Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-10 (2025) 1355–1360

Fair resource-constrained allocation of task-chains

Arianna Freda * **Gaia Nicosia** * **Andrea Pacifici** **

* *Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche, Università degli Studi Roma Tre, Roma, Italy (e-mail: {arianna.freda,gaia.nicosia}@uniroma3.it)*

** *Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università degli Studi di Roma Tor Vergata, Roma, Italy (e-mail: andrea.pacifici@uniroma2.it)*

Abstract: This work explores a novel resource allocation problem where a limited resource, such as machine time or budget, is distributed among multiple agents over discrete time slots. Each agent has indivisible unit demands and preferences for the order in which their units are served. The study aims to find fair solutions that balance agents' individual preferences and overall efficiency. To tackle this challenge, a Mixed-Integer Linear Programming (MILP) model is proposed to account for both demand allocation and order preferences. Computational experiments assess the model's effectiveness and evaluate the trade-off between fairness and efficiency. Results indicate that in small instances, fair solutions remain close to the system optimum with minimal efficiency loss. However, as complexity increases, maintaining fairness becomes significantly more costly.

Keywords: Job and activity scheduling, Multi-agent systems applied to industrial systems, Modelling and decision making in complex systems, Methodologies and tools for analysis of complexity, Mixed Integer Programming, Fairness

1. INTRODUCTION

In recent years, optimization problems have increasingly integrated multi-agent systems and fairness considerations, reflecting a growing awareness of the need to balance diverse objectives and interactions among stakeholders (see, e.g., Agnetis et al., 2025, 2015; Bertsimas et al., 2011; He et al., 2022; Naldi et al., 2019; Jiang et al., 2021). Within this framework, we examine a resource allocation problem where a resource—such as time on a manufacturing plant's processing machine or a company's project budget—is distributed among several distinct players, referred to as *agents*. The resource is allocated in discrete *parcels*, representing limited quantities over specific periods. The moments when this resource becomes available to the agents are termed *slots*. The allocation process for each agent is characterized by quantities indicating the amount of resource provided and the corresponding slots during which it is supplied. In this process, each agent requires the resource in a number of indivisible packets, i.e., given quantities of resource whose demand must be satisfied in full in one slot, meaning, for the entirety of that specific packet request, or not at all. Moreover, an agent has a preference on the (strict) order in which the packets must be served, i.e., the demand of each agent can be represented as a finite sequence of resource quantities, each corresponding to a packet.

This problem is related to several well-studied problems in the literature. For example, it shares similarities with the resource-constrained scheduling problem with temporal constraints as well with some special parallel batch scheduling problems. Also, the comparison between fair solutions and the system's optimal solution in multi-agent problems has been extensively explored in the literature. However, to the best of our knowledge, no previous study has addressed a problem exactly like ours, particularly in terms of resource distribution in discrete parcels over time.

The paper is organized as follows. After briefly discussing related literature in the next section, we outline the motivating application for addressing the problem in Section 3. Section 4 provides a formal definition of the problem, followed by the presentation of the integer linear programming model in Section 5. In Section 6, we detail the results of preliminary experiments. We draw some final considerations and future directions in Section 7.

2. RELATED LITERATURE

In recent years, there has been a growing focus on incorporating multi-agent systems and fairness principles into optimization problems across various domains. In resource allocation and scheduling, problems involving multiple agents have been explored from various perspectives, including the pursuit of Pareto optimal solutions, trade-off

10.1016/j.ifacol.2025.09.228

strategies, and approaches aimed specifically at achieving fairness in the distribution of resources.

Fairness concepts have long been integrated into telecommunication networks alongside optimization, particularly in resource allocation. However, QoS requirements can disadvantage lower-priority users, making dynamic bandwidth allocation and infrastructure upgrades essential to balancing fairness and performance (Haslak, 2020; Faruque, 2018; Sousa et al., 2020).

More recently, the principles of equitable resource allocation have also been applied to scheduling. Jobs are divided among two or more agents with distinct, often conflicting objectives. These objectives typically depend solely on the completion times of each agent's specific jobs (Agnetis et al., 2014). In the context of resource-constrained project scheduling, a number of problems in which different agents compete for the usage of the scarce resource with time-dependent costs have been considered. These types of problems consists of scheduling a set of activities subject to precedence and resource constraints, minimizing the makespan and the total cost for resource usage (Alcaraz et al., 2022; Rodríguez-Ballesteros et al., 2024). Notably, the limited resource distributed over time, as discussed in this paper, also bears similarities to certain parallel-batching scheduling problems, where a machine processes unit-time jobs grouped into batches, each of which may have a different size (Liu et al., 2009). Parallel-batching problems have been extended to scenarios involving multiple agents (see, e.g., Gao et al., 2019; He et al., 2022; Li and Yuan, 2012). In such cases, the focus has typically been on optimizing shared resources while accounting for the distinct requirements or constraints of each agent.

A few multi-agent scheduling studies explore the role of a third-party *supervisor* (see, e.g., Marini et al., 2013) enforcing rules in order to mitigate anarchic agents' behavior and ensure acceptable performance. In particular, a supervisor can ensure that no agent is excessively disadvantaged. To quantify the trade-off between achieving fairness and optimizing efficiency, the concept of *Price of Fairness*, introduced by Bertsimas et al. (2011) and Caragiannis et al. (2009), is defined as the (relative) difference between the optimal value of a system's performance objective (e.g., total completion time) and the value of the same performance indicator under a fairness-constrained allocation. This notion has emerged as a critical consideration in different scheduling problems, highlighting the importance of equitable resource allocation across various scenarios (Agnetis et al., 2019; Cosmi et al., 2022; Heeger et al., 2022; Hermelin et al., 2025; Zhang et al., 2020).

While problems related to ours can be found in the literature, they generally differ in their structure, assumptions, or the specific interactions among diverse agents. This distinction underscores the novelty and significance of the present work, as it tackles a unique combination of features that have not been thoroughly examined before.

3. MOTIVATING APPLICATIONS

In this section, we present two applications that have inspired this study. The underlying decision problems are fundamentally similar, differing slightly in whether unused portions of the resource can be carried forward to future time periods. This distinction gives rise to two variants of our problem, which are described in greater detail in Section 4.

3.1 Competing departments of a manufacturing facility

In a company, multiple departments work on various projects, all of which rely on a *single shared* processing resource (such as a machine or a dedicated team). Each department's project consists of a series of tasks, arranged in a strict sequence that dictates their execution order. Each task requires a certain amount of the shared resource (for example, processing time) and carries a specific value or *weight*, representing the importance of completing that task as early as possible. The processing resource is available in a series of time slots with known durations. However, the exact timing of when these slots become available is uncertain.

Given this setup, the departments are in competition to have their tasks scheduled as early as possible. An earlier time slot is preferable because it ensures a task's timely completion. However, due to the uncertainty in the release times of these slots, the exact completion time of each task cannot be determined *a priori*. Therefore, departments focus on positioning their tasks in the sequence of slots, prioritizing earlier slots over later ones.

To effectively allocate tasks, the assignment must satisfy constraints that follows. (*i*) Resource Capacity Constraint: The total resource requirement of tasks assigned to a specific time slot must not exceed the slot's duration. (*ii*) Task Precedence Constraint: Tasks within a project must respect their predefined sequence. If task h precedes task i in the sequence, then i must be scheduled in the same or a later time slot than h. (*iii*) Non-Preemptive Tasks: Each task must be fully completed within the time slot to which it is assigned. Tasks cannot be split across multiple time slots. This setup ensures that tasks are processed efficiently while adhering to both resource constraints and the strict ordering required by each department's project structure.

Example. As an example, consider the case of $n = 4$ agents' projects A, B, C, and D, q_j tasks for agent j, and $T = 3$ time slots with $b_1 = 47$ and $b_2 = 19$ and $b_3 = 34$ respective durations (resource availabilities). The following table reports the tasks' processing time (resource requirements).

Table 1. Resource requirements of agents' tasks

Proj. j	q_j	$i=1$	$i=2$	$i=3$	$i=4$
A (grey)	4	8	5	8	4
B (yellow)	2	8	12	-	-
C (red)	4	7	7	8	3
D (blue)	3	5	10	6	-

Figure 1 gives an illustration of a feasible assignment for this situation. In the picture, the i-th task of agent j is indicated as j_i (j = A, B, C, D—for ease of legibility, each with an identifying color—and $i = 1, \ldots, q_j$). Task assignments are clearly readable in the figure and satisfy the above mentioned requirements. As it is shown in the

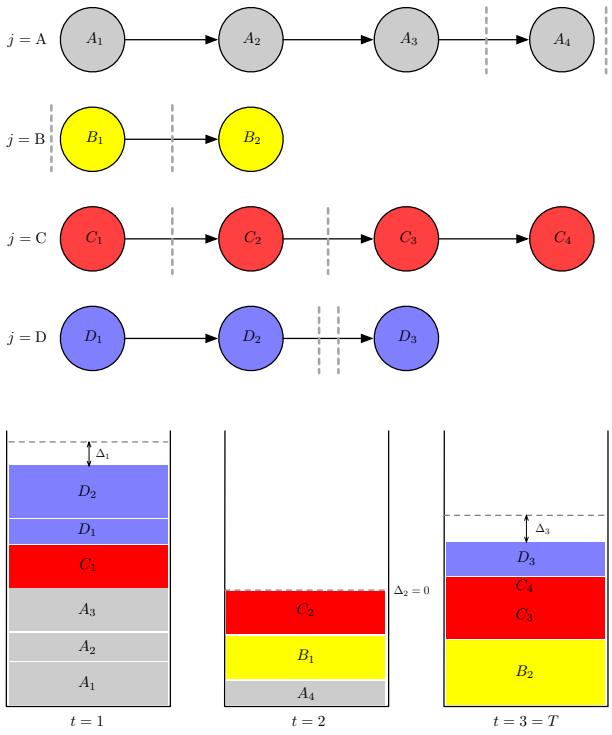

Fig. 1. Tasks allocation example

figure, one can observe that there are amounts of unused resource $\Delta_t > 0$ at slots $t = 1$ and $t = 2$.

3.2 Allocation of funds for academic positions

Departments within an academic institution (such as a university or faculty) submit funding requests to cover the costs of creating new research or professor positions in the near future or to facilitate career advancements for existing staff. Each department's requests are made in a strict sequence. For example, Department A first requests a budget (A1) to create a junior researcher position, followed by a budget (A2) to promote an associate professor to a full professor, and finally, after these prior requests, a budget (A3) for a new associate professor position. Similarly, Department B requests, in order, a budget (B1) to promote a senior researcher to associate professor, followed by a budget (B2) for a new full professor position.

At the same time, the faculty dean must decide how to fairly accommodate these requests from the departments, knowing that the funds (for example, from the government or other public and private institutions that support the faculty) will be received in limited installments, with known amounts but uncertain arrival times. For instance, the first installment consists of a budget of b_1, and the subsequent one provides a budget of b_2. While these installments are collectively sufficient to cover all departmental requests, it is necessary to determine which requests to fulfill with the first installment and which to defer to the second. Each request must be fully covered within a single installment, and the allocation must respect the strict sequence of departmental requests, the budget constraints, and the departments' preference to fulfill as many requests as possible with the first installment, given the uncertainty surrounding the availability of the second installment.

Differently from the case described in Section 3.1, here, possible unused quotes of an installment can be made available afterwards, and hence added to a successive part of the overall funding process.

4. PROBLEM STATEMENT

We are given a set J of n agents, where each agent $j \in J$ is associated with a set O^j consisting of q_j tasks, denoted as $j_1;\ j_2; \ldots; j_{q_j}$. Each task j_i has a resource requirement $p_i^j \in \mathbb{Z}_{\geq 0}$, for $i \in O^j$.

Table 2. Notation

Symbol	Meaning
J	set of n agents
O^j	set of q_j tasks of agent j
j_i	i-th task of agent j
p_i^j	resource requirement of task j_i
$\mathcal{T}$	set of T time slots
$\vartheta(j_i)$	time slot assigned to task j_i by supervisor $\mathcal{S}$
ϑ	overall allocation solution
$f^j(\vartheta)$	cost function for agent j for solution ϑ

The tasks for each agent $j \in J$ must be executed in strict sequential order. Specifically, if $1 \leq h < i \leq q_j$, then task j_i must be assigned to the same or a later time slot than task j_h.

We consider two scenarios in which a limited amount of a resource becomes available at discrete time slots $t \in \mathcal{T} = \{1, 2, \ldots, T\}$:

Non-resumable resource. For each time slot $t \in \mathcal{T}$, the available resource is fixed and equal to b_t.

Resumable resource. In addition to the resource b_t, any unused resource from the previous time slot $t-1$ is carried over and made available at time slot t, for all $t \in \mathcal{T} \setminus \{1\}$.

A supervising entity (referred to as the *supervisor*, $\mathcal{S}$) is responsible for determining the assignment of tasks to time slots. For each time slot $t \in \mathcal{T}$, the total resource demand of the tasks assigned to that slot must not exceed the available resource b_t.

Let $\vartheta(j_i) \in \mathcal{T}$ denote the time slot assigned to task j_i by $\mathcal{S}$, and let $\vartheta = \{\vartheta(j_i) : j \in J,\ i \in O^j\}$ represent the overall allocation decision. Each agent $j \in J$ aims to minimize their own cost function, which depends on the time slots assigned to their tasks and the tasks' weights. The contribution of task j_i to agent j's cost is simply given by the "completion time" $\vartheta(j_i)$, i.e., the time slot j_i is assigned to, which reflects a preference for earlier time slots. We consider as the cost function of agent j the average contribution of its tasks: $f^j(\vartheta) = \frac{1}{q_j} \sum_{i \in O^j} \vartheta(j_i)$.

At the same time, the supervisor $\mathcal{S}$ must allocate tasks while pursuing a twofold objective. On one hand, $\mathcal{S}$ seeks to ensure a fair distribution of costs among the agents. This fairness goal can be addressed by controlling the cost of the worst-off agent. A natural approach is to ensure that no agent's cost exceeds a specified threshold or, equivalently, to minimize the maximum cost among all agents:

$$\max_{j \in J} f^j(\vartheta). \tag{1}$$

On the other hand, $\mathcal{S}$ aims to avoid an excessive increase in the overall system cost, given by

$$\sum_{j \in J} f^j(\vartheta). \tag{2}$$

In conclusion, $\mathcal{S}$ seeks to determine the optimal resource allocation for each time slot, ensuring that agents' requests are satisfied in the specified order while minimizing the fairness measure defined in (1). We denote this "fair" solution as ϑ_F, while ϑ^* is the *system optimum*, i.e., the feasible solution minimizing the system cost (2).

This work focuses on determining and analyzing the resulting fair solutions in terms of computational cost (i.e., the CPU time required to achieve a solution with maximum fairness) and global system cost. We evaluate the price of fairness for our problem based on the results of our experiments, with an approach similar to that of Cosmi et al. (2022), where the average values of a ratio called the "instance price of fairness" ($iPoF$) are reported for various instance classes. However, unlike that study, we directly use the cost functions typically minimized in scheduling literature, avoiding the definition of utility functions (to be maximized) introduced by Bertsimas et al. (2011).

Roughly speaking, the $iPoF$ represents the relative cost incurred when transitioning from a globally optimal (minimum cost) solution to a fair solution. For an instance I, where ϑ_F is the fair solution and ϑ^* represents the system optimum, we compute the $iPoF(I)$ ratio associated with that specific instance. We slightly modify the definition of $iPoF$ provided by Cosmi et al. (2022) as follows:

$$iPoF(I) = \frac{\sum_{j \in J} f^j(\vartheta_F) - \sum_{j \in J} f^j(\vartheta^*)}{\sum_{j \in J} f^j(\vartheta_F)}. \tag{3}$$

To determine an assignment of agents' tasks to time slots, one can define a set of "cuts" on each chain representing an agent's sequence of tasks. A cut is a precedence arc that separates tasks allocated to two consecutive time slots. Since there are T time slots, $T-1$ cuts are sufficient to define any possible allocation. For example, in Figure 1, the cuts corresponding to the allocation of tasks for the four agents are depicted as dotted lines on their respective chains. Note that multiple cuts can correspond to the same precedence arc, effectively separating the same subsets of tasks (e.g., the arc between $D,2$ and $D,3$ in the figure).

Disregarding the time slot capacities b_t for all $t \in \mathcal{T}$, it is straightforward to observe that for an agent with q tasks, the number of possible allocations is equivalent to the number of multisubsets of size $T-1$ selected from a set of size q (i.e., combinations with repetition) that is $\left(\!\binom{q}{T-1}\!\right) = \binom{q+T-2}{T-1}$. Thus, the total number of solutions is given by:

$$O\left(\binom{\max_j\{q_j\} + T - 2}{T-1}^n\right),$$

for a set J of n agents. This number grows exponentially with increases in the number of agents, tasks, and time slots.

5. MIXED INTEGER PROGRAMMING MODEL

Hereafter, we present mixed integer programming models for our problem in its two cases (resumable and non-resumable). In this study, as we mentioned above, each agent j wants to minimize a cost function f^j expressing the average completion time of their tasks. Consequently, to ensure fairness, we consider the overall objective of minimizing a maximum function across all agents, representing the cost for the worse-off agent. This max function can be linearized using standard techniques to facilitate optimization.

We consider two sets of decision variables, namely, binary variables x^j_{it} indicating whether the i-th task of agent j is allocated to time slot t, and—for the resumable resource case—continuous variables u_t indicating the quantity of resource available at time slot t. The resulting MILP model for determining ϑ_F—for the resumable case—can be formulated as follows:

$$\min \max_{j \in J} \frac{1}{q_j} \sum_{i \in O^j} \sum_{t \in \mathcal{T}} t x^j_{it} \tag{4}$$

$$\text{s.t.} \sum_{t \in \mathcal{T}} x^j_{it} = 1 \qquad j \in J, i \in O^j \tag{5}$$

$$\sum_{i \in O^j} \sum_{j \in J} p^j_i x^j_{it} \le u_t \qquad t \in \mathcal{T} \tag{6}$$

$$u_t = b_t + u_{t-1} - \sum_{i \in O^j} \sum_{j \in J} p^j_i x^j_{it-1} \qquad t \in \mathcal{T} \setminus \{1\} \tag{7}$$

$$x^j_{it} \le \sum_{\tau \le t} x^j_{i-1,\tau} \qquad i \in O^j, t \in \mathcal{T} \tag{8}$$

$$x^j_{it} \in \{0,1\} \qquad i \in O^j, t \in \mathcal{T} \tag{9}$$

$$u_t \ge 0 \qquad t \in \mathcal{T}. \tag{10}$$

Constraints (5) are standard assignment constraints that assign every task to exactly one time slot. Constraints (6) ensure that the allocated demand does not exceed the available quantities u_t, which in turn are updated at every time slot t with the amount of resource available b_t and its utilization in the previous time slot $t-1$ (Constraints (7)). Finally, Constraints (8) ensure that if a task i of an agent j is assigned to a time slot t, then the preceding tasks of that agent must be assigned to slot t or a slot preceding t.

The above model can be easily adjusted for the non-resumable resource case scenario, where the remaining available resource cannot be carried over to the succeeding time slot. To this purpose we proceed by (i) eliminating the variable u_t, (ii) removing Constraints (7), and (iii) substituting Constraints (6) with the following

$$\sum_{i \in O^j} \sum_{j \in J} p^j_i x^j_{it} \le b_t \qquad t \in \mathcal{T} \tag{11}$$

6. COMPUTATIONAL EXPERIMENTS

Hereafter we describe the results of a set of computational experiments aimed at evaluating the quality of the proposed model in terms of efficiency and effectiveness when some characteristics of the input are varied.

The experiments have been designed as follows. We built and analyzed eight different classes of ten instances, each with similar characteristics regarding the number of agents n, average number of tasks per agent q, and the number

of time slots T. All data are pseudo-randomly generated taking the parameters specifics of the class in consideration. In all the generated instances we guarantee that the total available resource exceeds (by a small amount) the whole resource requested by the agents, i.e., $\sum_{t\in\mathcal{T}} b_t \gtrapprox \sum_{j\in J, t\in\mathcal{T}} p_t^j$. In each class, the parameters n, q, and T are integer values extracted—with uniform probability—from one of two different ranges, basically corresponding to "small" or "large" values. In our experiments we set the number of agents $n \in F = [2, 3]$ or $n \in M = [10, 15]$, the average number of an agent's tasks $q \in S = [1, 5]$ or $q \in L = [10, 30]$, and the number of time slots $T \in B = [2, 3]$ or $T \in E = [15, 20]$. This eventually renders 8 different classes, named after the corresponding intervals of the triple (n, q, T). (For instance, FLB corresponds to a class of instances with 2 or 3 agents each having between 10 and 30 tasks to be allocated to 2 or 3 time slots.) Additionally, to further stress the system, we built a class of larger-sized instances MLE* with $n = 50$, $q \in L = [10, 30]$, and $T = 50$.

The characteristics of the eight significant [1] classes of tests are reported in the following Table 3. The tests were

Table 3. Range of parameters per instance-class

Class	n	q_j's	T
FSE	2–3	1–5	15–20
FLB	2–3	10–30	2–3
FLE	2–3	10–30	15–20
MSB	10–15	1–5	2–3
MSE	10–15	1–5	15–20
MLB	10–15	10–30	2–3
MLE	10–15	10–30	15–20
MLE*	50	10–30	50

run on a computer equipped with a 3.70 GHz Intel®i7 processor and 32 GB of RAM. The mathematical programs for determining ϑ_F (and also ϑ^*) were solved using the CPLEX solver (version 12.9) with 8 threads.

The results of the preliminary computational tests are summarized in Tables 4–7. For each class of 10 instances, we report the following metrics: the average, minimum, and maximum CPU times; the number of instances solved to optimality; and the average optimality gap for instances that were not solved to optimality within the 300-seconds time limit.

The tests reveal that the number of time slots has a significant impact on CPU time. All instances with a small number of slots ($T \in B$) were solved within a few seconds in both the resumable and non-resumable resource settings. Consequently, we present the results for extended time horizons ($T \in E$) in separate tables, where metrics such as the optimality gap and the number of instances solved to optimality become meaningful. The data summarized in the tables highlight the following key insights:

[1] Since the b_t average values equal $\sum_{j\in J, t\in\mathcal{T}} p_t^j/T$, if we have more time slots than tasks, the average available resouce b_t is smaller than the average task requiremts p_i^j, which makes such a situation not significant in terms of our analyses. For this reason, in our experiments, we did not consider the scenario FSE ($n \in F = [2, 3]$, $q \in S = [1, 5]$, and $T \in E = [15, 20]$).

Table 4. Resumable res.: small T

Class	CPU time			iPoF
	average	min	max	
FSB	0.008	0.00	0.02	0.033
FLB	0.042	0.01	0.11	0.034
MSB	0.027	0.01	0.06	0.075
MLB	1.28	0.02	9.17	0.035

Table 5. Resumable res.: large T

Class	CPU time			# opt	% Gap	iPoF
	average	min	max			
FLE	187.65	5.33	300.00	4/10	1.08%	0.022
MSE	251.55	49.08	300.00	3/10	2.89%	0.15
MLE	300.00	300.00	300.00	0/10	10.40%	0.11
MLE*	300.00	300.00	300.00	0/10	60.62%	0.53

Table 6. Non-resumable res.: small T

Class	CPU time			iPoF
	average	min	max	
FSB	0.032	0.01	0.05	0.039
FLB	0.088	0.01	0.23	0.032
MSB	0.074	0.02	0.13	0.064
MLB	9.46	0.03	77.42	0.037

Table 7. Non-resumable res.: large T

Class	CPU time			# opt	% Gap	iPoF
	average	min	max			
FLE	49.55	0.02	300.00	9/10	2.65%	0.030
MSE	154.42	0.36	300.00	6/10	3.82%	0.16
MLE	300.00	300.00	300.00	0/10	12.68%	0.14
MLE*	300.00	300.00	300.00	0/10	60.84%	0.22

The size of time slots significantly affects computational performance. Shorter time horizons T (i.e., small numbers of time slots) lead to negligible CPU times and very low average iPoF values across both resource settings. Conversely, larger time horizons (T) substantially increase computational difficulty, resulting in longer CPU times and fewer optimal solutions within the time limit, even for simpler classes like FLE and MSE. Except for MLE*, optimality gaps remain below 13%, and iPoF values stay low regardless of the resource setting.

Class MLE* poses significant challenges, with optimality gaps exceeding 60% and the iPoF reaching its worst value of 0.53 in the resumable setting. This exceptionally high value is partly attributed to instances where the global optimum was unavailable. In such cases, we relied on lower bounds for the global optimum, resulting in only an upper bound for the iPoF.

Surprisingly enough, the resumable resource setting—allowing a higher degree of flexibility in the allocation of time-resource to tasks—does not make the problem (much) easier. Indeed, for small-sized instance (with smaller T values), the resumable scenario implies shorter computing times and gaps. However, one observes a slight increase in computation times and iPoF values, when the time horizon T becomes larger. More precisely, fair solutions are relatively close to the system optimum in small-sized instance classes (e.g., FSB, FLB, FLE) with low iPoF

values. This suggests that fairness can be achieved without significantly compromising overall efficiency, making the model a promising approach for real-world applications. (Of course, this evidence applies to the objective functions investigated in this preliminary study and it deserves additional study to extend it to a broader class of performance indicators and fairness measures.) However, as problem complexity grows (e.g., MSB, MLE), fairness becomes more costly, as reflected by higher iPoF values. This underscores the need for advanced techniques to balance fairness and computational efficiency in more complex scenarios.

7. CONCLUSION

Future research could focus on developing faster heuristic algorithms to improve solution times, as larger instances could not be solved to optimality within reasonable time limits. Our computational experiments indicate that while the MILP model effectively balances fairness and efficiency in small instances, its scalability is limited, making heuristic approaches a promising avenue for addressing larger cases. Additionally, one might explore the design of algorithms that account for the strategic behavior of individual agents in sequencing their tasks, potentially at the expense of other agents or the overall system efficiency. Our findings highlight that fair solutions tend to remain close to the system optimum in small instances, but fairness becomes increasingly costly as complexity grows. A bi-level programming approach as in Jiang et al. (2021) and Pferschy et al. (2021) could be employed to model these interactions, offering a structured way to capture the trade-off between fairness and efficiency in larger-scale problems.

REFERENCES

Agnetis, A., Billaut, J.C., Gawiejnowicz, S. Pacciarelli, D., and Soukhal, A. (2014). *Multiagent Scheduling Models and Algorithms*. Springer.

Agnetis, A., Chen, B., Nicosia, G., and Pacifici, A. (2019). Price of fairness in two-agent single-machine scheduling problems. *European Journal of Operational Research*, 276(1), 79–87.

Agnetis, A., Benini, M., Nicosia, G., and Pacifici, A. (2025). Trade-off between utility and fairness in two-agent single-machine scheduling. *European Journal of Operational Research*. doi:10.1016/j.ejor.2025.01.025.

Agnetis, A., Nicosia, G., Pacifici, A., and Pferschy, U. (2015). Scheduling two agent task chains with a central selection mechanism. *Journal of Scheduling*, 18(3), 243–261.

Alcaraz, J., Anton-Sanchez, L., and Saldanha-da-Gama, F. (2022). Bi-objective resource-constrained project scheduling problem with time-dependent resource costs. *Journal of Manufacturing Systems*, 63, 506–523.

Bertsimas, D., Farias, V., and Trichakis, N. (2011). The price of fairness. *Operations Research*, 59(1), 17–31.

Caragiannis, I., Kaklamanis, C., Kanellopoulos, P., and Kyropoulou, M. (2009). The efficiency of fair division. In *5th International Workshop on Internet and Network Economics, WINE 2009*, volume 5929 LNCS, 475–482. Springer.

Cosmi, M., Nicosia, G., and Pacifici, A. (2022). Computing fair solutions in single machine scheduling. *IFAC-PapersOnLine*, 55(10), 2185–2190. 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022.

Faruque, S. (2018). *Time Division Multiple Access (TDMA)*. Springer International Publishing.

Gao, Y., Yuan, J., Ng, C., and Cheng, T. (2019). A further study on two-agent parallel-batch scheduling with release dates and deteriorating jobs to minimize the makespan. *European Journal of Operational Research*, 273(1), 74–81.

Haslak, T. (2020). Weighted fair queuing as a scheduling algorithm for deferrable loads in smart grids. In V. Bertsch, A. Ardone, M. Suriyah, W. Fichtner, T. Leibfried, and V. Heuveline (eds.), *Advances in Energy System Optimization. ISESO 2018*, Trends in Mathematics. Birkhäuser, Cham.

He, C., Wu, J., and Lin, H. (2022). Two-agent bounded parallel-batching scheduling for minimizing maximum cost and makespan. *Discrete Optimization*, 45, 100698.

Heeger, K., Hermelin, D., Mertzios, G.B., Molter, H., Niedermeier, R., and Shabtay, D. (2022). Equitable scheduling on a single machine. *Journal of Scheduling*, 26(2), 209–225.

Hermelin, D., Molter, H., Niedermeier, R., Pinedo, M., and Shabtay, D. (2025). Fairness in repetitive scheduling. *European Journal of Operational Research*. doi: https://doi.org/10.1016/j.ejor.2024.12.052.

Jiang, Y., Wu, X., Chen, B., and Hu, Q. (2021). Rawlsian fairness in push and pull supply chains. *European Journal of Operational Research*, 291(1), 194–205.

Li, S. and Yuan, J. (2012). Unbounded parallel-batching scheduling with two competitive agents. *Journal of Scheduling*, 15(5), 629 – 640.

Liu, L., Ng, C., and Cheng, T. (2009). Bicriterion scheduling with equal processing times on a batch processing machine. *Computers and Operations Research*, 36(1), 110 – 118.

Marini, C., Nicosia, G., Pacifici, A., and Pferschy, U. (2013). Strategies in competing subset selection. *Annals of Operations Research*, 207(1), 181 – 200.

Naldi, M., Nicosia, G., Pacifici, A., and Pferschy, U. (2019). Profit-fairness trade-off in project selection. *Socio-Economic Planning Sciences*, 67, 133–146.

Pferschy, U., Nicosia, G., Pacifici, A., and Schauer, J. (2021). On the Stackelberg knapsack game. *European Journal of Operational Research*, 291(1), 18–31.

Rodríguez-Ballesteros, S., Alcaraz, J., and Anton-Sanchez, L. (2024). Metaheuristics for the bi-objective resource-constrained project scheduling problem with time-dependent resource costs: An experimental comparison. *Computers & Operations Research*, 163, 106489.

Sousa, I., Queluz, M.P., and Rodrigues, A. (2020). A survey on QoE-oriented wireless resources scheduling. *Journal of Network and Computer Applications*, 158, 102594.

Zhang, Y., Zhang, Z., and Liu, Z. (2020). The price of fairness for a two-agent scheduling game minimizing total completion time. *Journal of Combinatorial Optimization*, 44, 2104–2122.