INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 33 (2026) 892–925
DOI: 10.1111/itor.70052

An allocation rule for connection scheduling problems

Laura Davila-Pena[a,b,*], Peter Borm[c], Ignacio García-Jurado[d] and Jop Schouten[c]

[a] *Centre for Logistics and Sustainability Analytics (CeLSA), Department of Analytics, Operations and Systems, Kent Business School, University of Kent, Canterbury CT2 7PE, UK*
[b] *MODESTYA Research Group, Department of Statistics, Mathematical Analysis and Optimization, Faculty of Mathematics, Universidade de Santiago de Compostela, Campus Vida, Santiago de Compostela 15782, Spain*
[c] *Department of Econometrics & Operations Research, Tilburg University, P.O. Box 90153, Tilburg 5000 LE, the Netherlands*
[d] *CITMAga, MODES Research Group, Department of Mathematics, Universidade da Coruña, Campus de Elviña, A Coruña 15071, Spain*
E-mail: l.davila-pena@kent.ac.uk[Davila-Pena]; p.e.m.borm@tilburguniversity.edu[Borm]; ignacio.garcia.jurado@udc.es[García-Jurado]; j.schouten@tilburguniversity.edu[Schouten]

Received 16 July 2024; received in revised form 28 February 2025; accepted 29 April 2025

Abstract

This paper studies so-called connection scheduling problems, a type of interactive operations research problem. A connection scheduling problem combines aspects from the minimum cost spanning tree and sequencing problems. Given a graph, we aim to first establish a connection order on the players such that the total cost of connecting them to a source is minimal and second to find a fair cost allocation of such an optimal order among the players involved. We restrict our attention to connection scheduling problems on trees and propose a recursive method to solve these tree connection scheduling problems integrated with an allocation approach. This latter mechanism consistently and recursively uses benchmark endogenous myopic orders to determine potential cost savings, which will then be appropriately allocated. Interestingly, the transition process from a benchmark myopic order to an optimal one will be smooth using the switching of blocks of agents based on the basic notion of merge segments.

Keywords: cooperation; sequencing problems; connection scheduling problems; cost allocation

1. Introduction

As argued in Bergantiños et al. (2014), there are many real-life problems that require the construction of infrastructures to connect a set of agents to a source, either directly or indirectly. Examples include urban water supply from a general reservoir, connection to a supercomputer of users in a given region, communication with a coordination center for smart homes in a city, and energy

*Corresponding author.

supply to customers in a smart grid. This last example is of the utmost relevance for improving the sustainability of energy consumption and is generating a wide variety of operational problems that can be solved by mathematical programming; see, for instance, Bai et al. (2024). It involves the construction of infrastructures to connect all its customers to the grid, the objective being to connect all of them in a way that minimizes the total time involved. The construction time can be interpreted as a cost to be minimized. To address this problem, the standard minimum cost spanning tree (MCST) setting has been widely applied (see, e.g., Curiel, 1997, or Bergantiños and Lorenzo, 2004). In this way, the focus is on determining so-called MCSTs. A tree is a set of edges such that there is a single path from the source to each of the agents, and the cost of a tree is the sum of the costs of all the edges belonging to it. MCSTs can be computed in a polynomial time, and the most common methods are Kruskal's algorithm (Kruskal, 1956) and Prim's algorithm (Prim, 1957). A further issue of relevance in an interactive optimization setting is the allocation of costs among the different agents involved. An adequate allocation will serve to establish and maintain cooperation between the agents. In the context of MCST problems, Claus and Kleitman (1973) were the first to address this issue. Since then, numerous studies have approached the cost allocation problem from a game-theoretic perspective. Notable contributions to this area include the works of Granot and Huberman (1981, 1984), Feltkamp et al. (1994), Moretti et al. (2004), and Estévez-Fernández and Reijnierse (2014). Bergantiños and Vidal-Puga (2021) is a recent review within the context of MCST problems that relates the rules defined through cooperative games with rules defined from the problem itself.

Looking back at the smart grid example, it is often crucial for the agents to have continuous access to energy. As a result, they must contract an external service provider until the grid reaches them. Each agent has an associated coefficient that represents the cost per unit of time the infrastructure connecting the agent to the grid remains under construction. The total construction time depends on *when* the agent is connected to the grid. For example, if agent 2 connects through agent 1, the infrastructure for agent 1 must be built first, followed by the infrastructure linking agent 1 to agent 2. Thus, the total time to connect agent 2 would be the sum of the construction times for the infrastructure connecting agent 1 and the infrastructure connecting agent 1 to agent 2. The goal is to minimize the total aggregate costs, rather than just the overall construction time for the entire project. Situations such as these we call connection scheduling problems (CSPs).

One issue we would like to highlight is the proximity of our problem to a sequencing problem, of which we will now give a brief description. In deterministic one-machine sequencing problems, a set of jobs needs to be processed on a machine. Each of these jobs is identified with one agent and is associated with it: a processing time, that is, the time needed by the machine to process that specific job, and a cost function, which indicates how costly it is for that agent to spend a unit of time in the system. The main objective in sequencing problems consists of finding an optimal order, that is, an order on the jobs that minimizes the total aggregated costs of all agents. The cost of an agent will naturally depend on its completion time, and this dependence is linear in the classical model (Smith, 1956). There are, however, numerous variants of sequencing problems that allow for an adaptation to real situations. One of them is the sequencing problem with precedence constraints, where some jobs need to be processed before others, as first analyzed in Baker (1971). Other variants can be found in, for example, Cheng and Gupta (1989) and Adamopoulos et al. (1999). In most interactive sequencing problems, an initial order is assumed as a starting point and the focus is on the allocation of cost savings with respect to this initial order. For the classical model, Curiel

et al. (1989) introduce and axiomatically characterize an allocation rule, the *equal gain splitting rule* (EGS rule), based on neighbor switches to derive an optimal order from the initial one, which was later generalized in Hamers et al. (1996). Sequencing problems have been widely studied from a game-theoretic perspective; see, for instance, Curiel et al. (2002) for a survey and Saavedra-Nieves et al. (2025) for a more recent contribution.

A CSP is related to an MCST problem. However, in a CSP, the costs are computed in a different way. In particular, the order in which the edges are activated has a substantial effect on the cost in our setup, whereas this order is irrelevant in calculating the costs in an MCST problem. Besides, the CSP is closely linked to sequencing problems with precedence constraints. Although interactive sequencing problems with precedence constraints have been treated in the literature before (see Hamers et al., 2005), the approach under which we will study them here has, to the best of our knowledge, never been adopted. In our setting, we start from a graph $(N \cup \{0\}, E)$, where N is the set of nodes corresponding to the agents, 0 is the source node that must serve all agents, and E is a set of edges connecting the nodes. Each edge has a specific activation time, and each agent has a cost depending on the time it gets connected. We will aim, on the one hand, to find an optimal connection order on the agents such that the corresponding total aggregate connection costs over all players are minimized. Note that a connection order on the agents induces an activation order on the edges. On the other hand, we aim to find a fair allocation of these minimal costs. As we will explain below, we will tackle our first objective by reformulating existing algorithms in a way that allows us to also address the second goal, which constitutes the main contribution of our work. Recall that we are trying to allocate costs in a setting without initial connection rights, and thus we will start by using an endogenous myopic order that serves as a natural benchmark for approximating the individual costs in a setting without cooperation and information sharing. In a subsequent step, the cost savings obtained in going from the myopic to the optimal order are allocated based on a recursive switching procedure where our reformulations of previous optimization algorithms play an essential role. To the best of our knowledge, we are the first to design a cost allocation procedure for sequencing problems with precedence constraints that relies on these optimization methods.

In this paper, we start by motivating the CSP and by formally describing the general problem in detail. The difficulty in obtaining an optimal connection order for the CSP in general results in the restriction to CSPs on trees. CSPs on trees are special instances of sequencing problems with precedence constraints. More precisely, the networks that we will consider are series-parallel graphs (cf. Lawler, 1978), and more specifically, so-called chains and outtrees (cf. Prot and Bellenguez-Morineau, 2018). Works such as Horn (1972), Adolphson and Hu (1973), Sidney (1975), and Lawler (1978) provide algorithms that optimally solve one-machine sequencing problems with such precedence relations. For a comprehensive review of complexity results, we recommend Prot and Bellenguez-Morineau (2018). In our case, we broaden the scope of existing research by incorporating and focusing on the aspect of cost allocation. We first discuss a procedure to find an optimal order for network structures consisting of two lines arising from the source, integrated with an allocation approach. For the optimization part, the proposed solution algorithm for these 2-lines CSPs is a reformulation of the work of Sidney (1975) for parallel chains. We will, however, provide a modified and more elaborate procedure with additional ingredients like merge segments to allow for a more natural intuitive interpretation of the various steps in the algorithm. These elements are specifically introduced for the design of the subsequent cost allocation procedure. The 2-lines optimization algorithm and allocation procedure serve as the basis to recursively solve n-lines and

general tree CSPs. Please note that also, the "from top to bottom" recursion technique that we use for trees is different from the recursion approach from Sidney (1975). As stated in the previous paragraph, the allocation procedure starts by allocating costs according to a greedy endogenous myopic order as a first over-approximation. We will show that it is possible to go from this myopic order to an optimal order by nonnegative savings by recursively switching blocks of agents appropriately selected on the basis of merge segments. Our specific choices guarantee that, in each step, the two exchanged blocks play an identical role, while also all players in the same block are equally important. Therefore, we will recursively share cost savings in an equal way among players with the same role, which aligns with the ideas behind the EGS rule (Curiel et al., 1989) to fairly allocate the cost savings in going from an arbitrary initial order to an optimal one in a classical sequencing setting.

The remainder of this paper is organized as follows. Section 2 focuses on the general CSP. Section 3 provides a solution algorithm and an allocation rule for 2-lines CSPs. Sections 4 and 5 generalize the previous procedures to n-lines and tree CSPs, respectively. Finally, Section 6 summarizes the main conclusions of this work.

2. Problem description and motivation

The problem we address in this paper is the following. There is a node 0 that provides a certain service, a finite set of nodes N that must be (directly or indirectly) connected to 0 in order to receive the service, and a set of edges $E \subseteq \{\{i, j\} \mid i, j \in N \cup \{0\}, i \neq j\}$ such that $(N \cup \{0\}, E)$ is a connected graph. The edges are initially "inactive," and each has an activation time. In addition, each node has an associated parameter indicating the unit cost of not being connected to node 0. The objective pursued is twofold:

1. First, we aim to choose a *connection order* on the nodes of N so that the sum of the costs over all nodes, the *total connection cost*, is as low as possible. This order will lead to a set of edges in E to be activated that connect all nodes to 0. Once we have found an optimal solution, we know the costs required to connect each node of N to 0 according to that solution: the *minimal connection cost*.
2. The second objective is to propose a fair allocation of the minimal connection cost among the nodes of N.

Let us now treat a simple example that motivates the need to address these two issues and also to better appreciate the complexity of the problem at hand.

Example 1. Consider a situation with three customers (1, 2, and 3) that need to be connected to a smart grid with service node 0 as soon as possible. The graph in Fig. 1 represents the possible connections between the customers and the times required to activate each of these connections. The unit cost parameter associated with each customer is 1. As mentioned above, the first objective is to connect the three customers to node 0 while minimizing the total connection cost (which can also be interpreted as the total weighted completion time). Table 1 displays the possible connection orders on the customers, along with the corresponding edges to be sequentially activated, the

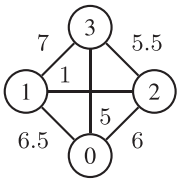


Fig. 1. Example of a CSP.

Table 1
Possible orders for the CSP from Fig. 1

Order	Edges to be activated	Individual cost vector	Total cost
(1,2,3)	{0, 1}, {1, 2}, {0, 3}	(6.5, 7.5, 12.5)	26.5
(1,3,2)	{0, 1}, {0, 3}, {1, 2}	(6.5, 12.5, 11.5)	30.5
(2,1,3)	{0, 2}, {2, 1}, {0, 3}	(7, 6, 12)	25
(2,3,1)	{0, 2}, {0, 3}, {2, 1}	(12, 6, 11)	29
(3,1,2)	{0, 3}, {0, 1}, {1, 2}	(11.5, 12.5, 5)	29
(3,2,1)	{0, 3}, {3, 2}, {2, 1}	(11.5, 10.5, 5)	27

individual cost vector (whose coordinates represent the cost of customers 1, 2, and 3, respectively), and the total aggregated costs.

By considering the six possible orders, one sees that the optimal connection order is (2,1,3). Correspondingly, the edges are activated in the order {0, 2}, {2, 1}, and {0, 3}. The optimal connection cost is 25, which is obtained by adding the costs of the nodes, 6 for customer 2, 7 for customer 1, and 12 for customer 3. Note that the greedy myopic order (i.e., the order in which at each step the edge incident on the component containing 0 with the lowest time is constructed) is not optimal. This myopic order is (3,2,1), which has an associated cost of 27. Nevertheless, this myopic order could be used as a benchmark order for cost allocation by subtracting an appropriate allocation of the savings of 2 from the myopic benchmark individual costs 11.5, 10.5, and 5.

Below, we will formally present the problem under consideration as well as some essential definitions to address it. From now on, we will use the terms machine and players instead of source and nodes, respectively.

A *connection scheduling problem*, CSP, can be summarized by a tuple $\mathcal{G} = (N, 0, E, \gamma, \alpha)$, where N is a finite set of jobs or players, 0 represents the machine, E is a set of available (precedence) edges between players and machine, that is, $E \subseteq \{\{i, j\} \mid i, j \in N \cup \{0\}, i \neq j\}$, such that $(N \cup \{0\}, E)$ is a connected graph, $\gamma : E \to (0, \infty)$ with $\gamma_{ij} = \gamma(\{i, j\})$, representing the activation time of the edge $\{i, j\} \in E$, and, finally, $\alpha : N \to (0, \infty)$, with $\alpha(i)$ representing the linear cost coefficient to spend one time unit in the system for player $i \in N$.

A processing or connection order is described by a bijection $\sigma : N \to \{1, \ldots, |N|\}$, and $\Pi(N)$ denotes the set of all processing orders. The main assumption is that a player $i \in N$ can only be processed by the machine if all players on a path in E from i to the machine have been processed before. A processing order $\sigma \in \Pi(N)$ is called feasible if the aforementioned condition is met for all players. Given $\sigma \in \Pi(N)$ and $i \in N$, let $P_\sigma(i) = \{j \in N \mid \sigma(j) < \sigma(i)\}$ denote the set of predecessors

of i in σ. Also, let $P_\sigma^0(i) = P_\sigma(i) \cup \{0\}$. Formally, $\sigma \in \Pi(N)$ is feasible if, for all $i \in N$, there exists $j \in P_\sigma^0(i)$ such that $\{i, j\} \in E$. Let $\Pi(N, E) \subseteq \Pi(N)$ denote the set of all feasible orders.

Definition 1. *Let $(N, 0, E, \gamma, \alpha)$ be a CSP, and let $\sigma \in \Pi(N, E)$. Given $k \in \{1, \ldots, |N|\}$, we recursively define the* completion time of player $i = \sigma^{-1}(k)$ with respect to σ, $C_i(\sigma)$, *as follows:*

$$C_i(\sigma) = \begin{cases} \gamma_{0i} & \text{if } k = 1 \\ C_{\sigma^{-1}(k-1)}(\sigma) + \min\left\{\gamma_{ij} \mid j \in P_\sigma^0(i) \text{ and } \{i, j\} \in E\right\} & \text{if } k > 1. \end{cases}$$

Given $i \in N$, we define the cost of player i with respect to σ, $c_i(\sigma)$, *as follows:*

$$c_i(\sigma) = \alpha(i) \cdot C_i(\sigma).$$

Let $c(\sigma) = (c_i(\sigma))_{i \in N}$ denote the individual cost vector with respect to σ. We define the total cost of σ, $TC(\sigma)$, *as follows:*

$$TC(\sigma) = \sum_{i \in N} c_i(\sigma). \tag{1}$$

Among other things, this paper aims to determine an optimal order $\hat{\sigma} \in \Pi(N, E)$ that minimizes the total cost among all feasible processing orders. It is important to note that the problem of finding an optimal connection order for general graphs $(N \cup \{0\}, E)$ is hard (cf. Lawler, 1978). The optimal solution in Example 1 already shows some unexpected peculiarity, such as the edge $\{0, 2\}$ being preferred to connect customer 2 with the service node, instead of the edge $\{3, 2\}$ that also connects it (because the edge $\{0, 3\}$ belongs to the optimal solution) and has a lower associated time (5.5 instead of 6). This highlights the potential complexity of the CSP. In this paper, we will focus on CSPs such that $(N \cup \{0\}, E)$ is a tree.

It should be stressed that by restricting the problem to trees, there will be only one path between any two nodes. With the purpose of simplifying the notation, the CSPs treated from now on will be denoted by a tuple $(N, 0, E, \gamma, \alpha)$, where γ is now a function on N. In particular, for $\gamma : N \to (0, \infty)$ we have that $\gamma(i) = \gamma_{ip_E^0(i)}$, where $p_E^0(i)$ is the immediate predecessor of i on the unique path in E from the machine to i. In this way, it is easily seen that equation (1) can be reformulated as

$$TC(\sigma) = \sum_{k=1}^{|N|} \gamma(\sigma^{-1}(k)) \cdot \left(\sum_{\{j \in N \mid \sigma(j) \geq k\}} \alpha(j) \right),$$

for all $\sigma \in \Pi(N, E)$, where $\sigma^{-1}(k)$ represents the player positioned at index k in the connection order.

3. 2-Lines CSPs

In this section, we describe and analyze 2-lines CSPs. We will first tackle the question of how to find an optimal connection order for these problems, and second how to allocate the cost of such a minimal connection order among the players involved.

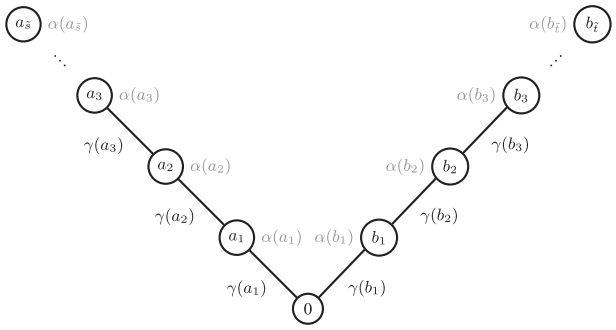


Fig. 2. Graphical representation of a 2-lines CSP.

3.1. Optimal orders

A CSP $(N, 0, E, \gamma, \alpha)$ is called a *2-lines CSP* if there exists a partition $\langle A, B\rangle$ of N with $A = \{a_1, \ldots, a_{\tilde{s}}\}$ and $B = \{b_1, \ldots, b_{\tilde{t}}\}$ with $\tilde{s} + \tilde{t} = |N|$, such that

$$E = \big\{\{0, a_1\}, \{a_1, a_2\}, \ldots, \{a_{\tilde{s}-1}, a_{\tilde{s}}\}\big\} \cup \big\{\{0, b_1\}, \{b_1, b_2\}, \ldots, \{b_{\tilde{t}-1}, b_{\tilde{t}}\}\big\}.$$

The sets A and B are called branches. For this particular case, a feasible order is described by a bijection $\sigma : A \cup B \to \{1, 2, \ldots, \tilde{s} + \tilde{t}\}$ such that

$$\sigma(a_k) < \sigma(a_l) \Rightarrow k < l \quad \text{and} \quad \sigma(b_k) < \sigma(b_l) \Rightarrow k < l.$$

Let $\Pi(A \cup B, E)$ denote the set of all such feasible orders. A graphical representation of a 2-lines CSP can be seen in Fig. 2.

Definition 2. *Let $(N, 0, E, \gamma, \alpha)$ be a 2-lines CSP, and let $\sigma \in \Pi(A \cup B, E)$. We define a* segment of A from h to l *as the following subset of A:*

$$Q_{hl} = \{a_h, a_{h+1}, \ldots, a_l\},$$

where $1 \le h \le l \le \tilde{s}$. Analogously, we define a segment of B from h to l *as the following subset of B:*

$$R_{hl} = \{b_h, b_{h+1}, \ldots, b_l\},$$

where $1 \le h \le l \le \tilde{t}$.

When we do not specify which branch a certain segment belongs to, we will use the notation X or Y instead of Q and R. A segment from the beginning of a branch is called a *head*.

Definition 3. *Given a segment X, we define the* cost-weighted average time per edge of X, $CAT(X)$, *as*

$$CAT(X) = \frac{\sum_{i \in X} \gamma(i)}{\sum_{i \in X} \alpha(i)}.$$

Algorithm 1. Algorithm to solve a 2-lines CSP

0. Initialize $k = 1, i = 1, l = 1, l' = 1$.
1. Consider the 2-lines CSP $(N, 0, E, \gamma, \alpha)$.
   - If $\dfrac{\gamma(a_1)}{\alpha(a_1)} \leq \dfrac{\gamma(b_1)}{\alpha(b_1)}$, select pivot $P_i = \{a_1\}$. Recall that $\langle A, B\rangle$ is the partition of N as detailed above.
   - Else, select pivot $P_i = \{b_1\}$.
2. (a) If $P_i \subseteq A$, take $l \leftarrow l + 1$.
   - If $l \leq \tilde{t}$, compare $CAT(P_i)$ to $CAT(R_{1l})$. Recall that $R_{1l} = \{b_1, \ldots, b_l\}$ (see Definition 2).
     - If $CAT(P_i) \leq CAT(R_{1l})$, go back to step 2.
     - If $CAT(P_i) > CAT(R_{1l})$, then $i \leftarrow i + 1$, $P_i = R_{1l}$. Go back to step 2.
   - Else, go to step 3.

   (b) If $P_i \subseteq B$, take $l' \leftarrow l' + 1$.
   - If $l' \leq \tilde{s}$, compare $CAT(P_i)$ to $CAT(Q_{1l'})$. Recall that $Q_{1l'} = \{a_1, \ldots, a_{l'}\}$ (see Definition 2).
     - If $CAT(P_i) \leq CAT(Q_{1l'})$, go back to step 2.
     - If $CAT(P_i) > CAT(Q_{1l'})$, then $i \leftarrow i + 1$, $P_i = Q_{1l'}$. Go back to step 2.
   - Else, go to step 3.
3. Set $M_k = P_i$.

   (a) If $M_k \subseteq A$.
   - If $A \setminus M_k = \emptyset$, then $M_{k+j} = \{b_j\}$ for all $j \in \{1, \ldots, \tilde{t}\}$. The algorithm is finished.
   - Else, let $A = r(A \setminus M_k)$, where $r : A \setminus M_k \to A$ is a renumbering function such that $r(a_h) = a_{h-l'}$, for all $h \in \{l' + 1, \ldots, \tilde{s}\}$. Let $\gamma(a_h) = \gamma(a_{h+l'})$ and $\alpha(a_h) = \alpha(a_{h+l'})$ for all $h \in \{1, \ldots, \tilde{s} - l'\}$. Set $N = A \cup B$, $k \leftarrow k + 1$ and $i \leftarrow i + 1$, and reset $l = 1$ and $l' = 1$. Go back to step 1.

   (b) If $M_k \subseteq B$.
   - If $B \setminus M_k = \emptyset$, then $M_{k+j} = \{a_j\}$ for all $j \in \{1, \ldots, \tilde{s}\}$. The algorithm is finished.
   - Else, let $B = \tilde{r}(B \setminus M_k)$, where $\tilde{r} : B \setminus M_k \to B$ is a renumbering function such that $r(b_h) = b_{h-l}$, for all $h \in \{l + 1, \ldots, \tilde{t}\}$. Let $\gamma(b_h) = \gamma(b_{h+l})$ and $\alpha(b_h) = \alpha(b_{h+l})$ for all $h \in \{1, \ldots, \tilde{t} - l\}$. Set $N = A \cup B$, $k \leftarrow k + 1$ and $i \leftarrow i + 1$, and reset $l = 1$ and $l' = 1$. Go back to step 1.

Moreover, given a segment X, we define its urgency, *$U(X)$, as*

$$U(X) = \frac{1}{CAT(X)}.$$

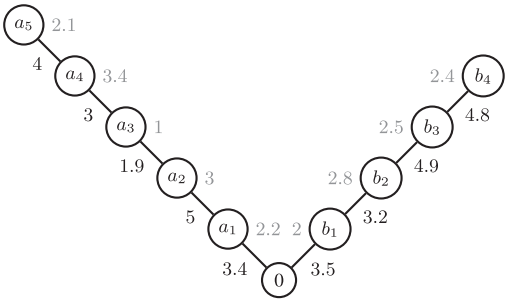
Given a 2-lines CSP, $(N, 0, E, \gamma, \alpha)$, our objective is to solve the following problem:

$$\begin{aligned} \min \quad & TC(\sigma) \\ \text{s.t.} \quad & \sigma \in \Pi(A \cup B, E). \end{aligned}$$

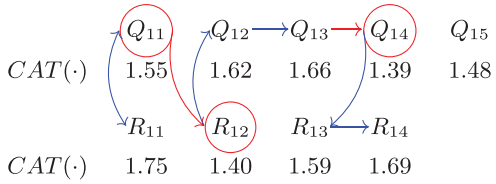
In Algorithm 1, we formally present an algorithm to solve 2-lines CSPs. Steps 1–3 constitute an iteration of the algorithm. The output of Algorithm 1 is a *merge order*:

$$\hat{\sigma} = (M_1, M_2, \ldots, M_{m-1}, M_m),$$

with $m \geq 2$, where $M_1, M_2, \ldots, M_m$ are called *merge segments*. It could be possible that both M_k and M_{k+1} belong to the same branch (because they might be merged at different steps). Of course, a merge order $\hat{\sigma}$ corresponds to one order on all players. If we do not want to highlight the merge

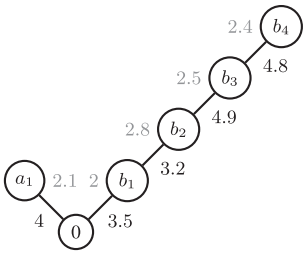


(a) Graphical representation

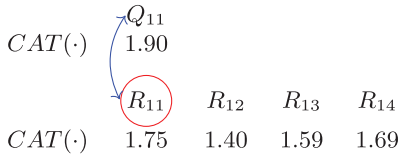


(b) Steps of Algorithm 1

Fig. 3. First iteration.



(a) Graphical representation



(b) Steps of Algorithm 1

Fig. 4. Second iteration.

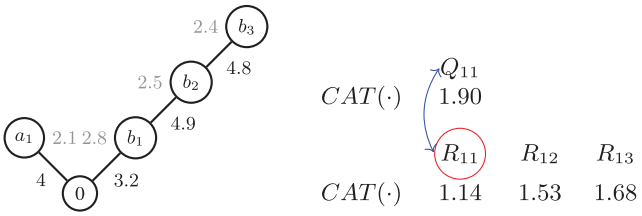
segments, we will use the notation $\hat{\tau}$ for this order. Note that Algorithm 1 has a time complexity of $O((\tilde{s}+\tilde{t})^2)$.

Example 2 shows the main ideas of this algorithm.

Example 2. Consider the 2-lines CSP presented in Fig. 3a. Figure 3b shows the steps to be followed by Algorithm 1 in the first iteration. First, $CAT(Q_{11})$ and $CAT(R_{11})$ are compared, and Q_{11} is selected as the first pivot. Since $CAT(Q_{11}) > CAT(R_{12})$, the pivot is changed to R_{12} (red arrows represent pivot transitions). After making the appropriate comparisons (see blue arrows), Q_{14} is chosen as the next pivot, and, since there are no other segment in branch B with a lower CAT than Q_{14}, it becomes the first merge segment, that is, $M_1 = Q_{14}$. We merge Q_{14} to the machine and renumber the nodes. This leads to the second iteration, presented in Fig. 4. Since $CAT(R_{11}) < CAT(Q_{11})$, R_{11} is selected as the first pivot of this iteration and becomes the second merge segment, as there is no other possible comparison. The same happens in the third iteration, as can be seen in Fig. 5. After adequately renumbering the nodes, Q_{11} and R_{11} are compared in the fourth iteration, as a result of which Q_{11} is selected as the pivot (Fig. 6). This still needs to be compared with R_{12}. Since $CAT(Q_{11}) \leq CAT(R_{12})$, Q_{11} becomes a merge segment. Finally, we are left with one branch, so each individual is considered a separate merge segment.

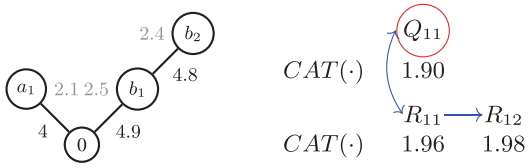
The algorithm outputs the following merge order:

$$\hat{\sigma} = (M_1, M_2, M_3, M_4, M_5, M_6),$$



(a) Graphical representation (b) Steps of Algorithm 1

Fig. 5. Third iteration.



(a) Graphical representation (b) Steps of Algorithm 1

Fig. 6. Fourth iteration.

where $M_1 = \{a_1a_2a_3a_4\}$, $M_2 = \{b_1\}$, $M_3 = \{b_2\}$, $M_4 = \{a_5\}$, $M_5 = \{b_3\}$, and $M_6 = \{b_4\}$, which has an associated cost of $TC(\hat{\sigma}) = 381.33$. Note that these merge segments are not the same as the so-called Sidney components (see Hamers et al., 2005 for details). It can be checked that the Sidney components would be $S_1 = \{a_1a_2a_3a_4\}$, $S_2 = \{b_1b_2\}$, $S_3 = \{a_5\}$, and $S_4 = \{b_3b_4\}$. For both merge segments and the Sidney components, we use this specific notation to emphasize the order of the players.

Next, we will show a series of results to prove that Algorithm 1 leads to an optimal order.

Remark 1. Given a set $Z \subseteq N$, we will use the notation $\gamma[Z] = \sum_{i \in Z} \gamma(i)$ and $\alpha[Z] = \sum_{i \in Z} \alpha(i)$. Hence, if X is a segment we can write $CAT(X) = \frac{\gamma[X]}{\alpha[X]}$.

The following proposition shows that an optimal order cannot have two consecutive segments of the same branch separated by nodes of the other branch when the CAT of the one segment is greater than the CAT of the other segment.

Proposition 1. *Let $(N, 0, E, \gamma, \alpha)$ be a 2-lines CSP. Let X and Y be two segments that belong to the same branch, and let Z be a segment from the other branch. If $CAT(X) > CAT(Y)$, then*

$$\tau = (\sim, X, Z, Y, \sim)$$

is not optimal.

Proof. See the Appendix. □

Definition 4. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 for a 2-lines CSP, let τ be a feasible order, and let $k \in \{1, 2, \ldots, m\}$. A* component of M_k in τ *is defined as a maximal connected*

subset of M_k with respect to τ. Denote by M_k/τ the set of components of M_k in τ, *and let $M_k/\tau = \{G_1, G_2, G_3, \ldots, G_{m_k}\}$ be the different components in the order they appear in τ, that is*

$$\tau = (\sim, G_1, \ldots, G_2, \ldots, G_3, \ldots, G_{m_k}, \sim).$$

Obviously, all merge segments and their components are segments. The following lemma presents different properties of merge segments and their components. Specifically, we will see that the last component is decisive for a certain segment to become a merge segment.

Lemma 1. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 for a 2-lines CSP. Take $k \in \{1, 2, \ldots, m\}$ such that $|M_k| > 1$. Let τ be a feasible order such that $|M_k/\tau| > 1$. Then, the following holds:*

(i) $CAT(G_{m_k}) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1})$;
(ii) $CAT(G_{m_k}) < CAT(M_k)$;
(iii) $CAT(G_{m_k}) < CAT(G_1)$.

Proof. See the Appendix. □

Given a nonconnected merge segment, the following lemma guarantees that there will be at least one pair of consecutive components in a feasible order such that the CAT of the first component is strictly greater than the CAT of the next component.

Lemma 2. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 for a 2-lines CSP. Take $k \in \{1, 2, \ldots, m\}$ such that $|M_k| > 1$. And let τ be a feasible order such that $|M_k/\tau| > 1$. Then, there exists $k \in \{2, 3, \ldots, m_k\}$ such that $CAT(G_{k-1}) > CAT(G_k)$.*

Proof. See the Appendix. □

The following lemma shows that there can be no optimal order in which merge segments have more than one component.

Lemma 3. *Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 for a 2-lines CSP. It holds that the elements of $M_k, k \in \{1, \ldots, m\}$, are consecutive in any optimal order.*

Proof. See the Appendix. □

The proposition below states that at least one optimal order has to start with the first merge segment obtained by applying Algorithm 1.

Proposition 2. *Let $(N, 0, E, \gamma, \alpha)$ be a 2-lines CSP. Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 for such a problem. There always exists an optimal order that starts with M_1.*

Proof. See the Appendix. □

The following proposition shows that the specific structure of an optimal order leads to an optimal order of a subproblem.

Proposition 3. *Let $(N, 0, E, \gamma, \alpha)$ be a 2-lines CSP, and let $\tilde{\sigma}^N = (M_1, \ldots, M_m)$ be the output of Algorithm 1. Let τ^N be an optimal order. If τ^N starts with M_1, it holds that $\tau^N|_{N \setminus M_1}$ is an optimal order for the subproblem on $N \setminus M_1$.*

Proof. See the Appendix. □

The next result describes a reverse version of Proposition 3: if we have a specific optimal order for a subproblem, we can derive an optimal order for the general problem.

Lemma 4. *Let $(N, 0, E, \gamma, \alpha)$ be a 2-lines CSP, and let $\tilde{\sigma}^N = (M_1, \ldots, M_m)$ be the output of Algorithm 1. Let $\tau^{N\setminus M_1}$ be an optimal order for the problem on $N \setminus M_1$. It holds that $\tau^N = (M_1, \tau^{N\setminus M_1})$ is an optimal order for $(N, 0, E, \gamma, \alpha)$.*

Proof. See the Appendix. □

We present below the main result of this subsection, which indicates that Algorithm 1 always leads to an optimal order.

Theorem 1. *Let $(N, 0, E, \gamma, \alpha)$ be a 2-lines CSP, and let $\hat{\tau}$ be the order provided by Algorithm 1. Then, $TC(\hat{\tau}) \leq TC(\tau)$ for all $\tau \in \Pi(A \cup B, E)$.*

Proof. See the Appendix. □

3.2. Allocating the minimal cost

This subsection introduces the κ rule as a cost allocation rule for 2-lines CSPs. The κ rule takes as a benchmark for allocating costs a greedy endogenous myopic connection order and will subtract a specific allocation vector of the maximal cost savings as given by the block splitting rule (BSR), which will be described later. The underlying allocation procedure is closely tied to the theoretical results presented in Subsection 3.1. In particular, the merge segments will be the foundation of the allocation procedure that we will discuss. Using their properties, we will carefully choose those blocks of agents to be sequentially switched when going from the benchmark myopic order to the optimal one. As we will see, these choices guarantee not only efficiency, but also nonnegative savings at each switching step and a fair treatment to all involved players. Below, we explain the ideas behind the κ rule in more detail.

Let $\mathcal{G} = (N, 0, E, \gamma, \alpha)$ be a 2-lines CSP. The order that we will use as a benchmark point is an endogenous and myopic order τ_0. It will depend on the particular problem we are considering, in the following way: at each step, the machine selects the player that has a higher urgency, always taking into account the existing precedence relations. For expositional simplicity, we will assume that $\frac{\alpha(i)}{\gamma(i)} \neq \frac{\alpha(j)}{\gamma(j)}$ for all $i, j \in N$, $i \neq j$, that is, all players' urgencies are different. Thus, τ_0 is unique. Moreover, given an optimal order $\hat{\tau}$, the total amount that will be saved is $g^N = TC(\tau_0) - TC(\hat{\tau})$. The allocation approach starts from the benchmark order, τ_0, and "repairs" it until the optimal order found by Algorithm 1, $\hat{\tau}$, is reached. In order to guarantee that these repairs lead to nonnegative cost savings and there is a local incentive to perform each step, we exchange blocks of players related to the merge segments.

Below, we provide a 2-lines CSP that illustrates the ideas behind our allocation procedure.

Example 3. Consider the 2-lines CSP from Example 2. In order to compute the benchmark myopic order, we start by comparing $\frac{\gamma(a_1)}{\alpha(a_1)} = \frac{3.4}{2.2} = 1.55$ to $\frac{\gamma(b_1)}{\alpha(b_1)} = \frac{3.5}{2} = 1.75$. Since the urgency of player a_1 is higher, we select a_1 as the first player of τ_0. The second step consists of comparing $\frac{\gamma(a_2)}{\alpha(a_2)} = 1.67$

to $\frac{\gamma(b_1)}{\alpha(b_1)} = 1.75$, leading to a_2 being the second player in τ_0. We repeat these comparisons until all players have been included in τ_0. The benchmark order is $\tau_0 = (a_1, a_2, b_1, b_2, a_3, a_4, a_5, b_3, b_4)$, with $TC(\tau_0) = 387.29$ and $c(\tau_0) = (7.48, 25.2, 17, 68, 50.4, 23.8, 42.28, 72.25, 80.88)$, where the coordinates in $c(\tau_0)$ follow the natural order prescribed by the branches. We have seen in Example 2 that $\hat{\tau} = (a_1, a_2, a_3, a_4, b_1, b_2, a_5, b_3, b_4)$ is an optimal order, with $TC(\hat{\tau}) = 381.33$. Thus, τ_0 is not optimal, and there are savings of $g^N = 387.29 - 381.33 = 5.96$ from τ_0 to $\hat{\tau}$.

How to allocate the savings of going from τ_0 to $\hat{\tau}$? Which players should be compensated for such savings? We know that we cannot switch b_2 with a_3 (the first point at which we have two consecutive players from different branches that are misplaced with respect to the optimal order) since this leads to a negative switch: in the benchmark order b_2 goes before a_3, this means that b_2 has a higher urgency than a_3. But we can switch specific consecutive blocks of players simultaneously. In this particular case, we could switch $X = \{b_1 b_2\}$ with $Y = \{a_3 a_4\}$, thus obtaining the optimal order. The gain resulting from switching them is denoted by g_{XY} and equals 5.96. Regarding a savings allocation rule, we propose to allocate $\frac{1}{2} g_{XY}$ to all players in X equally and $\frac{1}{2} g_{XY}$ to all players in Y equally. Here, our proposal would be to allocate a saving of $\frac{1}{2} \cdot \left(\frac{5.96}{2}\right) = 1.49$ to each of the players a_3, a_4, b_1, and b_2. Subsequently, these savings should be subtracted from $c(\tau_0)$, i.e., the κ cost allocation rule would lead to the following cost allocation vector: $(7.48, 25.2, 15.51, 66.51, 50.4, 22.31, 40.79, 72.25, 80.88)$.

The above example raises the following question: How do we choose these blocks in general in a unique way such that nonnegative switching gains are guaranteed in each step? The determination of these blocks cannot be carried out simply by observing the orders τ_0 and $\hat{\tau}$ but will be done iteratively. To do that, we will present in detail the *block splitting rule* (BSR). The key point of this approach is to determine, at each step, which blocks are to be swapped. Note that given two consecutive blocks, X and Y with X before Y, the gain resulting from switching them is

$$g_{XY} = \alpha[Y] \cdot \gamma[X] - \alpha[X] \cdot \gamma[Y] = \alpha[Y] \cdot \alpha[X] \cdot \frac{\gamma[X]}{\alpha[X]} - \alpha[X] \cdot \alpha[Y] \cdot \frac{\gamma[Y]}{\alpha[Y]}$$
$$= \alpha[Y] \cdot \alpha[X] \cdot (CAT(X) - CAT(Y)).$$

The merge segments play a fundamental role in defining the BSR, since by conveniently using their properties along with Algorithm 1 we will be able to guarantee nonnegative savings at each iteration. Thus, this procedure consists of two main stages: first, we will repair those merge segments whose players are not consecutive, and, second, reorder them as in $\hat{\tau}$. To this end, we will need to consider $\hat{\sigma}$, the corresponding merge order to $\hat{\tau}$. Algorithm 2 shows the scheme of this procedure.

Let X^ι and Y^ι be the misplaced blocks switched at iteration ι of Algorithm 2. Then, we define:

$$\mathrm{BSR}^\iota(\tau_0, \hat{\sigma}) = \frac{1}{2} g_{X^\iota Y^\iota} \left(\frac{1}{|X^\iota|} e^{X^\iota} + \frac{1}{|Y^\iota|} e^{Y^\iota} \right),$$

where, for $S \subseteq N$, e^S is the vector in $\mathbb{R}^N$ satisfying $e_i^S = 1$ if $i \in S$ and $e_i^S = 0$ otherwise. $\mathrm{BSR}^\iota(\tau_0, \hat{\sigma})$ represents the allocation obtained at iteration ι, hence $\mathrm{BSR}(\tau_0, \hat{\sigma}) = \sum_{\iota=1}^{I} \mathrm{BSR}^\iota(\tau_0, \hat{\sigma})$, where I represents the total number of iterations needed. By defining the BSR in this way, we guarantee that those agents who play the same role in the switch are equally compensated.

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies.

Algorithm 2. Algorithm to allocate the gains of a 2-lines CSP

0. Obtain τ_0 and apply Algorithm 1 to get $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$. Initialize $k = 1$, $r = 1$, $\iota = 1$, and $\tau' = \tau_0$.
1. (a) If $|M_k/\tau'| = 1$, take $k \leftarrow k + 1$.
   - If $k \leq m - 1$, go back to step 1.
   - If $k = m$, go to step 2.

   (b) If $|M_k/\tau'| > 1$, take $\tilde{k} \in \{2, \ldots, m_k\}$ such that $CAT(G_{\tilde{k}-1}) > CAT(G_{\tilde{k}})$ (we know there exists such a pair of components from Lemma 2). It is clear that between $G_{\tilde{k}-1}$ and $G_{\tilde{k}}$ there are only players from the other branch, i.e.,

   $$\tau' = (\sim, G_{\tilde{k}-1}, Z, G_{\tilde{k}}, \sim),$$

   where Z is a segment from the opposite branch of M_k. By Proposition 1, we know that either the order $\tau_1 = (\sim, G_{\tilde{k}-1}, G_{\tilde{k}}, Z, \sim)$ or the order $\tau_2 = (\sim, Z, G_{\tilde{k}-1}, G_{\tilde{k}}, \sim)$ has a lower total cost than τ'. Take $\tau'' = \arg\min_\tau \{TC(\tau) : \tau \in \{\tau_1, \tau_2\}\}$.
   - If $\tau'' = \tau_1$, then the blocks that have been switched are Z and $G_{\tilde{k}}$. Players from Z should receive $\frac{1}{2|Z|} g_{ZG_{\tilde{k}}}$, while players from $G_{\tilde{k}}$ should receive $\frac{1}{2|G_{\tilde{k}}|} g_{ZG_{\tilde{k}}}$.
   - If $\tau'' = \tau_2$, then the blocks that have been switched are $G_{\tilde{k}-1}$ and Z. Players from $G_{\tilde{k}-1}$ should receive $\frac{1}{2|G_{\tilde{k}-1}|} g_{G_{\tilde{k}-1}Z}$, while players from Z should receive $\frac{1}{2|Z|} g_{G_{\tilde{k}-1}Z}$.

   Set $\tau' = \tau''$, and take $\iota \leftarrow \iota + 1$. Go back to step 1.
2. (a) If $r \leq m$, consider the bijection

   $$\rho : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, m\}$$
   $$i \mapsto \rho(i) = j,$$

   such that $\tau' = (M_{\rho(1)}, \ldots, M_{\rho(m)})$. We need to go from τ' to $\hat{\tau}$.
   i. If $\rho(r) = r$, take $r \leftarrow r + 1$. Go back to step 2.
   ii. If $\rho(r) \neq r$, take $\tilde{r} \in \{r + 1, \ldots, m\}$ such that $\rho(\tilde{r}) = r$ (this means that M_r is on position $\tilde{r}$, i.e., $M_r = M_{\rho(\tilde{r})}$). By Algorithm 1, it holds that

   $$CAT(M_r) \leq CAT\left(M_r^{\cup}\right),$$

   where $M_r^{\cup} = \bigcup_{l=r}^{\tilde{r}-1} M_{\rho(l)}$. Hence, the order

   $$\tau'' = (\sim, M_{\rho(\tilde{r})}, M_{\rho(r)}, \ldots, M_{\rho(\tilde{r}-1)}, M_{\rho(\tilde{r}+1)}, \sim)$$

   that consists of moving $M_{\rho(\tilde{r})}$ to the front of $M_{\rho(r)}$ (so that $M_{\rho(\tilde{r})} \equiv M_r$ is now on position r) has lower total cost than τ'. The blocks that have been switched are $M_r^{\cup}$ and M_r. Allocate $\frac{1}{2|M_r^{\cup}|} g_{M_r^{\cup} M_r}$ to the players in $M_r^{\cup}$ and $\frac{1}{2|M_r|} g_{M_r^{\cup} M_r}$ to the players in M_r. Set $\tau' = \tau''$, and take $r \leftarrow r + 1$ and $\iota \leftarrow \iota + 1$. Go back to step 2.

   (b) If $r > m$, then $\tau' = \hat{\tau}$ and $I = \iota - 1$. The algorithm is finished. The outputs of the algorithm are the allocation and the misplaced blocks at each iteration $\iota \in \{1, \ldots, I\}$.

Next, we present the definition of the cost allocation rule for 2-lines CSPs.

Definition 5. *We define the cost allocation rule, κ, by setting*

$$\kappa(\mathcal{G}) = c(\tau_0) - BSR(\tau_0, \hat{\sigma}),$$

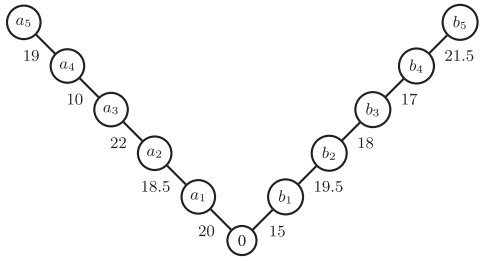


Fig. 7. Example of a 2-lines CSP.

Table 2
Steps of Algorithm 2 to go from τ_0 to $\hat{\sigma}$

ι	Case	Switched blocks	Resulting order	Gain	$\text{BSR}^{\iota}(\tau_0, \hat{\sigma})$
1	Nonconnected merge segment $M_2/\tau_0 = \{\{a_1a_2\}, \{a_3a_4\}\}$	$\{a_3a_4\}$ $\{b_5\}$	$(M_1, M_3, M_4, M_2, M_6, M_5)$	11	(0, 0, 2.75, 2.75, 0, 0, 0, 0, 0, 5.5)
2	Non-ordered merge segments	M_2 $M_3 \cup M_4$	$(M_1, M_2, M_3, M_4, M_6, M_5)$	6.5	(0.813, 0.813, 0.813, 0.813, 0, 0, 1.083, 1.083, 1.083, 0)
3	Non-ordered merge segments	M_5 M_6	$\hat{\sigma}$	2.5	(0, 0, 0, 0, 1.25, 0, 0, 0, 1.25)

for a 2-lines CSP $\mathcal{G}$.

The following example shows how to obtain the κ rule for a 2-lines CSP by applying the above procedure.

Example 4. Consider the 2-lines CSP $\mathcal{G}$ from Fig. 7. Assuming that $\alpha(i) = 1$ for all $i \in A \cup B$, these numbers are not incorporated in the figure for clarity.

The benchmark order would be $\tau_0 = (b_1, b_2, b_3, b_4, a_1, a_2, b_5, a_3, a_4, a_5)$, with an associated individual cost vector of $c(\tau_0) = (89.5, 108, 151.5, 161.5, 180.5, 15, 34.5, 52.5, 69.5, 129.5)$ and total cost of $TC(\tau_0) = 992$. The optimal solution provided by Algorithm 1 would be $\hat{\tau} = (b_1, a_1, a_2, a_3, a_4, b_2, b_3, b_4, a_5, b_5)$, with an associated cost of $TC(\hat{\tau}) = 972$. The corresponding merge order is $\hat{\sigma} = (M_1, M_2, M_3, M_4, M_5, M_6)$, where $M_1 = \{b_1\}$, $M_2 = \{a_1a_2a_3a_4\}$, $M_3 = \{b_2b_3\}$, $M_4 = \{b_4\}$, $M_5 = \{a_5\}$, and $M_6 = \{b_5\}$.

Table 2 summarizes the steps followed by Algorithm 2 to go from τ_0 to $\hat{\sigma}$. Note that, in the first iteration, M_2 is nonconnected in τ_0. In fact, $M_2/\tau_0 = \{G_1, G_2\}$, where $G_1 = \{a_1a_2\}$ and $G_2 = \{a_3a_4\}$. These components have $Z = \{b_5\}$ in between. By Lemma 1, we know that $CAT(G_1) > CAT(G_2)$. By Proposition 1, we know that either moving G_2 to the front of Z or moving G_1 to the back of Z would lead to positive savings. Thus, we need to compare the following two orders:

$$\tau_1 = (b_1, b_2, b_3, b_4, a_1, a_2, \underbrace{a_3, a_4}_{G_2}, \overbrace{b_5}^{Z} a_5) \quad \text{and} \quad \tau_2 = (b_1, b_2, b_3, b_4, \overbrace{b_5}^{Z}, \underbrace{a_1, a_2}_{G_1}, a_3, a_4, a_5).$$

Since $TC(\tau_1) = 981$ and $TC(\tau_2) = 996.5$, we choose $\tau' = \tau_1$. Hence, blocks Z and G_2 have been switched, as shown in Table 2. The resulting saving of 11 needs to be split equally between the two blocks that have been exchanged, and subsequently equally to the number of players in each block. Thus, players a_3, a_4, and b_5 have savings of 2.75, 2.75, and 5.5, respectively. After this iteration, all the merge segments are connected and all that remains is to reorder them sequentially, as is done in iterations 2 and 3. The savings allocation process is the same: first, between the blocks that need to be switched and, then, between players of each block.

Therefore, the final savings allocation rule provided by Algorithm 2 is the following:

$$\mathrm{BSR}(\tau_0, \hat{\sigma}) = \sum_{\iota=1}^{3} \mathrm{BSR}^{\iota}(\tau_0, \hat{\sigma}) = (0.813, 0.813, 3.563, 3.563, 1.25, 0, 1.083, 1.083, 1.083, 6.75).$$

Hence,

$$\kappa(\mathcal{G}) = (88.687, 107.187, 147.937, 157.937, 179.25, 15, 33.417, 51.417, 68.417, 122.75).$$

The vector above is a proposal for allocating the total cost of an optimal order. One could also view the outcome of $\kappa(\mathcal{G})$ in the following way. Consider the individual cost vector corresponding to the optimal order, $c(\hat{\tau}) = (35, 53.5, 75.5, 85.5, 159, 15, 105, 123, 140, 180.5)$. To reach $\kappa(\mathcal{G})$, we can take a vector of compensations, $(53.687, 53.687, 72.437, 72.437, 20.25, 0, -71.583, -71.583, -71.583, -57.75)$. Players a_1, a_2, a_3, a_4, and a_5 have positive numbers, which indicate that they have to pay compensations for being handled earlier than other players in the optimal order, with respect to what was supposed to happen according to the benchmark order. On the contrary, players b_2, b_3, b_4, and b_5 get compensated for delaying their processing in $\hat{\tau}$ compared to τ_0.

To conclude this subsection, we make two remarks on the κ rule.

As illustrated in Example 4, the κ rule for a particular problem can be interpreted as the cost that each player would have to pay if, instead of adopting the myopic connection order τ_0, the optimal connection order $\hat{\tau}$ is adopted. A particularly interesting feature is that all players benefit from this transition, since, by construction, $\mathrm{BSR}(\tau_0, \hat{\sigma}) \geq 0$, and therefore, $\kappa(\mathcal{G}) \leq c(\tau_0)$. In addition, considering that the optimal order $\hat{\tau}$ is to be adopted, the compensation that each player should receive over the cost given by $c(\hat{\tau})$ is $\kappa(\mathcal{G}) - c(\hat{\tau})$.

To better appreciate our rule, it is important to understand that implicit in its definition is that the players start from a connection order whose costs they must initially assume, and that the rule distributes the savings fairly when moving to an optimal order. In this paper, we assume that the benchmark order (τ_0) is greedy and endogenous and is given by the player's urgencies. It is important to note that in the CSP setting the optimal connection order ($\hat{\tau}$) is, in general, not based on such a greedy motivation. For this reason, allocating the costs directly based on the optimal connection order will not guarantee an intrinsically stable allocation procedure. Instead, to improve upon this procedural stability aspect, we consider a greedy benchmark cost allocation vector ($c(\tau_0)$) and a careful compensation procedure based on coalitional considerations to allocate further cost savings.

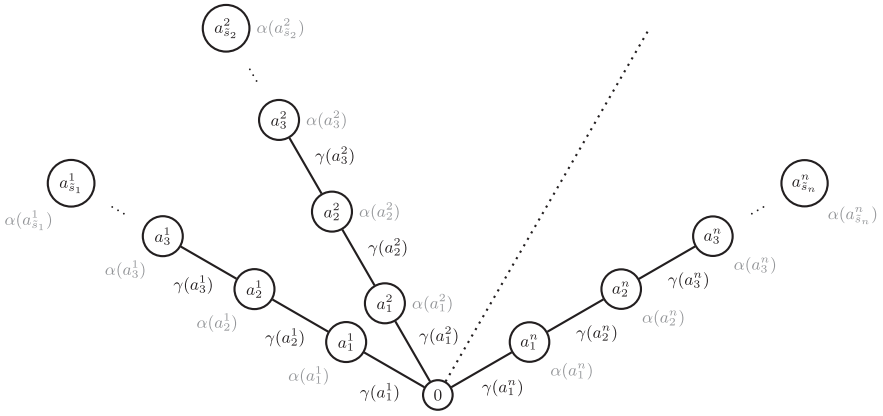


Fig. 8. Graphical representation of an n-lines CSP.

4. n-Lines CSPs

This section generalizes the optimization and allocation results for the 2-lines CSPs to n-lines CSPs.

4.1. Optimal orders

A CSP $(N, 0, E, \gamma, \alpha)$ is called an *n-lines CSP* if there exists a partition $\langle A^1, \ldots, A^n \rangle$ of N with $A^k = \{a^k_1, \ldots, a^k_{\tilde{s}_k}\}$ for all $k \in \{1, \ldots, n\}$ with $\sum_{k=1}^n \tilde{s}_k = |N|$, such that

$$E = \bigcup_{k=1}^{n} \left\{ \{0, a^k_1\}, \{a^k_1, a^k_2\}, \ldots, \{a^k_{\tilde{s}_k - 1}, a^k_{\tilde{s}_k}\} \right\}.$$

As for the 2-lines CSPs, the sets $A^k, k \in \{1, \ldots, n\}$, are called branches. A feasible order is described by a bijection $\sigma : N \to \{1, 2, \ldots, |N|\}$ such that $\sigma(a^k_h) < \sigma(a^k_l) \Rightarrow h < l$, for all $k \in \{1, \ldots, n\}$. Let $\Pi(N, E)$ denote the set of all such feasible orders. The definitions regarding the segments and *CATs* can be directly extended to this generalization. A graphical representation of an n-lines CSP is provided in Fig. 8.

To solve an n-lines CSP, we propose an algorithm that combines the concept of recursion with Algorithm 1. This procedure is based on the following idea: given an n-lines CSP $(N, 0, E, \gamma, \alpha)$, we consider a 2-lines CSP $(A^h \cup A^l, 0, E|_{A^h \cup A^l}, \gamma, \alpha)$, where $h, l \in \{1, \ldots, n\}$. Note that $E|_{A^h \cup A^l}$ is the restriction of E to the players of $A^h \cup A^l$. $\hat{\sigma}_{hl}$ will denote the output of Algorithm 1 for such a subproblem in which the merge segments are specified. $\hat{\tau}_{hl}$ will refer to the corresponding order of all players involved. Furthermore, $A^{hl}_{\hat{\tau}}$ will denote the branch formed by the nodes from A^h and A^l following the order specified by $\hat{\tau}_{hl}$. To enhance readability and conceptual clarity, we do not provide a fully formalized version of the proposed algorithm, which has a time complexity of $O(n \cdot |N|^2)$. Instead, Algorithm 3 presents an intuitive description of the procedure.

The following example illustrates how to apply Algorithm 3 to solve an n-lines CSP.

Algorithm 3. Algorithm to solve an n-lines CSP

1. Consider an n-lines CSP $(N, 0, E, \gamma, \alpha)$. Select branches A^1 and A^2.
2. Apply Algorithm 1 to solve the corresponding 2-lines CSP, $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$. This leads to an optimal order, $\hat{\tau}_{12}$. Replace A^1 and A^2 with the branch $A^{12}_{\hat{\tau}}$. We get a new problem with one branch less. Renumber the branches adequately.
3. (a) If there are still more than two branches left, go back to step 1.
   (b) If there are two branches left, apply Algorithm 1. The order obtained is the solution.

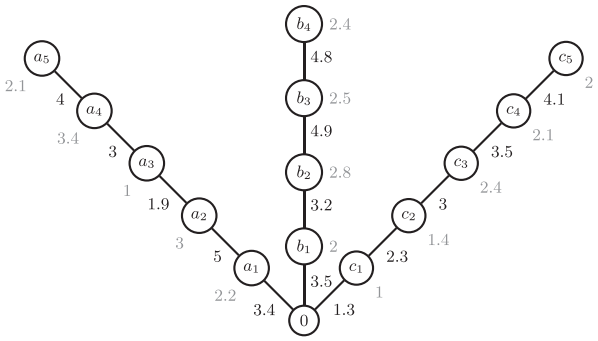


Fig. 9. Example of a 3-lines CSP.

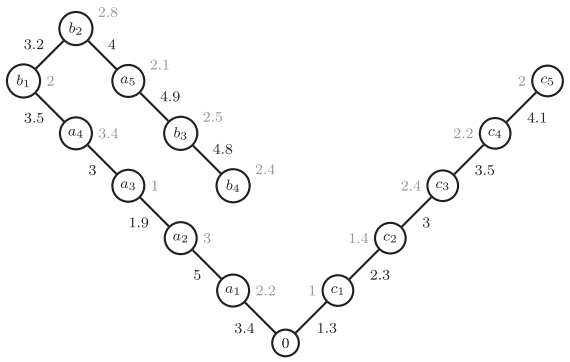


Fig. 10. Resulting 2-lines CSP after solving $(A \cup B, 0, E|_{A \cup B}, \gamma, \alpha)$.

Example 5. Consider the 3-lines CSP presented in Fig. 9. We will avoid renumbering the nodes for a better understanding of the algorithmic process. Also, we will denote the branches by A, B, and C for clarity, instead of A^1, A^2, and A^3, respectively.

As indicated in Algorithm 3, we first select branches A and B. The 2-lines CSP $(A \cup B, 0, E|_{A \cup B}, \gamma, \alpha)$ has been solved in Example 2, leading to the optimal order $\hat{\tau}_{12} = (a_1, a_2, a_3, a_4, b_1, b_2, a_5, b_3, b_4)$. We replace branches A and B with one branch respecting the order of $\hat{\tau}_{12}$, as illustrated in Fig. 10. Next, we solve this 2-lines CSP by applying Algorithm 1. The

output would be

$$\hat{\tau} = (c_1, a_1, a_2, a_3, a_4, c_2, c_3, b_1, b_2, c_4, a_5, b_3, b_4, c_5),$$

finishing Algorithm 3.

Along similar lines as in Lemma 3 one can show the following result.

Lemma 5. *Let $(N, 0, E, \gamma, \alpha)$ be an n-lines CSP, and $\hat{\sigma}_{12} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 for the 2-lines CSP $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$. It holds that the elements of $M_k, k \in \{1, \ldots, m\}$, are consecutive in any optimal order for $(N, 0, E, \gamma, \alpha)$.*

Moreover, the result below guarantees that obtaining an optimal order for an n-lines CSP, the first two branches can be replaced by one branch that reflects the optimal order found by Algorithm 1 for the corresponding 2-lines subproblem.

Proposition 4. *Let $(N, 0, E, \gamma, \alpha)$ be an n-lines CSP, and let τ_{12}^* be the output of Algorithm 1 for the 2-lines CSP $(A^1 \cup A^2, 0, E|_{A^1 \cup A^2}, \gamma, \alpha)$. There exists an optimal order $\hat{\tau}$ for $(N, 0, E, \gamma, \alpha)$ such that $\hat{\tau}(i) < \hat{\tau}(j)$ for all $i, j \in A^1 \cup A^2$ for which $\tau_{12}^*(i) < \tau_{12}^*(j)$.*

Proof. See the Appendix. □

The following result states that Algorithm 3 leads to an optimal order.

Theorem 2. *Let $(N, 0, E, \gamma, \alpha)$ be an n-lines CSP, and let $\hat{\tau}$ be an order provided by Algorithm 3. Then, $TC(\hat{\tau}) \leq TC(\tau)$ for all $\tau \in \Pi(N, E)$.*

Proof. See the Appendix. □

4.2. Allocating the minimal cost

We will illustrate how to extend the ideas of the allocation procedure as described by κ for 2-lines CSPs into a rule on n-lines CSPs. Consider an n-lines CSP $\mathcal{G} = (N, 0, E, \gamma, \alpha)$. Again, take as a starting point of the allocation mechanism a benchmark endogenous myopic order on all players, τ_0^N, in the same way as before. Let $\hat{\tau}^N$ be the optimal order provided by Algorithm 3. The total savings of $g^N = TC(\tau_0^N) - TC(\hat{\tau}^N)$ need to be adequately subtracted from $c(\tau_0^N)$ to obtain the final cost allocation. To determine the proportion of g^N for which each player is responsible, we apply a procedure similar to that in Algorithm 3: we will recursively allocate the local savings obtained at the 2-lines CSPs that comprise our n-lines. The sum of these local savings, however, is not necessarily equal to g^N because as we process each subsequent 2-lines CSP, some players may already have contributed to savings in a previous step, leading to overlapping contributions. Instead, we use these numbers to determine the *relative* importance of the different subproblems (and, mainly, of the players involved in these problems) in the final savings obtained.

Let $\tau_0^k, k \in \{2, \ldots, n\}$ be the benchmark order for the 2-lines CSP induced by branches $A^{1\ldots k-1}$ and A^k, and let $\hat{\tau}^k$ and $\hat{\sigma}^k$ be the optimal order and its corresponding merge order provided by Algorithm 1 for such a 2-lines CSP. The local cost savings are defined by $g^k = TC(\tau_0^k) - TC(\hat{\tau}^k)$, and they will be allocated in the same way as before for each subproblem. That is, given $k \in \{2, \ldots, n\}$,

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies.

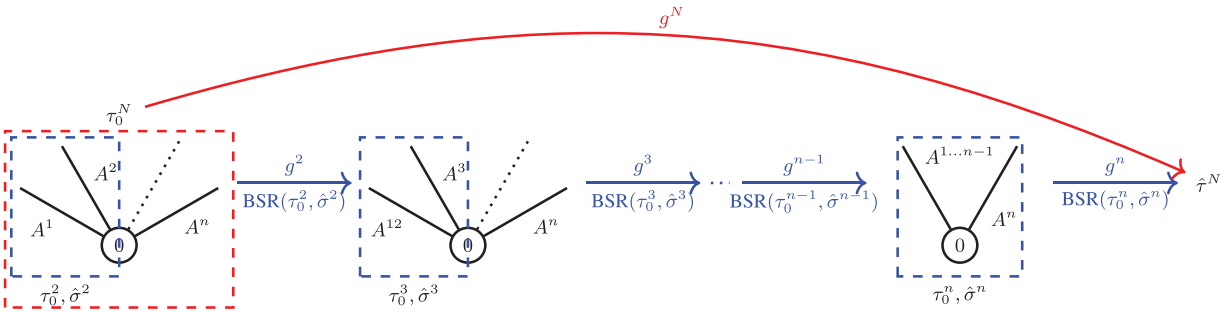


Fig. 11. Steps of the allocation procedure in an n-lines CSP.

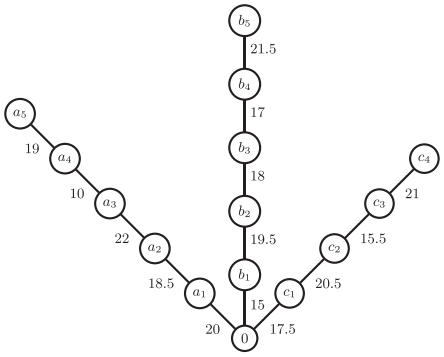


Fig. 12. Example of a 3-lines CSP.

going from τ_0^k to $\hat{\sigma}^k$ leads to a saving of g^k that is allocated among the players involved using Algorithm 2, thus obtaining BSR$(\tau_0^k, \hat{\sigma}^k)$. Each of these vectors is now complemented with 0s on those coordinates that refer to noninvolved players. Figure 11 summarizes this procedure.

Next, we present the definition of the cost allocation rule for n-lines CSPs.

Definition 6. *We define the cost allocation rule, κ, by setting:*

$$\kappa(\mathcal{G}) = c(\tau_0^N) - \frac{g^N}{\sum_{k=2}^n g^k} \cdot \sum_{k=2}^n BSR(\tau_0^k, \hat{\sigma}^k),$$

for an n-lines CSP $\mathcal{G}$.

The following example illustrates how to obtain the cost allocation rule for a 3-lines CSP.

Example 6. Consider the 3-lines CSP presented in Fig. 12. The general benchmark order would be $\tau_0^N = (b_1, c_1, b_2, b_3, b_4, a_1, a_2, c_2, c_3, c_4, b_5, a_3, a_4, a_5)$, with an associated individual cost vector of $c(\tau_0^N) = (107, 125.5, 226, 236, 255, 15, 52, 70, 87, 204, 32.5, 146, 161.5, 182.5)$ and the total cost of $TC(\tau_0^N) = 1900$.

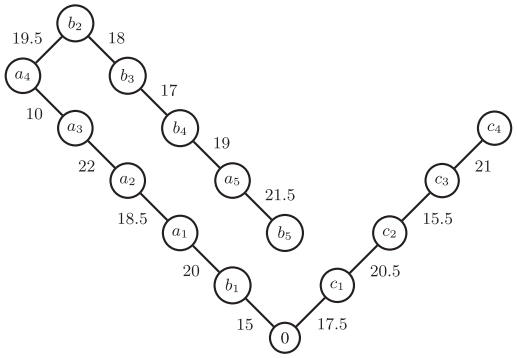


Fig. 13. Resulting 2-lines CSP.

Table 3
Steps of Algorithm 2 to go from τ_0^3 to $\hat{\sigma}^3$

ι	Case	Switched blocks	Resulting order	Gain	$\text{BSR}^{\iota}(\tau_0^3, \hat{\sigma}^3)$
1	Non-connected merge segment $M_3/\tau_0^3 = \{\{a_1a_2\}, \{a_3a_4\}\}$	$\{c_2c_3c_4\}$ $\{a_3a_4\}$	$(M_1, M_2, M_3, M_4, M_9, M_5, M_6, M_7, M_8, M_{10})$	18	(0, 0, 4.5, 4.5, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3)
2	Non-ordered merge segments	M_9 M_5	$(M_1, M_2, M_3, M_4, M_5, M_9, M_6, M_7, M_8, M_{10})$	1.5	(0, 0, 0, 0, 0, 0, 0, 0.75, 0, 0, 0, 0, 0, 0, 0, 0.75)
3	Non-ordered merge segments	M_9 M_6	$(M_1, M_2, M_3, M_4, M_5, M_6, M_9, M_7, M_8, M_{10})$	3	(0, 0, 0, 0, 0, 0, 0, 0, 1.5, 0, 0, 0, 0, 0, 1.5)
4	Non-ordered merge segments	M_9 M_7	$(M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_9, M_8, M_{10})$	4	(0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2)
5	Non-ordered merge segments	M_9 M_8	$\hat{\sigma}^3$	2	(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1)

First, select branches A and B. From Example 4, we know that $g^2 = 20$ and $\text{BSR}(\tau_0^2, \hat{\sigma}^2) = (0.813, 0.813, 3.563, 3.563, 1.25, 0, 1.083, 1.083, 1.083, 6.75, 0, 0, 0, 0)$.

Now, select branches $A \cup B$ and C. The local benchmark order for the resulting 2-lines CSP of Fig. 13 is given by $\tau_0^3 = (b_1, c_1, a_1, a_2, c_2, c_3, c_4, a_3, a_4, b_2, b_3, b_4, a_5, c_5)$, with $TC(\tau_0^3) = 1887.5$, and $\hat{\tau}^3 = (b_1, c_1, a_1, a_2, a_3, a_4, c_2, c_3, b_2, b_3, b_4, a_5, c_4, b_5)$, with $TC(\hat{\tau}^3) = 1859$. Thus, $g^3 = 28.5$. Also, $\hat{\tau}^N = \hat{\tau}^3$, so there is a total saving of $g^N = TC(\tau_0^N) - TC(\hat{\tau}^N) = 1900 - 1859 = 41$ that needs to be adequately allocated among the players. Note that $g^N \neq g^2 + g^3$. The corresponding merge order to $\hat{\tau}^3$ is given by $\hat{\sigma}^3 = (M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_{10})$, with $M_1 = \{b_1\}$, $M_2 = \{c_1\}$, $M_3 = \{a_1a_2a_3a_4\}$, $M_4 = \{c_2c_3\}$, $M_5 = \{b_2\}$, $M_6 = \{b_3\}$, $M_7 = \{b_4\}$, $M_8 = \{a_5\}$, $M_9 = \{c_4\}$, and $M_{10} = \{b_5\}$. Table 3 summarizes the steps of Algorithm 2 to go from τ_0^3 to $\hat{\sigma}^3$.

Therefore,

$$\text{BSR}(\tau_0^3, \hat{\sigma}^3) = (0, 0, 4.5, 4.5, 1, 0, 0.75, 1.5, 2, 0, 0, 3, 3, 8.25).$$

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies.

Hence,

$$\kappa(\mathcal{G}) = (107, 125.5, 226, 236, 255, 15, 52, 70, 87, 204, 32.5, 146, 161.5, 182.5)$$
$$- \frac{41}{20 + 28.5} \cdot [(0.813, 0.813, 3.563, 3.563, 1.25, 0, 1.083, 1.083, 1.083, 6.75, 0, 0, 0, 0)$$
$$+(0, 0, 4.5, 4.5, 1, 0, 0.75, 1.5, 2, 0, 0, 3, 3, 8.25)]$$
$$= (106.313, 124.813, 219.184, 229.184, 253.098, 15, 50.45, 67.816, 84.394, 198.294, 32.5, 143.464, 158.964, 175.526).$$

5. Tree CSPs

We will now extend the results we have seen for n-lines CSPs to the case of tree CSPs, both for optimization and cost allocation.

5.1. Optimal orders

A CSP $(N, 0, E, \gamma, \alpha)$ is called a *tree CSP* if $(N \cup \{0\}, E)$ is a tree.

Definition 7. *Let $(N, 0, E, \gamma, \alpha)$ be a tree CSP. We define the* degree of a node *$a \in N$, $deg(a)$, as the number of edges incident on that node.*

Definition 8. *Let $(N, 0, E, \gamma, \alpha)$ be a tree CSP. A* subsource *will be either the machine, 0, or a node with a degree of at least 3. Let $\mathcal{S}$ be the set of subsources.*

Given the subsources of a tree, we are interested in knowing their *level*.

Definition 9. *Let $(N, 0, E, \gamma, \alpha)$ be a tree CSP. The* level $\ell(s)$ of a subsource *$s \in \mathcal{S}$ is the number of subsources in the path between 0 and s, including 0 and s. Thus, the machine 0 is the only subsource with level 1.*

We assume an ordering on the subsources, from level 1 to the highest level, v. Given a level $l \in \{2, \ldots, v\}$, there are m_l subsources. Thus, we can write

$$\mathcal{S} = \{0, s_1^2, \ldots, s_{m_2}^2, s_1^3, \ldots, s_{m_3}^3, \ldots, s_1^v, \ldots, s_{m_v}^v\},$$

where s_k^l denotes the kth subsource from level l. Figure 14 provides an illustration of the subsources of a tree and their levels.

The theoretical results for n-lines CSPs can be extended to the general case of trees. In particular, given a tree CSP, the elements of the merge segments obtained when solving a 2-lines CSP of the highest level remain consecutive in any optimal order. Furthermore, there always exists an optimal order that maintains the order induced by the aforementioned subproblem. With all these ingredients, it is immediate to prove that Algorithm 4 leads to an optimal order. Here, a recursive methodology is adopted, starting with the n-lines CSPs at the highest level. For each of them, the 2-lines CSPs that comprise it are solved recursively until these n-lines are converted into a single line, thus reducing the dimension. For ease of presentation, an informal description is provided in Algorithm 4, which does not intend to be mathematically exhaustive.

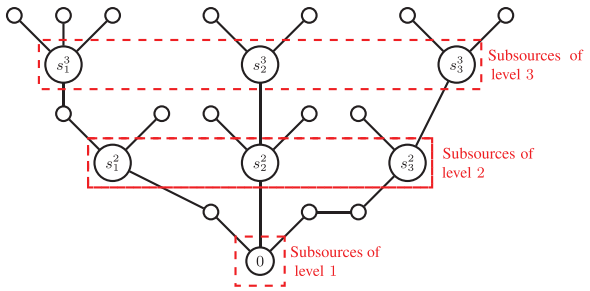


Fig. 14. Sketch of the subsources of a tree and their levels.

Algorithm 4. Algorithm to solve a tree CSP

1. Consider the tree CSP $(N, 0, E, \gamma, \alpha)$.
2. Repeat for all $k \in \{1, \ldots, m_v\}$. Consider the $n_k^v -$ lines CSP arising from s_k^v. Apply Algorithm 3 to obtain $\hat{\tau}^{N_{s_k^v}}$. Replace the n_k^v branches arising from s_k^v with one branch using the order of $\hat{\tau}^{N_{s_k^v}}$. Renumber the nodes adequately and update $\mathcal{S} = \mathcal{S} \setminus \{\cup_{k=1}^{m_v} s_k^v\}$ and $v = v - 1$. We have a new tree CSP with the highest level reduced by 1 since all the subproblems induced at subsources of the previous highest level have been converted into lines.
3. (a) If $\mathcal{S} \neq \{0\}$, go back to step 1.
   (b) If $\mathcal{S} = \{0\}$, solve the resulting n-lines CSP with Algorithm 3. The order obtained is the solution.

Let $n^{\max} = \max\{\deg(0), \max_{s \in \mathcal{S} \setminus \{0\}} \deg(s) - 1\}$. Since each induced n-lines CSP has a time complexity of $O(n^{\max}|N|^2)$, and there are $|\mathcal{S}|$ such induced n-lines CSPs, the overall time complexity of Algorithm 4 is $O(|\mathcal{S}| \cdot n^{\max} \cdot |N|^2)$. Now, we state the following result.

Theorem 3. *Let $(N, 0, E, \gamma, \alpha)$ be a tree CSP, and let $\hat{\tau}$ be an order provided by Algorithm 4. Then, $TC(\hat{\tau}) \leq TC(\tau)$ for all feasible order τ.*

Proof. This proof can be done by induction on the highest level of subsources, v, following similar arguments to those seen in the proofs of Theorems 1 and 2. □

5.2. Allocating the minimal cost

To appropriately extend the κ rule to the context of tree CSPs, we need to contemplate the local savings generated in each subsource, in a similar way to how we made the transition from 2-lines to n-lines CSPs.

Let $\mathcal{G} = (N, 0, E, \gamma, \alpha)$ be a tree CSP. For $s \in \mathcal{S}$, we define $N_s = F(s) \cup \{s\}$, where $F(s)$ is the set of followers of s with respect to 0 in the graph $(N \cup \{0\}, E)$. For every subsource $s \in \mathcal{S}$, we consider an induced n-lines CSP on N_s, $(N_s, 0, E|_{N_s}, \gamma, \alpha)$, where all initial branches with respect to s in E have been recursively replaced by a line that corresponds to an optimal order with respect to this branch. Naturally, if $\ell(s) = v$, then $(N_s, 0, E|_{N_s}, \gamma, \alpha)$ is already an n-lines CSP and we call it a subproblem at the highest level. Also, given $s \in \mathcal{S}$, let $\tau_0^{N_s}$ and $\hat{\tau}^{N_s}$ denote the corresponding benchmark order

and the optimal order provided by Algorithm 3 for $(N_s, 0, E|_{N_s}, \gamma, \alpha)$, respectively. Subsequently, the *stand-alone cost savings* $w(s)$ *with respect to* s are defined by

$$w(s) = TC(\tau_0^{N_s}) - TC(\hat{\tau}^{N_s}).$$

Thus, $w(s)$ can be interpreted as local savings made at subsources. It should be noted that the sum $\sum_{s\in\mathcal{S}} w(s)$ is not necessarily equal to the total savings $g^N = TC(\tau_0^N) - TC(\hat{\tau}^N)$. As done for n-lines CSPs, these stand-alone savings help in determining the relative importance of each subsource to realize g^N.

We will follow a recursive procedure by first solving the subproblems of the highest level. Once these problems have been solved, the highest level of the tree CSP has been reduced by 1, and we repeat the process. Hence, we will always start from an n-lines CSP, which will depend on the specific subsource we are considering. This is reflected in the notation by writing $n(s)$, $g^k(s)$, $\tau_0^k(s)$, $\hat{\sigma}^k(s)$, and $\hat{\tau}^k(s)$.

Next, we present the definition of the cost allocation rule for tree CSPs.

Definition 10. *We define the cost allocation rule,* κ*, by setting:*

$$\kappa(\mathcal{G}) = c(\tau_0^N) - g^N \cdot \sum_{s\in\mathcal{S}} \frac{w(s)}{\sum_{t\in\mathcal{S}} w(t)} \cdot \frac{1}{\sum_{k=2}^{n(s)} g^k(s)} \cdot \sum_{k=2}^{n(s)} BSR(\tau_0^k(s), \hat{\sigma}^k(s)), \tag{2}$$

for a tree CSP $\mathcal{G}$*.*

Figure 15 displays a flowchart of the allocation procedure for tree CSPs.

We conclude this section with two comments. With respect to κ, we emphasize that the objective behind it is to distribute in a fair way the savings that occur when players agree to change from a benchmark connection order to an optimal one. In this case, *fair* means that it must take into account the structure of the problem and that it must compensate all players with respect to the connection cost they had to bear in the benchmark order. In this paper, we assume that the benchmark order is the one given by the players' urgencies (τ_0), but it could be any other compatible with the connection graph. Specifically, if we write

$$\kappa^{\tau}(\mathcal{G}) = c(\tau^N) - g^N \cdot \sum_{s\in\mathcal{S}} \frac{w(s)}{\sum_{t\in\mathcal{S}} w(t)} \cdot \frac{1}{\sum_{k=2}^{n(s)} g^k(s)} \cdot \sum_{k=2}^{n(s)} \mathrm{BSR}(\tau^k(s), \hat{\sigma}^k(s)),$$

for $\tau \in \Pi(N, E)$, then $\kappa(\mathcal{G}) = \kappa^{\tau_0}(\mathcal{G})$. This general construction clearly guarantees that the following important property is satisfied.

Proposition 5. *Let* $\mathcal{G} = (N, 0, E, \gamma, \alpha)$ *be a tree CSP, and let* $\tau \in \Pi(N, E)$*. It holds that*

$$\kappa^{\tau}(\mathcal{G}) \leq c(\tau).$$

As future work, it would be of special interest to find characterizing properties that the proposed allocation rule κ^{τ_0} (or κ^{τ}) satisfies.

Furthermore, the procedure presented for calculating the κ rule for 2-lines, n-lines, and tree CSPs was limited to the case where all players have different urgencies and, consequently, there is a single

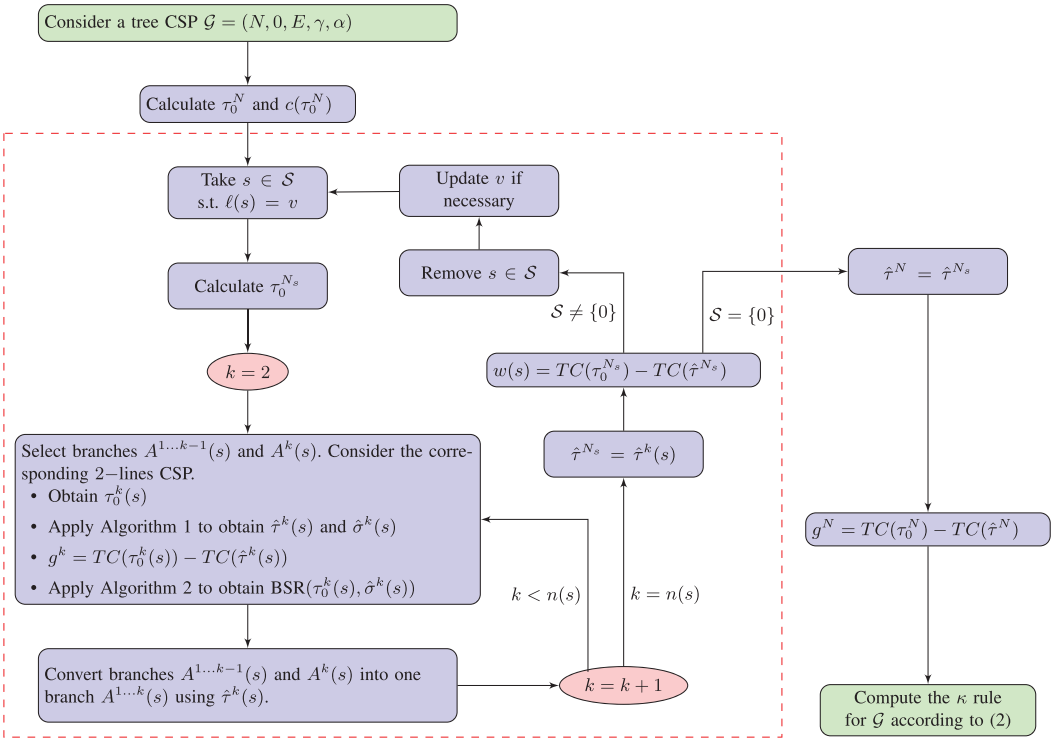


Fig. 15. Flowchart of the allocation procedure for tree CSPs. The dashed red box contains the steps of Algorithm 3.

benchmark order (also for all local problems). In the following, we will give some general indications on how to proceed when there are ties. First, we consider all possible benchmark orders. We assume that, at each step, the machine chooses with an equal probability between all jobs with the highest urgency. Thus, given the set of all possible benchmark orders, we will also have a probability distribution on this set. Now, given a fixed benchmark order, we proceed in the same way as explained above, obtaining a specific allocation proposal (which will now depend on that benchmark order). The only difference is that, when solving the *n*-lines CSPs associated with each subsource, the (local) benchmark order in these subproblems will not be recalculated according to possible ties that may exist. Instead, we will take the restriction of the (general) benchmark order that we are considering to the players involved in that subproblem. Naturally, the final cost allocation vector is defined as the weighted average of all the benchmark-specific allocation proposals, where the weights are the probabilities of each possible benchmark order.

6. Final discussion on the general CSP

Due to the complexity of solving the general CSP model, we focused on three increasingly complex situations: 2-lines, *n*-lines, and tree CSPs. For each, we proposed solution algorithms to find optimal connection orders and cost allocation procedures for the corresponding minimal connection costs. To study the *general* CSP from both an optimization and allocation perspective, different

Algorithm 5. Algorithm to solve a general CSP

0. Consider a general CSP $\mathcal{G} = (N, 0, E, \gamma, \alpha)$. Initialize $\hat{\tau} = \emptyset$, $TC(\hat{\tau}) = \infty$, and $\hat{T} = \emptyset$.
1. Apply `find_sp`$(N \cup \{0\}, E)$ to find $\mathcal{T}$, the set of all spanning trees of $(N \cup \{0\}, E)$.
2. (a) If $\mathcal{T} \neq \emptyset$, take $T \in \mathcal{T}$. Apply Algorithm 4 to $\mathcal{G}|_T$, leading to an optimal order τ for $\mathcal{G}|_T$.
   - If $TC(\tau) < TC(\hat{\tau})$, then $TC(\hat{\tau}) = TC(\tau)$, $\hat{\tau} = \tau$, and $\hat{T} = T$. Let $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T\}$. Go back to step 2.
   - Else, let $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T\}$. Go back to step 2.

   (b) If $\mathcal{T} = \emptyset$, then the algorithm is finished. The order obtained, $\hat{\tau}$, is the solution, and $\hat{T}$ is the spanning tree that led to $\hat{\tau}$.

approaches can be considered. Let $\mathcal{G} = (N, 0, E, \gamma, \alpha)$ be a CSP, where $(N \cup \{0\}, E)$ is the connected graph induced by $\mathcal{G}$. One possible approach is to generate the set of all spanning trees of $(N \cup \{0\}, E)$, denoted as $\mathcal{T}$, and apply Algorithm 4 to each tree CSP $\mathcal{G}|_T$ derived from restricting $\mathcal{G}$ to a spanning tree $T \in \mathcal{T}$. The solution would be the optimal order with the minimal total cost among such restricted tree CSPs. Let `find_sp` be an algorithm that finds all spanning trees of a connected graph. Algorithm 5 outlines the proposed procedure. Regarding algorithms for finding all spanning trees, we refer the reader to the review by Chakraborty et al. (2019), which extensively analyzes and classifies various approaches based on computational time, application areas, and other characteristics. Considering that CSPs stem from applications in energy supply and telecommunications, which are typically modeled by sparse graphs, we believe that the test and select method proposed by Onete and Onete (2010) could be a suitable option in this case. For cost allocation in the general CSP $\mathcal{G}$, we then would propose applying the κ rule from equation (2) to the tree CSP $\mathcal{G}|_{\hat{T}}$, where $\hat{T}$ is the spanning tree that led to the globally optimal connection order in Algorithm 5.

Acknowledgments

Laura Davila-Pena's research was funded by the Ministry of Education, Culture and Sports of Spain (contract FPU17/02126). This work is part of the R&D projects MTM2017-87197-C3-1-P, MTM2017-87197-C3-3-P, PID2021-124030NB-C31, and PID2021-124030NB-C32, granted by MICIU/AEI/10.13039/501100011033/ and by "ERDF A way of making Europe"/EU. This research was also funded by the Xunta de Galicia (Grupos de Referencia Competitiva ED431C 2021/24 and ED431C 2020/14). The authors would like to thank the editor-in-chief, the associate editor, and the three anonymous referees for their valuable comments, which helped improve an earlier version of this paper.

References

Adamopoulos, G.I., Pappis, C.P., Karacapilidis, N.I., 1999. Job sequencing with uncertain and controllable processing times. *International Transactions in Operational Research* 6, 5, 483–493.

Adolphson, D., Hu, T.C., 1973. Optimal linear ordering. *SIAM Journal on Applied Mathematics* 25, 3, 403–423.

Bai, F., Zhang, C., Zhang, X., 2024. Intelligent optimal demand response implemented by blockchain and cooperative game in microgrids. *International Transactions in Operational Research* 31, 6, 3704–3731.

Baker, K.R., 1971. *Single machine sequencing with weighting factors and precedence constraints*. Unpublished work. Department of Industrial Engineering, University of Michigan.

Bergantiños, G., Gómez-Rúa, M., Llorca, N., Pulido, M., Sánchez-Soriano, J., 2014. A new rule for source connection problems. *European Journal of Operational Research* 234, 3, 780–788.

Bergantiños, G., Lorenzo, L., 2004. A non-cooperative approach to the cost spanning tree problem. *Mathematical Methods of Operations Research* 59, 3, 393–403.

Bergantiños, G., Vidal-Puga, J., 2021. A review of cooperative rules and their associated algorithms for minimum-cost spanning tree problems. *SERIEs* 12, 1, 73–100.

Chakraborty, M., Chowdhury, S., Chakraborty, J., Mehera, R., Pal, R.K., 2019. Algorithms for generating all possible spanning trees of a simple undirected connected graph: an extensive review. *Complex & Intelligent Systems* 5, 265–281.

Cheng, T.C.E., Gupta, M.C., 1989. Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research* 38, 2, 156–166.

Claus, A., Kleitman, D.J., 1973. Cost allocation for a spanning tree. *Networks* 3, 4, 289–304.

Curiel, I., 1997. Minimum cost spanning tree games. In *Cooperative Game Theory and Applications*. Springer, Boston, MA, pp. 129–148.

Curiel, I., Hamers, H., Klijn, F., 2002. Sequencing games: a survey. In Borm, P. and Peters, H. (eds), *Chapters in Game Theory: In honor of Stef Tijs*. Springer, Boston, MA, pp. 27–50.

Curiel, I., Pederzoli, G., Tijs, S., 1989. Sequencing games. *European Journal of Operational Research* 40, 3, 344–351.

Estévez-Fernández, A., Reijnierse, H., 2014. On the core of cost-revenue games: Minimum cost spanning tree games with revenues. *European Journal of Operational Research* 237, 2, 606–616.

Feltkamp, V., Tijs, S., Muto, S., 1994. On the irreducible core and the equal remaining obligations rule of minimum cost spanning extension problems. Working paper. Center for Economic Research 106, Tilburg University, Tilburg, the Netherlands.

Granot, D., Huberman, G., 1981. Minimum cost spanning tree games. *Mathematical Programming* 21, 1, 1–18.

Granot, D., Huberman, G., 1984. On the core and nucleolus of the minimum cost spanning tree games. *Mathematical Programming* 29, 3, 323–347.

Hamers, H., Klijn, F., van Velzen, B., 2005. On the convexity of precedence sequencing games. *Annals of Operations Research* 137, 1, 161–175.

Hamers, H., Suijs, J., Tijs, S., Borm, P., 1996. The split core for sequencing games. *Games and Economic Behavior* 15, 2, 165–176.

Horn, W., 1972. Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal on Applied Mathematics* 23, 2, 189–202.

Kruskal, J.B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* 7, 1, 48–50.

Lawler, E.L., 1978. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics* 2, 75–90.

Moretti, S., Branzei, R., Norde, H., Tijs, S., 2004. The P-value for cost sharing in minimum cost spanning tree situations. *Theory and Decision* 56, 1, 47–61.

Onete, C.E., Onete, M.C.C., 2010. Enumerating all the spanning trees in an un-oriented graph-a novel approach. In *2010 XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD)*, Gammarth, Tunisia. IEEE, Piscataway, NJ, 1–5.

Prim, R.C., 1957. Shortest connection networks and some generalizations. *The Bell System Technical Journal* 36, 6, 1389–1401.

Prot, D., Bellenguez-Morineau, O., 2018. A survey on how the structure of precedence constraints may change the complexity class of scheduling problems. *Journal of Scheduling* 21, 1, 3–16.

Saavedra-Nieves, A., Mosquera, M.A., Fiestras-Janeiro, M.G., 2025. Sequencing situations with position-dependent effects under cooperation. *International Transactions in Operational Research* 32, 3, 1620–1640.

Sidney, J.B., 1975. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research* 23, 2, 283–298.

Smith, W.E., 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3, 1–2, 59–66.

Appendix A.

Proof of Proposition 1. Consider:

$$\tau_1 = (\sim, X, Y, Z, \sim);$$
$$\tau_2 = (\sim, Z, X, Y, \sim),$$

where Z is either moved to the back of Y or to the front of X in τ_1 and τ_2 in relation to τ, respectively. For τ_1, this means that completion times of all players in Z are increased by the sum of all processing times of players in Y, $\gamma[Y]$, and completion times of all players in Y are decreased by $\gamma[Z]$. Additionally, players in $N \setminus \{Y \cup Z\}$ remain with the same predecessors in τ and τ_1 and thus their completion times and costs do not change. Therefore, the difference in the total costs is

$$\begin{aligned} TC(\tau) - TC(\tau_1) &= \alpha[Y] \cdot \gamma[Z] - \alpha[Z] \cdot \gamma[Y] = \alpha[Y] \cdot \alpha[Z] \cdot \frac{\gamma[Z]}{\alpha[Z]} - \alpha[Z] \cdot \alpha[Y] \cdot \frac{\gamma[Y]}{\alpha[Y]} \\ &= \alpha[Y] \cdot \alpha[Z] \cdot (CAT(Z) - CAT(Y)). \end{aligned}$$

Using similar arguments as above for τ and τ_2,

$$\begin{aligned} TC(\tau) - TC(\tau_2) &= \alpha[Z] \cdot \gamma[X] - \alpha[X] \cdot \gamma[Z] = \alpha[Z] \cdot \alpha[X] \cdot \frac{\gamma[X]}{\alpha[X]} - \alpha[X] \cdot \alpha[Z] \cdot \frac{\gamma[Z]}{\alpha[Z]} \\ &= \alpha[Z] \cdot \alpha[X] \cdot (CAT(X) - CAT(Z)). \end{aligned}$$

Suppose, for the sake of contradiction, that τ is optimal. Then,

$$TC(\tau) - TC(\tau_1) \leq 0 \quad \text{and} \quad TC(\tau) - TC(\tau_2) \leq 0,$$

and hence, using the equalities above,

$$CAT(X) \leq CAT(Z) \leq CAT(Y),$$

which is a contradiction. Thus, τ cannot be optimal. □

Proof of Lemma 1. Since $|M_k| > 1$, it holds that M_k is not the first pivot in Algorithm 1. In order for M_k to become the new pivot and, hence, the merge segment, it must have a lower CAT than the previous pivot. Furthermore, the CAT of the previous pivot is less than or equal to the CAT of the combination of the first $m_k - 1$ components of M_k. This might be a direct comparison, but it could also be an indirect comparison via several other pivots. That is,

$$CAT(M_k) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1}). \tag{A1}$$

Then,

$$\frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]} \underset{(A1)}{<} \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]},$$

and, consequently,

$$\gamma[G_{m_k}] \cdot \left(\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]\right) < \left(\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]\right) \cdot \alpha[G_{m_k}]. \quad \text{(A2)}$$

Hence,

$$\frac{\gamma[G_{m_k}]}{\alpha[G_{m_k}]} < \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]}.$$

Thus, $CAT(G_{m_k}) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k-1})$, proving i).

To prove ii), we add $\gamma[G_{m_k}] \cdot \alpha[G_{m_k}]$ on both sides of equation (A2):

$$\begin{aligned}&\gamma[G_{m_k}] \cdot \left(\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}]\right) + \gamma[G_{m_k}] \cdot \alpha[G_{m_k}] \\ &< \left(\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}]\right) \cdot \alpha[G_{m_k}] + \gamma[G_{m_k}] \cdot \alpha[G_{m_k}],\end{aligned}$$

which results in

$$\begin{aligned}&\gamma[G_{m_k}] \cdot \left(\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]\right) \\ &< \left(\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]\right) \cdot \alpha[G_{m_k}].\end{aligned}$$

Consequently,

$$\frac{\gamma[G_{m_k}]}{\alpha[G_{m_k}]} < \frac{\gamma[G_1] + \gamma[G_2] + \cdots + \gamma[G_{m_k-1}] + \gamma[G_{m_k}]}{\alpha[G_1] + \alpha[G_2] + \cdots + \alpha[G_{m_k-1}] + \alpha[G_{m_k}]},$$

and thus $CAT(G_{m_k}) < CAT(G_1 \cup G_2 \cup \cdots \cup G_{m_k}) = CAT(M_k)$, proving (ii).

To prove (iii), note that $CAT(G_{m_k}) < CAT(M_k) < CAT(G_1)$, where the first inequality follows from (ii) and the second inequality from Algorithm 1: we know that $CAT(G_1)$ is higher than the CAT of a segment containing all players in the opposite branch, either by direct or indirect comparison. Otherwise, G_1 had become a merge segment by itself. □

Proof of Lemma 2. Suppose for the sake of contradiction that $CAT(G_{k-1}) \leq CAT(G_k)$ for all $k \in \{2, 3, \ldots, m_k\}$. That is,

$$CAT(G_1) \leq CAT(G_2) \leq \cdots \leq CAT(G_{m_k-1}) \leq CAT(G_{m_k}).$$

This implies that $CAT(G_1) \leq CAT(G_{m_k})$, contradicting (iii) of Lemma 1. Hence, there exists $k \in \{2, 3, \ldots, m_k\}$ such that $CAT(G_{k-1}) > CAT(G_k)$. □

Proof of Lemma 3. Let τ be an optimal order. Suppose that there exists $k \in \{1, \ldots, m\}$ such that players from M_k are separated by other players in τ. Also consider the set of components of M_k in τ, that is, $M_k/\tau = \{G_1, G_2, \ldots, G_{m_k}\}$, so we would have

$$\tau = (\sim, G_1, \ldots, G_2, \ldots, G_{m_k}, \sim).$$

From Lemma 2, we know that there exists $\tilde{k} \in \{2, \ldots, m_k\}$ such that $CAT(G_{\tilde{k}-1}) > CAT(G_{\tilde{k}})$. Rewrite τ as follows:

$$\tau = (\sim, G_{\tilde{k}-1}, Z, G_{\tilde{k}}, \sim),$$

where Z is a segment from the other branch. From Proposition 1, τ is not optimal, which is a contradiction. Hence, $\left|M_k/\tau\right| = 1$ for all $k \in \{1, \ldots, m\}$. □

Proof of Proposition 2. Consider $\tau \in \Pi(A \cup B, E)$ such that τ does not start with M_1. We will first prove that there always exists τ^* starting with M_1 such that $TC(\tau^*) \leq TC(\tau)$.

Assume, without loss of generality, that $M_1 \subseteq A$. During Algorithm 1, we compared the CAT of M_1 with the $CATs$ of all possible segments R_{1l}, $l \in \{1, \ldots, \tilde{t}\}$. It holds that

$$\begin{aligned} &CAT(M_1) \leq CAT(R_{11});\\ &CAT(M_1) \leq CAT(R_{12});\\ &\qquad\vdots\\ &CAT(M_1) \leq CAT(R_{1\tilde{t}}). \end{aligned} \tag{A3}$$

In case $|M_1/\tau| = 1$, M_1 is a connected component in τ. If τ does not start with M_1, then it must start with some players from branch B, followed by M_1:

$$\tau = (R_{1\ell}, M_1, \sim),$$

where $1 \leq \ell \leq \tilde{t}$. Now, consider the following order:

$$\tau_1 = (M_1, R_{1\ell}, \sim),$$

in which we have swapped the positions of M_1 and $R_{1\ell}$. Then,

$$\begin{aligned} TC(\tau) - TC(\tau_1) &= \alpha[M_1] \cdot \alpha[R_{1\ell}] \cdot CAT(R_{1\ell}) - \alpha[R_{1\ell}] \cdot \alpha[M_1] \cdot CAT(M_1)\\ &= \alpha[M_1] \cdot \alpha[R_{1\ell}] \cdot (CAT(R_{1\ell}) - CAT(M_1)) \underset{(A3)}{\geq} 0, \end{aligned} \tag{A4}$$

and hence, τ is not better than τ_1.

In case $|M_1/\tau| > 1$, then we can write τ as

$$\tau = (\sim, G_1, \ldots, G_2, \ldots, G_3, \ldots, G_{m_1}, \sim),$$

with $m_1 > 1$, where $M_1/\tau = \{G_1, G_2, G_3, \ldots, G_{m_1}\}$ denotes the set of components of M_1 in τ.

From Lemma 3, τ cannot be optimal since the elements of M_1 are not consecutive. In particular, we know there exists a bijection

$$\rho : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, m\}$$

such that $\tau' = (M_{\rho(1)}, \ldots, M_{\rho(m)})$ is an optimal order. Take $\hat{i} \in \{1, \ldots, m\}$ such that $\rho(\hat{i}) = 1$. Next consider

$$\tau^* = (M_{\rho(\hat{i})}, M_{\rho(1)}, M_{\rho(2)}, \ldots, M_{\rho(\hat{i}-1)}, M_{\rho(\hat{i}+1)}, \ldots, M_{\rho(m)}).$$

Then,

$$TC(\tau^*) \leq TC(\tau') < TC(\tau),$$

where the first inequality follows from a similar reasoning that leads to (A4) and the second inequality from Lemma 3. □

Proof of Proposition 3. Without loss of generality, we assume that $M_1 \subseteq A$. If $M_1 = A$, then it is clear that $\tau^N = (M_1, b_1, b_2, \ldots, b_{\tilde{t}})$, where $\tau^N|_{N\setminus M_1} = (b_1, b_2, \ldots, b_{\tilde{t}})$ is the optimal order of the 1-line CSP with a set of players $N \setminus M_1 = \{b_1, b_2, \ldots, b_{\tilde{t}}\}$. Let us suppose now that $M_1 \subsetneq A$. For the sake of contradiction, assume that $\tau^N|_{N\setminus M_1}$ is not an optimal order for the aforementioned subproblem. Then, there would exist $\hat{\tau}^{N\setminus M_1}$ such that

$$TC(\hat{\tau}^{N\setminus M_1}) - TC(\tau^N|_{N\setminus M_1}) < 0. \tag{A5}$$

Consider the order $\hat{\tau}^N = (M_1, \hat{\tau}^{N\setminus M_1})$. Note that

$$TC(\hat{\tau}^N) - TC(\tau^N) = TC(\hat{\tau}^{N\setminus M_1}) - TC(\tau^N|_{N\setminus M_1}) \underset{(A5)}{<} 0,$$

which is a contradiction because τ^N is optimal. Thus, $\tau^N|_{N\setminus M_1}$ is an optimal order for the problem with a set of players $N \setminus M_1$. □

Proof of Lemma 4. From Proposition 2, we know there exists an optimal order $\hat{\tau}^N$ for $(N, 0, E, \gamma, \alpha)$ that starts with M_1, so $\hat{\tau}^N = (M_1, \hat{\tau}^N|_{N\setminus M_1})$. From Proposition 3, $\hat{\tau}^N|_{N\setminus M_1}$ is an optimal order for the subproblem with a set of players $N \setminus M_1$. For the sake of contradiction, suppose that τ^N is not optimal. Then,

$$TC(\hat{\tau}^N) - TC(\tau^N) = TC(\hat{\tau}^N|_{N\setminus M_1}) - TC(\tau^{N\setminus M_1}) < 0,$$

which contradicts $\tau^{N\setminus M_1}$ being optimal for the problem with a set of players $N \setminus M_1$. □

Proof of Theorem 1. The proof uses induction to the number of players, $|N|$.

Consider $|N| = 2$. In such a case, there are two possible orders, $\tau_1 = (a_1, b_1)$ and $\tau_2 = (b_1, a_1)$. Note that

$$TC(\tau_1) = (\alpha(a_1) + \alpha(b_1)) \cdot \gamma(a_1) + \alpha(b_1) \cdot \gamma(b_1);$$
$$TC(\tau_2) = (\alpha(b_1) + \alpha(a_1)) \cdot \gamma(b_1) + \alpha(a_1) \cdot \gamma(a_1),$$

and thus

$$TC(\tau_1) - TC(\tau_2) = \alpha(b_1) \cdot \gamma(a_1) - \alpha(a_1) \cdot \gamma(b_1) = \alpha(b_1) \cdot \alpha(a_1) \cdot \left(\frac{\gamma(a_1)}{\alpha(a_1)} - \frac{\gamma(b_1)}{\alpha(b_1)}\right). \tag{A6}$$

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies.

Algorithm 1 compares $\frac{\gamma(a_1)}{\alpha(a_1)}$ to $\frac{\gamma(b_1)}{\alpha(b_1)}$ in order to choose the first merge segment, which in this case will consist of a single node. From (A6), we can see that the optimal order will be determined by the exact same comparison, thus Algorithm 1 leads to an optimal order.

Now, assume that Algorithm 1 leads to an optimal order if the number of players is $k < |N|$.

Now, take $k = |N|$. Let $\hat{\sigma} = (M_1, M_2, \ldots, M_m)$ be the output of Algorithm 1 corresponding to $\hat{\tau}$. Naturally, $\hat{\sigma}|_{N\setminus M_1} = (M_2, \ldots, M_m)$ will be an output of our procedure for the problem with a set of players $N \setminus M_1$. Using our induction hypothesis, $\hat{\sigma}|_{N\setminus M_1}$ is optimal for such a subproblem. From Lemma 4, the order $(M_1, \hat{\sigma}|_{N\setminus M_1})$ is optimal. Clearly, $\hat{\sigma} = (M_1, \hat{\sigma}|_{N\setminus M_1})$, finishing the proof. □

Proof of Proposition 4. Let $\sigma_{12}^* = (M_1, \ldots, M_m)$ be the output of Algorithm 1 for $(A^1 \cup A^2, 0, E|_{A^1\cup A^2}, \gamma, \alpha)$ that has τ_{12}^* as its associated order, and let τ' be an optimal order for $(N, 0, E, \gamma, \alpha)$. From Lemma 5, we know that the elements of $M_k,\ k \in \{1, \ldots, m\}$, are consecutive in τ'. Assume that τ' does not respect the relative order induced by τ_{12}^*, and let $M_k, M_l,\ k, l \in \{1, \ldots, m\}$ be the first merge segments such that $\tau_{12}^*(M_k) > \tau_{12}^*(M_l)$ but $\tau'(M_k) < \tau'(M_l)$. Note that M_k and M_l necessarily belong to different branches. We also have that

$$CAT(M_l) \leq CAT(M_k). \tag{A7}$$

Thus, between M_k and M_l in τ', players from all branches except that of M_l can be present. Let $M_k^1, M_k^2, \ldots, M_k^q$ be the maximal connected segments from the branch of M_k that are between M_k and M_l in τ', and let $Z^1, Z^2, \ldots, Z^q, Z^{q+1}$ be the (potential) maximal connected sets of nodes from $\mathcal{A} \setminus \{A^1 \cup A^2\}$ in τ'. There are four possible cases:

(i) $\tau' = (\sim, M_k, Z^1, M_k^1, Z^2, M_k^2, \ldots, Z^q, M_k^q, Z^{q+1}, M_l, \sim)$,
(ii) $\tau' = (\sim, M_k, Z^1, M_k^1, Z^2, M_k^2, \ldots, Z^q, M_k^q, M_l, \sim)$,
(iii) $\tau' = (\sim, M_k, M_k^1, Z^1, M_k^2, \ldots, Z^{q-1}, M_k^q, Z^q, M_l, \sim)$,
(iv) $\tau' = (\sim, M_k, M_k^1, Z^1, M_k^2, \ldots, Z^{q-1}, M_k^q, M_l, \sim)$.

We will only consider (i), since the other cases can be treated in an analogous way. We will first show that

$$\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} \leq CAT(M_k^{\tilde{q}}) \leq \frac{\gamma[Z^{\tilde{q}+1}]}{\alpha[Z^{\tilde{q}+1}]}, \tag{A8}$$

for all $\tilde{q} \in \{1, \ldots, q+1\}$. Suppose that (A8) does not hold. Then, there exists a $\tilde{q} \in \{1, \ldots, q+1\}$ such that

$$\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} > AC(M_k^{\tilde{q}}), \tag{A9}$$

or

$$AC(M_k^{\tilde{q}}) > \frac{\gamma[Z^{\tilde{q}+1}]}{\alpha[Z^{\tilde{q}+1}]}. \tag{A10}$$

Consider the order τ_1 as a modification of τ' in which the position of $Z^{\tilde{q}}$ and $M_k^{\tilde{q}}$ is swapped. Then,

$$TC(\tau') - TC(\tau_1) = \alpha[M_k^{\tilde{q}}] \cdot \alpha[Z^{\tilde{q}}] \cdot \left(\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} - CAT(M_k^{\tilde{q}}) \right) \underset{(A9)}{>} 0,$$

which contradicts τ' from being optimal. Thus, (A9) cannot hold. Analogously, consider the order τ_2 as a modification of τ' in which the position of $M_k^{\tilde{q}}$ and $Z^{\tilde{q}+1}$ is swapped. Then,

$$TC(\tau') - TC(\tau_2) = \alpha[Z^{\tilde{q}+1}] \cdot \alpha[M_k^{\tilde{q}}] \cdot \left(CAT(M_k^{\tilde{q}}) - \frac{\gamma[Z^{\tilde{q}+1}]}{\alpha[Z^{\tilde{q}+1}]} \right) \underset{(A10)}{>} 0,$$

which contradicts τ' from being optimal. Thus, (A10) cannot hold. This proves (A8). Using similar arguments, it can be shown that

$$CAT(M_k) \leq \frac{\gamma[Z^1]}{\alpha[Z^1]} \quad \text{and} \quad \frac{\gamma[Z^{q+1}]}{\alpha[Z^{q+1}]} \leq CAT(M_l). \tag{A11}$$

From (A8) and (A11), it follows that

$$CAT(M_k) \leq \frac{\gamma[Z^1]}{\alpha[Z^1]} \leq CAT(M_k^1) \leq \cdots \leq \frac{\gamma[Z^q]}{\alpha[Z^q]} \leq CAT(M_k^q) \leq \frac{\gamma[Z^{q+1}]}{\alpha[Z^{q+1}]} \leq CAT(M_l). \tag{A12}$$

By combining (A7) and (A12), we obtain that $CAT(M_k) = CAT(M_l)$, which in consequence leads to

$$CAT(M_k) = \frac{\gamma[Z^1]}{\alpha[Z^1]} = CAT(M_k^1) = \cdots = \frac{\gamma[Z^q]}{\alpha[Z^q]} = CAT(M_k^q) = \frac{\gamma[Z^{q+1}]}{\alpha[Z^{q+1}]} = CAT(M_l). \tag{A13}$$

Next, consider

$$\hat{\tau} = (\sim, M_l, M_k, Z^1, M_k^1, Z^2, M_k^2, \ldots, Z^q, M_k^q, Z^{q+1}, \sim),$$

which results from moving M_l to the front of M_k in τ'. Note that

$$\begin{aligned}
TC(\tau') - TC(\hat{\tau}) = {} & \alpha[M_l] \cdot \left(\alpha[M_k] \cdot CAT(M_k) + \textstyle\sum_{\tilde{q}=1}^{q} \alpha[M_k^{\tilde{q}}] \cdot CAT(M_k^{\tilde{q}}) + \sum_{\tilde{q}=1}^{q+1} \alpha[Z^{\tilde{q}}] \cdot \frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} \right) \\
& - \alpha[M_l] \cdot \left(\alpha[M_k] + \textstyle\sum_{\tilde{q}=1}^{q} \alpha[M_k^{\tilde{q}}] + \sum_{\tilde{q}=1}^{q+1} \alpha[Z^{\tilde{q}}] \right) \cdot CAT(M_l) \\
= {} & \alpha[M_l] \cdot \alpha[M_k] \cdot (CAT(M_k) - CAT(M_l)) + \alpha[M_l] \cdot \textstyle\sum_{\tilde{q}=1}^{q} \alpha[M_k^{\tilde{q}}] \cdot (CAT(M_k^{\tilde{q}}) - CAT(M_l)) \\
& + \alpha[M_l] \cdot \textstyle\sum_{\tilde{q}=1}^{q+1} \alpha[Z^{\tilde{q}}] \cdot \left(\frac{\gamma[Z^{\tilde{q}}]}{\alpha[Z^{\tilde{q}}]} - CAT(M_l) \right) \\
\underset{(A13)}{=} {} & 0.
\end{aligned}$$

This implies that $\hat{\tau}$ is an optimal order. □

Proof of Theorem 2. We will use induction in the number of branches, n.

If $n = 2$, Algorithm 3 coincides with Algorithm 1.

Suppose that Algorithm 3 leads to an optimal solution for all $k < n$.

Now, take $k = n$. We can select branches A^1 and A^2 and apply Algorithm 1 to obtain a relative order, τ_{12}. Using Proposition 4, there exists an optimal order $\hat{\tau}$ for $(N, 0, E, \gamma, \alpha)$ that maintains the order induced from branches A^1 and A^2. We can convert these two branches into one, A^{12}_{τ}, reducing the dimension of our problem by 1. Now we have an $(n-1)$-lines CSP, for which it is clear that $\hat{\tau}$ is also an optimal order. By the induction hypothesis, Algorithm 3 leads to an optimal solution, τ^*, for the $(n-1)$-lines CSP. It is straightforward to prove that τ^* is also an optimal solution for the n-lines CSP. For if not, there would exist an optimal order $\tilde{\tau}$ such that $TC(\tilde{\tau}) < TC(\tau^*)$. Thus, $TC(\hat{\tau}) = TC(\tilde{\tau}) < TC(\tau^*)$, contradicting the induction hypothesis. □