Computational Optimization and Applications (2022) 83:525–553
https://doi.org/10.1007/s10589-022-00406-7

Minimum cost *b*-matching problems with neighborhoods

I. Espejo[1] · **R. Páez**[1] · **J. Puerto**[2,3] · **A. M. Rodríguez-Chía**[1]

Received: 10 November 2021 / Accepted: 28 July 2022 / Published online: 21 August 2022

Abstract
In this paper, we deal with minimum cost *b*-matching problems on graphs where the nodes are assumed to belong to non-necessarily convex regions called neighborhoods, and the costs are given by the distances between points of the neighborhoods. The goal in the proposed problems is twofold: (i) finding a *b*-matching in the graph and (ii) determining a point in each neighborhood to be the connection point among the edges defining the b-matching. Different variants of the minimum cost *b*-matching problem are considered depending on the criteria to match neighborhoods: perfect, maximum cardinality, maximal and the *a*–*b*-matching problems. The theoretical complexity of solving each one of these problems is analyzed. Different mixed integer non-linear programming formulations are proposed for each one of the considered problems and then reformulated as Second Order Cone formulations. An extensive computational experience shows the efficiency of the proposed formulations to solve the problems under study.

Keywords Matching · Neighborhoods · SOC formulations

1 Introduction

Matching problems are among the most well-studied problems in combinatorial optimization. The original motivations of the problem were minimizing transportation costs and optimally assigning personnel to job positions, although many more applications of matching problems can be formulated as optimization problems defined on networks. Matching theory has been even used in the solution of linear systems. In Schenk et al. [33] proposed an alternative technique to the traditional pivoting strategies to factorize matrices based on symmetric weighted matchings.

✉ I. Espejo
inmaculada.espejo@uca.es

[1] Departamento de Estadística e Investigación Operativa, Universidad de Cádiz, Cádiz, Spain

[2] Departamento de Estadística e Investigación Operativa, Universidad de Sevilla, Seville, Spain

[3] IMUS Instituto de Matemáticas, Universidad de Sevilla, Seville, Spain

Recently, matching theory has been used for modeling and analyzing wireless resource allocation in a wireless network [24], where a matching was essentially an allocation between resources and users. Many real world applications are usually modelled throughout the use of networks. However, very often the exact location of the nodes is unknown or simply, we have to choose a location for the node within a given region. Thus, the assumption of modelling the nodes by points with a priori known location should be revised. In this sense, we propose to consider a region where the location of a node can lie, that is usually referred in the literature as *neighborhood*, see Arkin and Hassin [3], Dorrigiv et al. [14], Gentilini et al. [20] among others. In this paper, we deal with matching problems on graphs where nodes are points of their corresponding neighborhoods. For simplicity, we will say that the nodes of the graph are represented by neighborhoods.

A matching of a graph is a subgraph in which each node has a degree of at most one. A *maximum matching* is a matching of maximum cardinality, that is, a matching with the greatest possible number of edges. If each edge has an associated weight, the *maximum weight matching* is a matching which maximizes the sum of the weights. Alternatively, edges may have costs, and the *minimum cost matching* consists of finding a matching with the lowest possible cost, satisfying some specific characteristic. In particular, *the minimum cost maximal matching* consists of finding a maximal matching (no other edge can be added to it while keeping the property of being a matching) which minimizes the sum of the costs of all the edges in the matching. There are excellent surveys on Matching Theory by Gerards [21], Lovasz and Plummer [25], and Pulleyblank [32].

Generalizations of the matching problem consider more general degree constraints. These generalizations lead to so called *b-matchings* or *f-factors* [19, 21, 25]. The b-matching problems considered in this paper are those matching problems where each node is met by at most b edges and the same edge can be used at most once. Important applications of b-matching problems include the T-join problem, the Chinese postman problem, the 2-factor relaxation for the symmetric traveling salesman problem and capacitated vehicle routing [29]. The b-matching model is used to solve numerous problems in different areas, e.g. Tennenholtz [36] in the context of combinatorial auctions and Dong et al. [13] for energy allocation in sea surface monitoring problems. The problem of finding a b-matching of maximum cardinality has been widely studied [2, 18, 19]. The minimum-cost perfect bipartite b-matching problem has been studied in Cohen et al. [10]. Although the minimum cost maximal matching has been addressed by Bodur et al. [9], Taşkin and Ekim [35], Tural [37], to the best of our knowledge, the minimum cost maximal b-matching problem with $b > 1$ has not been studied.

The computational complexity of Matching problems, has been extensively studied in the literature. Edmond [17] proved that the maximum cardinality matching problem is solvable in polynomial-time and proposed a strongly polynomial algorithm for the maximum (minimum) weighted matching problem. The minimum cost maximal 1-matching problem is NP-hard in general graphs [9]. Vaidya [38] showed that Euclidean matching problems, in which the nodes are given as points in the plane and the weight of an edge between the two points is the distance between the two points in the plane, can be solved in $\mathcal{O}(n^{5/2}(\log n)^4)$

time, for n the number of nodes of the graph. When the points lie on the boundary of a convex polygon, a minimum weight perfect matching with Euclidean distances can be found in $\mathcal{O}(n \log n)$ time [28]. They also proved that this complexity is $\mathcal{O}(n\log^2 n)$ when the points lie on a simple non-convex polygon. Cunningham and Marsh [11] found the first polynomial time algorithm for finding an optimal b-matching of maximum cost and Anstee [2] presented a strongly polynomial algorithm for solving this problem. The maximum cardinality b-matching of maximum cost can be solved in $\mathcal{O}(nm\log^2 n)$ time [19], for m the number of edges of the graph.

As far as we know, matching problems on a graph with neighborhoods have not been investigated yet. Recently, some extensions of combinatorial optimization problems have been considered on a graph using convex neighborhoods to represent its nodes. In particular, the Traveling Salesman Problem with Neighborhoods (TSPN) consists of searching the minimum-length tour visiting just once each one of the neighborhoods. This problem was introduced by Arkin and Hassin [3] and studied when the neighborhoods are polygonal regions, discs or disjoint convex full dimension objects [12, 16, 20, 22]. As a generalization of the classical TSP problem, the Euclidean TSPN problem is also NP-hard. The Minimum Spanning Tree Problem with Neighborhoods (MSTPN) aims to identify a representative point from each neighborhood so that the MSTP of the set of representative points has the minimum total length among all possible spanning trees. The Minimum Spanning Tree Problem where neighborhoods are a set of disjoint disks, rectangles or second order cone representable, was analyzed in Blanco et al. [8], Dorrigiv et al. [14], Puerto and Valverde [31], Yang et al. [39]. The general case of this problem in the plane is NP-hard.

This paper concentrates on the Minimum Cost b-Matching Problems (MCbMP's) with neighborhoods instead of nodes, where the costs represent distances between points in each neighborhood. Depending on the criteria of matching neighborhoods, different variants of the MCbMP's are considered. In particular, we will study the minimum cost versions of: (i) the perfect b-matching problem (each neighborhoods is met by exactly b edges), (ii) the maximum cardinality b-matching problem (find a b-matching with the greatest possible number of edges), (iii) the maximal b-matching problem (no other edge can be added to it while keeping the property of being a b-matching) and (iv) the a–b-matching problem (in addition to the conditions of a b-matching, at least a-edges are incident in each neighborhood).

Applications of this kind of problems can be found in wireless network, where the goal is to serve the voice and/or data traffic demand of a particular geographic region (not always convex in structure). The traffic from and/or to a user device of a region is driven through a cell tower [30]. When the number of requests overhead the maximum capacity of the tower, the traffic is routed to the cell tower's antenna in another region, usually the closest one to reduce costs and guarantee the signal quality. Hence, when designing the wireless network, one may pose the location of a cell tower in each region (neighborhood) taking into account to which other tower will be connected in case of overhead the maximum capacity. The goal is matching neighborhoods and locating a cell tower in each of them to minimize the total sum of all the distances between the matched neighborhoods' towers. In order to prevent

that more than one tower overhead the maximum capacity, a number b could be set such that each neighborhood is matched to, at most, other b-neighborhoods instead of to only one. Depending on the criteria to find the b-matching with neighborhoods, the aforementioned problems arise. For instance, considering the maximum cardinality 2–4-matching criterion, the goal would be to find a matching taking into account that: (1) each tower can be covered, in the case of overhead its capacity, by at least two towers and at most four towers, (2) the higher number of towers to cover a possible overhead of another tower, the better for the quality of the connection among the towers.

To the best of our knowledge, this paper is the first attempt to deal with neighborhoods in matching problems and there is no previous results on the complexity of this kind of problems. We study different cases depending on b, the matching criterion and the type of neighborhoods. Table 1 summarizes the complexity results for some of the b-matching problems studied in this paper. This table allows us to compare the complexity in the case of nodes and neighborhoods. It is worth mentioning that matching problems that are strongly polynomial with nodes are still polynomially solvable with Second Order Cone (SOC) representable neighborhoods. However, they become NP-hard when non-convex polygons are considered as neighborhoods.

In spite of the hardness of some problems, we will provide mathematical programming tools for solving the aforementioned problems. In particular, we provide SOC based formulations for these problems with non-convex neighborhoods given as the union of SOC representable sets. The main reason for this is that they can be solved by interior point methods and, in general, are more efficient that Semidefinite Programming (SDP) problems. In addition, state-of-the-art solvers incorporate mixed integer non-linear implementations of SOC constraints. For further details on the SOC constraints, see the books of Ben-Tal and Nemirovski [5], Luemberger and Ye [27].

The rest of the paper is organized as follows. First, we describe the MCbMP's with all the variants studied here. Their complexity is analyzed in Sect. 3. In Sect. 4, different non-linear integer programming formulations are developed for the different variants of the b-matching problems. In Sect. 5, these formulations are rewritten as SOC programs in the case of non-convex neighborhoods given as the union of SOC representable sets. Before assessing their performance on randomly generated graphs in the computational results section, a procedure to generate initial solutions is described in Sect. 6. The paper ends with a section of conclusions and some possible future lines of research.

2 Minimum cost b-matching problems with neighborhoods

Let $G = (V, E)$ be an undirected graph where the nodes are represented by neighborhoods $\mathcal{N}_i$, with $i \in N := \{1, \ldots, n\}$, E be the set of edges between the neighborhoods of V, and $\delta(i)$ be the set of edges incident to $\mathcal{N}_i$. In this paper, for the sake of presentation, we will restrict to three types of neighborhoods: non-convex polygons, balls of ℓ_p-norms and convex polyhedra. Depending on the case the neighborhoods

Table 1 Complexity results for minimum cost matching problems

b	Minimum cost b-matching	Nodes	Convex neighborhood	Non-convex neighborhood
1	Perfect	Strongly Polynomial (Edmond [17])	Polynomial with SOC representable sets (Proposition 3.3)	NP-Hard with non-convex polygons (Proposition 3.2)
1	Maximum cardinality	Strongly Polynomial (Edmond [17])	Polynomial with SOC representable sets (Proposition 3.3)	NP-Hard with non-convex polygons (Proposition 3.2)
1	Maximal	NP-hard (Bodur et al. [9]). Polynomial in trees (Tural [37])	Polynomial in trees with SOC representable sets (Proposition 3.4)	
b	Perfect	Strongly polynomial (Anstee [2])		NP-Hard with non-convex polygons (Proposition 3.2)
b	Maximum cardinality	Strongly polynomial (Gabow [19])		NP-Hard with non-convex polygons (Proposition 3.2)
b	Maximal			
a–b	Maximum cardinality			NP-Hard with non-convex polygons (Proposition 3.2)
a–b	Maximal			

are encoded by the list of their consecutive vertices in case of non-convex polygons, the center, radius in case of a ℓ_p ball and by the set of inequalities describing the polyhedron. A *b-matching* $\mathcal{M} \subseteq E$ is a subset of edges such that each neighborhood in V is met by at most b edges in $\mathcal{M}$ and the same edge can be used at most once. In the b-matching problem, b is the so-called *node constraint* or *degree requirement* and it could be considered different for each neighborhood. Indeed, the maximum number of edges that can met $\mathcal{N}_i$ will be the minimum of $|\delta(i)|$ and b, and this can be different for each neighborhood. For the sake of simplicity and without loss of generality, we will assume the same value b for all the neighborhoods.

In the MCbMP's with neighborhoods the goal is twofold: (i) finding a b-matching in the underlying graph, and (ii) determining a point in each neighborhood to be the connection point among the edges defining the b-matching. We assume that the same point $z_i \in \mathcal{N}_i$ is used to calculate the distances with all the matched neighborhoods with $\mathcal{N}_i$. Note that this point could be anywhere in the neighborhood, even in its interior. Therefore, the sum of the lengths (costs) of the edges defined by these points being part of the b-matching is minimized. Indeed, to minimize the sum of all distances between matched neighborhoods, we have to simultaneously obtain the b-matching and points $z_i \in \mathcal{N}_i$, determining the distances between the neighborhoods. Note that the main difference with the MCbMP's with nodes is that in the version with neighborhoods, the distances are not part of the input, but part of the decision.

We introduce the following optimization problems:

(i) the minimum cost perfect b-matching problem (P_MCbMP) consists of finding the perfect b-matching of minimum cost, i.e., each neighborhoods is met by exactly b egdes.
(ii) the minimum cost maximum cardinality b-matching problem (M_{card}_MCbMP) consists of finding a b-matching with the maximum cardinality of minimum cost, i.e., with the greatest possible number of matched neighborhoods.
(iii) the minimum cost maximal b-matching problem (Max_MCbMP) consists of finding a maximal b-matching of minimum cost, i.e., no other edge can be added to it while keeping the property of being a b-matching.
(iv) the minimum cost maximum cardinality/maximal a–b-matching problem, denoted by M_{card}_MCabMP and Max_MCabMP respectively, with $a \in \mathbb{Z}$, $0 \leq a < b$, consists of finding a maximum cardinality/maximal b-matching of minimum cost where it is imposed that each neighborhood is matched to at least a neighborhoods. Observe that in a M_{card}_MCbMP or Max_MCbMP, a neighborhood could be not matched to any neighborhood. This situation is avoided in the a–b-matching problem with $a > 0$. When $a = 0$, the classical minimum cost maximum cardinality/maximal b-matching problem is obtained.

The following example illustrates the MCbMP's.

Example 2.1 Consider a graph $G = (V, E)$ in $\mathbb{R}^2$, with a set of nodes given by their coordinates $V = \{(40, 105), (63, 111), (103, 75), (66, 35), (36, 44), (58, 70)\}$, and the set of edges $E = \{(1, 2), (1, 3), (1, 6), (2, 3), (3, 6), (4, 5), (4, 6), (5, 6)\}$. An optimal

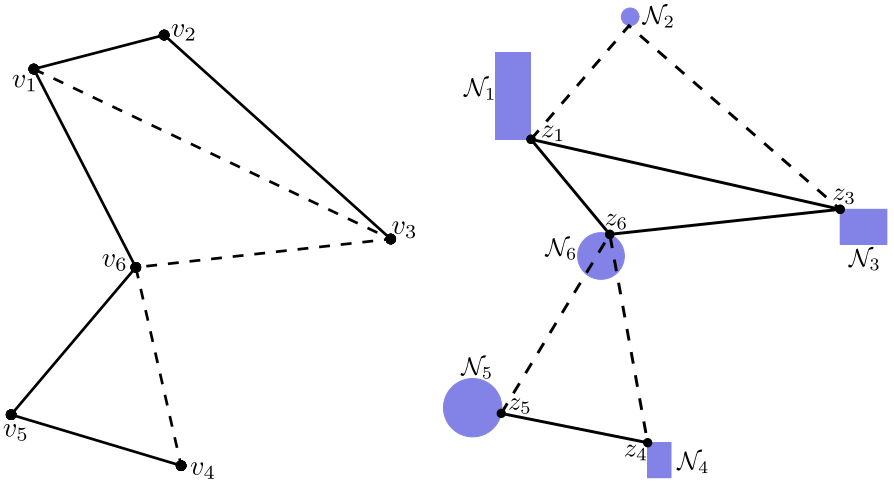
solution for the minimum cost maximal 2-matching problem using the Euclidean norm is given by the solid line in Fig. 1a, and the total cost is 182.32.

Now, we consider $V = \{\mathcal{N}_1, \ldots, \mathcal{N}_6\}$ a set of neighborhoods with $v_i \in \mathcal{N}_i$, for $i = 1, \ldots, 6$ and the same set of edges. For the sake of simplicity, the neighborhoods are defined as Euclidean disks and rectangles. The centers and radii of the disks are given by $\{\{(63, 111), 1.5\}, \{(36, 44), 5\}, \{(58, 70), 4\}\}$, and the lower left corner/upper right corner of the rectangles are $\{\{(40, 90), (46, 115)\}, \{(99, 72), (107, 78)\}, \{(66, 32), (70, 38)\}\}$. For solving the minimum cost maximal 2-matching problem using the Euclidean norm, we have to simultaneously obtain the maximal 2-matching and six points $z_i \in \mathcal{N}_i$, which determine the distances between the neighborhoods. The same point $z_i \in \mathcal{N}_i$ is used to calculate the distances with all the matched neighborhoods with $\mathcal{N}_i$.

The optimal matching with neighborhoods is given by the solid line in Fig. 1b, and the total cost is 140.83. For instance, $\mathcal{N}_1$ is matched to $\mathcal{N}_3$ and $\mathcal{N}_6$ and the point of the neighborhood $\mathcal{N}_1$ used to calculate the distances to $\mathcal{N}_3$ and $\mathcal{N}_6$ is the same point for both of them, z_1. Note that the distance between $\mathcal{N}_1$ and $\mathcal{N}_6$ using z_1 and z_6 is not the closest distance between neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_6$.

Comparing the optimal solutions of the maximal 2-matching problem with/without neighborhoods, we observe that $\mathcal{N}_2$ is not matched to any neighborhood in the optimal solution of the maximal 2-matching problem with neighborhoods (Fig. 1b). However, in the optimal solution without neighborhoods (Fig. 1a), v_2 is matched to v_1 and v_3. This shows the influence in the optimal solution of the points chosen in each neighborhood to determine the distances.

To illustrate the different b-matching problems introduced above, we consider the following example.



(a) Without neighborhoods **(b)** With neighborhoods

Fig. 1 The minimum cost maximal 2-matching problem with euclidean distances

Example 2.2 Consider the same graph of Example 2.1.

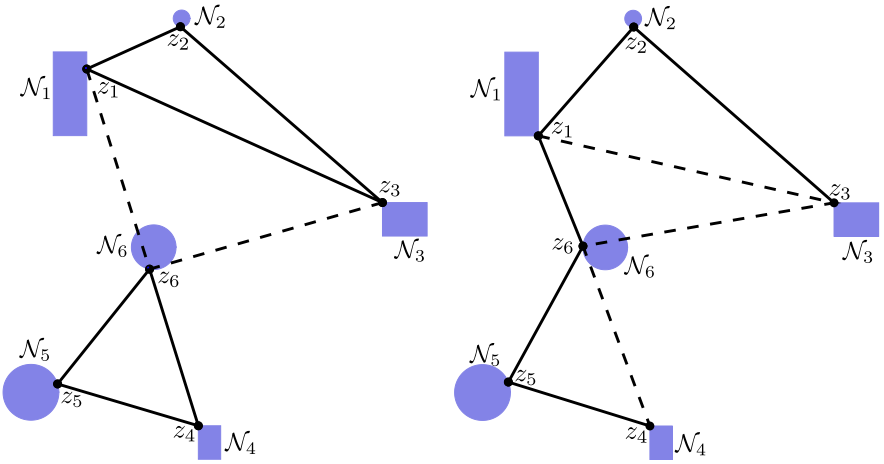
An optimal solution of the perfect 2-matching problem is given by the six solid lines in Fig. 2a and the total cost associated with this matching is 206.66. This is also the optimal solution of the maximum cardinality 2-matching problem. Note that this solution is maximal, and contains two edges more than the solution of the minimum cost maximal 2-matching problem given in Fig. 1b with total cost 140.83.

The optimal solution of the minimum cost maximal 2-matching given in Fig. 1b allows a neighborhood not to be matched to any other (see neighborhood $\mathcal{N}_2$). If we impose that all the neighborhoods must be matched to at least another one (1–2-matching problem), the optimal solution of the maximal 1–2-matching of minimum cost is given in Fig. 2b. The total cost associated with this matching is then 149.37. Note that now, neighborhood $\mathcal{N}_6$ is matched to $\mathcal{N}_1$ and $\mathcal{N}_5$ whereas in the optimal solution of the minimum cost maximal 0–2-matching $\mathcal{N}_6$ was matched to $\mathcal{N}_1$ and $\mathcal{N}_3$.

Observe that whenever the minimum cost perfect b-matching problem is feasible, the minimum cost maximum cardinality b-matching problem has the same solution (see Fig. 2a). However, the former can be unfeasible and the latter is always feasible, see Fig. 3.

3 Comparing the MCbMP's and analyzing the complexity

After describing in Sect. 2 the different b-matching problems of minimum cost studied in this paper, we show the relationship between the optimal total cost of all of them. Let f_b^P, f_b^{MC}, f_b^M be the optimal objective value of the P_MCbMP,



(a) Perfect and Maximum Cardinality 2-matching

(b) Maximal 1-2-matching

Fig. 2 2-matching problems of minimum cost

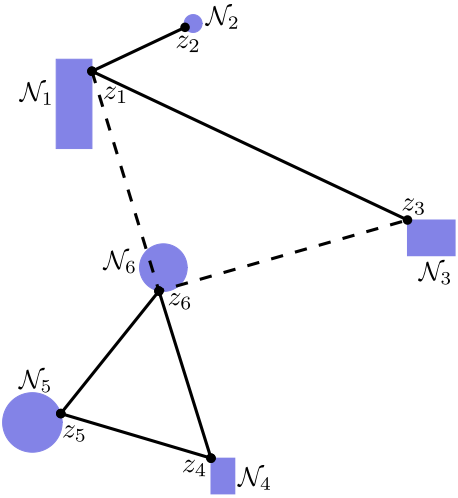


Fig. 3 The minimum cost maximum cardinality 2-matching problem

M_{card}_MCbMP and Max_MCbMP, respectively, and f_{ab}^{MC}, f_{ab}^{M} the corresponding version of the a–b-matching.

Proposition 3.1 *The following relationships are satisfied:*

(i) $f_b^M \leq f_b^{MC}$ *and* $f_b^{MC} = f_b^P$, *if P_MCbMP is feasible.*
(ii) $f_b^M \leq f_{ab}^M$.
(iii) $f_b^{MC} \leq f_{ab}^{MC}$.

Proof This result follows from the fact that the solutions of the perfect b-matching are feasible solutions of b-matching of maximum cardinality problems. Moreover, the feasible solutions of the latter are also feasible solutions of the maximal b-matching problem. On the other hand, the feasible solutions of the a–b-matching are feasible solutions of the corresponding minimum cost b-matching problem. □

Figure 4 shows the relationship among feasible solutions sets of the MCbMP's depending on the values of a and b.

In the following, we analyze the complexity of the different variants of the MCbMP's depending on b and the structure of the neighborhoods.

Proposition 3.2 *The P_MCbMP and* M_{card}*_MCbMP are NP-hard with non-convex polygons.*

Proof We consider a graph $G = (V, E)$ where $V = \{\mathcal{N}_i : i = 1, \ldots, k\}$, with $\mathcal{N}_1$ a point, $\mathcal{N}_i$ $(i \neq 1)$ given by non-convex polygons, and

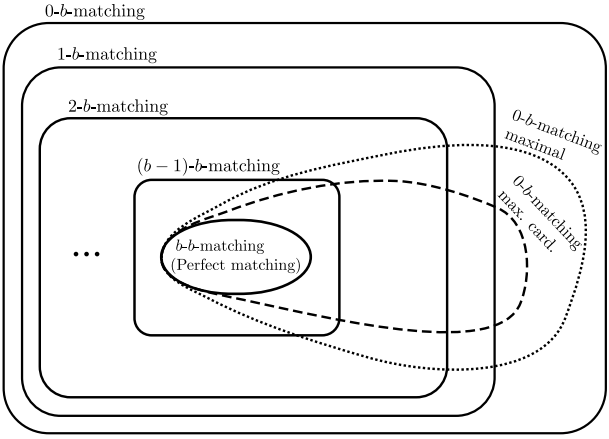


Fig. 4 Relationship among feasible solutions sets of the MCbMP's

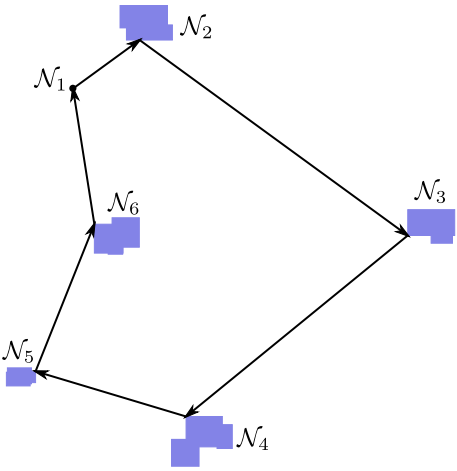


Fig. 5 Minimum cost perfect/maximum cardinality 2-matching problem with non-convex neighborhoods

$E = \{(1,2),(2,3),(3,4),\ldots,(k-1,k)(k,1)\}$ (Fig. 5). The minimum cost perfect/maximum cardinality 2-matching problem is equivalent to the Touring Polygons problem [15] which consists of finding a shortest tour (shortest cycle) that starts at $\mathcal{N}_1$, visits the polygons in the given order, and ends again at $\mathcal{N}_1$. Since the polygons are non-convex, the Touring Polygons problem is NP-hard for any metric ℓ_p, $p \geq 1$ [1, 15]. Therefore, the P_MCbMP and M_{card}_MCbMP are NP-hard and the result holds. □

From Proposition 3.2, we have the following result.

Corollary 3.1 *The M_{card}_MCabMP is NP-hard with non-convex polygons.*

Observe that (Max_MCbMP) and (Max_MCabMP) are NP-hard because the minimum cost maximal 1-matching problem where the nodes are points is NP-hard.

However, when $b = 1$ the complexity of some versions of the minimum cost 1-matching problems become polynomial in some cases.

Proposition 3.3 *The P_MC1MP and M_{card}_MC1MP with SOC-representable neighborhoods are solvable in polynomial time.*

Proof When $b = 1$, each neighborhood is met by at most one edge and the distances between neighborhoods are independent of one another. Therefore, we can previously determine the distances between neighborhoods (solving minimum distances problems between neighborhoods) and then solving a classical matching problem in a graph with nodes using these computed distances as the costs or distances of the edges between the nodes.

Provided that the distances between neighborhoods can be computed in polynomial time, since these distances can be reduced to solve a SOC programming problem and the version of the minimum cost perfect and maximum cardinality matching problems are solvable in polynomial time, the result follows. □

Proposition 3.4 *The Max_MC1MP on trees with SOC-representable neighborhoods is solvable in polynomial time.*

Proof The case of Max_MC1MP is solvable in polynomial time on trees [37]. Following similar arguments to the ones given in Proposition 3.3, the result holds. □

Note that the complexity given in the previous two results are polynomial in the sense of the complexity of interior point algorithm for solving SOC representable problems. Moreover, the results given in Propositions 3.3 and 3.4 could be extended to the case of any neighborhood, not only SOC-representable, provided that the distances between neighborhoods could be computed in polynomial time.

4 Formulations for MCbMP's

In this section we present mixed integer non-linear programming formulations for the different MCbMP's defined in Sect. 2. All formulations use the following sets of decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if } (i,j) \text{ belongs to the matching,} \\ 0, & \text{otherwise,} \end{cases} \qquad \forall (i,j) \in E, (i < j).$$

$$z_i : \text{continuous variables to represent the point selected in each neighborhood,} \qquad \forall i \in N.$$

Moreover, we denote $||\cdot||_q$ the standard ℓ_q-norm with $q \in [1, \infty)$. In this section, constraints related to the structure of the neighborhoods are not explicitly

formulated, and we will refer to them just by $z_i \in \mathcal{N}_i$. In Sect. 5, we will formulate them properly.

4.1 The minimum cost perfect *b*-matching problem

This section deals with the P_MCbMP, where each neighborhood $\mathcal{N}_i$ is met by exactly b edges. This problem can be formulated as follows:

$$(\mathrm{F}_P) \min \sum_{\substack{(i,j)\in E \\ i<j}} ||z_i - z_j||_q x_{ij} \tag{1}$$

$$\begin{aligned} s.t. \sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} &= b, \quad \forall i \in N, \\ x_{ij} &\in \{0,1\}, \quad \forall i, j(i<j) \in N, \\ z_i &\in \mathcal{N}_i, \quad \forall i \in N. \end{aligned} \tag{2}$$

The objective function (1) accounts for the sum of the length of the matching edges which is determined by the distance between the chosen points in each neighborhood. Constraints (2) define the perfect b-matching constraints, i.e., each neighborhood meets exactly b edges.

4.2 The minimum cost *b*-matching problem with maximum cardinality

The M_{card}_MCbMP can be formulated as a maximum weight b-matching problem as follows:

$$\begin{aligned} (\mathrm{F}_{card}) \max & \sum_{\substack{(i,j)\in E \\ i<j}} (M - ||z_i - z_j||_q) x_{ij} \\ s.t. & \sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} \leq b, \quad \forall i \in N, \\ & x_{ij} \in \{0,1\}, \quad \forall i, j(i<j) \in N, \\ & z_i \in \mathcal{N}_i, \quad \forall i \in N, \end{aligned} \tag{3}$$

where $M := \max_{\substack{i,j\in N \\ z_i\in\mathcal{N}_i, z_j\in\mathcal{N}_j}} \{||z_i - z_j||_q\} + 1$.

This problem has been transformed in a maximum problem using a big-M parameter, in such a way that if an edge is not used, the penalization is huge. Therefore, the choice of the maximum number of edges is guaranteed. Family of constraints (3) is the classical family of b-matching constraints when the number of edges is being maximized.

4.3 The minimum cost maximal b-matching problem

In this section we propose two different formulations for the Max_MCbMP. First, we give the following definition. A neighborhood $\mathcal{N}_i \in V$ is said saturated if no other edge can met $\mathcal{N}_i$. For each $\mathcal{N}_i \in V$, the maximum number of edges that can met $\mathcal{N}_i$ is defined as $S_i = \min\{b, |\delta(i)|\}$. Thus, a neighborhood $\mathcal{N}_i$ is said saturated if $\sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} = S_i$.

The first formulation of the minimum cost maximal b-matching problem consider binary y-variables to keep under control when neighborhood i can be matched with neighborhood j (because they are not saturated), i.e.,

$$y_{ij} = \begin{cases} 1, & \text{if } (i,j) \text{ could be in the matching,} \\ 0, & \text{otherwise,} \end{cases} \quad \forall (i,j) \in E.$$

Hence, this problem can be formulated as follows:

$$\begin{aligned} (\text{F1}_{max}) \min \quad & \sum_{\substack{(i,j)\in E \\ i<j}} ||z_i - z_j||_q x_{ij} \\ s.t. \quad & \sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} \leq S_i, \quad \forall i \in N, \end{aligned} \tag{4}$$

$$S_i y_{ij} \geq S_i - \left(\sum_{\substack{k:(i,k)\in E \\ k\neq j; i<k}} x_{ik} + \sum_{\substack{k:(i,k)\in E \\ k\neq j; k<i}} x_{ki} \right), \quad \forall i \in N, j : (i,j) \in E, \tag{5}$$

$$x_{ij} \geq y_{ij} + y_{ji} - 1, \quad \forall (i,j) \in E, i < j, \tag{6}$$

$$\begin{aligned} & y_{ij} \in \{0,1\}, \quad \forall (i,j) \in E, \\ & x_{ij} \in \{0,1\}, \quad \forall (i,j) \in E, i < j, \end{aligned} \tag{7}$$

$$z_i \in \mathcal{N}_i, \quad \forall i \in N. \tag{8}$$

Constraints (4) ensures that the generated solution is a b-matching. Constraints (5) ensure that $y_{ij} = 1$ if neighborhood $\mathcal{N}_i$ is not saturated without using edge (i, j) ($\sum_{\substack{k:(i,k)\in E \\ k\neq j; i<k}} x_{ik} + \sum_{\substack{k:(i,k)\in E \\ k\neq j; k<i}} x_{ki} < S_i$). Constraints (6) ensure that $x_{ij} = 1$ if $y_{ij} = 1$ and $y_{ji} = 1$. That is, if neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ are not saturated, then $\mathcal{N}_i$ and $\mathcal{N}_j$ are matched.

An alternative formulation can be considered using the following binary variables [35]:

$$y_i = \begin{cases} 1, & \text{if } \mathcal{N}_i \text{ is saturated in the matching,} \\ 0, & \text{otherwise.} \end{cases} \quad \forall i \in N.$$

The alternative formulation for the minimum cost maximal b-matching problem using the variables defined above is given by

$$\begin{aligned} (\mathrm{F2}_{max})\ \min \quad & \sum_{\substack{(i,j)\in E \\ i<j}} ||z_i - z_j||_q x_{ij} \\ s.t. \quad & (4), (7), (8) \\ & x_{ij} \geq 1 - y_i - y_j, \quad \forall (i,j) \in E, i<j, \end{aligned} \tag{9}$$

$$\begin{aligned} & S_i y_i \leq \sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji}, \quad \forall i \in N, \\ & y_i \in \{0,1\}, \quad \forall i \in N. \end{aligned} \tag{10}$$

Constraints (9) enforce the condition that if $x_{ij} = 0$, neighborhood i or j has to be saturated, i.e., $y_i = 1$ or $y_j = 1$. Constraints (10) ensure that if neighborhood i is not saturated, $(\sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} < S_i)$ then $y_i = 0$.

The following constraints are valid inequalities for $(\mathrm{F2}_{max})$,

$$y_i \geq \sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} - S_i + 1, \quad \forall i \in N. \tag{11}$$

That is, if neighborhood $\mathcal{N}_i$ is saturated $(\sum_{\substack{j:(i,j)\in E \\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E \\ j<i}} x_{ji} = S_i)$, then $y_i = 1$.

Further constraints can be added to reinforce the formulation in cases where $|\delta(i)| = 1$:

$$y_j \geq 1 - x_{ij} \quad \forall (i,j) \in E, |\delta(i)| = 1.$$

To the best of our knowledge, the Max_MCbMP with $b > 1$, for the standard case (by considering a set of nodes), has not been studied. Therefore, formulations $(\mathrm{F1}_{max})$ and $(\mathrm{F2}_{max})$ after removing (8) are also valid for solving the standard minimum cost maximal b-matching problem if we consider singletons instead of neighborhoods.

4.4 The minimum cost *a*–*b*-matching problems

This section is devoted to the minimum cost b-matching problems where additional constraints are imposed to the matching in order to guarantee each neighborhood $\mathcal{N}_i$ is matched to at least a neighborhoods, with $1 \leq a \leq b$. These models are called minimum cost a–b-matching problems.

Formulations for the Max_MCabMP and M_{card}_MCabMP are obtained by adding the following constraints to the corresponding formulations given in the previous subsections:

$$\sum_{\substack{j:(i,j)\in E\\ i<j}} x_{ij} + \sum_{\substack{j:(i,j)\in E\\ j<i}} x_{ji} \geq a, \forall i \in N,$$

and are denoted by (F^{ab}_{max}) and (F^{ab}_{card}), respectively.

5 SOC formulations for the MCbMP's with neighborhoods

Observe that all the formulations of the different versions of the MCbMP's given in the previous section can be written as follows:

$$\begin{aligned} \min & \sum_{\substack{(i,j)\in E\\ i<j}} ||z_i - z_j||_q x_{ij} - \sum_{\substack{(i,j)\in E\\ i<j}} M x_{ij} \\ & s.t.[MC] \\ & x_{ij} \in \{0,1\}, \quad \forall (i,j) \in E, i<j, \\ & z_i \in \mathcal{N}_i, \quad \forall i \in N, \end{aligned} \tag{12}$$

where [MC] is the set of Matching Constraints given in each formulation depending on the problem (perfect b-matching, maximum cardinality b-matching, maximal b-matching and a–b-matching) and $M := \max_{\substack{i,j\in N\\ z_i\in\mathcal{N}_i, z_j\in\mathcal{N}_j}} \{||z_i - z_j||_q\} + 1$, for the case of the maximum cardinality version, and 0 otherwise.

One of the main difficulty of the problem above lies in the term $||z_i - z_j||_q x_{ij}$ of the objective function (12). Let us define a new set of variables $d_{ij} \geq 0$ as $d_{ij} = ||z_i - z_j||_q x_{ij}$, $\forall (i,j) \in E$, $i<j$. Then, the formulations for the MCbMP's presented in the previous section can be reformulated as follows:

$$\begin{aligned} (\mathrm{F}_{general})\ \min & \sum_{\substack{(i,j)\in E\\ i<j}} d_{ij} - \sum_{\substack{(i,j)\in E\\ i<j}} M x_{ij} \\ & s.t.[MC] \\ & ||z_i - z_j||_q \leq d_{ij} + M_{ij}(1 - x_{ij}), \quad \forall (i,j) \in E, i<j, \end{aligned} \tag{13}$$

$$\begin{aligned} & d_{ij} \geq 0, \quad \forall (i,j) \in E, \\ & x_{ij} \in \{0,1\}, \quad \forall (i,j) \in E, i<j, \\ & z_i \in \mathcal{N}_i, \quad \forall i \in N, \end{aligned} \tag{14}$$

where $M_{ij} := \max_{z_i\in\mathcal{N}_i, z_j\in\mathcal{N}_j} \{||z_i - z_j||_q\}$. Although the reformulation has a linear objective function, constraints (13) are not linear since they are considering a ℓ_q-norm. However, this type of constraints have a representation as Second Order Cone (SOC) inequalities, see Blanco et al. [7]. Most of the state-of-the-art solvers are capable to efficiently solve optimization problems involving SOC constraints by means of quadratic constraints with positive definite matrices.

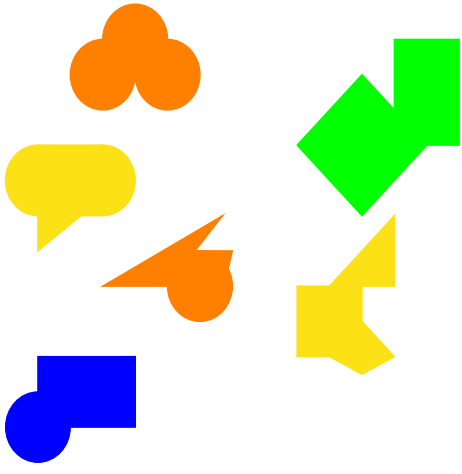
So far, we have focused our attention in the modelling aspects of the objective function and constraints to model the corresponding matching criterion under study. And, we have obtained a linear objective function and a set of constraints with a SOC representation. However, the characterization of the neighborhoods have not been addressed yet. Particularly, we will concentrate on neighborhoods given by the union of SOC-representable sets [26], more precisely, polyhedra and ℓ_q balls. Different transformations of ($F_{general}$) (one for each matching criterion) will be carried out to accommodate it to the use of second order cone optimization. The main reason for this is that it can be solved by interior point methods and, in general, they are more efficient that Semidefinite Programming (SDP) problems. For further details on the SOC constraints, see books Ben-Tal and Nemirovski [5], Luemberger and Ye [27].

5.1 SOC formulations for the MCbMP's with some non-convex neighborhoods

In this subsection, we consider the case where the neighborhoods belong to a family of non-convex sets given by a finite union of polyhedrons and ℓ_q-norm balls as Fig. 6. We will use results from disjunctive programming [4, 34] to formulate the MCbMP's with these neighborhoods.

Let us consider a set of neighborhoods $\mathcal{N}_i$, with $i \in N$, defined as $\mathcal{N}_i = \left(\bigcup_{j\in P^i} \mathcal{P}_j^i\right) \cup \left(\bigcup_{k\in D^i} \mathcal{D}_k^i\right)$ where $\mathcal{P}_j^i$ is a full-dimensional polyhedron given by $\mathcal{P}_j^i := \{x \in \mathbb{R}^m : A_j^i x \leq b_j^i\}$, whose recession cone is the null vector, with A_j^i a real matrix and $b_j^i \in \mathbb{R}$ for any $j \in P^i := \{1, \ldots, p^i\}$, and $\mathcal{D}_k^i := \{x \in \mathbb{R}^m : ||x - c_k^i||_q \leq r_k^i\}$ where $||\cdot||_q$ denotes the standard ℓ_q-norm with $q \in [1, \infty)$, $q \in \mathbb{Q}$, and $c_k^i \in \mathbb{R}^m$ and $r_k^i \in \mathbb{R}$ for any $k \in D^i := \{1, \ldots, d^i\}$. Observe that neighborhoods $\mathcal{N}_i$ are not convex in general.

Fig. 6 Different neighborhoods



Let z_i be a point in the neighborhood $\mathcal{N}_i$. Then z_i belongs to a polyhedron $\mathcal{P}^i_j$, for some $j \in P^i$, or a ℓ_q-norm ball $\mathcal{D}^i_k$, for some $k \in D^i$. We will use the following variables to distinguish points in the polyhedron or ℓ_q-norm ball.

- Continuous variables $u^i_j \in \mathbb{R}^m$ to represent a point in the polyhedron $\mathcal{P}^i_j$.
- Continuous variables $w^i_k \in \mathbb{R}^m$ to represent a point in the ℓ_q-norm ball $\mathcal{D}^i_k$.
- Binary variables t^i_j equal to 1 if z_i is in the polyhedron $\mathcal{P}^i_j$ and 0 otherwise.
- Binary variables s^i_k equal to 1 if z_i is in the ℓ_q-norm ball $\mathcal{D}^i_k$ and 0 otherwise.

The following proposition give a characterization of a point belonging to a neighborhood.

Proposition 5.1 *$z_i \in \mathcal{N}_i$ if and only if*

$$z_i = \sum_{j \in P^i} u^i_j + \sum_{k \in D^i} (w^i_k - c^i_k(1 - s^i_k)), \tag{15}$$

$$A^i_j u^i_j \leq b^i_j t^i_j, \quad j \in P^i, \tag{16}$$

$$||w^i_k - c^i_k||_q \leq r^i_k s^i_k, \quad k \in D^i, \tag{17}$$

$$\sum_{j \in P^i} t^i_j + \sum_{k \in D^k} s^i_k = 1, \tag{18}$$

$$t^i_j, s^i_k \in \{0, 1\}, \quad j \in P^i, k \in D^i. \tag{19}$$

Proof ($\Rightarrow$) Let $z_i \in \mathcal{N}_i$. Then, z_i belongs to a polyhedron $\mathcal{P}^i_{j_0}$, for some $j_0 \in P^i$, or a ℓ_q-norm ball $\mathcal{D}^i_{k_0}$, for some $k_0 \in D^i$ (if z_i belongs to the intersection of both, we only consider one of the two cases).

- if $z_i \in \mathcal{P}^i_{j_0}$, we consider $t^i_{j_0} = 1$, $t^i_j = 0$ and $u^i_j = \mathbf{0} \in \mathbb{R}^m$, $\forall j(\neq j_0) \in P^i$, $u^i_{j_0} = z_i$, $s^i_k = 0$ and $w^i_k = c^i_k$, $\forall k \in D^i$. Hence, (15)–(19) are satisfied.
- if $z_i \in \mathcal{D}^i_{k_0}$, we consider $s^i_{k_0} = 1$, $s^i_k = 0$ and $w^i_k = c^i_k$ $\forall k(\neq k_0) \in D^i$, $w^i_{k_0} = z_i$, $t^i_j = 0$ and $u^i_j = \mathbf{0} \in \mathbb{R}^m$, $\forall j \in P^i$. Hence, (15)–(19) are satisfied.

($\Leftarrow$) Let z_i a point satisfying (15)–(19). From (18) and (19), there exists either $j_0 \in P^i$ such that $t^i_{j_0} = 1$ and $t^i_j = 0$ $\forall j(\neq j_0) \in P^i$ and $s^i_k = 0$ $\forall k \in D^i$, or there exists $k_0 \in D^i$ such that $s^i_{k_0} = 1$ and $s^i_k = 0$, $\forall k(\neq k_0) \in D^i$ and $t^i_j = 0$, $\forall j \in P^i$.

In the first case, on the one hand, since $t^i_{j_0} = 1$ and $t^i_j = 0$ $\forall j(\neq j_0) \in P^i$ then, by (16), $u^i_{j_0} \in \mathcal{P}^i_{j_0}$ and $A^i_j u^i_j \leq 0$, $\forall j(\neq j_0) \in P^i$. Since any vector satisfying $A^i_j u^i_j \leq 0$, $\forall j(\neq j_0) \in P^i$ would be a vector of recession cone of $\mathcal{P}^i_j$ and, by hypothesis, this is

the null vector, then $u_j^i = \mathbf{0} \in \mathbb{R}^m$, $\forall j(\neq j_0) \in P^i$. Moreover, on the other hand, if $s_k^i = 0$ $\forall k \in D^i$, by (17), we have that $||w_k^i - c_k^i||_q = 0$, $\forall k \in D^i$, that is, $w_k^i = c_k^i$. Therefore, from (15), we obtain that $z_i = u_{j_0}^i \in \mathcal{P}_{j_0}^i$. That is, z_i belongs to the polyhedron $\mathcal{P}_{j_0}^i$.

Analogously in the second case, if we have that there exists $k_0 \in D^i$ such that $s_{k_0}^i = 1$ and $s_k^i = 0$, $\forall k(\neq k_0) \in D^i$ and $t_j^i = 0$, $\forall j \in P^i$, then we would obtain that $z_i = w_{k_0}^i \in \mathcal{D}_{k_0}^i$, that is, z_i belongs to the ℓ_q-norm ball $\mathcal{D}_{k_0}^i$. □

Therefore, (14) can be replaced by (15)–(19). Hence, since constraints (17) are SOC-representable, ($F_{general}$) is a mixed integer SOC-formulation when the neighborhoods are defined by a finite union of polyhedrons and ℓ_q-norm balls, and the distances are also measured with a ℓ_q-norm. We could even consider neighborhoods with different ℓ_q-norms or distances between neighborhoods using simultaneously different ℓ_q -norms, with $q \in [1, \infty)$.

6 Procedure for generating an initial solution

In this section we describe a procedure to generate an initial solution for solving the MCbMP's with neighborhoods.

This procedure starts by considering a fixed point in each neighborhood, $v_i \in \mathcal{N}_i$. Now, $V = \{v_1, \ldots, v_n\}$. The distances $\delta_{ij} = ||v_i - v_j||_q$ between these points are calculated. Afterwards, we solve the standard version (without neighborhoods) of the MCbMP's by considering these points:

$$
\begin{aligned}
(F_{point}) \min \; & \sum_{\substack{(i,j)\in E \\ i<j}} \delta_{ij} - \sum_{\substack{(i,j)\in E \\ i<j}} M x_{ij} \\
s.t. \quad & [MC] \\
& x_{ij} \in \{0,1\}, \quad \forall (i,j) \in E, i < j,
\end{aligned}
$$

where $M := \max_{v_i, v_j \in V}\{||v_i - v_j||_q\} + 1$. Notice that in the case of the Max_MCbMP, this model include binary y-variables.

Let $\bar{x}_v$ (and $\bar{y}_v$ for ($F2_{max}$)) the optimal solution of the above problem. This is a partial feasible solution of the general formulation with neighborhoods ($F_{general}$). We provide these values of the x-variables (y-variables) as feasible starting solution for the ($F_{general}$) model.

The goodness of the initial solution generated by the procedure will depend on the points chosen in each neighborhood. Figure 1 showed that the optimal matching of the problems with/without neighborhoods is not the same and depends on the choice of the points. This procedure is tested in the next section.

7 Computational study

A computational experiment was carried out to assess the usefulness of the given formulations for solving the proposed MCbMP's with neighborhoods. Tests are performed on instances with n neighborhoods defined in $\mathbb{R}^m$ (with $m = 2$ or 3). We consider two kind of neighborhoods: convex sets (ℓ_2-norm unit balls and polyhedra) and non-convex sets defined as the union of three of these convex sets. We have considered the number of neighborhoods $n \in [10, 26]$, $b \in [1, 5]$ and the Euclidean norm to measure the distances. Graphs with different settings have been considered. The expected edge density of the graph (measured as $\frac{2|E|}{n\cdot(n-1)}$, see Taşkin and Ekim [35]) takes values 0.4, 0.6, 0.8 and 1 (complete graph).

All the formulations were coded in Python 3.8, and solved using Gurobi 9.1.2. [23] in a Windows 10 Server with an Intel Xeon W-2245 processor at 3.9 GHz and 256 MB of RAM and setting by default all cuts and presolve strategies of the solver. A limit of one hour of CPU time was set in all the experiments.

The procedure described in Sect. 6 for getting an initial feasible solution is tested. The results obtained by using the solution procedure are indicated with *Proc*.

7.1 Computational results with non-convex neighborhoods

First, different tests are performed on non-convex neighborhoods. These neighborhoods were generated by considering the union of three convex sets of the following:

- In the case of the plane: ℓ_2-norm disks with different radii randomly generated, and rectangles with sides parallel to the axis and lengths randomly generated.
- In the 3D-space, ℓ_2-norm balls with different radii randomly generated and polyhedra whose faces are rectangles randomly generated.

The neighborhoods were generated taking into account different factors such as size, overlapping, and spatial distribution among them. First, n random points in $[0, 100]^m$ are generated. Each point will belong to one of the neighborhoods and will be in at least one of the three sets of this neighborhood. The minimum distance among the randomly generated points is calculated (ℓ), and a fixed percentage f (factor) of ℓ is calculated. The diameters of ℓ_2-norm balls and the sides of the rectangles will have a randomly generated length in $(0, f \cdot \ell)$.

Similar procedures to generate instances have been used in the literature of the combinatorial optimization problems with neighborhoods (see Blanco et al. [8], Gentilini et al. [20] among others).

A preliminary study with $n = \{12, 14, 16, 18, 20\}$ and $b = \{1, 2, 3, 4, 5\}$ for a complete graph in the plane was carried out. Five different instances were generated for each combination of n and b-values. Figure 7 displays the percentage of instances solved (within a given time) for the different formulations proposed here with the procedure (solid line) and without procedure (dashed line). As can be seen, this percentage increases when the solution procedure is considered for

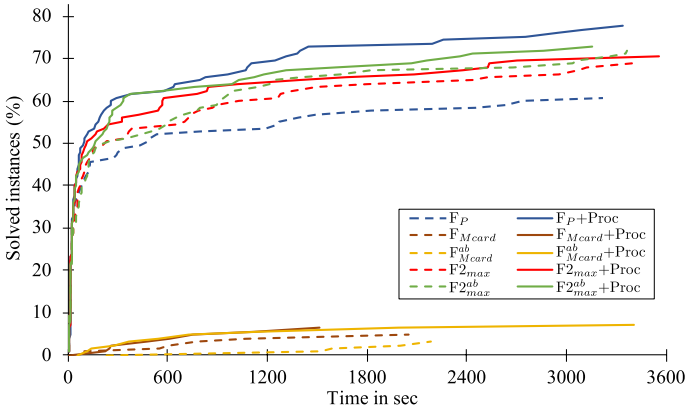


Fig. 7 Percentage of instances solved within given time for all the formulations in a complete graph

all the MCbMP's studied. In the case of the P_MCbMP (blue line) this percentage increases up to 20%. The hardest problem to be solved is M_{card}_MCbMP (brown and yellow lines) where less than 10% of instances were solved to optimality after 1 hour. We think it is due to the big-*M* parameter defined in the objective function. Moreover, the behaviour of ($F2_{max}$) and ($F2^{ab}_{max}$) are quite similar with both formulations.

In this figure does not appear formulation ($F1_{max}$) because provides worse computational results than ($F2_{max}$) for solving Max_MCbMP (see Fig. 8, which displays, in logarithmic scale, the average of the times for different values of *n* and *b* of both formulations).

As consequence of this first experiment, we decided focus our computational study on formulations solving the largest percentage of instances within the time limit. Since the behaviour of ($F2_{max}$) and ($F2^{ab}_{max}$) are quite similar, we concentrate on (F_P) and ($F2_{max}$) providing an initial solution. The CPU time to obtain an initial solution using the procedure in Sect. 6 is around 0.1–0.2 seconds on most instances.

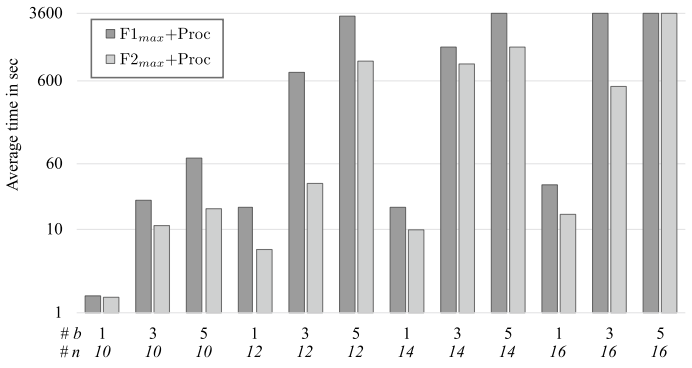


Fig. 8 Average time (logarithmic scale) for ($F1_{max}$) and ($F2_{max}$)

Table 2 Average results for (F_P) in complete graphs

n	b	CPU	#	G_{BS}
12	1	9.3	5/5	
	2	13.2	5/5	
	3	80.9	5/5	
	4	905.8	5/5	
	5	1959.4	3/5	32.5
14	1	12.7	5/5	
	2	16.1	5/5	
	3	155.0	5/5	
	4	807.2	4/5	2.0
	5	948.7	4/5	4.0
16	1	16.2	5/5	
	2	35.3	5/5	
	3	86.3	5/5	
	4	2187.2	4/5	4.0
	5	2587.6	2/5	3.0
18	1	18.0	5/5	
	2	75.6	5/5	
	3	845.0	5/5	
	4	2431.9	3/5	2.0
	5	3106.2	2/5	4.0
20	1	26.6	5/5	
	2	454.9	5/5	
	3	2326.8	2/5	5.0
	4	1H	0/5	4.8
	5	1H	0/5	5.4

Table 2 summarizes the results for (F_P). We concentrated on complete graphs (density equal to one) due to the number of unfeasible problems obtained when graphs of other densities were used. The first two columns are the values of n and b, respectively. The column CPU reports the average computing time (in seconds) to attain optimality. Notice that, here, a CPU time equal to 1H has been attributed for the instances that could not be solved within this limit, therefore times are underestimated. Column # gives the number of solved instances. Whenever the time limit of 1 hour is reached without certifying optimality, a gap between the lower and upper objective bound is calculated. Thus, if zu is the upper bound (the best solution found) and zl is the lower bound, then gap is defined as $100\,|zu - zl|/|zu|$. When $zu = 0$ and $zl \neq 0$, the gap is defined to be infinity. Column G_{BS} reports the average percentage gap of those instances where the gap is not infinity (a superscript indicates the number of these instances).

The first observation that comes after running the experiment is that the computational times increase with the size of n and b. Moreover, the number of unsolved instances increases with b.

Table 3 Average results for ($F2_{max}$)

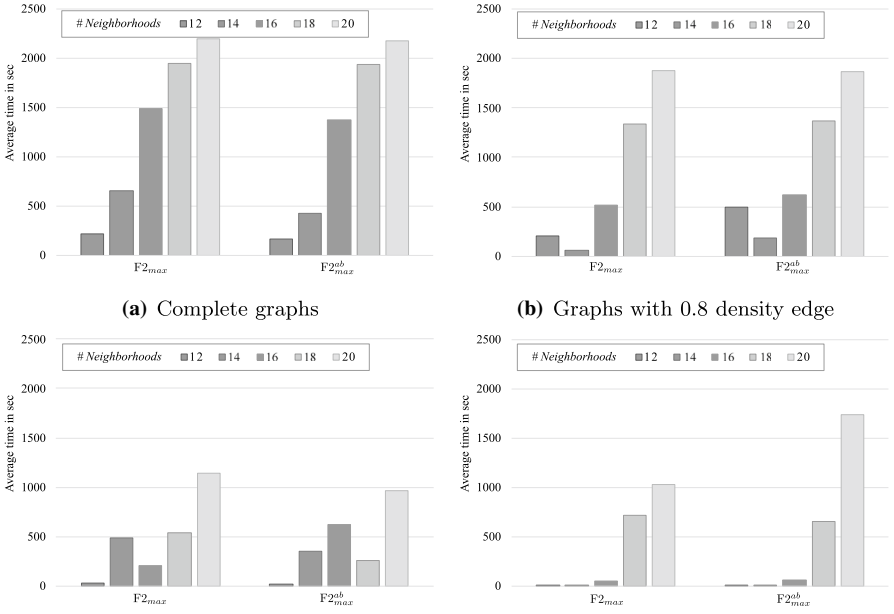
		Complete graph			Density=0.8			Density=0.6			Density=0.4		
n	b	CPU	#	G_{BS}	CPU	#	G_{BS}	CPU	#	G_{BS}	CPU	#	G_{BS}
12	1	5.7	5/5		2.0	5/5		1.1	5/5		2.0	5/5	
	2	14.0	5/5		11.8	5/5		17.6	5/5		3.9	5/5	
	3	34.6	5/5		17.6	5/5		12.2	5/5		5.0	5/5	
	4	40.4	5/5		30.6	5/5		77.6	5/5		6.3	5/5	
	5	992.0	4/5	1.8	938.2	4/5	35.4	22.2	5/5		6.7	5/5	
14	1	9.88	5/5		8.3	5/5		6.4	5/5		1.9	5/5	
	2	22.4	5/5		15.6	5/5		12.7	5/5		5.8	5/5	
	3	914.5	5/5		30.5	5/5		18.5	5/5		11.0	5/5	
	4	1029	4/5	5.9	233.2	5/5		1298.4	4/5	17.2	12.2	5/5	
	5	1460	4/5	3.9	258.4	5/5		1126.4	4/5	29.0	6.8	5/5	
16	1	14.7	5/5		8.9	5/5		10.9	5/5		4.3	5/5	
	2	36.3	5/5		24.9	5/5		18.5	5/5		15.3	5/5	
	3	493.3	5/5		47.2	5/5		36.3	5/5		139.2	5/5	
	4	3297.0	1/5	3.7	860.9	5/5		84.7	5/5		75.9	5/5	
	5	1H	0/5	5.0	1608.0	3/5	4.8	899.7	4/5	43.3	21.9	5/5	
18	1	16.9	5/5		13.9	5/5		11.7	5/5		8.0	5/5	
	2	95.6	5/5		35.1	5/5		25.9	5/5		23.8	5/5	
	3	2067.0	5/5		592.5	5/5		97.1	5/5		437.1	5/5	
	4	1H	0/5	4.5	2406.9	2/5	3.6	904.3	5/5		1625.3	3/5	14.3
	5	1H	0/5	7.1	2855.2	2/5	3.2	1671.0	3/5	23.5	1475.4	3/5	8.5
20	1	23.4	5/5		18.9	5/5		15.2	5/5		8.8	5/5	
	2	796.9	5/5		99.6	5/5		33.6	5/5		19.2	5/5	
	3	2929.0	1/5	6.4	1829.4	3/5	6.0	842.7	5/5		37.0	5/5	
	4	1H	0/5	9.2	1H	0/5	7.7	1308.6	4/5	6.0	2837.6	2/5	42.5
	5	1H	0/5	11.2	1H	0/5	7.3	3372.1	2/5	4.1	2186.4	2/5	35.0

Table 3 shows the results for ($F2_{max}$) with graphs of different densities. In all the experiments the valid inequalities (11) have been included (their inclusion provided better computational results in a preliminary study). As it can be seen, the computational times increase with the density of the graph. Solving instances in complete graphs is harder than solving instances in graphs with lower densities. This can be better appreciated in Figs. 9 and 10, which display the averages of the CPU times for different values of n and b, respectively, depending on the density of the graph. In addition, an extension of the experiment was carried out to evaluate up to what size of instances we could solve with ($F2_{max}$). Table 4 shows that even for lower densities, with $n \geq 22$, and $b \geq 3$, the instances are hard to solve. Moreover, they show the behaviour of ($F2^{ab}_{max}$) with $a = 1$ (we proved different values of a obtaining similar results).

Another computational study was still carried out to test three-dimensional instances with (F_P) and ($F2_{max}$) in complete graphs (see Table 5). We observe that

Table 4 Average results for ($F2_{max}$) with higher n

		Density = 0.6			Density = 0.4		
n	b	CPU	#	G_{BS}	CPU	#	G_{BS}
22	1	22.4	5/5		16.1	5/5	
	2	1696.4	4/5	6.0	59.7	5/5	
	3	1H	0/5	15.0	2091.8	3/5	4.5
24	1	31.4	5/5		21.9	5/5	
	2	2470.4	2/5	8.0	528.9	5/5	
	3	1H	0/5	20.0	2959.4	1/5	7.5
26	1	50.0	5/5		25.4	5/5	
	2	1H	0/5	14.4	330.2	5/5	
	3	1H	0/5	24.2	1H	0/5	14.0



(a) Complete graphs

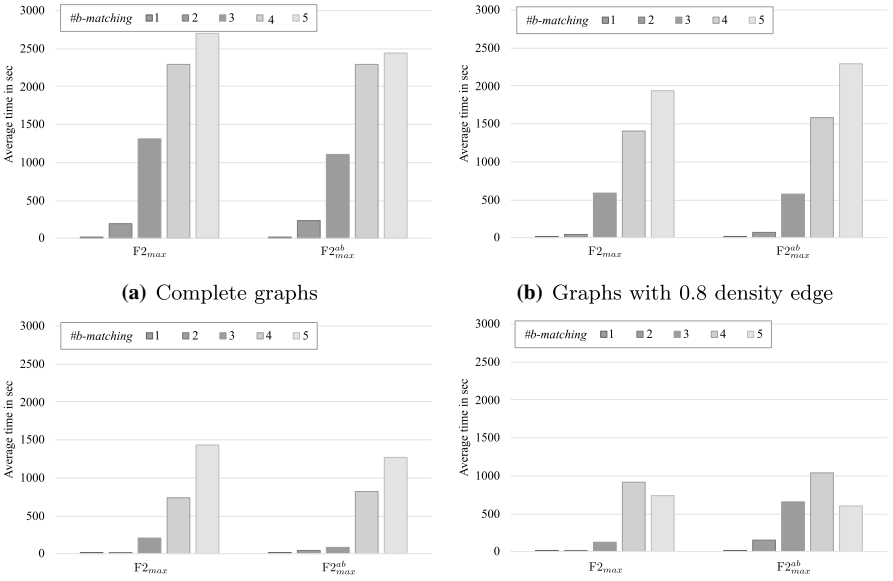
(b) Graphs with 0.8 density edge

(c) Graphs with 0.6 density edge

(d) Graphs with 0.4 density edge

Fig. 9 Average time by number of neighborhoods

three-dimensional instances are harder to solve than two-dimensional ones, given the same number of neighborhoods, n. See $n = 18$ and $b = 3$. the case of (F_P), the average time for solving 2D-instances was 845 seconds, whereas three of the 3D-instances could not be solved in one hour for the same size. For ($F2_{max}$), the average time of 2D-instances tested was 2067 seconds, whereas none of the five 3D-instances could be solved in one hour.



(a) Complete graphs

(b) Graphs with 0.8 density edge

(c) Graphs with 0.6 density edge

(d) Graphs with 0.4 density edge

Fig. 10 Average time by size of *b*

Table 5 Average results for (F_P) and ($F2_{max}$) in complete graphs with 3D-instances

		F_P			$F2_{max}$		
n	b	CPU	#	G_{BS}	CPU	#	G_{BS}
12	1	23.0	5/5		23.1	5/5	
	3	829.3	4/5	–	122.7	5/5	
	5	1H	0/5	$^{(3)}5.5$	3517.8	1/5	$^{(1)}6.1$
14	1	26.8	5/5		28.6	5/5	
	3	1660.3	3/5	3.5	946.9	5/5	
	5	1H	0/5	$^{(4)}10.0$	1H	0/5	$^{(1)}9.5$
16	1	32.5	5/5		37.8	5/5	
	3	1H	0/5	$^{(2)}8.3$	3196.9	1/5	$^{(2)}5.0$
	5	1H	0/5	$^{(3)}10.0$	1H	0/5	9.7
18	1	47.8	5/5		45.8	5/5	
	3	3158.8	2/5	9.3	1H	0/5	11.5
	5	1H	0/5	$^{(4)}6.0$	1H	0/5	$^{(2)}14.7$
20	1	55.7	5/5		53.4	5/5	
	3	1H	0/5	12.1	1H	0/5	14.3
	5	1H	0/5	13.4	1H	0/5	$^{(1)}21.3$

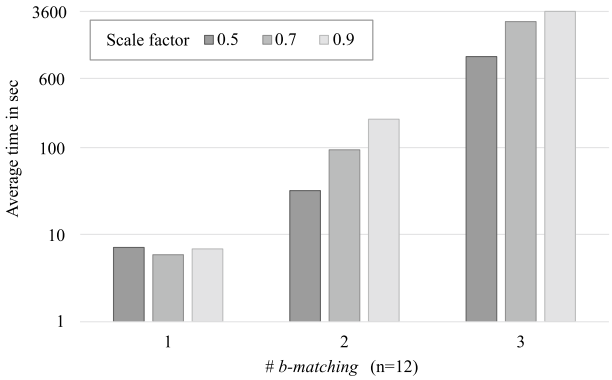


Fig. 11 Time average (logarithmic scale) for ($F2_{max}$) neighborhoods (n = 12) with different scale factors

We have noticed by testing different instances that overlapping does not influence the overall CPU time. Nevertheless, the size of the neighborhoods may influence the difference in CPU time. In order to illustrate the effect of the size of the neighborhoods, Fig. 11 shows the averaged time (logarithmic scale) for ($F2_{max}$) with $n = 12$ with different scale factors (0.5, 0.7 and 0.9). This confirms that instances with larger neighborhoods are harder to solve than instances with smaller ones.

7.2 Computational results with convex neighborhoods

A second study was carried out to compare the performance of (F_P) and ($F2_{max}$) with convex neighborhoods. We concentrate on the plane and consider two kind of instances: all the neighborhoods are ℓ_2-norm disks or all are squares (see Blanco et al. [8], Blanco [6] for similar instances). First, n random points in $[0, 100]^m$ are generated. In the first set of instances, these points will be the centers for disks. In the second set of instances, they will be the centers of the squares. With the same randomly generated points, the two set of instances are obtained. The radius of the disks is 4 units, and the side length of the squares is 6 units. Five different set of points were generated for each combination of $n = \{12, 14, 16, 18, 20, 22, 24, 26\}$ and $b = \{1, 3, 5\}$.

The average results with both set of instances are given in Table 6. The results show that solving instances with ℓ_2-norm disks is much harder than with squares. See $n = 20$ and $b = 5$. (F_P) and ($F2_{max}$) did not solve to optimality any of the instances with disks in 1 hour. However, when the instances are squares, the time is 10.2 for (F_P) and 64.7 for ($F2_{max}$). This is due the fact that when neighborhoods are squares, the constraints are all linear.

8 Concluding remarks

This work can be considered as another step forward to address combinatorial optimization problems on graphs with neighborhoods instead of nodes. Different versions of the *b*-matching problem with minimum cost have been studied: the perfect

Table 6 Average results for (F_P) and ($F2_{max}$) in complete graphs with disks vs. squares neighborhoods

		F_P						$F2_{max}$					
		Disks			Squares			Disks			Squares		
n	b	CPU	#	G_{BS}	CPU	#	G_{BS}	CPU	#	G_{BS}	CPU	#	G_{BS}
12	1	2.1	5/5		0.1	5/5		1.4	5/5		0.1	5/5	
	3	8.8	5/5		0.3	5/5		23.6	5/5		0.9	5/5	
	5	63.6	5/5		0.3	5/5		78.1	5/5		1.6	5/5	
14	1	3.9	5/5		0.2	5/5		2.1	5/5		0.2	5/5	
	3	39.9	5/5		0.6	5/5		380.3	5/5		2.2	5/5	
	5	2630.2	2/5	3.0	1.7	5/5		749.7	5/5		3.0	5/5	
16	1	6.1	5/5		0.2	5/5		3.2	5/5		0.2	5/5	
	3	35.0	5/5		1.0	5/5		122.0	5/5		3.9	5/5	
	5	2745.6	2/5	1.3	2.7	5/5		2476.2	5/5		7.5	5/5	
18	1	9.2	5/5		0.4	5/5		5.5	5/5		0.5	5/5	
	3	480.7	5/5		2.9	5/5		1850.4	4/5	8.6	13.0	5/5	
	5	2947.3	1/5	3.0	5.1	5/5		1H	0/5	4.9	37.0	5/5	
20	1	5.0	5/5		0.4	5/5		5.3	5/5		0.5	5/5	
	3	999.5	4/5	3.0	2.9	5/5		779.9	5/5		8.7	5/5	
	5	1H	0/5	3.0	10.2	5/5		1H	0/5	5.1	64.7	5/5	
22	1	9.8	5/5		0.4	5/5		12.7	5/5		0.5	5/5	
	3	2527.9	3/5	3.5	4.3	5/5		1H	0/5	9.6	17.7	5/5	
	5	1H	0/5	3.6	11.5	5/5		1H	0/5	12.8	303.1	5/5	
24	1	13.4	5/5		0.4	5/5		9.1	5/5		0.6	5/5	
	3	1H	0/5	4.4	6.6	5/5		1H	0/5	11.4	59.7	5/5	
	5	1H	0/5	(1)16.8	52.7	5/5		1H	0/5	16.5	1346.6	4/5	0.6
26	1	30.9	5/5		0.5	5/5		23.0	5/5		0.9	5/5	
	3	1H	0/5	5.4	13.1	5/5		1H	0/5	16.9	197.0	5/5	
	5	1H	0/5	(3)31.5	27.7	5/5		1H	0/5	15.1	2421.8	2/5	0.9

matching problem, the maximum cardinality matching problem, the maximal matching problem and the *a*–*b*-matching problem. The property of the standard matching problems (without neighborhoods) of being solvable in polynomial-time, is not guaranteed with neighborhoods. We have proved that, in general, all the versions of the *b*-matching with non-convex neighborhoods studied here are NP-hard, even for non-convex polygons.

Different non-linear integer programming formulations for the minimum cost *b*-matching problems with all the versions presented in this paper have been developed and then reformulated as mixed integer SOC formulations. We have performed a series of computational experiments in order to compare the performance of the given formulations, as well as to explore the limitations of each one of them. For this, we have generated several batteries of instances with different settings. The extensive computational experience shows the usefulness of the formulations

to solve the proposed problems. We have developed an initial solution procedure that allows to reduce the computational times and solve to optimality some initially unsolved instances.

The findings of this paper can be the basis of further research on some other combinatorial optimization problems with neighborhoods. It would be interesting to explore alternative algorithms for solving the MCbMP's, as Benders decomposition methods or good matheuristics to be able to solve larger instances.

Acknowledgements This research has been partially supported by the Spanish Ministerio de Ciencia y Tecnologia, Agencia Estatal de Investigacion and Fondos Europeos de Desarrollo Regional (FEDER) via project PID2020-114594GB-{C21,C22}, FEDER-US-1256951, FEDER-UCA18-106895, Junta de Andalucía P18-FR-1422, CEI-3-FQM331, FQM-331, FQM-355 and NetmeetData: Ayudas Fundación BBVA a equipos de investigación científica 2019.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability The data that support the findings of this study are available from the corresponding author upon request.

Declarations

Conflict of interests The authors declare that they have no conflict of interest.

References

1. Ahadi, A., Mozafari, A., Zarei, A.: Touring disjoint polygons problem is NP-hard. In: Widmayer, P., Xu, Y., Zhu, B. (eds.) COCOA 2013. Lecture Notes in Computer Science, vol. 8287, pp. 351–360. Springer, Cham (2013)
2. Anstee, R.P.: A polynomial algorithm for *b*-matchings, an alternative approach. Inf. Process. Lett. **24**, 153–157 (1987)
3. Arkin, E.M., Hassin, R.: Approximation algorithms for the geometric covering salesman problem. Discrete Appl. Math. **55**(3), 197–218 (1994)
4. Balas, E.: Disjunctive programming. Ann. Discrete Math. **5**, 3–51 (1979)
5. Ben-Tal, A., Nemirovski, A.: Lectures on modern convex optimization. In: Analysis, algorithms and engineering applications. SIAM Series (2001)
6. Blanco, V.: Ordered p-median problems with neighborhoods. Comput. Optim. Appl. **73**, 603–645 (2019)
7. Blanco, V., Puerto, J., El-Haj Ben-Ali, S.: Revisiting several problems and algorithms in continuous location with ℓ_τ norms. Comput. Optim. Appl. **58**(3), 563–595 (2014)
8. Blanco, V., Fernández, E., Puerto, J.: Minimum spanning trees with neighborhoods: mathematical programming formulations and solution methods. Eur. J. Oper. Res. **262**(3), 863–878 (2017)

9. Bodur, M., Ekim, T., Taskin, Z.C.: Decomposition algorithms for solving the minimum weight maximal matching problem. Networks **62**, 273–287 (2013)
10. Cohen, M.B., Madry, A., Sankowski, P., Vladu, A.: Negative-weight shortest paths and unit capacity minimum cost flow in $O(m^{10/7}logW)$ Time. In: Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, pp. 752–771 (2017)
11. Cunningham, W.H., Marsh, A.B.: A primal algorithm for optimum matching. In: Balinski, M.L., Hoffman, A.J. (eds.) Polyhedral Combinatorics. Mathematical Programming Studies, vol. 8, pp. 50–72. Springer, Berlin, Heidelberg (1978)
12. De Berg, M., Gudmundsson, J., Katz, M.J., Levcopoulus, C., Overmars, M.H., Van der Stappen, A.F.: TSP with neighborhoods of varying size. J. Algorithms **57**(1), 22–36 (2005)
13. Dong, L., Kang, X., Pan, M., Zhao, M., Zhang, F., Yao, H.: B-matching-based optimization model for energy allocation in sea surface monitoring. Energy **192**, 116618 (2020)
14. Dorrigiv, R., Fraser, R., He, M., Kamali, S., Kawamura, A., López-Ortiz, A., Seco, D.: On minimum-and maximum-weight minimum spanning trees with neighborhoods. Theory Comput. Syst. **56**(1), 220–250 (2015)
15. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: Conference Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 473–482 (2003)
16. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. J. Algorithms **48**(1), 135–159 (2003)
17. Edmonds, J.: Paths, trees, and flowers. Can. J. Math. **17**(3), 449–467 (1965)
18. Emek, Y., Kutten, S., Shalom, M., Zaks, S.: Hierarchical b-Matching. In: Bureš, T., et al. (eds.) Theory and Practice of Computer Science. SOFEM 2021. Lecture Notes in Computer Science, vol. 12607, pp. 189–202. Springer, Cham (2021)
19. Gabow, H.N.: Data structures for weighted matching and extensions to b-matching and f-factors. ACM Trans. Algorithms (TALG) **14**(3), 1–80 (2018)
20. Gentilini, I., Margot, F., Shimada, K.: The travelling salesman problem with neighbourhoods: MINLP solution. Optim. Methods Softw. **28**(2), 364–378 (2013)
21. Gerards, A.M.H.: Matching. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) Handbooks in Operations Research and Management Science. Network Models, vol. 7, pp. 135–224. Elsevier, North-Holland (1995)
22. Gudmundsson, J., Levcopoulos, C.: A fast approximation algorithm for TSP with neighborhoods. Nord. J. Comput. **6**(4), 473–482 (1999)
23. Gurobi Optimization Inc. Gurobi Optimizer Reference Manual. http://www.gurobi.com (2021) Accessed 7 November 2021
24. Han, Z., Gu, Y., Saad, W.: Matching Theory for Wireless Networks. Wireless Networks Series. Springer, Cham (2017)
25. Lovasz, L., Plummer, M.D.: Matching Theory. Annals of Discrete Mathematics, vol. 29. North Holland (1986)
26. Lobo, M., Vandenberghe, L., Boyd, S., Lebret, H.: Applications of second-order cone programming. Linear Algebra Appl. **284**, 193–228 (1998)
27. Luemberger, D.G., Ye, Y.: Linear and Nonlinear Programming. International Series in Operations Research and Management Science. Springer, New York (2008)
28. Marcotte, O., Suri, S.: Fast matching algorithms for points on a polygon. SIAM J. Comput. **20**(3), 405–422 (1991)
29. Miller, D.L.: A matching based exact algorithm for capacitated vehicle routing problems. ORSA J. Comput. **7**(1), 1–9 (1995)
30. Millman National Land Services: What Is a Cell Tower and How Does a Cell Tower Work? https://millmanland.com/company-news/what-is-a-cell-tower-and-how-does-a-cell-tower-work (2021). Accessed 7 November 2021
31. Puerto, J., Valverde, C.: Routing for unmanned aerial vehicles, touring dimensional sets. Eur. J. Oper. Res. (2021). https://doi.org/10.1016/j.ejor.2021.06.061
32. Pulleyblank, W.R.: Matchings and extensions. In: Graham, R.L., Grötschel, M., Lovász, L. (eds.) Handbook of Combinatorics, vol. 1, pp. 179–232. Elsevier, Amsterdam (1995)
33. Schenk, O., Wächter, A., Hagemann, M.: Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. Comput. Optim. Appl. **36**, 321–341 (2007)

34. Sherali, H.D., Shetty, C.M.: Optimization with disjunctive constraints. Lecture Notes in Economics and Mathematical Systems, vol. 181. Springer, Heidelberg (1980)
35. Taşkin, Z.C., Ekim, T.: Integer programming formulations for the minimum weighted maximal matching problem. Optim. Lett. **6**, 1161–1171 (2012)
36. Tennenholtz, M.: Tractable combinatorial auctions and b-matching. Artif. Intell. **140**(1–2), 231–243 (2002)
37. Tural, M.K.: Maximal matching polytope in trees. Optim. Method. Softw. **31**(3), 471–478 (2016)
38. Vaidya, P.M.: Geometry helps in matching. SIAM J. Comput. **18**, 1201–1225 (1989)
39. Yang, Y., Lin, M., Xu, J., Xie, Y.: Minimum spanning tree with neighborhoods. In: Kao, M.Y., Li, X.Y. (eds.) Algorithmic Aspects in Information and Management. AAIM 2007. Lecture Notes in Computer Science, vol. 4508. Springer, Berlin (2007)