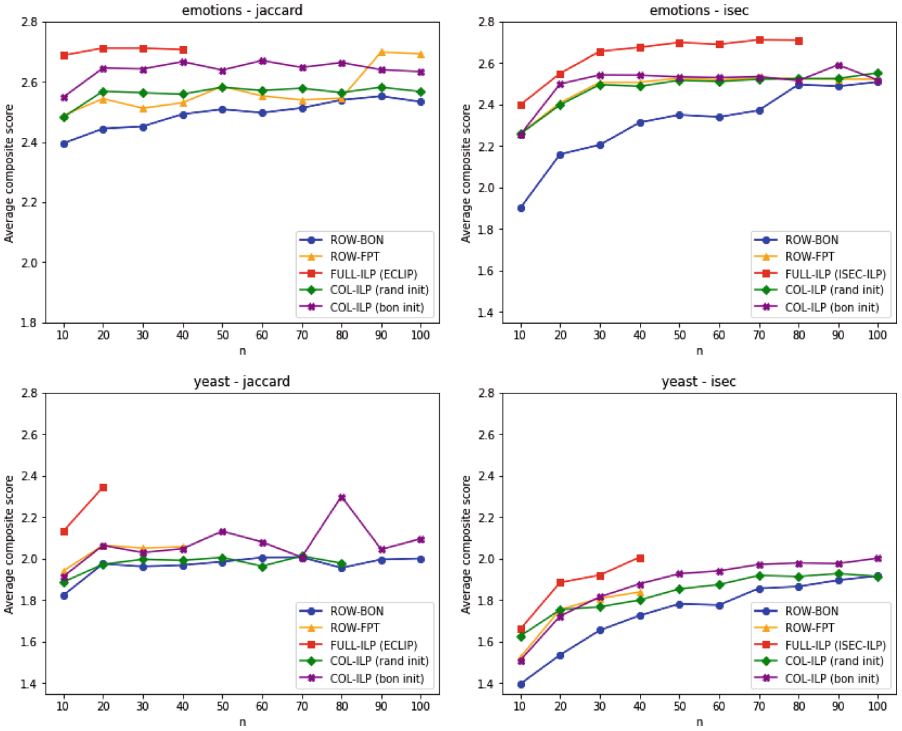

Fig. 3. The mean composite score on EMOTIONS (top) and YEAST (bottom) subsets of different sizes. **left**: the Jaccard variant, **right**: the set-intersection variant.

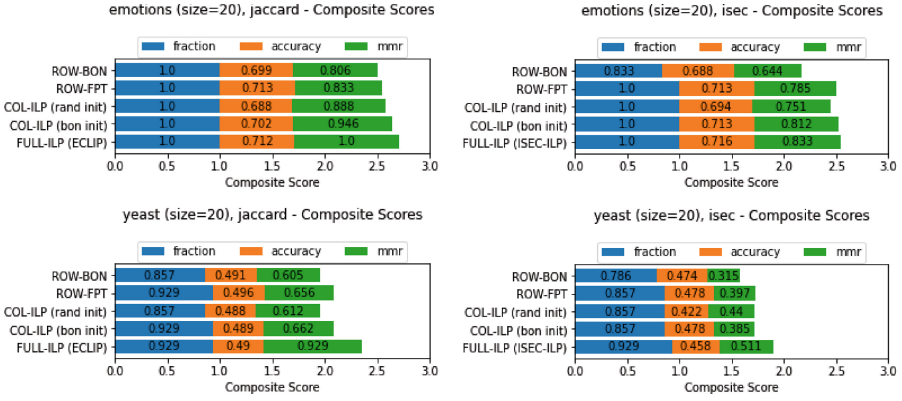

Fig. 4. The mean composite scores on EMOTIONS (top) and YEAST (bottom) with 20 elements. **left**: the Jaccard variant, **right**: the set-intersection variant.

5 Conclusions and Future Work

We have presented novel approaches for OCC that combine greedy iterations with exact ILP-based solutions for each iteration. Our *COL-ILP* method was shown to be both practical and effective, outperforming other algorithms on two datasets in an array of measures. Future work will be to generalize our methods to larger datasets and evaluate them on real data where the no true solution is available.

Acknowledgements. RS was supported by a research grant from the Israel Science Foundation (grant no. 715/18).

References

1. de Andrade, C.E., Resende, M.G.C., Karloff, H.J., Miyazawa, F.K.: Evolutionary algorithms for overlapping correlation clustering. In: Arnold, D.V. (ed.) Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, 12–16 July 2014, pp. 405–412. ACM (2014). https://doi.org/10.1145/2576768.2598284
2. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Mach. Learn. **56**(1–3), 89–113 (2004). https://doi.org/10.1023/B:MACH.0000033116.57574.95
3. Bonchi, F., Gionis, A., Ukkonen, A.: Overlapping correlation clustering. Knowl. Inf. Syst. **35**(1), 1–32 (2013). https://doi.org/10.1007/s10115-012-0522-9
4. Brohée, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinform. **7**, 488 (2006). https://doi.org/10.1186/1471-2105-7-488
5. Chierichetti, F., Kumar, R., Pandey, S., Vassilvitskii, S.: Finding the Jaccard median. In: Charikar, M. (ed.) Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, 17–19 January 2010, pp. 293–311. SIAM (2010). https://doi.org/10.1137/1.9781611973075.25
6. Demaine, E.D., Emanuel, D., Fiat, A., Immorlica, N.: Correlation clustering in general weighted graphs. Theor. Comput. Sci. **361**(2–3), 172–187 (2006). https://doi.org/10.1016/j.tcs.2006.05.008
7. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001(December), pp. 3–8, 2001. Vancouver, British Columbia, Canada], pp. 681–687. MIT Press (2001). https://proceedings.neurips.cc/paper/2001/hash/39dcaf7a053dc372fbc391d4e6b5d693-Abstract.html
8. Gartzman, D., Sharan, R.: Exact algorithms for overlapping correlation clustering. Master's thesis, Tel Aviv University (2016)
9. Hu, L.Z., et al.: Epic: software toolkit for elution profile-based inference of protein complexes. Nat. Methods **16**(8), 737–742 (2019). https://doi.org/10.1038/s41592-019-0461-4
10. Johansson, F.D., Chattoraj, A., Bhattacharyya, C., Dubhashi, D.P.: Weighted theta functions and embeddings with applications to max-cut, clustering and summarization. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett,

R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015(December), pp. 7–12, 2015. Montreal, Quebec, Canada, pp. 1018–1026 (2015). https://proceedings.neurips.cc/paper/2015/hash/4c27cea8526af8cfee3be5e183ac9605-Abstract.html
11. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: In 5-th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
12. Miettinen, P.: On the positive-negative partial set cover problem. Inf. Process. Lett. **108**(4), 219–221 (2008). https://doi.org/10.1016/j.ipl.2008.05.007
13. Nepusz, T., Yu, H., Paccanaro, A.: Detecting overlapping protein complexes in protein-protein interaction networks. Nat. Methods **9**(5), 471–472 (2012). https://doi.org/10.1038/nmeth.1938
14. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. In: Kucera, L. (ed.) Graph-Theoretic Concepts in Computer Science, 28th International Workshop, WG 2002, Cesky Krumlov, Czech Republic, 13–15 June 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2573, pp. 379–390. Springer (2002). https://doi.org/10.1007/3-540-36379-3_33
15. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.P.: Multi-label classification of music into emotions. In: Bello, J.P., Chew, E., Turnbull, D. (eds.) ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, 14–18 September 2008, pp. 325–330 (2008). http://ismir2008.ismir.net/papers/ISMIR2008_275.pdf
16. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-09823-4_34
17. Wang, R., Liu, G., Wang, C., Su, L., Sun, L.: Predicting overlapping protein complexes based on core-attachment and a local modularity structure. BMC Bioinform. **19**(1), 1–15 (2018). https://doi.org/10.1186/s12859-018-2309-9