



Clustering on k -edge-colored graphs[☆]



E. Angel^a, E. Bampis^{b,c,*}, A. Kononov^d, D. Paparas^e, E. Pountourakis^f,
V. Zissimopoulos^g

^a IBISC, Université d'Evry, France

^b Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France

^c CNRS, UMR 7606, LIP6, F-75005, Paris, France

^d Sobolev Institute of Mathematics, Novosibirsk, Russia

^e Computer Science Department, Columbia University, USA

^f EECS Department, Northwestern University, USA

^g Department of Informatics & Telecommunications, National and Kapodistrian University of Athens, Greece

ARTICLE INFO

Article history:

Received 23 February 2014

Received in revised form 25 March 2016

Accepted 18 April 2016

Available online 12 May 2016

Keywords:

Clustering

Approximation algorithms

Coloring

ABSTRACT

We study the Max k -colored clustering problem, where given an edge-colored graph with k colors, we seek to color the vertices of the graph so as to find a clustering of the vertices maximizing the number (or the weight) of matched edges, i.e. the edges having the same color as their extremities. We show that the cardinality problem is NP-hard even for edge-colored bipartite graphs with a chromatic degree equal to two and $k \geq 3$. Our main result is a constant approximation algorithm for the weighted version of the Max k -colored clustering problem which is based on a rounding of a natural linear programming relaxation. For graphs with chromatic degree equal to two we improve this ratio by exploiting the relation of our problem with the MAX 2-AND problem. We also present a reduction to the maximum-weight independent set (IS) problem in bipartite graphs which leads to a polynomial time algorithm for the case of two colors.

© 2016 Published by Elsevier B.V.

1. Introduction

We consider the following problem: we are given an edge-colored graph $G = (V, E)$, where every edge e is labeled with one color among $\{1, 2, \dots, k\}$ and it is associated with a weight w_e . We are interested in coloring every vertex of the graph with one of the k available colors so as to create at most k clusters. Each cluster corresponds to the subgraph induced by the vertices colored with the same color. Given a coloring of the vertices, an edge is called *matched* if its color is the same as the color of both its extremities. Our goal is to find a clustering of the vertices maximizing the total weight of the matched edges of the graph. We call this problem the *Max k -colored clustering problem* and we denote it as MAX- k -CC.

Our model has similarities with the centralized version of the information-sharing model introduced by Kleinberg and Ligett [3,7]. In their model, the edges are not colored and two adjacent nodes share information only if they are colored with the same color. As they mention, one interesting extension of their model would be the incorporation of different categories of information. The use of colors in our model goes in this direction. Every edge-color corresponds to a different information

[☆] An extended abstract of this work has been presented in MFCS 2013.

* Corresponding author at: Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France.

E-mail addresses: Eric.Angel@ibisc.fr (E. Angel), Evripidis.Bampis@lip6.fr (E. Bampis), alvenko@math.nsc.ru (A. Kononov), paparas@cs.columbia.edu (D. Paparas), Emmanouil.Pountourakis@eecs.northwestern.edu (E. Pountourakis), vassilis@di.uoa.gr (V. Zissimopoulos).

category and two adjacent vertices share information if their color is the same as the color of the edge that connects them. While the centralized version of the information-sharing problem of Kleinberg and Ligett is easy to solve, we show that the introduction of colors in the edges of the graph renders the problem NP-hard. In this paper, we focus on the centralized variant of our problem and we study its approximability. Studying our problem from a game theoretic point of view would be an interesting direction for future work. Our problem is also related to the classical correlation clustering problem [2,6].

1.1. Related works and our contribution

In Section 2, we formulate the problem as an integer linear program and we propose a $1/e^2$ -approximation ratio algorithm which is based on a randomized rounding of its linear programming relaxation.¹ Our problem can be formulated as the combinatorial allocation problem studied by Feige and Vondrák in [4]. We may consider each color as a player and each vertex as an item. The items have to be allocated to competing players by a central authority with the goal of maximizing the total utility provided to the players. Every player (color) has utility functions derived from the different subsets of vertices. In their work, Feige and Vondrák considered subadditive, fractional subadditive and submodular functions [4]. It is easy to see that in our case the function is supermodular and hence, their method cannot be directly applied. At the end of Section 2, we show that in the special case where the chromatic degree² of the graph is equal to two, our problem is a special case of the MAX 2-AND problem [9]. We show in Section 3 that the cardinality Max k -colored problem is strongly NP-hard by a reduction from MAX-2-SAT, even for bipartite graphs with chromatic degree equal to two, whenever the number of colors is any constant number $k \geq 3$. In Section 4, we present a reduction to the maximum-weight independent set (IS) problem in bipartite graphs which allows us to get an optimal polynomial-time algorithm for the case of two colors. Furthermore, we exploit this idea to get a $\frac{2}{k}$ -approximation algorithm whose approximation ratio is better than the ratio of the constant-approximation algorithm presented in Section 3 for any $k \leq 14$.

2. A constant approximation algorithm

As the problem is strongly NP-hard (see Section 3), in the first part of this section, we present a constant-factor approximation algorithm for our problem, while in the second part we focus on graphs with chromatic degree equal to two.

For every vertex i of the graph and for every available color c , we introduce a variable x_{ic} which is equal to one if i is colored with color c and zero otherwise. Also, for every edge $e = [i, j]$, we introduce a variable z_{ij} which is equal to one if both extremities are colored with the same color as e , and zero otherwise. We obtain the following ILP:

$$\begin{aligned} & \max \sum_e w_e z_e \\ & z_e \leq x_{ic}, \quad \forall c\text{-colored edge } e \text{ and } i \in e \\ & \sum_c x_{ic} = 1, \quad \forall i \\ & x_{ic}, z_e \in \{0, 1\}, \quad \forall i, c, e. \end{aligned}$$

By relaxing the integrality constraints, we get the linear relaxation of this ILP that we denote by LP.

Our algorithm works in k iterations by considering each color c , $1 \leq c \leq k$, independently from the others, and so the order in which the colors are considered does not matter. By $l(c)$, we denote the number of c -colored edges in the graph. When an edge is chosen this means that its two extremities get the color of this edge. Since in general a vertex is adjacent to edges of different colors, a vertex may get more than one color. We want to avoid such situations and the way the algorithm assigns colors to vertices is designed to minimize the number of such conflicts.

The algorithm is given below.

Algorithm RR

Phase I:

Solve the linear program LP, and let z_e^* be the values of variables z_e .

For each color c

Order non decreasingly the c -colored edges $e_1, \dots, e_{l(c)}$ according to their z_e^* values.

Let us assume that we have $z_{e_1}^* \leq z_{e_2}^* \leq \dots \leq z_{e_{l(c)}}^*$.

Let r be a random value in $[0, 1]$.

Choose edges e with $z_e^* > r$.

End For

¹ Recently, Ageev and Kononov [1] improved the analysis of our algorithm and they showed that our algorithm is in fact a 0.25 -approximation algorithm. They have also proposed a new algorithm with a better approximation ratio of $7/23$.

² We define the *chromatic degree* of a vertex as the number of different colors which appear in its incident edges. The *chromatic degree* of an edge-colored graph is the maximum chromatic degree over all its vertices.

Phase II:

For each vertex v

If v gets no color or more than two colors, remove it (together with all its adjacent edges) from graph G .

End For

Let G' be the obtained graph.

For each vertex v in G'

If v got one color, then assign this color to it.

If v got two colors, then choose randomly one of them, each one with probability $1/2$.

End For

Notice that the algorithm does not assign colors to all vertices. Indeed, at the end of Phase I some vertex may get no colors and then in Phase II the algorithm assigns colors only for a subgraph G' of the initial graph G .

The following two Lemmas are straightforward to prove.

Lemma 1. For any edge e , the probability that e is chosen is z_e^* .

Notice that for a vertex v , it may be the case that none of its adjacent edges are chosen. In that case, v gets no color. But in general, several of its adjacent edges can be chosen, and the vertex v can get more than one colors. We denote by X_{vc} (resp. \bar{X}_{vc}) the event that v gets (resp. does not get) color c .

Lemma 2. For any vertex v , if there exists at least one c -colored edge incident to v , we have $Pr(X_{vc}) = z_{e'}^*$ where e' is the edge with the maximal value of z_e^* among all its incident c -colored edges.

Lemma 3. For any vertex v , one has $\sum_c Pr(X_{vc}) \leq 1$.

Proof. For any color c , let $e(c)$ be the c -colored edge which is incident to vertex v (if such an edge exists), and with the maximal value of z_e^* among all such edges. From Lemma 2, one has $Pr(X_{vc}) = z_{e(c)}^*$. Therefore, $\sum_c Pr(X_{vc}) \leq \sum_c z_{e(c)}^* \leq \sum_c X_{vc}^* = 1$. \square

As stated before, a vertex v can get more than one color during the execution of the algorithm. However, in general this number will be small. We have the following lemmas.

Lemma 4. Given a set of independent events such that the sum of their probabilities is less than or equal to 1, the probability of getting at most one of them is greater or equal to $2/e$.

Proof. We denote by X_i (for $1 \leq i \leq l$) the events. The probability of getting at most one of them is $p = \sum_{i=1}^l Pr(X_i) \prod_{i' \neq i} (1 - Pr(X_{i'})) + \prod_{i=1}^l (1 - Pr(X_i))$.

We are going to show that the minimum of this expression is obtained when each X value is equal to $1/l$. To simplify the presentation we denote by X_i , $i = 1, 2, \dots, l$, both the event X_i and its probability. Notice that we can assume that $\sum_{i=1}^l X_i = 1$ (If this is not the case, we can increase the value of one X_i in order to obtain the equality, the probability of getting at most one event will decrease, and if we show that it is at least $2/e$ it shows that it is also at least $2/e$ for the original value X_i).

Also, without loss of generality, we rename the variables X_i such that X_1 has the biggest value and X_2 has the smallest value. If $X_1 = X_2$ then all the values are equal to $1/l$. Let us now assume that $X_1 > X_2$. In that case we have:

$$\begin{aligned}
 p &= \sum_{i=1}^l X_i \prod_{j \neq i} (1 - X_j) + \prod_{j=1}^l (1 - X_j) \\
 &= X_1(1 - X_2) \prod_{j \geq 3} (1 - X_j) + X_2(1 - X_1) \prod_{j \geq 3} (1 - X_j) \\
 &\quad + (1 - X_1)(1 - X_2) \sum_{i \geq 3} X_i \prod_{\substack{j \neq i \\ j \geq 3}} (1 - X_j) + (1 - X_1)(1 - X_2) \prod_{j \geq 3} (1 - X_j) \\
 &= (1 - X_1 X_2) \underbrace{\prod_{j \geq 3} (1 - X_j)}_A + (1 - X_1)(1 - X_2) \underbrace{\sum_{i \geq 3} X_i \prod_{\substack{j \neq i \\ j \geq 3}} (1 - X_j)}_B \\
 &= (1 - X_1 X_2)A + (1 - X_1)(1 - X_2)B \\
 &= A + B + (B - A)X_1 X_2 - BX_1 - BX_2.
 \end{aligned}$$

Let $g(\Delta) = A + B + (B - A)(X_1 - \Delta)(X_2 + \Delta) - B(X_1 - \Delta) - B(X_2 + \Delta)$ for some small enough Δ . Next, we claim that this expression has a minimum when $X_1 - \Delta = X_2 + \Delta$.

One has

$$g(\Delta) = A + B + (B - A)(X_1X_2 - \Delta X_2 + \Delta X_1 - \Delta^2) - B(X_1 + X_2) \\ = (-\Delta X_2 + \Delta X_1 - \Delta^2)(B - A) + \text{constant}$$

and thus by getting the derivative, we get $g'(\Delta) = (-X_2 + X_1 - 2\Delta)(B - A)$, and $g'(\Delta) = 0$ for $\Delta = (X_1 - X_2)/2$.

We are going to show that $B - A < 0$.

$$B - A = \sum_{i \geq 3} X_i \prod_{\substack{j \neq i \\ j \geq 3}} (1 - X_j) - \prod_{j \geq 3} (1 - X_j) \\ = \underbrace{\left(\sum_{i \geq 3} \frac{X_i}{1 - X_i} - 1 \right)}_{<0} \underbrace{\prod_{j \geq 3} (1 - X_j)}_{>0}.$$

One has

$$\sum_{i \geq 3} \frac{X_i}{1 - X_i} \leq \sum_{i \geq 3} \frac{X_i}{1 - X_1} \leq 1,$$

since $\sum_{i=1}^l X_i = 1$, and this concludes the proof of our claim.

Consequently, the minimum value of p is obtained when every X_i is equal to $1/l$. In that case, the probability is $(\frac{l-1}{l})^{l-1} + (\frac{l-1}{l})^l \geq \frac{2}{e}$ for any $l \geq 2$. \square

Lemma 5. *At any time during the execution of the algorithm, for any vertex v , the probability that v gets at most one additional color until the end of Phase I of the algorithm is greater than or equal to $2/e$.*

Proof. The events: “ v gets color c ” for $1 \leq c \leq k$, are independent, and the sum of their probabilities is less than or equal to 1 according to Lemma 3. Therefore, the result follows from Lemma 4. \square

Proposition 1. *At any time during the execution of the algorithm, consider any edge $e = [u, v]$, and let us denote by p_u (resp. p_v) the probability that u (resp. v) gets at most one additional color until the end of Phase I of the algorithm. Let us denote by $p_{u \wedge v}$ the probability that both u and v get each one at most one additional color until the end of Phase I of the algorithm. Then, one has $p_{u \wedge v} \geq p_u \cdot p_v$.*

Proof. We have to prove that $p_{u \wedge v}$ reaches its smallest value when all vertices receive different colors. This is equivalent to the case where the vertices are colored independently. For the purpose of the proof, let us consider an artificial sequence of algorithms denoted by $\Sigma_0, \dots, \Sigma_k$, with Σ_0 being our algorithm.

The difference among these algorithms comes from the way in which the vertices get a color. Let us fix a color c . We consider two different procedures for assigning colors to the vertices. The *First procedure*, assigns the colors in the same way as our algorithm does. Let us recall how our algorithm works for just two vertices: Without loss of generality, we assume that there exist an edge e' adjacent to u with color c and an edge e'' adjacent to v with color c (in the case where the edge e' does not exist, the probability that u gets color c in the algorithm is equal to 0). Moreover e' (resp. e'') is the edge with the maximal value of $z_{e'}^*$ (resp. $z_{e''}^*$) among all c -colored edges incident to u (resp. v). For convenience, we set $p = z_{e'}^*$ and $q = z_{e''}^*$. Let us assume that $p \leq q$. From the algorithm and Lemma 2, we get that p is the probability that both u and v get color c in the Phase I of the algorithm. In turn $q - p$ is the probability that v gets color c assuming that u does not get color c and the probability that both vertices do not get color c is equal to $(1 - q)$. Using the *First procedure*, we color both vertices u and v (with color c) with probability p , and we color only vertex v with probability $q - p$. The *Second procedure* colors the vertices with color c independently. More precisely, we color vertex u with probability p , and we color vertex v with probability q .

In the algorithm Σ_0 , for each color c , $1 \leq c \leq k$, we use the *First procedure* for assigning colors to vertices. In the algorithm Σ_i , $1 \leq i \leq k$, for colors c such that $1 \leq c \leq i$ (resp. $i + 1 \leq c \leq k$) we use the *Second procedure* (resp. *First procedure*) for assigning those colors to vertices. Thus, in algorithm Σ_k , all colors are assigned to vertices using the *Second procedure*.

Let us fix any iteration (color) t , and let us analyze the behavior of those algorithms from iteration t until the end of their execution (at the end of Phase I), i.e. when colors $t, t + 1, \dots, k$ are assigned to vertices. Let us also consider any edge $e = [u, v]$. We denote by $p_u(\Sigma_i)$ (resp. $p_v(\Sigma_i)$) the probability that u (resp. v) gets at most one additional color from iteration t until the end of iteration k , for the algorithm Σ_i . Moreover, we denote by $p_{u \wedge v}(\Sigma_i)$ the probability that both u and v get at most one additional color each from iteration t until the end of iteration k , for the algorithm Σ_i . Notice that one has for any vertex v , $p_v(\Sigma_i) = p_v(\Sigma_0)$ for $1 \leq i \leq k$. Let us now prove that for any $1 \leq i \leq k - 1$, one has $p_{u \wedge v}(\Sigma_i) \geq p_{u \wedge v}(\Sigma_{i+1})$.

If $t \geq i + 2$, since both algorithms Σ_i and Σ_{i+1} use the *First procedure* to assign colors c to vertices, for $i + 2 \leq c \leq k$, they behave in the same way during iterations $t, t + 1, \dots, k$, and so $p_{u \wedge v}(\Sigma_i) = p_{u \wedge v}(\Sigma_{i+1})$.

We now assume that $1 \leq t \leq i + 1$. Algorithms Σ_i and Σ_{i+1} only differ in the way they assign color $i + 1$ to vertices. If there is no $(i + 1)$ -colored edge adjacent to either u or v , then again those two algorithms have the same behavior from iteration t to k and so $p_{u \wedge v}(\Sigma_i) = p_{u \wedge v}(\Sigma_{i+1})$. Let us assume now w.l.o.g. that there exists at least one $(i + 1)$ -colored edge which is adjacent to u . Recall that we denote by X_{vc} (resp. $\overline{X_{vc}}$) the event that v gets (resp. does not get) color c . We have the following probabilities:

	When $\Sigma = \Sigma_i$	When $\Sigma = \Sigma_{i+1}$
$Pr_{\Sigma}(X_{u,i+1} \wedge \overline{X_{v,i+1}})$	0	$p(1 - q)$
$Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge X_{v,i+1})$	$q - p$	$(1 - p)q$
$Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge \overline{X_{v,i+1}})$	$1 - q$	$(1 - p)(1 - q)$
$Pr_{\Sigma}(X_{u,i+1} \wedge X_{v,i+1})$	p	pq

Let us denote by A_0 (resp. B_0) the event which corresponds to the situation where vertex u (resp. v) gets no additional color when considering iterations (colors) in $\{t, t + 1, \dots, k\} \setminus \{i + 1\}$. Let us also denote by A_1 (resp. B_1) the event which corresponds to the situation where vertex u (resp. v) gets one additional color when considering iterations (colors) in $\{t, t + 1, \dots, k\} \setminus \{i + 1\}$. Since these events do not depend on the color $i + 1$, they have the same probability for algorithms Σ_i and Σ_{i+1} .

For $\Sigma \in \{\Sigma_0, \dots, \Sigma_k\}$, one has $p_{u \wedge v}(\Sigma) = Pr(A_0 \wedge B_0) + Pr(A_1 \wedge B_0) \cdot [Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge X_{v,i+1}) + Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge \overline{X_{v,i+1}})] + Pr(A_0 \wedge B_1) \cdot [Pr_{\Sigma}(X_{u,i+1} \wedge \overline{X_{v,i+1}}) + Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge \overline{X_{v,i+1}})] + Pr(A_1 \wedge B_1) \cdot [Pr_{\Sigma}(X_{u,i+1} \wedge X_{v,i+1})]$.

As stated above, $Pr(A_0 \wedge B_0)$, $Pr(A_1 \wedge B_0)$, $Pr(A_0 \wedge B_1)$, $Pr(A_1 \wedge B_1)$ are the same for Σ_i and Σ_{i+1} . In the following table, we give the remaining terms, with $A = Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge X_{v,i+1}) + Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge \overline{X_{v,i+1}})$, $B = Pr_{\Sigma}(X_{u,i+1} \wedge X_{v,i+1}) + Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge \overline{X_{v,i+1}})$, and $C = Pr_{\Sigma}(\overline{X_{u,i+1}} \wedge \overline{X_{v,i+1}})$.

	$\Sigma = \Sigma_i$	$\Sigma = \Sigma_{i+1}$
A	$(q - p) + (1 - q) = (1 - p)$	$q(1 - p) + (1 - p)(1 - q) = (1 - p)$
B	$(1 - q)$	$p(1 - q) + (1 - p)(1 - q) = (1 - q)$
C	$(1 - q)$	$(1 - p)(1 - q)$

A term-by-term comparison is sufficient for concluding that $p_{u \wedge v}(\Sigma_i) \geq p_{u \wedge v}(\Sigma_{i+1})$.

Since in algorithm Σ_k , all colors are assigned to vertices using the *Second procedure*, i.e. in an independent way, one has $p_{u \wedge v}(\Sigma_k) = p_u(\Sigma_k) \cdot p_v(\Sigma_k)$. Then $p_{u \wedge v} = p_{u \wedge v}(\Sigma_0) \geq p_{u \wedge v}(\Sigma_1) \geq \dots \geq p_{u \wedge v}(\Sigma_k) = p_u(\Sigma_k) \cdot p_v(\Sigma_k) = p_u(\Sigma_0) \cdot p_v(\Sigma_0) = p_u \cdot p_v$. \square

Corollary 1. *At any time during the execution of the algorithm, for any edge $e = [u, v]$, the probability that both u and v get at most one additional color each until the end of the algorithm is greater than or equal to $4/e^2$.*

Proof. It follows directly from Proposition 1 and Lemma 5. \square

Definition. An edge $e = [u, v]$ is *safe* if both extremities u and v are colored with the color of edge e and they get at most one additional color each.

Theorem 1. *The algorithm RR is $1/e^2 \simeq 0.135$ -approximate for MAX- k -CC.*

Proof. Let e be any edge of the graph G . We are going to evaluate the probability that edge e is matched in the solution returned by the algorithm. Let OPT be the sum of weights of the matched edges in an optimal solution. Since the linear program LP is a linear relaxation, we have $\sum_{e \in E} w_e z_e^* \geq OPT$. Since the colors are considered in an independent way by the algorithm, we can assume w.l.o.g. that edge e has color 1. This edge needs to be chosen in the first iteration of the algorithm (i.e. when color 1 is considered). This occurs with the probability z_e^* according to Lemma 1. Then during the remaining iterations until the end of the Phase I of the algorithm, i.e. when colors from 2 to k are considered, this edge must remain safe so that it belongs to graph G' . This occurs with a probability greater than or equal to $4/e^2$ according to Corollary 1. Thus, we have proved that each edge $e = [u, v]$ from G belongs to the graph G' with a probability greater than or equal to $4z_e^*/e^2$. We also know that if $e = [u, v]$ belongs to G' then each of the two vertices u and v got either one color, in this case it is the color of edge e , or two colors, and in this case one of them is the color of edge e . So assuming that e belongs to G' the probability that e is matched is at least $1/4$. Overall, this probability is equal to $4z_e^*/e^2 \times 1/4 = z_e^*/e^2$. Thus the expected cost of the solution returned by the algorithm is at least $\sum_{e \in E} w_e z_e^*/e^2 \geq OPT/e^2$. \square

This algorithm can be derandomized by the method of conditional expectations [8]. To do this, we can represent the execution of the algorithm with a tree T with k levels. Each node on level i represents a partial solution, in which only the i th first colors have been assigned. A path from the root of T to a vertex on level i determines the coloring of edges and vertices

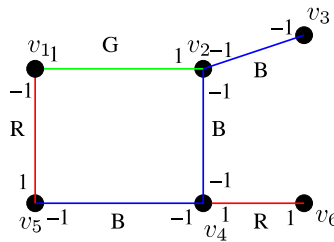


Fig. 1. An example of labeling.

with colors c_1, \dots, c_i . To apply the derandomization of the algorithm, we need to be able to calculate the exact conditional expectation of the number of matched edges using our randomized algorithm, in any node of this tree. This can be done in polynomial time. Then at each level, we follow the next vertex with the highest conditional expectation. We repeat this procedure, and from the property of the conditional expectations after k iteration, we reach a leaf of this tree and we find the desired solution.

The algorithm RR can also be used for the case where there are more than one color on the edges. It is sufficient to create parallel edges, i.e. one edge for each color.

2.1. Graphs with a chromatic degree equal to 2

In this case the Max k -colored clustering problem can be seen as a particular case of the problem MAX 2-AND [9]. Indeed, for each couple (v, e) , with e an edge adjacent to v , we define a label $l(v, e) \in \{-1, 1\}$. This set of labels must satisfy the following condition: For any vertex v , if e and e' are two edges adjacent to v with different colors, then $l(v, e) \neq l(v, e')$. Since we know that in the graph G' , for any vertex v , the set of its adjacent edges are colored with at most 2 colors, it is easy to define such a set of labels (i.e. for all edges e colored with the first (resp. second) color we set $l(v, e) = 1$ (resp. $l(v, e) = -1$). Notice that it is possible to have $l(u, e) \neq l(v, e)$ for an edge $e = [u, v]$. An example is given in Fig. 1.

An instance of MAX 2-AND is composed of a collection of clauses (with non-negative weights assigned to them) such that each clause is either of the form z_i or $z_i \wedge z_j$, where each z_i is either a boolean variable x_k or its negation \bar{x}_k . The goal is to find an assignment of the boolean variables x_1, \dots, x_n , in order to maximize the weight of the satisfied clauses. It is easy to see that any instance of our problem can be transformed to an equivalent MAX 2-AND instance for which an algorithm with an approximation ratio of 0.859 exists [9]. Indeed, for each vertex $v_i \in V$ we define a variable z_i and for each edge $e = [v_i, v_j]$ we define a clause $z_i \wedge z_j$, where z_i is either a boolean variable x_i if $l(v_i, e) = 1$ or its negation \bar{x}_i if $l(v_i, e) = -1$. For example, for the instance given in Fig. 1 we obtain the set of clauses: $x_1 \wedge x_2$ (for edge $[v_1, v_2]$), $\bar{x}_1 \wedge x_5$ (for edge $[v_1, v_5]$), $\bar{x}_5 \wedge \bar{x}_4$ (for edge $[v_5, v_4]$), and so on.

3. Complexity

In this part we show that the problem is NP-complete for bipartite graphs if we allow the initial coloring of the edges to contain three or more colors. Our reduction is from MAX-2-SAT.

Theorem 2. *The MAX-3-CC problem is NP-complete even for bipartite graphs with chromatic degree two and $w_e = 1$, for every edge e of the graph.*

Proof. Clearly the problem is in NP. Let us now give a polynomial time reduction R that maps any instance of the decision version of the MAX-2-SAT problem $\mathcal{I}_{\text{MAX-2-SAT}} = \langle \mathcal{X}, \mathcal{C}, B \rangle$ where $\mathcal{X} = \{x_1, \dots, x_n\}$ is a set of variables, $\mathcal{C} = \{c_1, \dots, c_m\}$ is a set of disjunctive clauses with exactly two literals, and $B \leq m$ is a positive integer, to an instance $R(\mathcal{I}_{\text{MAX-2-SAT}}) = \mathcal{I}_{\text{MAX-3-CC}} = \langle G, C, f, 3 \rangle$ for the MAX-3-CC problem. For the rest of the proof we assume that $C = \{R(\text{ed}), B(\text{lue}), G(\text{reen})\}$. The reduction is based on the gadgets presented in Figs. 2 and 3.

The gadgets. The set V is constructed as follows: For each variable x_i of the MAX-2-SAT formula we create a new node v_i and for each $c \in C$ we construct four nodes $v_{\text{up}}^c, v_{\text{down}}^c, v_{\text{left}}^c$ and v_{right}^c . Then for each clause, we add six edges based on whether both of the literals are positive or negative, or one of them is negative and the other positive.

First case: Assume that both of the literals are positive or both are negative i.e. the clause is either $c_k = (x_\alpha \vee x_\beta)$ or $c_k = (\bar{x}_\alpha \vee \bar{x}_\beta)$. Then we construct the gadgets in Fig. 2.

Second case: Assume that one literal is positive and the other is negative. That is, the clause is of the form $c_k = (x_\alpha \vee \bar{x}_\beta)$ or $c_k = (\bar{x}_\alpha \vee x_\beta)$. Respectively we construct the gadgets in Fig. 3.

Finally we set $P = 3B + 2(m - B)$, where m is the total number of clauses.

Example. For example, for the formula $(\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$ the graph that we construct is pictured in Fig. 4.

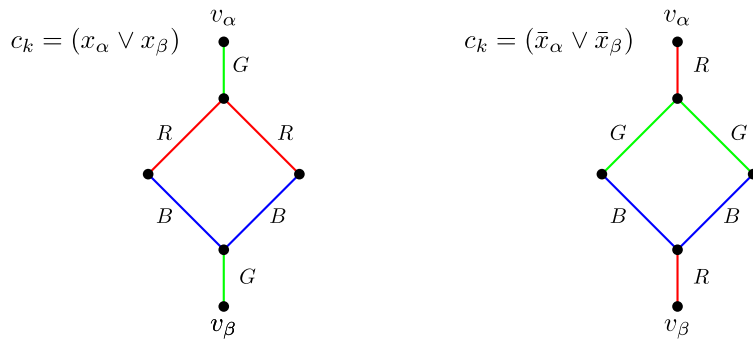


Fig. 2. The case when all literals are positive or all literals are negative.

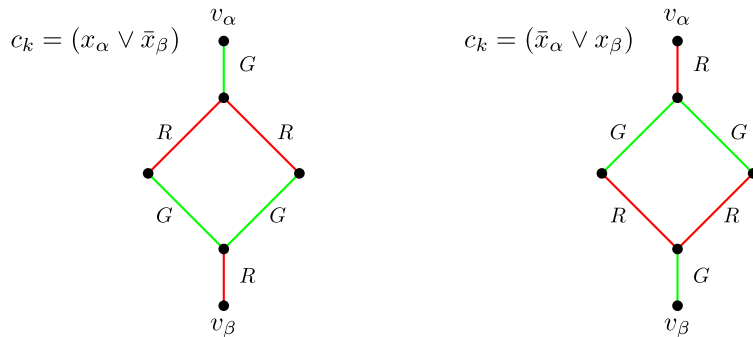


Fig. 3. The case when one literal is positive and one literal is negative.

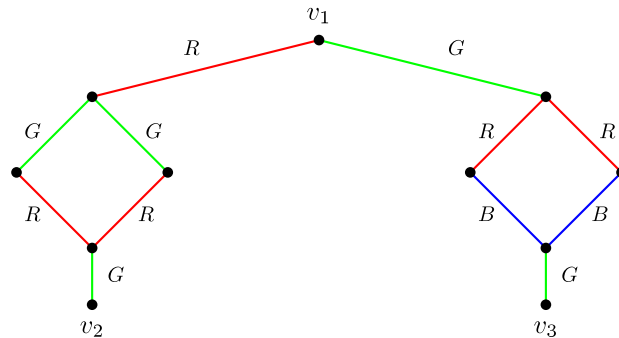


Fig. 4. The graph devised for the formula $(\bar{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$.

It is not difficult to check that the constructed graph does not contain any odd cycle and so it is bipartite. Also, for every vertex of the constructed graph the edges that are incident to this vertex are colored with at most two different colors, i.e. the chromatic degree of the graph is two. The next lemma follows from a simple case analysis.

Lemma 6. *The maximum contribution that any gadget can have is exactly 3 and is obtained when at least one of the nodes v_α and v_β has the same color as the edge that connects it with the rest of the gadget. If none of v_α and v_β has the same color with the edge that connects it with the rest of the gadget then the maximum contribution that can be achieved is 2.*

Lemma 7. *For an instance of the MAX-2-SAT problem $\mathcal{I}_{\text{MAX-2-SAT}} = \langle \mathcal{X}, C, B \rangle$, there is a truth-assignment that satisfies at least B clauses if and only if there is a clustering for the corresponding MAX-3-CC problem with contribution greater than or equal to $3 \cdot B + 2(m - B)$, where $m = |C|$ is the number of clauses.*

Proof. To prove the if direction, let T be a truth assignment that satisfies at least B clauses of a 2-SAT formula F . In the derived graph, color green all the nodes that correspond to variables that are true and red all the nodes that correspond to false variables. In this way, for each satisfied clause the corresponding gadget in the optimum clustering will have pay-off three.

Since each of the gadgets representing a satisfied clause will contribute three to the pay-off and the satisfied clauses are $L \geq B$ the total optimal contribution of these clauses will be $3 \cdot L$. The gadgets of the remaining $m - L$ clauses will have

optimal contribution 2 each and the total optimal contribution from the unsatisfied clauses will be $2 \cdot (m - L)$. Hence the total pay-off will be $2 \cdot (m - L) + 3 \cdot L = 2 \cdot m + L \geq 2 \cdot m + B = 3 \cdot B + 2(m - B)$.

For the opposite direction, suppose that the corresponding graph of a formula F has a partition with pay-off at least $3 \cdot B + 2(m - B) = 2 \cdot m + B$. Since each one of the gadgets contributes to the pay-off either 2 or 3, there must exist at least B gadgets with pay-off 3.

Let us assign the value true to the variables with green corresponding nodes and the value false to the remaining variables. Notice now that each one of the gadgets with pay-off three corresponds to a satisfied clause.

Since the gadgets with pay-off 3 are at least B , there are at least B clauses that are satisfied and the only if direction holds too. \square

4. A reduction to the independent set problem

In this section we will show that the colored clustering problem can be reduced to the IS problem in bipartite graphs.

Given an instance of the MAX- k -CC problem, we create the line graph G_{line} corresponding to the initial graph. We then construct a new graph G'_{line} by deleting the edges between the vertices of G_{line} that correspond to neighboring edges of the same color in G .

Lemma 8. *The MAX- k -CC problem has a clustering with pay-off P if and only if the graph G'_{line} has an independent set of size P .*

Proof. For the if direction, suppose that the initial problem has a partition with pay-off P . For this to happen there must exist a set \mathcal{L} of P edges with properly colored ends. Each edge $e \in \mathcal{L}$ is either adjacent to some other edges in \mathcal{L} and all have the same color or not adjacent with any other edge in \mathcal{L} . In either case, the vertex in G'_{line} that corresponds to e is not adjacent to any vertex corresponding to some other edge in \mathcal{L} , because in G'_{line} we have eliminated the edges between vertices corresponding to adjacent edges with the same color. Hence, the nodes of G'_{line} that correspond to edges in \mathcal{L} form an independent set of size P .

To prove the opposite direction, let us examine an instance of the induced problem that has an independent set of size P . The nodes that form the independent set correspond to edges of the initial graph that either are not adjacent or are adjacent and have the same color. Therefore it is possible to color the extremities of these edges with the same color as the edges themselves and hence to produce a solution with pay-off P , because there are P such edges. \square

For $k = 2$, the constructed graph is always bipartite. Indeed, in G'_{line} we have eliminated the edges between nodes of the line graph G_{line} that correspond to edges of the same color in the initial graph G . So, while traversing any cycle of G'_{line} the color of the corresponding edge must change from node to node. Since there are only two different colors, any cycle must have even length and, therefore, the graph is bipartite. Notice that our reduction holds also for the weighted case.

As a result, given that a weighted independent set can be found in polynomial time in a bipartite graph, we get that the weighted MAX-2-CC is polynomially solvable.

For any $k \geq 3$ we can also derive from Lemma 8 a $\frac{2}{k}$ approximation algorithm for the weighted MAX- k -CC. Although, it is not a constant-approximation algorithm its ratio is better than $1/e^2$ for every $k \leq 14$. We use the following Theorem, from [5]: Let G be a weighted graph with n vertices and m edges; let k be an integer greater than one. If it takes only s steps to color the vertices of G in k colors, then it takes only $s + O(nm \log(n^2/m))$ steps to find an independent set whose weight is at least $2/k$ times the weight of an optimal independent set. In our case we have $s = 0$.

Acknowledgments

This work has been partially supported by the ANR project TODO (09-EMER-010), and by the project ALGONOW of the research funding program THALIS (co-financed by the European Social Fund-ESF and Greek national funds).

References

- [1] A.A. Ageev, A.V. Koonov, Improved Approximations for the Max k -Colored Clustering Problem, in: LNCS, vol. 8952, 2015, pp. 1–10.
- [2] N. Bansal, A. Blum, S. Chawla, Correlation Clustering, in: Machine Learning, vol. 56, 2004, pp. 89–113.
- [3] G. Ducoffe, D. Mazaauric, A. Chaintreau, Convergence of Coloring Games with Collusions. CoRR abs/1212.3782, 2012.
- [4] U. Feige, J. Vondrák, Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. FOCS 2006: 667–676.
- [5] Dorit S. Hochbaum, Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems, in: Hochbaum Dorit S. (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Company, 1997.
- [6] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, 1981.
- [7] J.M. Kleinberg, K. Ligett, Information-Sharing and Privacy in Social Networks. CoRR abs/1003.0469, 2010.
- [8] V. Vazirani, Approximation Algorithms, Springer, 2004.
- [9] U. Zwick, Analyzing the MAX 2-SAT and MAX DI-CUT approximation algorithms of Feige and Goemans, currently available from <http://www.cs.tau.ac.il/~zwick/online-papers.html>.