Integer Programming Models for the Multidimensional Assignment Problem with Star Costs

Jose L. Walteros[a,], Chrysafis Vogiatzis[a], Eduardo L. Pasiliao[b], Panos M. Pardalos[a]

[a]Department of Industrial and Systems Engineering
University of Florida
josewalt@buffalo.edu, chvogiat@ufl.edu, pardalos@ufl.edu

[b]AFRL Munitions Directorate
Air Force Research Laboratory
pasiliao@eglin.af.mil

Abstract

We consider a variant of the multidimensional assignment problem (MAP) with decomposable costs in which the resulting optimal assignment is described as a set of disjoint stars. This problem arises in the context of multi–sensor multi–target tracking problems, where a set of measurements, obtained from a collection of sensors, must be associated to a set of different targets. To solve this problem we study two different formulations. First, we introduce a continuous nonlinear program and its linearization, along with additional valid inequalities that improve the lower bounds. Second, we state the standard MAP formulation as a set partitioning problem, and solve it via branch and price. These approaches were put to test by solving instances ranging from tripartite to 20–partite graphs of 4 to 30 nodes per partition. Computational results show that our approaches are a viable option to solve this problem. A comparative study is presented.

Keywords: Combinatorial optimization, multidimensional assignment problem, star covering, multi–sensor multi–target tracking problem, graph partitioning, branch and price.

1. Introduction

The multidimensional assignment problem (MAP), originally introduced by Pierskalla (1968), aims to minimize the overall cost of assignment when matching elements from $\mathcal{N} = \{N_1, \ldots, N_n\}$ ($n > 2$) disjoint sets of equal size m. It comes as a natural generalization of the two-dimensional Assignment Problem (AP), known to be polynomially solvable (Kuhn (1955); Edmonds and Karp (1972)). Among all the different generalizations of the MAP, the one considered in this paper is the *axial* MAP (hereafter referred to as MAP). In an axial MAP, each element of every set must be assigned to exactly one of m disjoint n-tuples, and each n-tuple must contain exactly one element of each set. Contrary to the AP, the MAP is known to be NP-hard (Karp (2010)) for all values of $n > 2$.

The MAP is usually presented as the following integer (0-1) program

$$(MAP): \min \quad \sum_{i_1 \in N_1} \sum_{i_2 = N_2} \cdots \sum_{i_n \in N_n} c_{i_1 i_2 \ldots i_n} x_{i_1 i_2 \ldots i_n} \tag{1}$$

$$s.t. \quad \sum_{i_2 \in N_2} \sum_{i_3 \in N_3} \cdots \sum_{i_n \in N_n} x_{i_1 i_2 \ldots i_n} = 1, \qquad i_1 \in N_1 \tag{2}$$

$$\sum_{i_1 \in N_1} \cdots \sum_{i_{s-1} \in N_{s-1}} \sum_{i_{s+1} \in N_{s+1}} \cdots \sum_{i_n \in N_n} x_{i_1 i_2 \ldots i_n} = 1, \quad i_s \in N_s, \quad s = 2, \ldots, n-1 \tag{3}$$

$$\sum_{i_1 \in N_1} \sum_{i_2 \in N_2} \cdots \sum_{i_{n-1} \in N_{n-1}} x_{i_1 i_2 \ldots i_n} = 1, \qquad i_n \in N_n \tag{4}$$

$$x_{i_1 i_2 \ldots i_n} \in \{0, 1\}, \qquad i_s \in N_s, \quad s = 1 \ldots n, \tag{5}$$

where, for every n-tuple $(i_1, i_2, \ldots, i_n) \in N_1 \times N_2 \times \cdots \times N_n$, variable $x_{i_1 i_2 \ldots i_n}$ takes the value of one if elements of the given n-tuple belong to the same assignment, and zero otherwise. The total assignment cost (1) is computed as the cost of matching elements from different sets together. As an example, an assignment which selects elements $(i_1, i_2, \ldots, i_n)$ to be grouped together would have a cost of $c_{i_1 i_2 \ldots i_n}$.

Depending on the definition of the assignment costs, there are several variations of the MAP that can be considered. These variations are mainly associated with cases where the assignment cost of each n-tuple can be decomposed as a function of all possible pairwise assignment costs between elements of different sets. That is, $c_{i_1 i_2 \ldots i_n} = f(c_{i_1 i_2}, \ldots, c_{i_n i_{n-1}})$, where $f : N_1 \times N_2 \cup \cdots \cup N_{n-1} \times N_n \to \mathbb{R}$ and $c_{i_s i_t}$ is the cost of assigning together elements $i_s \in N_s$ and $i_t \in N_t$, for $s \neq t$. In general, the main advantage of having decomposable cost functions is that there may be ways of tackling the problem without having to completely enumerate all of the different assignment costs, which can be exponentially many. Moreover, most of these MAP variations can be associated with a weighted n-partite graph, in which the elements are represented by the vertices of the graph, each of the edges describes the decision of assigning two elements within the same n-tuple, and the weights on the edges account for the corresponding assignment costs. We provide a detailed explanation of this representation in Section 2.

Based on the applications and the context of the problem, there are different definitions of the MAP with decomposable costs that can be found in the literature (Aneja and Punnen (1999); Bandelt et al. (1994); Burkard et al. (1996); Crama and Spieksma (1992); Malhotra et al. (1985); Kuroki and Matsui (2009)). In this paper we consider the case where each n-tuple of any feasible assignment is assumed to form a star (see, (Bandelt et al. (1994)). Nonetheless, since most of the extant literature concentrates on the case where the n-tuples form cliques, we also provide a brief description of the latter to emphasize the differences and enrich the discussion.

For the case of the cliques, a feasible assignment includes all possible pairwise connections within the elements of each tuple and thus, the cost of tuple $(i_1, i_2, \ldots, i_n) \in N_1 \times N_2 \times \cdots \times N_n$ is defined as the sum of all pairwise assignment costs. That is,

$$c_{i_1 i_2 \ldots i_n} = \sum_{s=1}^{n} \sum_{t=s+1}^{n} c_{i_s i_t} \tag{6}$$

On the other hand, for the case of the stars, one element of each tuple is assigned to be a center (or representative) and the other elements are considered to be the leafs (or legs) of the star. Note that, contrary to the case of the cliques, each tuple can generate many different star configurations, depending on which element is selected as the center. Assuming for example, that the center is element i_s, the cost of the induced star is the sum of the pairwise costs between i_s and the other elements of the tuple. In view of these multiple possible configurations, the cost of tuple $(i_1, i_2, \ldots, i_n) \in N_1 \times N_2 \times \cdots \times N_n$ is defined as the minimum cost among the costs of all the possible star configurations of the tuple. That is,

$$c_{i_1 i_2 \ldots i_n} = \min_{i_s \in \{i_1, i_2, \ldots, i_n\}} \left\{ \sum_{t \in \{1,2,\ldots,n\} \setminus \{s\}} c_{i_s i_t} \right\} \tag{7}$$

We name the aforementioned MAP version, the *multidimensional star assignment problem* (MSAP), because of the particular structure that each feasible assignment has. Despite the fact that this variant is often referred to as a particular case of the clique version (Bandelt et al. (1994)), we consider that it is relevant to state it in a separate form. We base our argument on the fact that there exist applications for which the use of this variant could be of benefit. Moreover, there are formulations and techniques specifically tailored to solve the MSAP.

Several methodologies have been proposed to solve different variants and generalizations of the MAP, including exact approaches, approximation algorithms, heuristics, and metaheuristics. In particular, given the inherent NP-hardness of the problem, heuristic approaches have gained practical interest over the years. These include greedy heuristics (Balas and Saltzman (1991)), generalized randomized adaptive search procedures (GRASP) (Murphey et al. (1999) and Robertson III (2001)), GRASP with path relinking (Aiex et al. (2005)), randomized algorithms (Oliveira and Pardalos (2004)), genetic algorithms (Gaofeng and Lim (2003)), memetic algorithms (Karapetyan and Gutin (2011b)), local search heuristics (Bandelt et al. (2004) and Karapetyan and Gutin (2011a)), simulated annealing (Clemons et al. (2004)), decomposition schemes (Vogiatzis et al. (2013)), Lagrangian based procedures (Balas and Saltzman (1991); Frieze and Yadegar (1981), and Poore and Robertson III (1997)), and branch-and-bound techniques (Larsen (2012), and Pasiliao et al. (2005)).

From the perspective of approximation algorithms, there exist the works of Crama and Spieksma (1992) and Bandelt et al. (1994). Furthermore, contributions to the study of the polyhedral structure of the MAP formulation and other generalizations can be found in Appa et al. (2006); Balas and Saltzman (1989); and Magos and Mourtos (2009). Finally, studies related to the asymptotic behavior of the expected optimal value of the MAP, as well as tools to perform probabilistic analysis of MAP instances are given in Krokhmal et al. (2007), Grundel et al. (2004), and Gutin and Karapetyan (2009).

Among all the proposed techniques listed above, we next focus our attention on approaches that are either proposed to tackle the MSAP, or that are designed to solve generalizations of the MAP, and thus can also be used to solve this problem. To solve the MSAP, Crama and Spieksma (1992) introduced an approximation algorithm designed to solve the three-dimensional case (i.e., $n = 3$). The proposed algorithm consists of sequentially solving two linear assignment problems. First, the elements of set N_1 are assigned to the ones of set N_2 and then, the resulting pairs are assigned to the elements of set N_3. The authors proved that if the pairwise assignment costs satisfy the triangle inequality, the proposed algorithm produces a $\frac{1}{2}$ approximation. Moreover, noting that this algorithm can produce three different solutions by simply varying the assignment order of the sets (e.g., assigning first the elements of N_1 to the ones of set N_3 and then, assigning the resulting pairs to the elements of set N_2), Crama and Spieksma proved that selecting the best of the three solutions yields a $\frac{1}{3}$ approximation.

In a subsequent study, Bandelt et al. (1994) proposed two type of heuristics, namely the hub and the recursive heuristics. They can be viewed as generalizations of the approach proposed by Crama and Spieksma (1992), but designed to solve the general n–dimensional case. The authors also provide an upper bound on the ratio between the cost of the solutions produced by these heuristics and the cost of the optimal solution.

As mentioned before, the MSAP is a particular case of the MAP and therefore, it can be solved using formulation (1)–(5). The polyhedral studies introduced by Balas and Saltzman (1989), for the three-dimensional case, and by Appa et al. (2006) and Magos and Mourtos (2009), for a more general version of the MAP, can be used to enhance (1)–(5) via the introduction of cutting planes. Using formulation (1)–(5) to solve the MSAP has one main drawback. It requires that all possible star costs be generated beforehand. This could be problematic because the total number of possible stars grows exponentially with the size of the problem (see Section 3). To circumvent this issue, it is possible to embed (1)–(5) within a branch–and–price scheme (see Section 3). Therefore, instead of enumerating all possible stars from the beginning, those are generated via column generation, in case they are considered suitable. The downside of this approach, though, is that mixing cutting planes and column generation is in general a difficult task (Barnhart et al. (1998); Desaulniers et al. (2011), and Lübbecke and Desrosiers (2005)).

For additional information about the MAP and its variations, we refer the reader to the surveys provided by Burkard et al. (1998); Burkard and Çela (1999); Burkard (2002); Gilbert and Hofstra (1988); Pardalos and Pitsoulis (2000); Pentico (2007), and Spieksma (2000)).

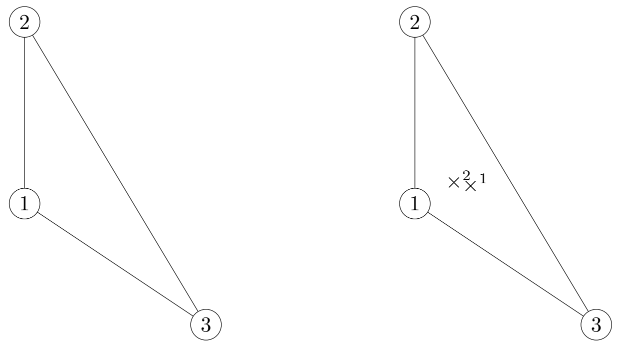
This paper is inspired by the context of multi–sensor multi–target tracking problems, that involve the assignment of a series of sensor observations into a set of different targets. The relationship between these problems and the MAP, has been stated and studied by many authors including Bandelt et al. (2004); Chummun et al. (2001); Deb et al. (1993, 1997); Morefield (1977); Murphey et al. (1999); Poore (1994), and Pusztaszeri et al. (1996) among others.

There are several contexts in which using star costs can prove beneficial when solving multi–sensor multi–target tracking problems. In particular, when the assignment costs have metric properties (i.e, nonnegativity, symmetry, and subadditivity), it is interesting to see that in some cases, considering all pairwise costs within the assignments (e.g., the clique case) is not necessary to obtain a valid solution. We provide two examples with costs that satisfy those properties and show that the star cost variation is a valuable tool to solve those problems. We would like to emphasize though, that the star costs are not always a better or worse option compared to other versions such as the clique costs. Instead, it can be seen as an alternative that also provides additional information, and that can contribute to a better analysis depending on the context.

In the first example we aim to identify a set of land mines that are planted on a field. To find the location of the mines, a drone is sent to fly over the field emitting a signal. Once the signal reaches each of the mines, it bounces back and is analyzed by the sensors of the drone. After the drone has flown over the field a number of times and its sensors have collected the set of different signals (several of those associated with each of the mines), it is possible to calculate a set of estimated locations where the mines could be located. The idea behind solving a MAP is to associate the locations that are close to each other, which would help pinpoint the actual positions of the mines. In this context, the assignment costs represent the Euclidean distances between the estimated locations and, since the costs satisfy the triangle inequality, not all the costs need to be considered to obtain a valid association.

Consider the simple case depicted in Figure 1a, where the nodes represent three estimated locations that where assigned together. Here, the center of the star is clearly node 1. Notice that, the cost of the legs (i.e., edges $(1, 2)$ and $(1, 3)$) already contains information about the cost of edge $(2, 3)$. In this example, since we expect that cost c_{23} satisfies $\max\{c_{12}, c_{13}\} \leq c_{23} \leq c_{12} + c_{13}$, some information about c_{23} is already considered within the value of $c_{12} + c_{13}$. In some situations, for instance in the presence of a faulty sensor, adding c_{23} could contribute to obtain a wrong assignment, especially when this value is close to its upper bound.

Moreover, the expected position of the mine is often considered to be a concurrency point associated with the positions that are in the same n-tuple, generally the centroid. On the other hand, if star costs are used, the region around the center of the stars can be seen as the zone where the mines are most likely to be located. Under the triangle inequality assumption, it is easy to prove that many of the triangle concurrency points (e.g., the centroid, the incenter, or the circumcenter) are always closer to the center of the star. For instance, observe the locations of the centroid ($\times^1$) and the incenter ($\times^2$) of Figure 1b (this can also be generalized for $n > 3$). This implies that, if the assignments (i.e., the n-tuples) produced by both the clique or the star versions are the same, then the conclusion regarding the positions of the mines would be very similar. Following this idea, letting C_{MSAP} and C_{MAP} be the optimal assignment costs of the MSAP and the clique MAP, respectively. It can be seen that, for the 3–dimensional case, the inequality $C_{MSAP} \leq C_{MAP} \leq 2C_{MSAP}$ holds. This is easily generalized for the n–dimensional case to $C_{MSAP} \leq C_{MAP} \leq (n-1)C_{MSAP}$, which implies that, in case a solution for the clique version ir required, the MSAP generates both upper and lower bounds on the optimal value.



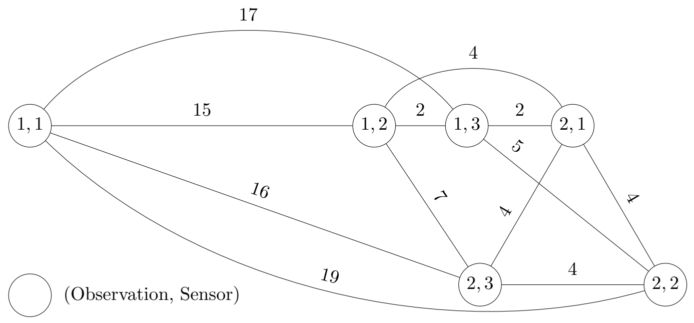
(a) Estimated locations

(b) The positions of the centroid and the incenter

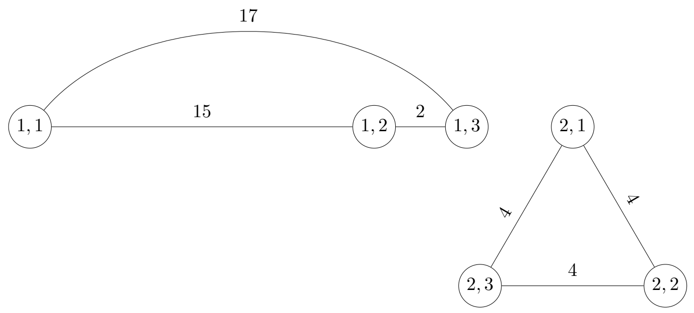
Figure 1: Example of three estimated locations

Figure 2 represents a case in which one of the observations of a given sensor (node (1,1)) is wrongly assigned to a position far away from the other observations. In this example, we assume that there are two mines in the field and six estimated positions calculated by three different sensors. First, in Figure 1b, note that, if the representative locations of the mines are assumed to be the centroids of the cliques, those positions are strongly biased by the wrong observation and then, are positioned significantly farther away from any of the original observations. This particular issue is depicted in clique $\{(1,1),(1,2),(1,3)\}$. On the other hand, in Figure 1c it can be seen that this effect is reduced when using a star assignment. We make the remark that similar examples can also can be constructed to point out negative effects of using the star costs. Therefore, in this context it is often considered solving the problem with both the star and the clique costs to have more information and obtain a better analysis.

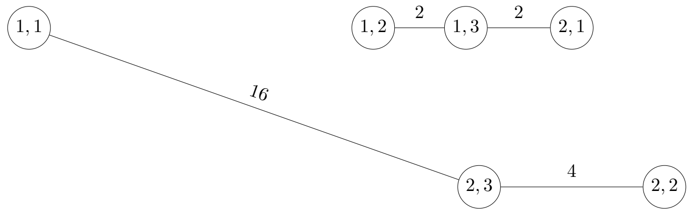
The second example is a case where the costs do not represent Euclidean distances and hence, the concept of the geometric center does not have a proper interpretation. Assume we have a group of antennas that are placed to intercept a collection of encrypted messages that we want to decode. Because of the interference and the noise of the environment, when the messages are received by the antennas they are somehow disrupted and thus not perfect for decoding. Each of the antennas can potentially receive a different version of each of the messages. The idea is to identify which of the received versions are the ones that most closely resemble the correct (transmitted) ones, so they can be sent for decoding. Here, since there is no way to compare the disrupted versions with the correct messages, the decision of which versions are finally sent to decode should be made by analyzing the dissimilarities between the received messages. In many contexts like this, the decoding process could be rather difficult. Hence, it is desired to select only one of the disrupted versions of each of the messages. Assuming that the messages are encrypted in some kind of alphabet (e.g., binary codes), using the centroid of the disrupted versions is not a valid approach here. In this framework, the use of the star costs could be of benefit, because the messages associated with the centers of the stars can be the ones selected for decoding (representative). For this problem, the costs can be



(a) Assignment costs of the sensor observations



(b) Optimal clique assignment



(c) Optimal star assignment

Figure 2: Example of a 3–sensor 2–target tracking problem

assumed to be the number of different characters between the messages, the sum of how far apart in the alphabet the different characters of the messages are, or any desired correlation metric.

The examples that we provided represent two cases of many others for which the MSAP arises. We would like to emphasize that the models introduced in this paper are intended to tackle more general instances of the MSAP and, therefore, can be used to solve not only these problems, but a multitude of others as well. The paper is outlined as follows. In section 2, we formally state the MSAP and propose a continuous nonlinear formulation and its linearization. Further, we introduce five families of valid inequalities to strengthen the formulation. Then, in section 3 we reformulate (1)–(5) as a set partitioning problem, and solve it via branch and price. In section 4 we present the computational results over a series of instances ranging in size from tripartite to 20–partite graphs of 4 to 30 nodes per partition. We then proceed to compare the solutions from both approaches. Last, in section 5 we summarize our findings, and offer insights for future research on this topic.

2. The multidimensional star assignment problem (MSAP)

Given a collection $N_1, \ldots, N_n$ of n disjoint node sets of equal size m and the corresponding collection of pairwise arc sets $E_{12}, \ldots E_{n-1,n}$, where $E_{st} = N_s \times N_t$ for $s < t$, let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a

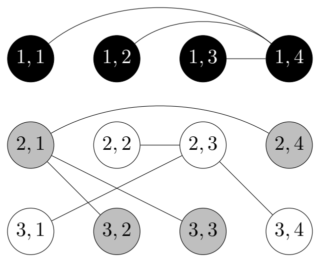


Figure 3: A valid star assignment for a graph with $n = 4$ and $m = 3$

complete n–partite graph, where $\mathcal{N} = \bigcup_{s=1}^{n} N_s$ and $\mathcal{E} = \bigcup_{s<t} E_{st}$. We refer to the ith node of set N_s with the duple (i, s) and use the quadruple (i, s, j, t), to represent the edge between nodes (i, s) and (j, t). Also, let c_{ij}^{st} be a cost value associated with edge (i, s, j, t) that accounts for the cost of assigning nodes (i, s) and (j, t) to the same star. Let a valid star be defined as a subgraph of $\mathcal{G}$ such that, (1) it contains exactly one node from each set $N_1, \ldots, N_n$; (2) one of the nodes (the center) is connected to all of the other nodes of the star (the leafs); and (3) there are no connections between any pair of leafs. Then, the MSAP aims for a set of m disjoint valid stars that cover all the nodes in $\mathcal{G}$. An example of a valid star assignment of an instance, where $n = 4$ and $m = 3$ is given in Figure 2. The three valid stars are colored gray, black, and white. The MSAP is known to be NP-hard (Crama and Spieksma (1992)).

2.1. *Continuous nonlinear formulation*

Let variables $\mathbf{z}$ and $\mathbf{x}$ be defined as follows:

$$z_i^s = \begin{cases} 1, & \text{if node } (i, s) \in \mathcal{N} \text{ is a star center} \\ 0, & \text{otherwise,} \end{cases}$$

and

$$x_{ij}^{st} = \begin{cases} 1, & \text{if nodes } (i, s) \text{ and } (j, t) \in \mathcal{N} \text{ belong to the same star} \\ 0, & \text{otherwise.} \end{cases}$$

Observe that $x_{ij}^{st} = x_{ji}^{ts}$ for all (i, s, j, t), and $x_{ij}^{ss} = 0$, since $(i, s, j, s) \notin \mathcal{E}$. The initial formulation is presented in (8)-(15), where the objective function (8) aims to minimize the overall assignment cost. Constraints (9) ensure that, if node (i, s) is a center, it must be connected to $(n - 1)$ nodes. Conversely, if it is not a center, then it should be connected to one node. Constraints (10)–(11) guarantee that if node (i, s) is a center, it must be connected to exactly one node from each set N_t, for all $t \neq s$. Further, constraints (12) enforce that there are exactly m stars in the optimal solution. Nonlinear constraints (13) guarantee that each node $(i, s) \in \mathcal{N}$ is either connected to a center, or is a center itself, and hence, it can only be connected to leafs. Last, (14)-(15) define the domain of variables $\mathbf{z}$ and $\mathbf{x}$.

$$(INLP) : \min \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{s=1}^{n} \sum_{\{t=1,\ldots,n: s<t\}} c_{ij}^{st} x_{ij}^{st} \tag{8}$$

$$\text{s.t.} \sum_{j=1}^{m} \sum_{t=1}^{n} x_{ij}^{st} = 1 + (n-2) z_i^s, \qquad \forall i = 1, \ldots, m \quad s = 1, \ldots, n \tag{9}$$

$$\sum_{j=1}^{m} x_{ij}^{st} \leq 1, \qquad \forall i = 1, \ldots, m \quad s, t = 1, \ldots, n \tag{10}$$

$$\sum_{j=1}^{m} x_{ij}^{st} \geq z_i^s, \qquad \forall i = 1, \ldots, m \quad s, t = 1, \ldots, n \tag{11}$$

$$\sum_{i=1}^{m} \sum_{s=1}^{n} z_i^s = m \tag{12}$$

$$z_i^s + \sum_{j=1}^{m} \sum_{t=1}^{n} x_{ij}^{st} z_j^t = 1, \qquad \forall i = 1, \ldots, m \quad s = 1, \ldots, n \tag{13}$$

$$z_i^s \in \{0, 1\}, \qquad \forall i = 1, \ldots, m \quad s = 1, \ldots, n \tag{14}$$

$$x_{ij}^{st} \in \{0, 1\}, \qquad \forall i = 1, \ldots, m \quad s = 1, \ldots, n. \tag{15}$$

Let $RNLP$ be the continuous relaxation of formulation (8)–(15). That is, both $\mathbf{z}$ and $\mathbf{x}$ are allowed to take fractional values between zero and one. We now proceed to prove that in any feasible solution of $RNLP$, all $\mathbf{z}$ variables take integer values. For this proof, assume we have a feasible solution $(\mathbf{z}, \mathbf{x})$. We refer to node (i, s) as white if $z_i^s = 0$, black if $z_i^s = 1$, or gray if $z_i^s \in (0, 1)$.

Lemma 1. *In any feasible solution of $RNLP$, a white node can only be connected to black nodes.*

PROOF. See Appendix A. □

Lemma 2. *In any feasible solution of $RNLP$, a black node can only be connected to white nodes.*

PROOF. See Appendix A. □

Based on Lemmata 1 and 2, we can deduce that, if there exist gray nodes, then they are only connected to other gray nodes.

Lemma 3. *If there exist gray nodes in a feasible solution, then the number of those is a multiple of n.*

PROOF. See Appendix A. □

Theorem 1. *In any feasible solution of the $RNLP$, all $\mathbf{z}$ variables are binary (i.e., there cannot exist gray nodes).*

PROOF. See Appendix A. □

Contrary to the case of the $\mathbf{z}$ variables, it is easy to see that there can be feasible solutions where some of the $\mathbf{x}$ variables are fractional. Consider the example presented in Figure 4. It is easy to see that the fractional solution depicted in Figure 4c, satisfies all constraints in (9)–(13). Moreover, such a solution is actually a convex combination of integral solutions displayed in Figures 4a and 4b. Note that we are able to construct this combination because the star centers in both integral solutions 4a and 4b are the same. Because of the result presented in Theorem 1, a strictly convex combination between two integral solutions with different star centers would yield fractional $\mathbf{z}$ variables, and therefore an infeasible solution. We now formally prove that if there exists an optimal solution where the $\mathbf{x}$ are fractional, there is also an alternative integral solution.

Theorem 2. *If $RNLP$ is feasible, then there always exists an optimal solution of $RNLP$, where all $\mathbf{x}$ variables take integer values.*

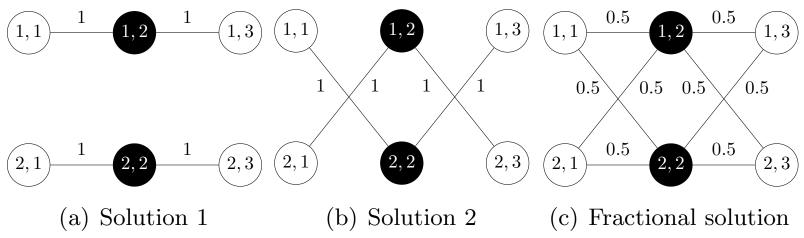


Figure 4: Example of a fractional solution

Proof. See Appendix A. $\square$

As a result from Theorems 1 and 2, constraints (14) and (15) can be relaxed in $INLP$, leaving the continuous nonlinear optimization problem $RNLP$, henceforth referred to only as NLP. To solve this formulation we can use any available optimizer that handles nonlinear programs. Although, in spite of having a continuous formulation, rather than an integer one, nonlinear constraints (13) still pose a difficult challenge because of their non-convex nature. As an alternative approach, instead of solving NLP, we propose using a standard linearization technique that involves introducing additional variables to replace the bilinear terms in constraints (13). Furthermore, from the results presented in Theorems 1 and 2, we derive additional valid inequalities that strengthen the proposed linear formulation. A description of the linearization and the valid inequalities follows.

2.2. Linearization

The bilinear terms of constraints (13) represent the greatest difficulty of the NLP formulation. Thus, we apply a standard linearization technique by replacing those terms with additional variables ($\mathbf{w}$). Unfortunately, by relaxing the nonlinear constraints, we lose the integrality properties described in Theorems 1 and 2. Hence, the resulting formulation is a *mixed integer linear program* (MIP). On top of that, this reformulation comes with an overhead of $O(n^2m^2)$ variables and constraints. The proposed reformulation follows:

$$(MIP) : \min \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{s=1}^{n}\sum_{\{t=1,\dots,n:s<t\}} c_{ij}^{st}x_{ij}^{st} \tag{16}$$

$$\begin{aligned}
&\text{s.t. } (9)-(12) && && (17)\\
&w_{ij}^{st} \le z_j^t, && \forall i,j=1,\dots,m \quad s,t=1,\dots,n && (18)\\
&w_{ij}^{st} \le x_{ij}^{st}, && \forall i,j=1,\dots,m \quad s,t=1,\dots,n && (19)\\
&w_{ij}^{st} \ge z_j^t + x_{ij}^{st} - 1, && \forall i,j=1,\dots,m \quad s,t=1,\dots,n && (20)\\
&z_i^s + \sum_{j=1}^{m}\sum_{t=1}^{n} w_{ij}^{st} = 1, && \forall i=1,\dots,m \quad s=1,\dots,n && (21)\\
&w_{ij}^{st} \in [0,1], && \forall i=1,\dots,m \quad s=1,\dots,n && (22)\\
&x_{ij}^{st} \in [0,1], && \forall i=1,\dots,m \quad s=1,\dots,n && (23)\\
&z_i^s \in \{0,1\}, && \forall i=1,\dots,m \quad s=1,\dots,n, && (24)
\end{aligned}$$

where variable w_{ij}^{st} represents the bilinear term $x_{ij}^{st}z_j^t$, for all $(i,s,j,t) \in \mathcal{E}$, and constraints (18)–(20) are the linearization inequalities. Note that, contrary to the case of the $\mathbf{x}$ variables, $w_{ij}^{st} = x_{ij}^{st}z_j^t \neq x_{ij}^{st}z_i^s = w_{ji}^{ts}$. Thus, both variables w_{ij}^{st} and w_{ji}^{ts} must be included. Finally, using a similar argument as in Theorem 2, it is easy to see that in the above formulation we can relax the integrality constraints of variables $\mathbf{x}$ and $\mathbf{w}$.

2.3. Valid inequalities

In this subsection, we introduce five families of valid inequalities that strengthen the *MIP* formulation. Since we no longer have the nonlinear set of constraints (13), it is possible (and likely) that the linear relaxation of this formulation produces fractional solutions (i.e., gray nodes). To cope with this issue, we can use the results from Lemmata 1 and 2 to define the following inequalities.

Proposition 1. *For any edge $(i, s, j, t) \in \mathcal{E}$, the inequality $x_{ij}^{st} \leq 2 - z_i^s - z_j^t$ is valid.*

Proof. From constraints (13), for nodes (i, s) and (j, t), we obtain

$$z_i^s + x_{ij}^{st} z_j^t \leq 1$$

and

$$z_j^t + x_{ij}^{st} z_i^s \leq 1.$$

Adding both inequalities yields

$$2 \geq z_i^s + z_j^t + x_{ij}^{st}(z_i^s + z_j^t). \tag{25}$$

Furthermore, since $x_{ij}^{st} \in [0, 1]$ and $z_i^s \in [0, 1]$, we have that

$$x_{ij}^{st} z_i^s \geq x_{ij}^{st} + z_i^s - 1$$

and

$$x_{ij}^{st} z_j^t \geq x_{ij}^{st} + z_j^t - 1.$$

Adding up both inequalities, results in

$$x_{ij}^{st}(z_i^s + z_j^t) \geq 2x_{ij}^{st} + z_i^s + z_j^t - 2. \tag{26}$$

Finally, from (25) and (26), we obtain

$$x_{ij}^{st} \leq 2 - z_i^s - z_j^t. \tag{27}$$

□

Inequality (27) comes from the fact that in any feasible solution of the *MIP* there are no connections between black nodes (star centers). Clearly, if both variables z_i^s and z_j^t take the value of one, the corresponding variable x_{ij}^{st} cannot be positive.

Proposition 2. *For any edge $(i, s, j, t) \in \mathcal{E}$, the inequality $x_{ij}^{st} \leq z_i^s + z_j^t$ is valid.*

Proof. From Lemma 1, the proposed inequality:

$$x_{ij}^{st} \leq z_i^s + z_j^t \tag{28}$$

is trivially derived. □

Similarly as before (cf. Proposition 1), in any feasible solution of the MIP there are no connections between white nodes (leafs). Thus, if both variables z_i^s and z_j^t are zero, the corresponding variable x_{ij}^{st} must also be zero.

There are two interesting features of inequalities (27) and (28). First, the size of both families is $O(n^2m^2)$, which implies that both can be directly included in the MIP without the need of a separation algorithm. Second, as proved in Theorem 3, if constraints (27) and (28) are included in formulation (16)–(24), it is possible to relax constraints (21), and therefore remove the extra $\mathbf{w}$ variables. The resulting formulation, referred to as $MIPa$, not only has less variables and constraints than MIP, but also produces a better dual bound.

$$
\begin{aligned}
(MIPa): \min \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{s=1}^{n}\sum_{\{t=1,\dots,n: s<t\}} & c_{ij}^{st}x_{ij}^{st} && && (29)\\
\text{s.t. } (9)-(12) & && && (30)\\
x_{ij}^{st} \le z_i^s + z_j^t, & && \forall i,j = 1,\dots,m \quad s,t=1,\dots,n && (31)\\
x_{ij}^{st} \le 2 - z_i^s - z_j^t, & && \forall i,j = 1,\dots,m \quad s,t=1,\dots,n && (32)\\
z_i^s \in \{0,1\}, & && \forall i = 1,\dots,m \quad s=1,\dots,n && (33)\\
x_{ij}^{st} \in \{0,1\}, & && \forall i = 1,\dots,m \quad s=1,\dots,n. && (34)
\end{aligned}
$$

Theorem 3. *$MIPa$ is a valid formulation for the MSAP.*

Proof. Since $MIPa$ includes constraints (9)–(12), it suffices to show that any optimal solution of this formulation satisfies that, (1) there are no gray nodes (i.e., $z_i^s \in \{0,1\}, \forall (i,s) \in N$) and (2) the only connections that can exist are between white and black nodes.

Clearly, $MIPa$ contains the integrality constraints for the $\mathbf{z}$ variables. Thus, no gray nodes would exist. Furthermore, observe that constraints (31) guarantee that no connection exists between white nodes. Hence, white nodes can only be connected to black ones. Similarly, constraints (32) guarantee that no connection exists between two black nodes, hence black nodes can only be connected to white ones. □

We now describe three additional families of valid inequalities.

Proposition 3. *For any node $(i,s) \in \mathcal{N}$, the inequality*

$$
z_i^s + \sum_{j=1}^{m}\sum_{t=1}^{n} q_{ij}^{st} \ge 1, \tag{35}
$$

where $q_{ij}^{st} \in \{x_{ij}^{st}, z_j^t\}$ is valid.

Proof. From nonlinear constraints (13), it is easy to see that any integral solution of $MIPa$ satisfies the following equation:

$$
z_i^s + \sum_{j=1}^{m}\sum_{t=1}^{n} \min\{x_{ij}^{st}, z_j^t\} = 1.
$$

Since by definition, $q_{ij}^{st} \ge \min\{x_{ij}^{st}, z_j^t\}$, all integral feasible solutions satisfy the proposed inequality. □

We refer to this family as the *minimum sum* inequalities. Note that, for each node (i,s), the number of inequalities of this family is $O(2^{mn})$. Thus, we propose the separation procedure presented in Algorithm 1. Given a fractional solution $(\bar{\mathbf{z}},\bar{\mathbf{x}})$, for each node (i,s), we search for the inequality that is violated the most. For this purpose, for each (j,t) such that $(i,s,j,t)\in\mathcal{E}$, we select as q_{ij}^{st} in (35) the variable associated with the minimum value between $\bar{z}_j^t$ and $\bar{x}_{ij}^{st}$. We repeat this procedure for all $(i,s)\in\mathcal{N}$, adding all violated inequalities.

Algorithm 1 `minSumSeparation`$(\mathcal{G},i,s,\bar{\mathbf{z}},\bar{\mathbf{x}})$

```
1:  $sum = \bar{z}_i^s$
2:  $cut = z_i^s$
3:  for all $(j,t)$ such that $(i,s,j,t)\in\mathcal{E}$ do
4:      if $\bar{z}_j^t < \bar{x}_{ij}^{st}$ then
5:          $sum \leftarrow sum + \bar{z}_j^t$
6:          $cut \leftarrow cut + z_j^t$
7:      else
8:          $sum \leftarrow sum + \bar{x}_{ij}^{st}$
9:          $cut \leftarrow cut + x_{ij}^{st}$
10:     end if
11: end for
12: if $sum < 1$ then
13:     return $cut \geq 1$
14: else
15:     return $null$
16: end if
```

For the last two families of valid inequalities, consider the fractional solution depicted in Figure 5. This is an optimal solution of the linear relaxation of $MIPa$. First, observe the 4-cycle formed by nodes $(2,1),(1,3),(1,4),(2,2)$. Clearly, in a feasible integral solution the number of arcs within the cycle cannot be greater than two. We analyze this inequality in Proposition 4.

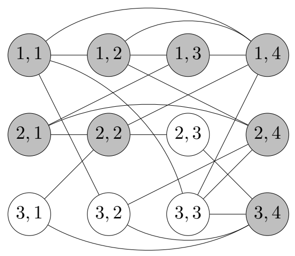


Figure 5: Example of a fractional solution of the linear relaxation of $MIPa$

Proposition 4. *For any 4–cycle involving nodes $(i,s),(j,t),(k,u)$, and (l,v), $x_{ij}^{st}+x_{jk}^{tu}+x_{kl}^{uv}+x_{il}^{sv}\leq 2$ is a valid inequality.*

Proof. Assume that there is an integral solution where $x_{ij}^{st}+x_{jk}^{tu}+x_{kl}^{uv}+x_{il}^{sv}>2$. Hence, there exist at least three arcs in the 4–cycle for which the respective $\mathbf{x}$ variables are one. In turn, this implies that at least two nodes have two connections each. From (9), it can be deduced that these two nodes need to be black (since they have more than one connection). Hence, all the possible node configurations can be summarized with the two cases, shown in Figure 6. The cases that are not included in this figure correspond to mirrored versions of the ones that are presented.

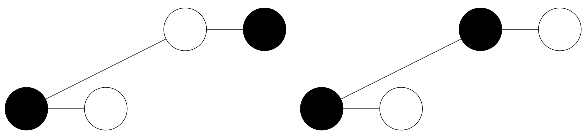

(a) Case 1 (b) Case 2

Figure 6: Cycle cut example for Proposition 4

Case 1 can be dismissed because a white node is connected to two black nodes, contradicting (9). Similarly, case 2 cannot happen because it contradicts the fact that two black nodes can never be connected (see Proposition 2). □

We refer to this family as the *4-cycle* inequalities. Since the number of inequalities of this family is $O(m^4n^4)$, we can find all violations by total enumeration.

Finally, observe the triplet formed by nodes $(1,3), (2,1)$, and $(2,2)$ in Figure 5. It is easy to see that all three nodes cannot be black and simultaneously have any connection between them. We analyze this inequality in Proposition 5.

Proposition 5. *For any triplet of nodes $(i,s), (j,t), (k,u)$, where node (i,s) is the center of the triplet, $x_{ij}^{st} + x_{ik}^{su} + z_i^s + z_j^t + z_k^u \leq 3$ is a valid inequality.*

Proof. Since there cannot be any connection between two black nodes or two white nodes, it is easy to see that the only feasible options can be summarized by the ones depicted in Figure 7. As before, we do not include mirrored configurations, or the cases where the inequality is trivially satisfied (e.g., the case where all the nodes are white and there are no connections between them). Note that in all this cases, if we sum up the corresponding **z** variables of all three nodes, and the **x** of the arcs connecting the center with the leafs, the resulting summation is always less or equal than three. □

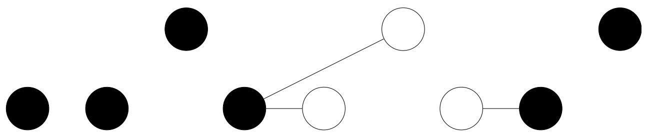

(a) Case 1 (b) Case 2 (c) Case 3

Figure 7: Triplet cut example for Proposition 5

We described the above inequality for the case where node (i,s) is the center of the triplet. Although, note that it is possible to obtain two alternative inequalities by choosing any of the other two nodes as the center. We refer to this family as the *triplet* inequalities and, since the total number of posible triplets is $O(n^3m^3)$, we can find violations by total enumeration. Finally, note that this family of inequalities can be easily generalized for k–tuples, where $k > 3$. However, in practice they are less effective and harder to evaluate, as the total number of k–tuples if $O(n^km^k)$.

3. Set partitioning formulation

From formulation (1)–(5), it is easy to see that the MSAP is a particular case of the set partitioning problem, and thus can be formulated as such. The set up of this formulation is as follows. Let K be the set of all feasible stars that can be used to partition $\mathcal{G}$. For each star $k \in K$, let a_{is}^{k} be a parameter that takes the value of one if star k covers node (i,s) and zero, otherwise. Let y_k be a binary variable that takes the value of one if star k is selected and zero, otherwise. Further, let $\phi(k)$ be a function that returns the center of star k, and finally, let c_k be the cost of star k. That is, assuming that $\phi(k) = (i,s)$, the cost of star k is given by,

$$c_k = \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: t \neq s\}} c_{ij}^{st} a_{jt}^{k}. \tag{36}$$

Then, the resulting set partitioning (SP) formulation can be defined as follows:

$$(SP): \min \sum_{k \in K} c_k y_k \tag{37}$$
$$\text{s.t.} \quad \sum_{k \in K} a_{is}^{k} y_k = 1, \qquad \forall i = 1,\ldots,m \quad s = 1,\ldots,n \tag{38}$$
$$y_k \in \{0,1\}, \qquad \forall k \in K. \tag{39}$$

Note that constraint set (38) is created by aggregating all constraints in (1)–(5). Hence, both formulations are equivalent. The advantage of the SP over $MIPa$ is the fact that the star configuration, required by any feasible solution of the MSAP, is implicitly guaranteed by the definition of set K. In other words, since all the elements in K are valid stars, the remaining decision is finding the right partition. However, the downside is that $|K|$ grows exponentially as the size of $\mathcal{G}$ increases (see Proposition 6). For this reason, (37)–(39) has far more variables than can be reasonably handled and therefore solving it explicitly can be computationally impractical.

Proposition 6. *The total number of feasible stars is nm^n.*

Proof. To be a feasible star, its center must be connected to exactly one node from each of the other $n-1$ sets. Since there are m nodes in each set, the total possible stars centered at any given node is m^{n-1}. Furthermore, since the total number of nodes is nm and each of them can be the center of a star, the total number of stars is nm^n. □

To overcome this difficulty, we solve the problem via branch and price (Barnhart et al. (1998) and Lübbecke and Desrosiers (2005)). That is, at each node of the branch-and-bound tree, we solve a linear relaxation of SP, defined over a manageable subset of stars $K' \subset K$. We refer to this formulation as the *restricted master problem* (RMP). If the current set of stars K' is not sufficient to declare optimality, we proceed to generate additional stars via column generation. Furthermore, if the optimal solution at the end of the column generation phase is not integral, we branch to reduce the solution space, eliminating undesired fractional solutions. The RMP is as follows:

$$(RMP): \min \sum_{k \in K'} c_k y_k \tag{40}$$
$$\text{s.t.} \quad \sum_{k \in K'} a_{is}^{k} y_k = 1, \qquad \forall i = 1,\ldots,m \quad s = 1,\ldots,n \tag{41}$$
$$0 \leq y_k \leq 1, \qquad \forall k \in K', \tag{42}$$

3.1. Generating the initial set of columns

Before starting the column generation phase, we require to construct an initial set of stars K' for which RMP is feasible. There are different options for constructing this set. One can, for example, use a typical initialization method such as the two-phase or the penalization methods (see Bazaraa et al. (2009)). Alternatively, we can initialize K' with any feasible solution produced heuristically. This is in general a preferred option because a good initial solution provides RMP with a high quality upper bound, which in turn may help prune some unnecessary branches in the branch-and-bound tree. To find the initial set K', we propose a very simple greedy algorithm that generates iteratively a set of valid stars. For this purpose, we define a feasible star p to be a minimum cost star of $\mathcal{G}$, if $c_p \leq c_k$, for all $k \in K$. Assume we begin the algorithm with an empty set K'. Then, we sequentially add to K' a minimum cost star of graph $\mathcal{G} \setminus K'$. We repeat this process until all the nodes are covered. A full description of this heuristic is presented in algorithms 2–4.

Algorithm 2 `minCenteredStar`$(\mathcal{G}, m, n, i, s)$

```
1: $S \leftarrow \{(i,s)\}$
2: $cost \leftarrow 0$
3: for all $t = 1, \dots, n$, where $t \neq s$ do
4:     $l \in \arg\min_{j=1,\dots,m}\{c_{ij}^{st}\}$
5:     $S \leftarrow S \cup \{(l,t)\}$
6:     $cost \leftarrow cost + c_{il}^{st}$
7: end for
8: return $[S, cost]$
```

Algorithm 3 `minStar`(G, m, n)

```
1: $S^* \leftarrow \emptyset$
2: $cost^* \leftarrow \infty$
3: for all $s = 1, \dots, n$ do
4:     for all $i = 1, \dots, m$ do
5:         $[S, cost] \leftarrow$ minCenteredStar$(G, m, n, i, s)$
6:         if $cost < cost^*$ then
7:             $cost^* \leftarrow cost$
8:             $S^* \leftarrow S$
9:         end if
10:     end for
11: end for
12: return $[S^*, cost^*]$
```

We now prove that the proposed heuristic produces a feasible solution in $O(m^3n^2)$. We first derive the complexity of identifying a minimum cost star of a given graph $\mathcal{G}$ and then, we prove the correctness and provide the overall complexity.

Lemma 4. *Given a graph $\mathcal{G}$, finding a minimum cost star takes $O(m^2n^2)$.*

Proof. First, assume we know a priori the center of the minimum cost star. Let (i,s) be that node. Since the center must be connected to exactly one node from each set $N_t \in \mathcal{N} \setminus N_s$, we obtain the minimum cost star by finding a node $(\bar{j}, t)$ such that $\bar{j} \in \arg\min_{j=1,\dots,m}\{c_{ij}^{st}\}$, from each of the

Algorithm 4 `findInitialSet`(G)

```
1:  $K' \leftarrow \emptyset$
2:  $G' \leftarrow G$
3:  $cost^* = 0$
4:  $m' = m$
5:  **while** $G' \neq \emptyset$ **do**
6:      $[S, cost] =$ minStar$(G', m', n)$
7:      $K' \leftarrow S$
8:      $cost^* \leftarrow cost^* + cost$
9:      $G' \leftarrow G' \setminus S$
10:     $m' = m' - 1$
11: **end while**
```

remaining $n-1$ sets. Furthermore, each set has in total m nodes. Thus, finding the minimum cost star centered at (i, s) takes $O(nm)$ (see Algorithm 2).

Now, observe that since there are nm nodes in $\mathcal{G}$, we can repeat the same process for each node, selecting the overall minimum cost star in $O(m^2n^2)$ (see Algorithm 3). □

Theorem 4. *Given a graph $\mathcal{G}$, the proposed heuristic finds a feasible set K' in $O(m^3n^2)$.*

Proof. We initialize Algorithm 4 with a graph $\mathcal{G}' = \mathcal{G}$. First, note that at each iteration of the while loop (steps 5–11), we generate a valid star comprised by exactly one node from each set $N_1, \ldots, N_n$. Since we remove from $\mathcal{G}'$ the nodes of such star, we guarantee that those nodes are not covered by other stars in further iterations. Moreover, at each iteration, the number of nodes in $\mathcal{G}'$ decreases by n. In view of $\mathcal{G}'$ having initially mn nodes, by iteration m all the nodes must be covered by one star. Finally, finding the minimum cost star takes $O(m^2n^2)$. Thus, generating K' takes $O(m^3n^2)$. □

Naturally, there are additional elements that can be used for improving the running time of Algorithm 4 and obtaining additional stars to add in K'. For example, after each run of Algorithm 2, we can temporarily store the minimum cost star centered at each node $(i, s) \in \mathcal{G}$. Then, in further iterations we can warm start the algorithm initializing the search with those stars. Moreover, noting that each of these minimum cost stars is a member of K, we can include some of them in the initial set K'. However, since we would like to avoid having a large initial set of columns, or worse including poor candidates, in practice we impose a limit on the size of the initial set K'. Thus, besides adding the solution found by Algorithm 4, we keep an ordered set of the best αm generated stars and include those in K'. We have tested different values for α, being $\alpha = \{2, 3, 4\}$ the ones that produced the best results. We only use this limit when generating the initial set of columns. Once in the column generation stage, we relax this limit.

3.2. Optimality conditions and new candidate stars generation

While solving RMP, it is possible (and in general common) that the initial set of stars K' does not include an optimal partition of $\mathcal{G}$. Moreover, it is also possible that we cannot even find an optimal solution of the linear relaxation, using the stars in K'. This is not a particular issue of the MSAP as it happens often with most problems that are solved using set partitioning formulations (Barnhart et al. (1998)).

To find the optimal solution of the linear relaxation, we use a column generation approach that introduces new stars to K' in case they are needed to declare optimality. Hence, at each iteration, we are required to find whether there exists a new star $k \in K \setminus K'$, that improves the current solution. In other words, we aim to find a variable y_k, for $k \in K \setminus K'$, such that its reduced cost is negative, or, from the dual perspective, a variable such that its corresponding dual constraint is violated.

Consider the dual of the RMP, defined as the relaxed dual problem RDP.

$$(RDP) : \max \sum_{i=1}^{m} \sum_{s=1}^{n} \alpha_i^s \tag{43}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \sum_{s=1}^{n} a_{is}^k \alpha_i^s \leq c_k, \qquad \forall k \in K' \tag{44}$$

Notice that (44) includes only the constraints associated with the stars in K'. This implies that, if we find a star $k \in K \setminus K'$, such that its corresponding constraint (44) is violated by the current solution, i.e., a star k for which $c_k - \sum_{i=1}^{m} \sum_{s=1}^{n} a_{is}^k \alpha_i^s < 0$, we can include such star in K'. In consequence, the optimality conditions of the column generation stage can be defined as follows.

$$c_k - \sum_{j=1}^{m} \sum_{t=1}^{n} a_{jt}^k \alpha_j^t \geq 0, \quad \forall k \in K \setminus K', \tag{45}$$

or, alternatively, from (36),

$$\sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: t \neq s\}} c_{ij}^{st} a_{jt}^k - \sum_{j=1}^{m} \sum_{t=1}^{n} a_{jt}^k \alpha_j^t = \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: t \neq s\}} c_{ij}^{st} a_{jt}^k - \alpha_i^s - \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: t \neq s\}} a_{jt}^k \alpha_j^t =$$
$$= -\alpha_i^s + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: t \neq s\}} (c_{ij}^{st} - \alpha_j^t) a_{jt}^k \geq 0, \quad (i,s) = \phi(k), \quad \forall k \in K \setminus K'. \tag{46}$$

To identify promising stars, we propose a similar approach to the one used to obtain the initial set K'. We first fix the center of the star at node (i,s) and then, search for stars that violate condition (46). We repeat the same process for each node $(i,s) \in \mathcal{G}$. Thus, given a center node (i,s), the proposed pricing subproblem $(P(i,s))$ is as follows:

$$(P(i,s)) : -\alpha_i^s + \min \sum_{j=1}^{m} \sum_{\{t=1\ldots,n: t \neq s\}} (c_{ij}^{st} - \alpha_j^t) a_{jt} \tag{47}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} a_{jt} = 1, \quad \forall t = 1, \ldots, n, \quad t \neq s \tag{48}$$

$$a_{jt} \in \{0,1\}, \quad \forall j = 1, \ldots m, \quad t = 1, \ldots, n, \quad t \neq s \tag{49}$$

where the objective function (47) minimizes the reduced cost of the star, constraints (48) guarantee that the center node (i,s) is connected with a node in each layer $t \neq s$, and constraints (49) define the domain of the variables.

Then, we solve a problem $P(i,s)$ for each node $(i,s) \in \mathcal{G}$, finding new candidate stars (possibly many) at each iteration. Note that, if for all nodes $(i,s) \in \mathcal{G}$, $-\alpha_i^s + \min \sum_{j=1}^{m} \sum_{t=1, t \neq s}^{n} (c_{ij}^{st} - \alpha_j^t) a_{jt} \geq 0$, we have an optimal solution for the RMP.

In principle, instead of solving a sequence of mn $P(i,s)$ subproblems, we could adapt Algorithm 3 to obtain the same result in $O(m^2 n^2)$. This approach will indeed work for solving the root node of the branch-and-bound tree. However, if the solution of the linear relaxation is not integral, we must proceed with the branching stage to eliminate fractional solutions. Then, the branching decisions must be enforced in the subproblem in the form of constraints to prevent generating stars that violate them. Because of these additional constraints, we cannot solely apply Algorithm 3. On the other hand, $P(i,s)$ can be easily updated to include the branching constraints. Nonetheless, solving sequentially mn of such integer programs, after including the branching constraints, could potentially hinder the column generation procedure. To circumvent this issue we will show that it is possible to reformulate $P(i,s)$ as a shortest path problem with side constraints. Before describing the proposed reformulation, we first introduce the branching rule.

3.3. Branching rule

One of the major difficulties that arise when solving a problem via branch and price is defining the branching rule. There are many reasons why standard branching (i.e., fixing a fractional variable to either zero or one) is undesirable in this context. First of all, when fixing a variable to one, the n nodes of the star corresponding to such a variable are covered. Since those nodes cannot be covered again, all the other stars in K containing at least one of those nodes are discarded (i.e., the corresponding variables are fixed to zero). Hence, the number of variables to be considered in that branch is significantly reduced. On the other hand, when a variable is fixed to zero, only one of exponentially many variables is fixed. This particular behavior results in a highly unbalanced, and thus inefficient, branching tree. Furthermore, when fixing a variable to zero, we must ensure that the pricing subproblem does not produce the same variable again. This is generally done by either finding the next best solution (possibly the nth best after n branches), or by including additional constraints in the subproblem. In both cases, solving the pricing problem becomes remarkably harder.

An alternative option, that is widely used when solving set partitioning problems, is the so-called Ryan-Foster branching rule (Barnhart et al. (1998); Ryan and Foster (1981)). Consider a fractional solution of the RMP. From constraints (41), it is easy to see that if a solution is fractional, there exists a node that is partially covered by at least two stars. Let y_k and y_l be two fractional variables (i.e., $y_k, y_l \in (0,1)$) such that the corresponding stars share node (i,s). Clearly, since stars k and l are different, there exists a node $(j,t) \neq (i,s)$ that is either covered by k and not by l, or vice versa. Thus, we can generate a branch in which we force both nodes (i,s) and (j,t) to be covered by the same star, and a second one that ensures that the nodes are covered by different stars. It is easy to see that the current fractional solution is no longer valid, while any feasible integer solution belongs to one of the two branches.

Once we generate the new branches, we need to enforce the respective branching decisions in the subsequent iterations. Instead of sequentially adding such constraints to the RMP, we can alternatively propagate them over set K'. For the branch that requires both nodes to be in different stars, we fix to 0 all the variables of the stars that cover both nodes simultaneously. Conversely, we fix to 0 all the variables of the stars that cover only one of these nodes, for the branch that requires both nodes to be together. Moreover, we must ensure that the pricing problem also satisfies the branching constraints. If we use formulation $P(i,s)$ as the pricing problem, these constraints are easily enforced as follows. Assume that (i,s) and (j,t) are the nodes of the current branching decision. If the subproblem we are solving is $P(i,s)$, we fix a_{jt} to be either one or zero for the cases where both nodes are required to be together or separated, respectively. For $P(j,t)$, the process is similar but fixing a_{is} instead. If we are solving the subproblem for other nodes different than (i,s) or (j,t), we introduce constraints $a_{is} = a_{jt}$ or $a_{is} + a_{jt} \leq 1$ in each of the branches, respectively.

3.4. Star generation as a shortest path problem with side constraints

In general, the quality of any column generation approach relies on the ability to generate new candidate variables in an efficient way. This is because the subproblem, that either declares optimality or generates new variables, may be solved a considerable number of times before terminating. Despite the fact that solving the subproblem via integer programming is sufficient for tackling small instances, in practice, the use of specialized algorithms is proven to yield better results (Feillet et al. (2004)). Furthermore, when possible, it is often desired to transform the subproblem into a shortest path problem because of the variety of different solution techniques that can be used to efficiently solve this kind of problems (Feillet et al. (2004); Irnich and Desaulniers (2005); Valério de Carvalho (1999)).

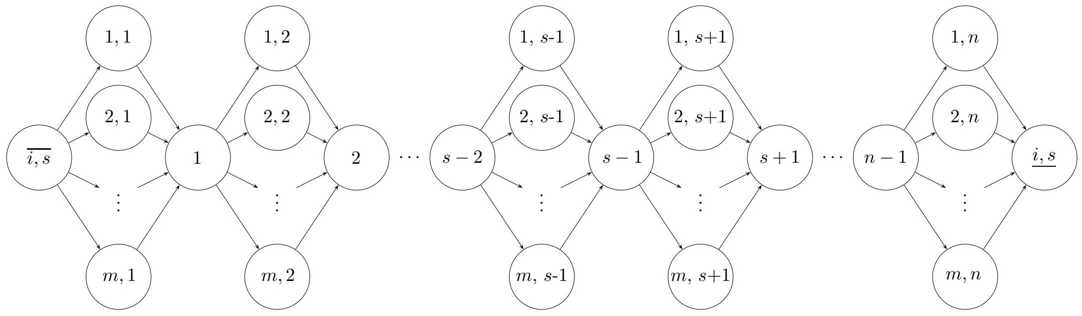
For this reason, as an alternative to solving $P(i,s)$ using integer programming techniques, we propose reformulating it as a shortest path problem over a directed acyclic network $\mathcal{H}(i,s) = (\mathcal{N}', \mathcal{A})$. The set up for this network is as follows. First, we introduce two nodes $(\overline{i,s})$ and $(\underline{i,s})$ referred to as the source and sink nodes, respectively. There is a set of nodes, $\bar{N}$, comprised of one node associated with each set $N_1 \ldots, N_{s-1}, N_{s+1}, \ldots, N_{n-1}$ (i.e., $\bar{N} = \{1, \ldots, s-1, s+1, \ldots, n-1\}$). The node set is defined as $\mathcal{N}' = \{(\overline{i,s}), (\underline{i,s})\} \cup \bar{N} \cup \mathcal{N} \setminus N_s$, and the arc set $\mathcal{A}$ is constructed as follows. First, if $s = 1$, then there is an arc between the source node $(\overline{i,s})$ and node $(j,2)$, for all $j = 1, \ldots, m$. Otherwise, if $s > 1$, there is an arc between the source node $(\overline{i,s})$ and node $(j,1)$, for all $j = 1, \ldots, m$. Second, there is an arc between nodes (j,t) and t, for all $j = 1, \ldots, m$, such that $t \in \bar{N}$. Third, there is an arc between nodes t and $(j, t+1)$, such that $1 < t < s-1$; an arc between nodes $s-1$ and $(j, s+1)$; and an arc between nodes t and $(j, t+1)$, such that $s < t < n$, for all $j = 1, \ldots, m$. Finally, if $s < n$, there is an arc between node (j,n) and the sink node $(\underline{i,s})$, for all $j = 1, \ldots, m$. Conversely, if $s = n$, there is an arc between node $(j, n-1)$ and the sink node $(\underline{i,s})$, for all $j = 1, \ldots, m$.

The cost of using the arc between nodes $(\overline{i,s})$ and (j,t) is $(c_{ij}^{st} - \alpha_j^t)$, for $j = 1, \ldots, m$ and $t = \{1, 2\}$, depending on the value of s. Further, the cost between nodes (j,t) and t is zero, and the cost between nodes t and (j,u) is $(c_{ij}^{su} - \alpha_j^u)$, for all $j = 1, \ldots, m$ and $u = s+1$, if $t = s-1$, or $u = t+1$, otherwise. Finally, the cost of using the arc between nodes (j,t) and $(\underline{i,s})$ is zero, for $j = 1, \ldots, m$ and $t = \{n-1, n\}$, also depending on the value of s. A graphical representation of $\mathcal{H}(i,s)$ is presented in Figure 8a.

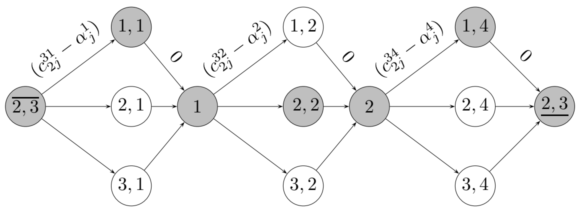
Note that the costs of the arcs do not depend on the nodes they emanate from. This is because such arcs represent the connection between (i,s) and the node at the head of the corresponding arc. Moreover, it is easy to see that each path from the source to the sink in $\mathcal{H}(i,s)$ can be associated with a star centered at (i,s). For instance, the path formed by the gray nodes in Figure 8b can be associated with the valid star shown in Figure 8c. Thus, if we subtract α_i^s from the cost of the shortest path between the source and sink nodes in $\mathcal{H}(i,s)$, we obtain the minimum reduced cost of the variable associated with the star that corresponds to the shortest path.

In the absence of branching constraints, since $\mathcal{H}(i,s)$ is acyclic and the number of arcs is $O(mn)$, finding a shortest path can be done in $O(mn)$ (Ahuja et al. (1993)). Thus, similarly as in Algorithm 3, solving the problem for all the nodes takes $O(m^2n^2)$. On the other hand, once the branching stage begins, finding the corresponding shortest path becomes a more difficult task. The advantage of the proposed representation is that we can include the additional branching constraints and solve the resulting shortest path with side constraints via Dynamic Programming (Irnich and Desaulniers (2005)). In the Appendix B we provide a brief description of the algorithms that we used to calculate the candidate stars.

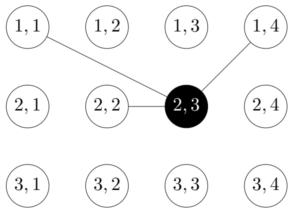
After solving all the $P(i,s)$ subproblems for all nodes $(i,s) \in \mathcal{N}$, it is possible to obtain several candidate stars with the same set of nodes, but with different centers. For instance, for the network



(a) Network $\mathcal{H}(i, s)$


$\overline{2,3}$
$(c^{31}_{2j} - \alpha^1_j)$
$(c^{32}_{2j} - \alpha^2_j)$
$(c^{34}_{2j} - \alpha^4_j)$
0
1, 1
2, 1
3, 1
1
1, 2
2, 2
3, 2
2
1, 4
2, 4
3, 4
$\underline{2,3}$

(b) $\mathcal{H}(2, 3)$ for $n = 4$ and $m = 3$



(c) Candidate star

Figure 8: Subproblem network

depicted in Figure 8c, assume that after solving $P(2, 3)$ we obtain the star described in the figure, and when solving $P(1, 4)$ we obtain a star with the same set of nodes but centered at $(1, 4)$. Clearly, from the definition of the star costs presented in (7), among those stars, we should only consider the one with the minimum cost. Thus, while solving the subproblems, we create a hash map that, in case more than one star is generated with the same set of nodes, keeps only the one with the minimum cost.

3.5. Stabilization techniques

Finally, it is well known that the SP formulation is highly degenerate. Note for example that for the MSAP case, SP has in total mn constraints (without including the bounds on the variables), whereas any feasible integer solution would only have m nonzero variables. Primal degeneracy is well known to cause slow convergence (Lübbecke and Desrosiers (2005)). One of the principal causes of this behavior is the instability of the dual variables. Unfortunately, they do not smoothly converge to their respective optima, as they constantly oscillate without a regular pattern. Since the columns generated depend on the dual variable values, effects on the quality of solutions and on the running time pose an important issue that must be considered.

Several methods have been proposed to overcome this issue (Agarwal et al. (1989); Marsten et al. (1975); Rousseau et al. (2007)). Besides applying such methods, it is also known that when using an interior point method to solve the RMP, the dual solutions that are produced are often more balanced (Bixby et al. (1992)), leading to a better performance. We are aware though, that stabilization strategies generally outperform this option. However, for the purpose of this paper, the combination of an interior point method and the fact that we often include several star candidates (possibly one for every P(i, s)), proves to be sufficient.

4. Computational results

To analyze the performance of the different formulations, we created a test bed of 37 problem sets, ranging n from 3 to 20 and m from 4 to 30. We refer to each set as $n\text{D}m$. Moreover, every set is comprised of 20 random instances generated using the method described in (Grundel and Pardalos (2005)), which was designed to produce difficult MAP instances. The full set of instances ($37 \times 20 = 740$ in total) can be found in `http://www.caopt.com/instances/MAP`.

The computational experiments were performed on a server with two AMD Opteron 6128 Eight-Core CPUs and 12 GB of RAM, running Linux x86_64, CentOS 5.9. Formulations MIP and $MIPa$ were implemented in C++ and solved using CPLEX 12.3, while the branch–and–price framework for the SP was coded in C and solved using SCIP (Berthold et al. (2012)).

In our experiments we compare the performance of formulations SP, MIP, and $MIPa$ without the minimum sum, 4-cycle, and triplet cuts, as well as the $MIPa$ including the cuts at (1) the root node ($MIPa$-R), and (2) all nodes of the branch-and-bound tree ($MIPa$-A). For all the approaches, we imposed a time limit of 7,200 seconds.

Table 1 reports the average computational time for all the problem sets, where the fastest times are presented in bold. In parentheses we include (1) the average optimality gap of the instances that were not solved by the time limit and (2) the number of instances that were not solved to optimality, if any. The gap was calculated as (Primal bound-Dual bound)/Primal bound. Table 2 compares the average number of nodes of the branch-and-bound tree, while also presenting the average number of inequalities that were produced for each of the three families of cuts. The average number of cuts generated at the root node is given in parentheses. For the sake of simplicity, these values were rounded up to the next integer.

First of all, examining Table 1 we can easily observe that MIP is generally outperformed. This is reasonable, since the overhead of adding the linearization variables renders it very slow. We can also observe that SP is very fast in the instances where n is small. However, as it increases, SP becomes significantly slower than the $MIPa$, with and without the cuts. Furthermore, in the smaller instances, the cut families introduced are not as effective, which is to be expected. Hence, the solution times of $MIPa$, $MIPa$-R, and $MIPa$-A are comparable. On the other hand, in larger instances, the effect of the cuts becomes visible, rendering $MIPa$-R a clear winner.

Note that, because of the complexity of the separation algorithms, $MIPa$-R is faster than $MIPa$-A. Nonetheless, as it can be seen in Table 2, the branch-and-bound tree for $MIPa$-A is remarkably smaller than $MIPa$-R, which in turn is smaller than $MIPa$. Finally, we report the number of cutting planes introduced at each problem set. Observe that all families of inequalities are used, being the minimum sum cuts the ones that appear more often.

Finally, Table 3 reports additional statistics about the performance of formulation SP, including the average number of nodes of the branch-and-price tree, the number of times the subproblem (i.e., the star generation algorithm) was executed, the number of candidate stars produced, and the average total time spent solving the subproblem. For the sake of simplicity, with the exception of the execution times, the values were rounded up to the next integer.

As can be seen in Table 3, the execution time of the subproblem is in average about 70% of the total running time of the algorithm. This is expected because each of the subproblem calls corresponds to solving a total of mn shortest path problems with side constraints. For this reason, the time required to solve the RMP compared to the one needed to generate the new candidate stars is almost negligible. Moreover, this percentage is even higher ($\approx 95\%$) for the instances that where not solved to optimality. This is not only because those are the problems with lager sizes, but also because the corresponding branch-and-bound trees have more nodes and therefore, more side constraints are introduced to the shortest path problems. It is also interesting to note that

Table 1: Computational times, optimality gaps and the number of unsolved instances.

Set	MIP	MIPa	MIPa-R	MIPa-A	SP
3D5	0.02	0.01	**0.00**	**0.00**	**0.00**
3D10	0.66	0.09	0.08	0.10	**0.00**
3D15	1.58	0.16	0.15	0.17	**0.01**
3D20	31.61	1.56	1.46	1.61	**0.01**
3D25	55.04	1.83	1.78	1.97	**0.02**
3D30	67.64	3.27	3.27	3.45	**0.03**
4D5	0.11	0.01	**0.00**	0.01	**0.00**
4D10	5.31	0.64	0.60	0.66	**0.03**
4D15	149.65	7.55	7.65	12.95	**0.16**
4D20	266.37	11.83	11.44	18.20	**0.32**
4D25	3,427.10	103.90	99.27	188.70	**2.68**
4D30	7,200.00 (11.31%;20)	1,193.00	836.20	1,447.00	**8.76**
5D5	0.90	0.20	0.18	0.22	**0.01**
5D10	55.25	4.45	4.39	6.31	**0.18**
5D15	581.67	31.36	29.42	45.13	**5.30**
5D20	7,200.00 (2.47%;20)	518.09	**196.80**	952.12	454.49
5D25	7,200.00 (14.42%;20)	7,152.00	**4,104.00**	7,200.00 (0.58%; 20)	**3,758.90** (0.88%;9)
6D5	4.16	0.62	0.59	0.75	**0.08**
6D10	110.96	10.04	**8.17**	17.81	13.60
6D15	7,200.00 (4.83%; 20)	592.29	**463.48**	1,017.20	2763.94 (0.05%;4)
7D5	5.60	0.98	**0.79**	1.21	0.86
7D10	1,745.90	79.61	**69.75**	143.13	1738.21 (0.30%;3)
7D15	7,200.00 (7.51%;20)	7,200.00 (1.02%;20)	6,120.00 (0.11%;1)	7,200.00 (1.03%;20)	**4,871.15**
8D5	21.11	1.96	1.45	3.19	**0.50**
8D10	5,209.30 (0.91%;3)	323.73	**229.08**	638.23	4,614.73 (2.01%;12)
9D5	26.13	2.79	**2.31**	3.18	2.65
9D10	7,200.00 (11.54%;20)	581.02	**389.30**	912.84	7,200.00 (9.34%;20)
10D5	105.35	11.34	**10.81**	17.61	16.34
10D10	7,200.00 (23.23%;20)	706.98	**631.12**	2,791.70	7,200.00 (14.91%;20)
11D5	152.45	22.77	**20.69**	31.48	1,344.10
11D10	7,200.00 (26.11%;20)	7,200.00 (2.03%;20)	**7,193.80 (0.46%;19)**	7,200.00 (10.08%;20)	7,200.00 (34.5%;20)
12D5	403.09	36.75	**31.81**	50.66	774.42
12D10	7,200.00 (31.92%;20)	7,200.00 (21.46%;20)	**7,200.00 (7.54%;20)**	7,200.00 (22.92%;20)	7,200.00 (39.22%;20)
15D5	2,137.28	68.63	**58.85**	185.11	7,200.00 (16.81%;20)
15D10	7,200.00 (43.33%;20)	7,200.00 (29.83%;20)	**7,200.00 (18.01%;20)**	7,200.00 (32.46%;20)	7,200.00 (41.42%;20)
20D4	7,200.00 (17.11%;20)	248.93	**239.57**	439.76	7,200.00 (18.34%;20)
20D5	7,200.00 (29.96%;20)	579.41	**415.41**	1,016.20	7,200.00 (29.60%;20)

the average execution time of each of the individual mn subproblems is very short. For example, consider the execution time of the 20D5 instances. The average time that takes evaluating each of the shortest paths is $6,625.83/(6,556 \times 20 \times 5) = 0.01$ seconds.

5. Concluding remarks

In this paper, we presented a reformulation of the MAP with decomposable costs, where each of the assignments is assumed to be a star. We proposed a continuous nonlinear formulation for the problem, and its linearization into a mixed integer program. In addition to that, we proposed a series of valid inequalities to strengthen the formulation. Last, we also implemented a branch–and–price framework to solve a set partitioning reformulation of the problem. All the approaches were compared, and the families of cuts introduced proved to be very efficient at tackling large–scale instances.

Our work stemmed from the multi–sensor multi–target tracking problem, for which the MAP is widely used. We show that our approach is also a viable option for solving these problems. An advantage of this formulation is the fact that the center of every star plays the important role of a representative measurement. Further, the effect of noisy sensors is alleviated, compared to employing the MAP formulation.

Table 2: Number of nodes and cuts generated.

	Number of nodes				Number of cuts		
Set	*MIP*	*MIPa*	*MIPa*-R	*MIPa*-A	Minimum sum	4-Cycle	Triplet
3D5	0	0	0	0	0 (0)	0 (0)	0 (0)
3D10	0	0	0	0	1 (0)	0 (0)	0 (0)
3D15	14	14	2	2	43 (37)	0 (0)	3 (2)
3D20	96	82	51	38	186 (43)	0 (0)	25 (3)
3D25	101	128	60	44	255 (104)	0 (0)	39 (11)
3D30	113	160	81	64	1,007 (376)	0 (0)	51 (19)
4D5	2	1	0	0	5 (3)	1 (0)	2 (1)
4D10	37	61	31	18	296 (28)	24 (8)	21 (4)
4D15	45	79	45	33	627 (231)	32 (8)	24 (6)
4D20	320	260	81	59	5,631 (384)	72 (8)	90 (6)
4D25	2,798	2,839	190	141	10,823 (491)	136 (24)	181 (10)
4D30	12,195	14,047	2,601	520	15,493 (1,097)	1,193 (31)	1,447 (24)
5D5	14	14	13	9	44 (24)	20 (3)	6 (1)
5D10	220	219	108	63	1,030 (80)	152 (8)	38 (4)
5D15	988	1,327	618	313	8,715 (154)	400 (17)	126 (6)
5D20	1,186	16,758	7,184	686	21,153 (460)	536 (26)	261 (15)
5D25	1,475	38,400	23,305	953	26,542 (1,239)	646 (29)	293 (16)
6D5	3	3	0	0	15 (12)	1 (0)	1 (0)
6D10	85	191	99	38	1,219 (57)	208 (24)	14 (0)
6D15	14,096	37,467	12,216	829	20,212 (230)	936 (25)	219 (4)
7D5	4	5	4	3	18 (14)	0 (0)	0 (0)
7D10	1,085	1,614	1,059	414	14,277 (72)	976 (8)	123 (1)
7D15	17,211	33,006	17,139	625	19,219 (676)	664 (40)	136 (3)
8D5	11	16	14	9	60 (7)	18 (3)	1 (0)
8D10	914	9,107	3,896	451	21,428 (127)	1,472 (16)	163 (1)
9D5	101	168	94	26	1,044 (24)	512 (8)	15 (0)
9D10	1,044	19,015	11,927	570	17,361 (241)	1,232 (18)	102 (2)
10D5	118	203	79	45	1,309 (49)	368 (9)	17 (1)
10D10	1,693	32,076	14,421	821	12,072 (115)	1,736 (14)	130 (3)
11D5	185	302	95	67	1,632 (58)	440 (8)	31 (1)
11D10	2,001	38,303	13,082	1,094	21,102 (122)	2,491 (10)	157 (3)
12D5	257	506	190	114	2,275 (56)	501 (6)	52 (1)
12D10	3,974	38,900	19,510	1,827	24,893 (445)	2,708 (9)	158 (2)
15D5	501	544	201	147	2,291 (37)	607 (12)	53 (1)
15D10	5,283	39,975	26,208	2,340	28,991 (1,207)	2,806 (44)	201 (3)
20D4	61	214	86	36	2,429 (216)	1,936 (32)	148 (1)
20D5	273	1,016	277	64	3,809 (309)	2,048 (33)	197 (1)

As far as future work is concerned, as seen in the computational results, the number of minimum sum cuts produced is very high, even though they are effective. A method to smartly choose the best of these cuts would result in a smaller model size, which could prove beneficial. Moreover, especially in large–scale instances, a decomposition scheme could be used in order to reduce the size of every subproblem, making it manageable in time. More specifically, Benders' decomposition, along with graph partitioning schemes, might be of interest.

Last, scientific interest has turned recently to the need for a decentralized algorithm for solving multi–sensor multi–target tracking problems. In this setting, every sensor decides on-the-fly which other sensors it is paired with. Such extensions would involve game theoretic approaches, as in (Marden and Shamma (2007)), or swarm optimization (Özbakir et al. (2010)). Another extension could focus on further investigating different potential MAP relaxations, such as the detection of disjoint trees, or quasicliques.

Acknowledgements

The authors are thankful to the editor and the anonymous referees for their insightful comments that contributed to improving the quality of the paper.

Table 3: Statistics of the SP formulation.

Set	Nodes	Subproblem calls	Variables	Subproblem time
3D5	1	5	30	0.00
3D10	1	8	91	0.00
3D15	1	9	192	0.00
3D20	2	11	277	0.00
3D25	2	12	393	0.01
3D30	2	14	540	0.01
4D5	2	8	55	0.00
4D10	10	29	254	0.01
4D15	42	94	535	0.09
4D20	44	100	808	0.19
4D25	213	340	1,245	2.02
4D30	401	578	1,621	7.13
5D5	5	18	123	0.00
5D10	44	111	588	0.09
5D15	309	533	1,285	4.28
5D20	2,177	2,792	2,410	442.58
5D25	7,822	9,344	4,238	3,690.60
6D5	18	55	313	0.03
6D10	344	704	1,456	11.17
6D15	3,869	5,436	4,370	2,690.06
7D5	24	87	476	0.06
7D10	2,890	4,673	4,966	1,683.83
7D15	3,513	5,896	7,608	4,699.18
8D5	44	153	778	0.25
8D10	3,915	7,298	9,674	4,453.21
9D5	176	540	1,834	1.75
9D10	3,140	7,782	19,681	6,903.88
10D5	523	1,528	4,177	12.65
10D10	3,968	10,930	29,839	7,004.02
11D5	1,344	3,894	9,425	192.02
11D10	1,868	7,650	35,122	7,048.93
12D5	1,731	5,249	13,175	737.12
12D10	2,011	7,911	38,777	7,098.14
15D5	2,612	11,901	52,510	6,582.72
15D10	821	4,780	39,019	6,328.91
20D4	1,322	9,307	63,592	6,641.76
20D5	846	6,556	62,033	6,625.83

Appendix A. Mathematical proofs

Proof of Lemma 1. Let node (i, s) be white. From (9) and (13), since $z_i^s = 0$, we have that

$$\sum_{j=1}^{m}\sum_{t=1}^{n} x_{ij}^{st} = 1$$

and,

$$\sum_{j=1}^{m}\sum_{t=1}^{n} x_{ij}^{st} z_j^t = 1.$$

Thus,

$$\sum_{j=1}^{m}\sum_{t=1}^{n} x_{ij}^{st} = \sum_{j=1}^{m}\sum_{t=1}^{n} x_{ij}^{st} z_j^t,$$

which can be rewritten as

$$\sum_{j=1}^{m}\sum_{\{t=1,\ldots,n:z_j^t=0\}} x_{ij}^{st} + \sum_{j=1}^{m}\sum_{\{t=1,\ldots,n:z_j^t>0\}} x_{ij}^{st} =$$

$$= \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: z_j^t = 0\}} x_{ij}^{st} z_j^t + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: z_j^t > 0\}} x_{ij}^{st} z_j^t = \sum_{\{j=1,\ldots,m: z_j^t > 0\}} x_{ij}^{st} z_j^t. \tag{A.1}$$

Moreover, since $x_{ij}^{st} \in [0,1]$ and $z_j^t \in [0,1]$,

$$\sum_{\{j=1,\ldots,m: z_j^t > 0\}} x_{ij}^{st} \geq \sum_{\{j=1,\ldots,m: z_j^t > 0\}} x_{ij}^{st} z_j^t.$$

Hence, from (A.1) we obtain

$$\sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: z_j^t = 0\}} x_{ij}^{st} = 0, \text{ and} \tag{A.2}$$

$$\sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: z_j^t > 0\}} x_{ij}^{st} = \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: z_j^t > 0\}} x_{ij}^{st} z_j^t. \tag{A.3}$$

Equations (A.2) and (A.3) imply that there are no connections between white nodes, and white and gray nodes, respectively. □

Proof of Lemma 2. Let node (i, s) be black, i.e., $z_i^s = 1$. From (9), and (13) respectively, we have

$$\sum_{j=1}^{m} \sum_{t=1}^{n} x_{ij}^{st} = n - 1, \text{ and} \tag{A.4}$$

$$\sum_{j=1}^{m} \sum_{t=1}^{n} x_{ij}^{st} z_j^t = \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st} = 0\}} x_{ij}^{st} z_j^t + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st} > 0\}} x_{ij}^{st} z_j^t =$$

$$\sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st} > 0\}} x_{ij}^{st} z_j^t = 0 \tag{A.5}$$

Now, notice that in order to satisfy (A.5), if $x_{ij}^{st} > 0$, then $z_j^t = 0$. This, in turn, implies that a black node cannot be connected to other black nodes, or gray nodes. □

Proof of Lemma 3. Assume there exist k black nodes. Then, each of those nodes is connected to $(n-1)$ white nodes, overall covering kn nodes in the graph. Hence, there must be exactly $(m-k)n$ gray nodes remaining to be covered. □

Proof of Theorem 1. Assume for a contradiction that there exist gray nodes. First, note that from Lemma 3, if there exists a gray node in $\mathcal{G}$, we can assume without loss of generality that every node is also gray. This is because we can remove all black and white nodes from the graph, and the remaining (gray) nodes still form an n–partite graph.

Since all nodes are gray, we have that $z_i^s \in (0, 1)$, for node (i, s). Further, constraint (13) can be rewritten as

$$z_i^s + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st} > 0\}}^{n} x_{ij}^{st} z_j^t = 1. \tag{A.6}$$

Now, consider the following expression

$$z_i^s + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st}>0\}} z_j^t. \tag{A.7}$$

Clearly, expression (A.7) must be either less than, equal to, or greater than 1. Thus, we divide the proof into these three cases.

Case 1. Let us assume that it is less than 1. It is easy to see that

$$z_i^s + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st}>0\}}^{n} z_j^t \geq z_i^s + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st}>0\}} x_{ij}^{st} z_j^t,$$

which implies that

$$z_i^s + \sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st}>0\}}^{n} x_{ij}^{st} z_j^t < 1,$$

contradicting (A.6).

Case 2. Assume (A.7) is equal to one. To satisfy constraint (A.6), all x_{ij}^{st} that are positive must be equal to one. Further, from constraint (11), we get that node (i,s) is connected to at least one node from each set N_t, $t \neq s$. Since all connections emanating from (i,s) satisfy $x_{ij}^{st} = 1$ we obtain

$$\sum_{j=1}^{m} \sum_{\{t=1,\ldots,n: x_{ij}^{st}>0\}}^{n} x_{ij}^{st} \geq n-1.$$

By assumption node (i,s) is gray. Hence, since $z_i^s < 1$, we would violate constraint (9).

Case 3. Consider the case where expression (A.7) is greater than one. First, note that $z_i^s > 0$ implies that node (i,s) is partially assigned to be part of a star centered at itself. Moreover, for each node (j,t) such that $x_{ij}^{st} > 0$, since $z_j^t > 0$, node (i,s) is also partially assigned to be part of a star centered at (j,t). Thus, expression (A.7) accounts for the number of stars that node (i,s) is assigned to, including the one centered at itself. We proved above that this expression cannot be less than or equal to one for any gray node. Hence, this implies that all nodes in $\mathcal{G}$ are associated with more than one star. By constraint (12) there are only m stars. Also, the sum of the $\mathbf{x}$ variables of each star must be equal to $n-1$. Thus, if (A.7) is greater than one for all nodes in $\mathcal{G}$, we would violate at least one of the constraints in (9).

The three contradictions described above imply that in any feasible solution of the $RNLP$ there are no gray nodes. □

Proof of Theorem 2. Theorem 1 implies that, in any feasible solution, there are exactly m black nodes. Without loss of generality, assume that all of those nodes are elements of the same set N_s (see Figure 4). Assume there is an optimal solution $(\mathbf{z}^*, \mathbf{x}^*)$, where $\mathbf{x}^*$ is fractional. Observe that if we fix $\mathbf{z} = \mathbf{z}^*$ in (8)–(15), the resulting formulation can be divided into $n-1$ independent linear assignment problems between the elements in N_s (the black nodes) and the elements in $N_t, t \neq s$, respectively (the white nodes at each set). Note, that since there always exists an integer optimal solution in a linear assignment problem, we can deduce that there is always an alternative optimal solution for the $RNLP$ where $x_{ij}^{st} \in \{0,1\}$, for all the edges in $\mathcal{G}$. □

Appendix B. Dynamic programming algorithm for finding candidate stars

There are different algorithms available in the literature to solve the shortest path problem with side constraints (e.g., Irnich and Desaulniers (2005); Lozano and Medaglia (2013); Santos et al. (2007)). We opted for adapting the dynamic programming approach described by Irnich and Desaulniers (2005), which, based on our experiments, was sufficient for our purposes. The proposed approach is as follows. Let Ω be the set of branching constraints. As mentioned in Section 3, each branching constraint $\omega \in \Omega$ has a pair of nodes (we refer to those as $\mathcal{N}(\omega) = \{(i,s),(j,t)\}$) associated with it and represents the branch in which the given nodes must be either covered by different stars (`DIFF`) or by the same star (`SAME`). Let $type(\omega)$ be a function that returns the type of the constraint. At any stage of the algorithm, we use $\mathcal{P}$ to refer to the set of intermediate candidate paths (i.e., paths that have not reached the sink node). Associated with each candidate path $P \in \mathcal{P}$, we use $P.cost$, $P.last$, and $P.l$ to represent the current cost, the last node, and the vector of labels of path P, respectively. The vector of labels $P.l$ contains one entry for each branching constraint $\omega \in \Omega$ and helps to determine if the candidate path is feasible. Depending on which type branching constraint ω is of and the nodes that comprise path P, label $P.l[\omega]$ can take the following possible values:

$$P.l[\omega] = \begin{cases} 2, & \text{if both nodes in } \mathcal{N}(\omega) \text{ are part of } P \text{ (only for } type(\omega) = \texttt{SAME}) \\ 1, & \text{if only one node in } \mathcal{N}(\omega) \text{ is part of } P \\ 0, & \text{if no node in } \mathcal{N}(\omega) \text{ is currently part of } P, \\ \text{-1}, & \text{if no node in } \mathcal{N}(\omega) \text{ can be added to } P \text{ (only for } type(\omega) = \texttt{SAME}). \end{cases}$$

All of the previous values of $P.l[\omega]$ are self-explanatory with the exception of $P.l[\omega] = -1$. Assuming that $\mathcal{N}(\omega) = \{(i,s),(j,t)\}$, this case corresponds to the situation where there is already a node (k,u) in P, such that either $u = s$ or $u = t$. Since in each star there must be exactly one node from each set, this implies that neither (i,s) nor (j,t) can be included in P.

If a feasible path P has reached the sink node and for each constraint $\omega \in \Omega$, such that $type(\omega) =$`SAME`, we have that $P.l[w]$ is either 0 or 2, we say that P is a *complete path*. Note that if P has reached the sink and there is still one of such constraints for which $P.l[w] = 1$, the path P is infeasible as it contains only one of the nodes of constraint ω. Finally, we name $\mathcal{Q}$ as the set of complete paths that have been generated. The proposed approach is presented in algorithms 5–7.

The `initialize` function (see Algorithm 5) creates a path that begins at the source node with an initial cost of $-\alpha_i^s$. It also updates the constraint labels according to the values described above.

The `append` function (see Algorithm 6) identifies if appending a node to the end of path P is a feasible option. In such case, the function updates the constraint labels as well as the cost. If the resulting path is not feasible, it returns an empty path. Note that, if the node v that is being added is either a member of set $\bar{N}$ or is equal to node $\underline{(i,s)}$(see step 2), the algorithm appends it at the end of P as doing so does not affect the feasibility of the path (i.e., v is not present in any constraint of set Ω).

The quality of Algorithm 7 is highly dependent on (1) which path is selected at each iteration (step 5) and (2) how and when the dominance test is performed (step 16). The selection criteria that we use is based on the fact that every complete path $P \in \mathcal{Q}$ will pass through the same number of nodes, i.e., it will have the same size (see Figure 8 for an example). This is because each complete path is associated with a feasible star and each feasible star has in total n nodes. Moreover, because of the particular structure of network $\mathcal{H}(i,s)$, it is always posible to keep a difference of at most one node between the sizes of all candidate paths in set $\mathcal{P}$. This can be done by always selecting in step

Algorithm 5 `initialize`(G, Ω, i, s)

```
1: $P \leftarrow \{(\overline{i, s})\}$
2: $P.cost \leftarrow -\alpha_i^s$
3: for all $\omega \in \Omega$ do
4:   if $(i, s) \in \mathcal{N}(\omega)$ then
5:     $P.l[\omega] = 1$
6:   else if $type(\omega) = \texttt{SAME}$   and   $\exists (j, t) \in \mathcal{N}(\omega)$ such that $t = s$ then
7:     $P.l[\omega] = -1$
8:   else
9:     $P.l[\omega] = 0$
10:  end if
11: end for
12: return $P$
```

5 a path P with the currently smallest number of nodes. The reason for this, is that performing the dominance test is easier when all the candidate paths are of the same size.

The objective of the dominance test (step 16) is to discard candidate paths that are dominated by other paths in $\mathcal{P}$. That is, if we have two paths P^1 and P^2 for which $P^1.last = P^2.last$, we say that P^1 dominates P^2 if:

- for all $\omega \in \Omega$ with $type(\omega) = \texttt{SAME}$, $P^1.l[\omega]$ and $P^2.l[\omega]$ are both equal to 1, or take the value of 0 or 2 (e.g., $P^1.l[\omega] = 0$ and $P^2.l[\omega] = 2$ is a valid possibility);
- for all $\omega \in \Omega$ with $type(\omega) = \texttt{DIFF}$, $P^1.l[\omega] \leq P^2.l[\omega]$;
- $P^1 cost \leq P^2.cost$.

Clearly, if P^1 dominates P^2, any complete path constructed by extending P^1 will yield, in the worst case, a path with the same cost as the one constructed by extending P^2 with the same set of nodes. For this reason, there is no gain in considering P^2 as a candidate path and thus, it can be removed from $\mathcal{P}$. Because the execution time of the dominance test depends on the number of branching constraints and the size of $\mathcal{P}$, based on our experiments, the best performance we obtained was when the dominance test was performed only after reaching each node $v \in \bar{N}$.

References

Agarwal, Y., Mathur, K., Salkin, H. M., 1989. A set-partitioning-based exact algorithm for the vehicle routing problem. Networks 19 (7), 731–749.

Ahuja, R. K., Magnanti, T. L., Orlin, J. B., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, New Jersey, USA.

Aiex, R. M., Resende, M. G. C., Pardalos, P. M., Toraldo, G., 2005. Grasp with path relinking for three-index assignment. INFORMS Journal on Computing 17 (2), 224–247.

Aneja, Y. P., Punnen, A. P., 1999. Multiple bottleneck assignment problem. European Journal of Operational Research 112 (1), 167–173.

Appa, G., Magos, D., Mourtos, I., 2006. On multi-index assignment polytopes. Linear Algebra and its Applications 416 (2–3), 224–241.

Balas, E., Saltzman, M. J., 1989. Facets of the three-index assignment polytope. Discrete Applied Mathematics 23 (3), 201–229.

Balas, E., Saltzman, M. J., 1991. An algorithm for the three-index assignment problem. Operations Research 39 (1), 150–161.

Bandelt, H. J., Crama, Y., Spieksma, F. C. R., 1994. Approximation algorithms for multi-dimensional assignment problems with decomposable costs. Discrete Applied Mathematics 49 (1–3), 25–50.

Algorithm 6 append(G, v, P)

```
1: if $v \in \bar{N}$ or $v = (\underline{i, s})$ then
2:    $P \leftarrow P \cup \{v\}$
3:    return $P$
4: end if
5: $(j, t) \leftarrow v$
6: for all $\omega \in \Omega$ do
7:    if $(j, t) \in \mathcal{N}(\omega)$ then
8:       if $type(\omega) = \texttt{DIFF}$ then
9:          if $P.l[\omega] = 1$ then
10:            return $\emptyset$
11:         else
12:            $P.l[\omega] \leftarrow 1$
13:         end if
14:      else
15:         if $P.l[\omega] = 1$ then
16:            $P.l[\omega] \leftarrow 2$
17:         else if $P.l[\omega] = 0$ then
18:            $P.l[\omega] \leftarrow 1$
19:         else
20:            return $\emptyset$
21:         end if
22:      end if
23:   else if $type(\omega) = \texttt{SAME}$ and $\exists (k, u) \in \mathcal{N}(\omega)$ such that $u = t$ then
24:      $P.l[\omega] = -1$
25:   end if
26: end for
27: $P \leftarrow P \cup v$
28: $P.cost \leftarrow P.cost + c_{ij}^{st} - \alpha_j^t$
29: return $P$
```

Bandelt, H. J., Maas, A., Spieksma, F. C. R., 2004. Local search heuristics for multi-index assignment problems with decomposable costs. The Journal of the Operational Research Society 55 (7), 694–704.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46 (3), 316–329.

Bazaraa, M. S., Jarvis, J. J., Sherali, H. D., 2009. Linear Programming and Network Flows. Wiley.

Berthold, T., Gamrath, G., Gleixner, A. M., Heinz, S., Koch, T., Shinano, Y., 2012. Solving mixed integer linear and nonlinear problems using the SCIP Optimization Suite. ZIB-Report 12–17, Zuse Institute Berlin, Takustr. 7, 14195 Berlin.

Bixby, R. E., Gregory, J. W., Lustig, I. J., Marsten, R. E., Shanno, D. F., 1992. Very large-scale linear programming: A case study in combining interior point and simplex methods. Operations Research 40 (5), 885–897.

Burkard, R. E., 2002. Selected topics on assignment problems. Discrete Applied Mathematics 123 (1–3), 257–302.

Burkard, R. E., Çela, E., 1999. Linear assignment problems and extensions. In: Du, D.-Z., Pardalos, P. M. (Eds.), Handbook of Combinatorial Optimization. Springer US, pp. 75–149.

Burkard, R. E., Çela, E., Pardalos, P. M., Pitsoulis, S., 1998. The quadratic assignment problem. In: Du, D.-Z., Pardalos, P. M. (Eds.), Handbook of Combinatorial Optimization. Springer US, pp. 241–338.

Burkard, R. E., Rudolf, R., Woeginger, G. J., 1996. Three-dimensional axial assignment problems with decomposable cost coefficients. Discrete Applied Mathematics 65 (1–3), 123–139.

Chummun, M. R., Kirubarajan, T., Pattipati, K. R., Bar-Shlom, Y., 2001. Fast data association using multidimensional assignment with clustering. Aerospace and Electronic Systems, IEEE Transactions on 37 (3), 898–913.

Algorithm 7 `findCandidateStar`(G, m, n, Ω, i, s)

```
1:  $P \leftarrow$ initialize$(G, \Omega, i, s)$
2:  $\mathcal{P} \leftarrow \{P\}$
3:  $\mathcal{Q} \leftarrow \emptyset$
4:  while $\mathcal{P} \neq \emptyset$ do
5:      select a path $P \in \mathcal{P}$ and remove it from $\mathcal{P}$
6:      for all node $v$ in the forward star of $P.last$ do
7:          $P' \leftarrow$ append $(G, v, P)$
8:          if $P' \neq \emptyset$ then
9:              if $P'$ is complete then
10:                 $\mathcal{Q} \leftarrow \mathcal{Q} \cup P'$
11:             else if $P'$ has not reached the sink then
12:                 $\mathcal{P} \leftarrow \mathcal{P} \cup P'$
13:             end if
14:         end if
15:     end for
16:     perform a dominance test to filter set $\mathcal{P}$
17: end while
18: if $\mathcal{Q} = \emptyset$ then
19:     return $\emptyset$
20: end if
21: find $P^* = \arg\min_{P \in \mathcal{Q}}\{P.cost\}$
22: reconstruct the candidate star $S$ from solution $P^*$
23: return $S$
```

Clemons, W. K., Grundel, D. A., Jeffcoat, D. E., 2004. Applying simulated annealing to the multidimensional assignment problem. In: Grundel, D. A., Murphey, R., Pardalos, P. M. (Eds.), Theory and algorithms for cooperative systems. World Scientific, pp. 45–61.

Crama, Y., Spieksma, F. C. R., 1992. Approximation algorithms for three-dimensional assignment problems with triangle inequalities. European Journal of Operational Research 60 (3), 273–279.

Deb, S., Pattipati, K. R., Bar-Shalom, Y., 1993. A multisensor-multitarget data association algorithm for heterogeneous sensors. Aerospace and Electronic Systems, IEEE Transactions on 29 (2), 560–568.

Deb, S., Yeddanapudi, M., Pattipati, K., Bar-Shalom, Y., 1997. A generalized SD assignment algorithm for multisensor-multitarget state estimation. Aerospace and Electronic Systems, IEEE Transactions on 33 (2), 523–538.

Desaulniers, G., Desrosiers, J., Spoorendonk, S., 2011. Cutting planes for branch-and-price algorithms. Networks 58 (4), 301–310.

Edmonds, J., Karp, R. M., 1972. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM) 19 (2), 248–264.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44 (3), 216–229.

Frieze, A. M., Yadegar, J., 1981. An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. The Journal of the Operational Research Society 32 (11), 989–995.

Gaofeng, H., Lim, A., 2003. A hybrid genetic algorithm for three-index assignment problem. In: The 2003 Congress on Evolutionary Computation, 2003. CEC '03. Vol. 4. pp. 2762–2768.

Gilbert, K. C., Hofstra, R. B., 1988. Multidimensional assignment problems. Decision Sciences 19 (2), 306–321.

Grundel, D. A., Oliveira, C. A. S., Pardalos, P. M., 2004. Asymptotic properties of random multidimensional assignment problems. Journal of Optimization Theory and Applications 122 (3), 487–500.

Grundel, D. A., Pardalos, P. M., 2005. Test problem generator for the multidimensional assignment problem. Computational Optimization and Applications 30 (2), 133–146.

Gutin, G., Karapetyan, D., 2009. A selection of useful theoretical tools for the design and analysis of optimization heuristics. Memetic Computing 1 (1), 25–34.

Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M. M. (Eds.), Column Generation. Springer US, pp. 33–65.

Karapetyan, D., Gutin, G., 2011a. Local search heuristics for the multidimensional assignment problem. Journal of Heuristics 17 (3), 201–249.

Karapetyan, D., Gutin, G., 2011b. A new approach to population sizing for memetic algorithms: A case study for the multidimensional assignment problem. Evolutionary Computation 19 (3), 345–371.

Karp, R. M., 2010. Reducibility among combinatorial problems. 50 Years of Integer Programming 1958-2008, 219–241.

Krokhmal, P. A., Grundel, D. A., Pardalos, P. M., 2007. Asymptotic behavior of the expected optimal value of the multidimensional assignment problem. Mathematical programming 109 (2), 525–551.

Kuhn, H. W., 1955. The hungarian method for the assignment problem. Naval research logistics quarterly 2 (1-2), 83–97.

Kuroki, Y., Matsui, T., 2009. An approximation algorithm for multidimensional assignment problems minimizing the sum of squared errors. Discrete Applied Mathematics 157 (9), 2124–2135.

Larsen, M., 2012. Branch and bound solution of the multidimensional assignment problem formulation of data association. Optimization Methods and Software 27 (6), 1101–1126.

Lozano, L., Medaglia, A. L., 2013. On an exact method for the constrained shortest path problem. Computers & Operations Research 40 (1), 378 – 384.

Lübbecke, M. E., Desrosiers, J., 2005. Selected topics in column generation. Operations Research 53 (6), 1007–1023.

Magos, D., Mourtos, I., 2009. Clique facets of the axial and planar assignment polytopes. Discrete Optimization 6 (4), 394–413.

Malhotra, R., Bhatia, H. L., Puri, M. C., 1985. The three dimensional bottleneck assignment problem and its variants. Optimization 16 (2), 245–256.

Marden, J., Shamma, J. S., 2007. Autonomous vehicletarget assignment: a game theoretical formulation. ASME Journal of Dynamic Systems, Measurement, and Control, 584–596.

Marsten, R. E., Hogan, W. W., Blankenship, J. W., 1975. The boxstep method for large-scale optimization. Operations Research 23 (3), 389–405.

Morefield, C. L., 1977. Application of 0-1 integer programming to multitarget tracking problems. Automatic Control, IEEE Transactions on 22 (3), 302–312.

Murphey, R. A., Pardalos, P. M., Pitsoulis, L., 1999. A parallel grasp for the data association multidimensional assignment problem. In: Pardalos, P. M. (Ed.), Parallel Processing of Discrete Problems. Vol. 106 of The IMA Volumes in Mathematics and its Applications. Springer New York, pp. 159–179.

Oliveira, C. A. S., Pardalos, P. M., 2004. Randomized parallel algorithms for the multidimensional assignment problem. Applied Numerical Mathematics 49 (1), 117–133.

Özbakir, L., Baykasoğlu, A., Tapkan, P., 2010. Bees algorithm for generalized assignment problem. Applied Mathematics and Computation 215 (11), 3782–3795.

Pardalos, P. M., Pitsoulis, L. S. E., 2000. Nonlinear assignment problems: algorithms and applications. Vol. 7. Springer.

Pasiliao, E. L., Pardalos, P. M., Pitsoulis, L. S., 2005. Branch and bound algorithms for the multidimensional assignment problem. Optimization Methods and Software 20 (1), 127–143.

Pentico, D. W., 2007. Assignment problems: A golden anniversary survey. European Journal of Operational Research 176 (2), 774–793.

Pierskalla, W. P., 1968. The multidimensional assignment problem. Operations Research 16 (2), 422–431.

Poore, A. B., 1994. Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. Computational Optimization and Applications 3 (1), 27–57.

Poore, A. B., Robertson III, A. J., 1997. A new lagrangian relaxation based algorithm for a class of multidimensional assignment problems. Computational Optimization and Applications 8 (2), 129–150.

Pusztaszeri, J.-F., Rensing, P. E., Liebling, T. M., 1996. Tracking elementary particles near their primary vertex: A combinatorial approach. Journal of Global Optimization 9 (1), 41–64.

Robertson III, A. J., 2001. A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. Computational Optimization and Applications 19 (2), 145–164.

Rousseau, L. M., Gendreau, M., Feillet, D., 2007. Interior point stabilization for column generation. Operations Research Letters 35 (5), 660 – 668.

Ryan, D. M., Foster, B. A., 1981. An integer programming approach to scheduling. Computer scheduling of public transport urban passenger vehicle and crew scheduling, 269–280.

Santos, L., Coutinho-Rodrigues, J., R., C. J., 2007. An improved solution algorithm for the constrained shortest path problem. Transportation Research Part B: Methodological 41 (7), 756 – 771.

Spieksma, F. C. R., 2000. Multi index assignment problems: complexity, approximation, applications. Nonlinear Assignment Problems: Algorithms and Applications 7, 1–12.

Valério de Carvalho, J. M., 1999. Exact solution of bin-packing problems using column generation and branch-and-bound. Annals of Operations Research 86 (0), 629–659.

Vogiatzis, C., Pasiliao, E. L., Pardalos, P. M., 2013. Graph partitions for the multidimensional assignment problem. Computational Optimization and Applications, (submitted).