

Title	Multi-agent scheduling for shared manufacturing systems
Authors	Duran, Ege;Ozturk, Cemalettin;O'Sullivan, Barry
Publication date	2025
Original Citation	Duran, E., Ozturk, C. and O'Sullivan, B. (2025) 'Multi-agent scheduling for shared manufacturing systems', Procedia Computer Science, 253, pp. 727–736. https://doi.org/10.1016/j.procs.2025.01.134
Type of publication	Article (peer-reviewed)
Link to publisher's version	10.1016/j.procs.2025.01.134
Rights	© 2025, the Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0) Peer-review under responsibility of the scientific committee of the 6th International Conference on Industry 4.0 and Smart Manufacturing - https://creativecommons.org/licenses/by-nc-nd/4.0
Download date	2026-10-09 12:39:12
Item downloaded from	https://hdl.handle.net/10468/17294

6th International Conference on Industry 4.0 and Smart Manufacturing

Multi-agent Scheduling for Shared Manufacturing Systems

Ege Duran^{a,*}, Cemalettin Ozturk^b, Barry O’Sullivan^c

^a*SFI Centre for Research Training in AI, School of Computer Science & IT
University College Cork, Ireland*

^b*Department of Process, Energy & Transport Engineering, Munster Technological University, Cork, Ireland*

^c*Insight SFI Research Centre for Data Analytics School of Computer Science & IT, University College Cork, Ireland*

Abstract

Advances in digitization and resource-sharing business models have created new opportunities for manufacturing companies, enhancing competitiveness and resilience. However, these benefits bring computational challenges in efficiently planning and scheduling shared resources. Therefore, there is a need for scalable and quick solutions for practical applications. Shared manufacturing systems share characteristics with parallel machine scheduling, allowing for the application of advancements from this domain. This research focuses on Multi-Agent Parallel Machine Scheduling (MAPMS) in shared manufacturing, specifically addressing scenarios involving two parallel machines and distinct agents managing exclusive, set of non-overlapping orders. The study introduces a novel multi-objective mixed integer programming (MIP) model for order scheduling across multiple facilities, accounting for sequence-dependent setup times between orders. It also proposes a new heuristic method designed for industrial use. Benchmark instances demonstrate the practicality of both the MIP model and heuristic, contributing valuable insights into MAPMS challenges in shared manufacturing environments.

© 2025 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0>)

Peer-review under responsibility of the scientific committee of the 6th International Conference on Industry 4.0 and Smart Manufacturing

Keywords:

Multiagent systems; Scheduling; planning; parallel machine

1. Introduction

The manufacturing industry continuously faces critical challenges such as supply chain disruptions, political instability, increasing competition, mass customization, and skill shortage. These challenges force the industry to be more efficient, agile, and resilient than ever. Collaboration between industry stakeholders is one of the opportunities to resolve these challenges as it enhances resource utilization, reduces manufacturing costs through economies of scale, and provides redundant capacities against disruptions while supporting the viability of small and medium-

* Corresponding author.

E-mail address: e.duran@cs.ucc.ie

sized enterprises (SMEs). Successful peer-to-peer (P2P) implementations of such collaborations in the last decade in ride-hailing and hospitality services (i.e. Uber, Air B&B), encouraged the manufacturing industry to implement this disruptive business model which resulted in a paradigm shift from traditionally isolated manufacturing capacities to shared manufacturing (SharedMfg) concept. An early example of SharedMfg systems is seen in the biopharma manufacturing industry ([2]) where today many other manufacturing companies, along with other industries implement the same model via common platforms such as Gaia-X ([7]) to improve their value chain collaboratively. Several case studies have been carried out to validate and demonstrate the potential of collaborative networked organizations (CNO) concepts, particularly in the manufacturing industry. Camarinha et al. [4] provided eight case studies of CNOs. VEN, founded in 2004 and financed by Yorkshire Forward and the EU, enables SMEs to collaborate on major agreements across many industries, facilitating access to research institutions [4]. The Groupe Renault Alliance, formed by Nissan, Daimler, and Mitsubishi, shares vehicle platforms, manufacturing sites, and distribution infrastructure [12]. SharedMfg concept has also been used in the UK health industry, particularly for relocation of manufacturing systems closer to clinical demand points which results in better control over quality, volume, and cost [13]. These examples demonstrate the significance and practical applicability of shared manufacturing within real-world scenarios. However, there are practical challenges arise such as acquiring cloud systems, implementing advanced analytics technologies, and ensuring data privacy. There are two main facilitators of SharedMfg systems; (1) providing the ICT infrastructure (namely, Manufacturing as a Service, MaaS) for seamless integration between stakeholders, and (2) efficient planning, scheduling, and coordination of shared resources for the benefit of both manufacturing capacity providers and customers in which each of them has their objectives. Therefore, from a computational perspective, the scheduling of SharedMfg systems can be classified as multi-agent systems. Furthermore, a multi-agent approach provides the flexibility needed to accommodate diverse manufacturing requirements. Each manufacturing stakeholder (i.e., agent) can independently optimize its operations to meet its specific goals, whether it's to minimize production time, reduce costs, or enhance product quality. The rest of the paper is structured as in the following. The next section, Section 2, reviews the recent literature on multi-agent scheduling. Section 3 outlines the problem definition. Section 4 and 5 introduce the MIP model and the heuristic algorithm respectively. Section 6 discusses the results of the numerical experiments, and the paper concludes with Section 7 with a discussion on future research directions.

2. Literature Survey

Multi-agent scheduling (MAS) ([1]) deals with sequencing and timing of value-added activities based on consumer preferences to enhance customer satisfaction ([3]). The advantages of MAS over standard scheduling include agents' interaction and autonomy, where each agent is responsible for a portion of the jobs and has distinct goals based on their business strategy ([16]). Agents interact and consult with each other to find a schedule that benefits both individuals and the group. The computational complexity of solving MAS problems is NP-hard ([5]). Therefore, researchers have developed computationally efficient exact and approximation algorithms for MAS systems with single ([14]) and parallel machines ([9], [16]) with practical extensions such as maintenance, job release dates.

Wu et al. [15] developed a branch and bound algorithm as an exact method (B&B) and tabu search to tackle multi-agent single-machine scheduling with two agents; total weighted number of tardy jobs and tardiness. Authors goal is to minimize resulting numbers for both agents. Kung et al. [10] proposed a B&B and three genetic algorithms for two-agent single-machine scheduling problem with job ready times. Wang et al. [14] investigated MAS on a single machine with release dates, developing a B&B and a hybrid discrete artificial bee colony algorithm for small and medium-sized instances, respectively. Additionally, the authors applied Deep Reinforcement Learning (DRL) to the MAS problem for the first time to solve real-time applications where jobs arrive dynamically. While the above research studies have made significant contributions to the field, it is important to note that industrial manufacturing processes often involve multiple machines. These are categorized in the literature as multi-agent parallel machine scheduling problems (MAPMS). Yu et al. [16] developed a heuristic algorithm for a scenario involving two agents and two parallel machines, specifically suited for discrete and small-scale manufacturing systems. Elvikis and Donatas [6] worked on two-agent scheduling on uniform parallel machines and aimed to minimize agents' makespan by employing a minimal complete Pareto set enumeration algorithm. Gu et al. [9] proposed an approximation algorithm to solve the same problem.

This research contributes to the literature both in terms of problem definition and the solution method. The MAS problem considered in this paper takes into account the planning of customer orders, distributing jobs of orders to suitable machines/facilities and scheduling those jobs with sequence-dependent setup times. Furthermore, a novel mathematical model and an efficient heuristic algorithm, that is adapted from [16] bring significant contributions to the field in terms of the solution of the MAPMS.

3. Problem Definition

In its simplest case, at least two manufacturing facilities are needed to establish a SharedMfg system which can be reduced into a system with two parallel machines with identical features. Hence, in the rest of the paper, SharedMfg scheduling will be referred to as Multi-Agent Parallel Machine Scheduling (MAPMS). Using the scheduling literature notation, ([8]), MAPMS can be formally classified as of P2|Co| $\sum T^A, C_{max}^B$. "P2" denotes the presence of two identical parallel machine environments. "Co" represents a distinctive job characteristic wherein competing agents are involved, and each agent exclusively possesses all the jobs assigned to it, with no common jobs among agents. The notation $\sum T^A, C_{max}^B$ signifies the optimality criteria for agents A and B, respectively ([16]). Each agent manages a set of orders, and each order may encompass two or more jobs with identical processing times. Jobs of an order can be split to both machines. $J^A = \{J_1^A, J_2^A, \dots, J_{n_A}^A\}$ and $J^B = \{J_1^B, J_2^B, \dots, J_{n_B}^B\}$ represent set of orders that are coordinated by Agent A and B respectively in which the total number of orders is $N = n_A + n_B$. While Agent A pursues minimizing the tardiness of orders it is coordinating, Agent B seeks to find a schedule that minimizes the makespan for its orders. There are two parallel machines in which jobs of orders can be processed in both and orders can be split into machines without preemption. That means all jobs of an order assigned to the same machine are processed consecutively. There might be sequence-dependent changeover times between different orders. While two agents are orchestrated, Agent A is given a higher priority than B. Hence the decision problem of MAPMS is finding an assignment of jobs of customer orders into parallel machines and scheduling while two conflicting agents try to minimize total tardiness and makespan of the production. In the following, a demonstrated example with 6 orders coordinated by each Agent A and B is used to illustrate MAPMS problem studied. Data used for the demonstration; due date (min), the number of jobs in each order, processing time of each job (min), and the sequence-dependent setup times between each order (min) are provided in Tables 1 and 2. In Table 2, D0 refers to the initial setup time for each order in each machine. The mathematical model presented in Section 4 finds the optimal schedule for the instance with total tardiness of 6 min. and 40 min. of makespan as illustrated by the Gantt chart in Figure 1. Note that X_A and X_B represent the index of order X coordinated by agent A and B respectively in first column of Table 1 while ST stands for the sequence-dependent setup time between orders.

Table 1: Demonstrated instance data

Order ID	Due date (d_i)	# of jobs (U_i)	processing time (p_i)
1.A	17	3	1
2.A	11	2	3
3.A	15	3	2
4.A	13	1	2
5.A	13	3	1
6.A	16	2	2
1.B	8	2	2
2.B	16	2	3
3.B	11	3	1
4.B	14	4	2
5.B	17	3	1
6.B	11	4	1

Table 2: Data for setup times

	J1_A	J2_A	J3_A	J4_A	J5_A	J6_A	J1_B	J2_B	J3_B	J4_B	J5_B	J6_B
D0	1	4	4	4	2	4	3	5	1	1	5	3
J1_A	-	2	2	1	2	5	4	1	4	1	3	4
J2_A	2	-	4	4	1	2	2	2	1	3	5	4
J3_A	3	5	-	1	1	5	1	5	2	5	2	3
J4_A	1	2	2	-	2	4	4	3	4	1	4	5
J5_A	3	5	4	5	-	5	4	5	5	5	1	5
J6_A	3	1	2	2	4	-	1	2	3	5	3	1
J1_B	3	3	1	2	1	3	-	4	3	4	3	2
J2_B	4	4	4	3	4	5	2	-	4	2	3	2
J3_B	3	4	1	4	3	4	1	1	-	4	3	4
J4_B	5	1	1	3	4	3	4	1	1	-	4	1
J5_B	3	1	5	4	5	1	5	4	4	5	-	4
J6_B	1	1	2	1	5	2	4	2	1	1	5	-

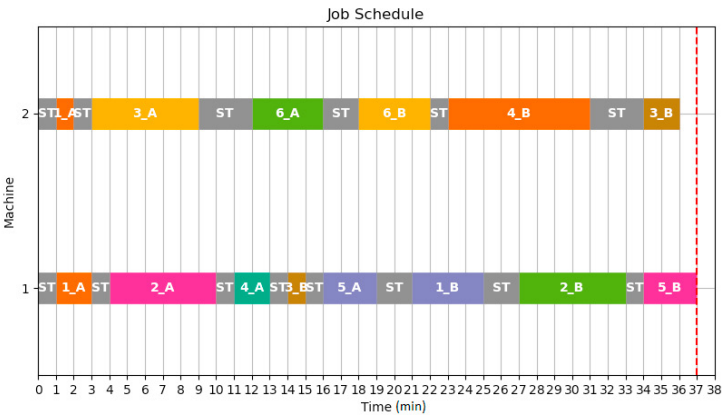


Fig. 1: Optimal solution of MAPMS for a small instance

- The main contributions of this paper are as follows:
1. A novel multi-objective Mixed Integer Programming model (MIP) for MAPMS with sequence-dependent setup times between orders is formulated.
 2. A scalable fast heuristic algorithm for industrial size applications of the problem developed.
 3. A randomly generated data set is used for benchmarking the developed MIP model and heuristic algorithm.

4. Problem Formulation

In this section, a generic MAPMS problem formulation with K parallel machines and sequence-dependent setup times between orders is presented. Sets, indices, input parameters, and decision variables along with the formulation of MAPMS are described in the following:

Set and Indices

N : set of orders, $i, j \in N = \{0, 1, 2, \dots, |N|\}$ in which 0 refers to the index of the dummy order scheduled first in each machine

K : set of machines, $k \in K = \{1, 2, \dots, |K|\}$

Parameters

U_i : number of jobs of order i

p_i : unit processing time of order i

d_i : due date of order i

M : a large number

$s_{i,j}$: machine independent sequence dependent setup time between orders i and j in which $s_{0,j}$ represents the setup time if order j is processed first. Note that $s_{i,j}=0$ if there is no setup time.

Decision Variables

$v_{i,k}$: number of jobs of order i is processed on machine k $V_{i,k} \in \mathbb{Z}^+$

$C_{i,k}$: completion time of order i on machine k $C_{i,k} \in \mathbb{Z}^+$

C_{max} : Maximum completion time, makespan, for Agent B

T_i : Tardiness of job i , $\forall i \in J_A$ $T_i \in \mathbb{R}^+$

$$x_{i,k} = \begin{cases} 1 & \text{if order } i \text{ processed on machine } k \\ 0 & \text{otherwise} \end{cases} \quad y_{i,j,k} = \begin{cases} 1 & \text{if order } i \text{ is processed immediately} \\ & \text{before order } j \text{ on machine } k \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Minimize } w * \sum_{i=1}^{i \in J_A} T_i + (1 - w) * C_{max} \quad (1)$$

$$\sum_{k \in K} v_{i,k} = U_i, \forall i \in \{1..N\} \quad (2)$$

$$x_{i,k} \leq v_{i,k}, \forall i \in \{1..N\}, \forall k \in K \quad (3)$$

$$v_{i,k} \leq x_{i,k} * U_i, \forall i \in \{1..N\}, \forall k \in K \quad (4)$$

$$\sum_{i=1}^N y_{0,i,k} \leq 1, \forall k \in K \quad (5)$$

$$N * \sum_{i=1}^N y_{0,i,k} \geq \sum_{i=1}^N x_{i,k}, \forall k \in K \quad (6)$$

$$\sum_{i=0, i \neq j}^N y_{i,j,k} = x_{j,k}, \forall j \in \{1..N\}, \forall k \in K \quad (7)$$

$$y_{i,j,k} \leq x_{i,k}, \forall i \in \{0..N\}, j \in \{1..N\}, i \neq j, \forall k \in K \quad (8)$$

$$\sum_{j=0, j \neq i}^N y_{i,j,k} = 1, \forall i \in \{1..N\}, \forall k \in K \quad (9)$$

$$C_{i,k} + M * (1 - y_{0,i,k}) \geq p_i * v_{i,k} + s_{0,i}, \forall i \in \{1..N\}, \forall k \in K \quad (10)$$

$$C_{j,k} + M(1 - y_{i,j,k}) \geq C_{i,k} + p_j * v_{j,k} + s_{i,j} \\ \forall i, j \in \{1..N\}, i \neq j, \forall k \in K \quad (11)$$

$$T_i \geq C_{i,k} - d_i, \forall i \in J_A, \forall k \in K \quad (12)$$

$$C_{max} \geq C_{i,k}, \forall i \in J_B, \forall k \in K \quad (13)$$

The objective function (1) is formulated to minimize tardiness for Agent A and makespan for Agent B by employing the weighted sum method ($w < 1$). Constraints (2) ensure that the total number of jobs of an order i processed in both machines equals the total number of jobs of that order. Constraints (3) and (4) give the lower and upper bounds for the number of jobs assigned to each machine for each order. Constraints (5) stipulate the scheduling of the dummy order (order index 0) as the first one in each machine. Constraints (6) assert that if a non-dummy order is assigned to a machine, it has to be immediately preceded by another order. Constraints (7) ensure that if a non-dummy order is either preceded or succeeded by an order in machine k , this order must be assigned to that machine. Constraints (8) and (9) ensure that an order can be preceded or succeeded by at most one another. Constraints (10) mandate that the completion time of the first non-dummy order in the sequence on any machine must be greater than or equal to the product of its processing time and the number of assigned jobs of that order to that machine and the required initial setup time. Constraints (11) guarantee that the completion time of a non-dummy order j on a machine must be greater than or equal to the sum of the completion time of the immediately preceding order, the total processing time of its jobs assigned to that machine and the sequence-dependent changeover time from the immediate predecessor order. Constraints (12) quantify tardiness for orders affiliated with Agent A, while constraints (13) compute the makespan for orders associated with Agent B.

5. Heuristic Algorithm

Although the MIP model in Section 4 guarantees the optimality for MAPMS, it is intractable for large instances due to NP-hard computational complexity ([5]). Hence, in this section, first, a greedy heuristic is presented for 2 parallel machines case ($K = 2$) without sequence-dependent setup times, as an improved version of ([16]), will be presented and then the algorithm will be extended to solve the cases with sequence-dependent setup times.

Overall heuristic method consists of a set of sub-heuristics which are focusing on individual agents and interact with each other as shown in Fig. 2. Algorithm 1 coordinates decisions on how many jobs of each order will be assigned to which machine and their schedule such that first, minimizing the total tardiness of orders of Agent A and then minimizing the makespan of orders of Agent B. The Algorithm 2 and Algorithm 3 are executed to distribute jobs of orders coordinated by Agent A and Agent B to the machines, respectively. Then, Algorithm 1 continues to make final adjustments aiming to minimize the agents' objectives, and returns the final scheduling and order assignment.

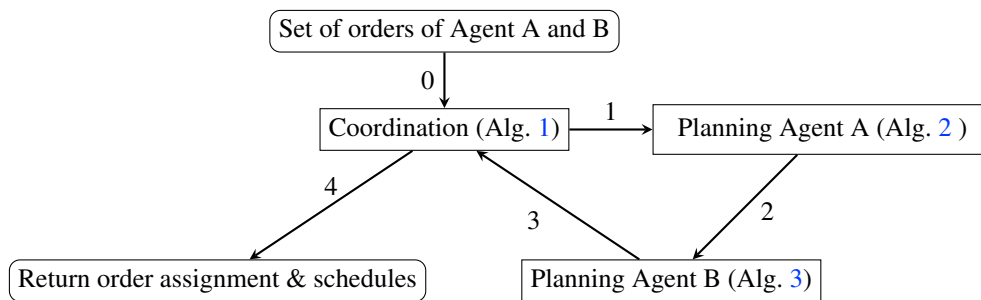


Fig. 2: Planning and scheduling heuristic flowchart

In coordination algorithm (Alg. 1), the initial assignment is generated by assigning all jobs of an order to the same machine, to minimize individual order completion times (steps 1-2). Then, individual planning and scheduling algorithms for each agent, Algorithm 2 and 3, are called for distributing jobs of each order and scheduling in Steps 3-4. Algorithm 2 excludes splitting the latest completed order of Agent A to machines due to the risk of deteriorating the makespan of the orders of Agent B, and hence planning and scheduling the latest completed orders in each machine is investigated in step 5. Step 6 returns the final order assignment and scheduling.

Once Algorithm 1 finds an initial schedule, Algorithm 2 tries to split and schedule orders associated with Agent A to parallel machines to minimize their tardiness. Algorithm 2 starts with identifying the tardy orders in the initial schedule at Step 1. If the last orders of machines belong to Agent A, they are removed from the consideration. Then, the list of tardy orders J^A is sorted according to the earliest tardiness time (step 2). For each order in J^A , k' is the

Algorithm 1 Coordination

- 1: Sort J^A with Earliest Due Date (EDD) and J^B with Longest Processing Time (LPT) dispatching rules, create a set of orders J , $J = J^A \cup J^B$
- 2: Compute minimum completion time for each order by considering two machines and assign the first order in J to machine k until all orders are assigned to a machine with minimum completion time.
- 3: Go to Algorithm 2
- 4: Go to Algorithm 3
- 5: Compare completion time of machines k and k' . If $k > k'$ then let j_i is the last order in machine k if there is an improvement opportunity by splitting jobs of order j_i to k' , split order j_i until the latest completion time of machine k' is more than or equal to one in k , vice versa. Otherwise go to **Step 6**
- 6: Return schedule

machine that order j_i is assigned (step 4) and k is the other machine (step 4). Let j_m be the job assigned to machine k , associated with Agent A, with a completion time less than or close to that of j_i . The split jobs of j_i will be scheduled just after j_m (step 5). The algorithm checks whether order j_i can be split from machine k' to machine k (Step 6). If jobs of order j_i can be split then, the number of jobs that can be split and the number of jobs needed to be split are calculated, q and t respectively (Step 7). If q is more than or equal to t then the algorithm tries to split jobs of order in machine k' to machine k until the due date of order i is equal or smaller than its completion time in machine k , otherwise the algorithm replicates the same steps until the completion time of order i in machine k is larger than machine k' (Step 8). If there is no improvement opportunity then the algorithm goes to step 3.

Algorithm 2 Planning and scheduling of tardiness agent (Agent A)

- 1: $R \leftarrow$ the last order $\forall k \in K, J^A = \{\forall i \in J^A \text{ if } T_i > 0\} \setminus R$
- 2: Sort J^A with earliest tardiness time
- 3: **for** j_i in J^A **do**
- 4: Let k' be the machine to which j_i is assigned and let k be the other machine.
- 5: Let m be an order which is assigned to k and belongs to Agent A with a completion time less than but close to j_i
- 6: Check if there is an improvement opportunity by splitting j_i and assigning to k next to order j_m
- 7: If there is an improvement then calculate the number of jobs that can be split, q , and the number of jobs needed to be split to meet the deadline, t , otherwise go to **Step 3**
- 8: If $q \geq t$, split jobs of order j_i until $C_{ik'}$ is less than its deadline, otherwise split jobs of order j_i until C_{ik} is less than C_{ik}
- 9: **end for**

The Algorithm 3 is executed to plan and schedule orders of Agent B. In step 1, order j_i is selected with maximum completion time among orders of Agent B. C_{max} is denoted the finishing time of the selected order j_i . Order j_m is selected from the other machine (k) which belongs to Agent B with a completion time less than but close to that of j_i (step 2). If there is an opportunity to minimize the makespan of Agent B by splitting jobs of j_i to machine k , makespan of Agent B is updated and referred as C_{update} (step 3). Jobs of j_i continue to be split until C_{update} is larger than C_{max} (Step 3).

Algorithm 3 Planning and scheduling of makespan agent (Agent B)

- 1: Select order j_i which belongs to Agent B with the maximum completion time (C_{max}) among orders of Agent B and k' refers to a machine that j_i is assigned and k refers to the other machine.
- 2: Let m be an order which is assigned to k and belongs to Agent B with a completion time less than but close to j_i
- 3: If there is an improvement opportunity by splitting jobs of j_i to machine k split it and update makespan for Agent B, C_{update} . Split jobs of j_i until C_{update} larger than C_{max} , otherwise go to the step 4 in Alg. 1

Complexity of the coordination algorithm (Algorithm 1) is dominated by the Algorithm 2 called in Step 3 which has $O(J_A)$ worst case complexity due to looping over all jobs of Agent A. Compared to the algorithm in the paper of [16], sorting orders of each agent separately and giving priority to Agent A orders in splitting, in Algorithm 1 provide better results as we numerically show in Section 5. Note that in contrast to [16] we also add a control mechanism in Algorithm 2 that allows changing the schedule only if there is an improvement in tardiness for Agent A.

5.1. Incorporating sequence-dependent setup times

Initial assignment and scheduling algorithm, Algorithm 1 is revised as in Algorithm 4 to accommodate sequence-dependent setup times. The revision incorporates an additional loop to find which order will be processed first in which machine. Steps 1 to 3 retrieve and assign the order with the minimum initial setup time as the first order to be processed for each machine, subsequently removing them from consideration. Algorithm 4 then continues with the same steps as in Algorithm 1. The completion times of jobs and orders in Algorithms 1, 2, and 3 are reformulated to include sequence-dependent setup times accordingly.

Algorithm 4 Coordination (Revised)

```

1: for  $k$  in  $K$  do
2:   Let  $j_i$  be an order with minimum initial setup time,  $\min(s_{0i} \mid \forall i \in N)$ , and assign to machine  $k$ 
3: end for
4: Repeat steps 1-6 in Alg. 1

```

6. Computational experiments

In this section, we provide numerical experiments to demonstrate the efficacy of the developed MIP model and heuristic for MAPMS for both cases with and w/o sequence-dependent setup times. 30 instances that are randomly generated based on the problem structure in [16] are used for benchmarking developed methods and can be accessed from GitHub¹. Note that while [16] provide one instance only, one of the contributions of this study is generating various sizes of instances for the MAPMS problem. Instances are classified as small, medium, and large based on the number of orders (N) which are 6, 12, and 18 respectively.

The number of orders coordinated by Agent A and B are assumed to be equal for each instance, $n_A = n_B$. To ensure the statistical significance of experiments, 10 random instances were generated for each of these three classes. The number of jobs for each order i , (U_i) is randomly generated with $U \sim [1, 4]$ while the processing time for each job (p_i) (min) is randomly generated with $U \sim [1, 3]$ which the lower and upper bounds for both are taken from [16]. The due date for each order (d_i) (min) is randomly generated with $U \sim [(d_{\min} * n) / k, (d_{\max} * n) / k]$ in which $d_{\min}$ and $d_{\max}$ refer to the minimum and maximum due date in the demonstrated example of [16], n is the total number of jobs in a given experiment and k is referring to total number of jobs in the problem given in the same paper. Sequence-dependent setup times, $s_{i,j}$ (min) are generated with $U \sim [1, 5]$. Note that w in the objective function of the MIP model is set to 0.8 for all instances.

The heuristic algorithm was implemented in Python 3.9 programming environment and executed on Dell Precision 3470 12th gen processor, Intel Core i7, 16GB RAM. CPLEX 22.1.0 is used for solving MIP instances on the same machine. Figure 3 shows the results obtained from the MIP model, base algorithm([16]), and heuristic, Algorithm 1 without setup time on 30 experiments. The MIP was executed within a time limit of 1800 seconds for medium and large-scale problems, while the algorithm achieved solutions within milliseconds. In small (smallest circles) and medium-scale (medium circles) instances, both MIP and Algorithm 1 produce solutions that are closely aligned. However, when large instances are considered (biggest circles), MIP outperforms heuristic in the most cases. When the Algorithm 1 (orange circles) and the base algorithm in ([16]) (green circles) are compared to each other, it is shown that the former, developed in this paper, provides superior results in all instances.

¹ <https://github.com/egedurann/Multi-agent-Scheduling-for-Shared-Manufacturing-Systems.git>

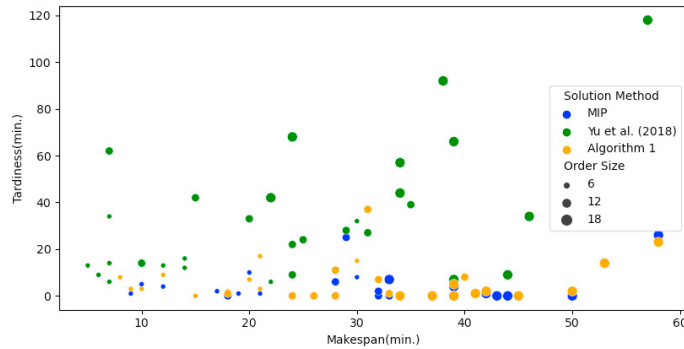


Fig. 3: Comparison for the methods without setups

Figure 4 presents the results of both the MIP model and the Algorithm 4. The MIP model performs better in general in smaller instances, providing optimal solutions in under a second as expected. However, the solution time hits the time limit of 1800 seconds for the MIP model in medium and large-size instances whereas the runtime of the heuristic algorithm is under one second in all instances thanks to its polynomial complexity. Close convergence of heuristic results to the MIP solutions is also observed in medium and large instances as well. In overall, 14 of 30 instances, heuristic results are better than MIP in terms of total tardiness of orders in J^A . Please note that, detailed numerical results are omitted due to limited space and can be accessed from the author's GitHub ⁶.

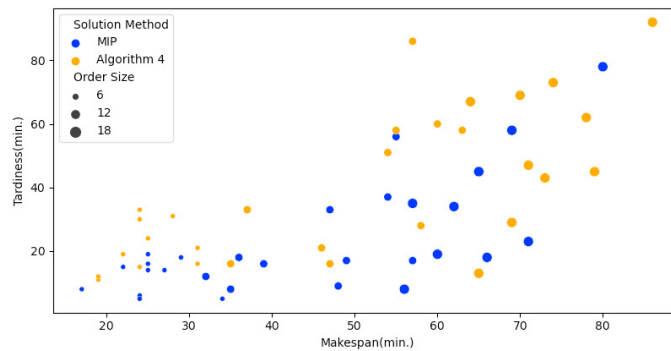


Fig. 4: Comparison MIP and Alg. 4

7. Conclusion

Shared manufacturing is a disruptive model that fosters collaboration between stakeholders to optimize resource use, enhance competitiveness, and strengthen supply chain resilience. Industries like machinery manufacturers, additive manufacturing facilities, and tier-1 and 2 automotive suppliers can quickly benefit from this model. This paper presents a multi-agent scheduling approach for coordinating shared manufacturing systems. A novel mathematical model and fast greedy heuristics are developed for the $P2|Co|\sum_T^A, C_{max}^B$ problem with sequence-dependent setup times. Testing on 30 randomly generated instances shows that while the MIP model performs well for small and medium instances, it struggles with larger ones within a 1800-second time limit. In contrast, the heuristic solves all instances in milliseconds with acceptable convergence to exact results. However, relying on static and deterministic data as well as lack of an improvement step in the heuristic algorithm are the biggest limitations of our approach. Therefore, this study can be extended in several ways. The proposed heuristic can also be enhanced with local search for improved

performance. To enhance the breadth and applicability of this research, a possible extension could be generalizing the methods developed into n machines/facility case. Rather than restricting the analysis to identical machines, a more realistic approach could entail extending methods to several practical scenarios such as non-identical machines and order release times, incorporating machine unavailability, preemption of orders, and transportation between machines/facilities to a common distribution center. Adding new agents to the carbon footprint of the schedule and energy consumption could empower sustainability and the circularity of the shared manufacturing systems. However, when additional factors are taken into account, the problem becomes more challenging to solve [11]. Therefore, alternative exact methods can be considered to overcome the computational intractability of the proposed MIP model for large instances such as combinatorial Bender's decomposition and constraint programming. Investigating large neighbourhood search metaheuristic could significantly improve the scalability of the proposed MIP model and convergence of the developed heuristic methods. Uncertainty in job processing and order arrival times as well as machine breakdowns some other extensions of the problem that would need incorporating simulation for most robust production plans and schedules. A reinforcement learning algorithm that is trained based on the data from such simulation would be also a promising further research direction to handle the problem in real-time.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland Centre for Research Training in Artificial Intelligence under Grant No. 18/CRT/6223 and 22/NCF/DR/11264, National Challenge Fund, Digital for Resilience Challenge.

References

- [1] Agnetis, A., Mirchandani, P.B., Pacciarelli, D., Pacifici, A., 2004. Scheduling problems with two competing agents. *Operations research* 52, 229–242.
- [2] Arnum, P.V., 2012. A manufacturing-capacity sharing model. URL: <https://www.biopharminternational.com/view/manufacturing-capacity-sharing-model/>.
- [3] Bai, D., Diabat, A., Wang, X., Yang, D., Fu, Y., Zhang, Z.H., Wu, C.C., 2022. Competitive bi-agent flowshop scheduling to minimise the weighted combination of makespans. *International Journal of Production Research* 60, 6750–6771.
- [4] Camarinha-Matos, L.M., Afsarmanesh, H., Galeano, N., Molina, A., 2009. Collaborative networked organizations—concepts and practice in manufacturing enterprises. *Computers & industrial engineering* 57, 46–60.
- [5] Cheng, T.E., Ng, C.T., Yuan, J., 2006. Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical computer science* 362, 273–281.
- [6] Elvikis, D., T'kindt, V., 2014. Two-agent scheduling on uniform parallel machines with min-max criteria. *Annals of Operations Research* 213, 79–94.
- [7] GAIA-X, 2022. Gaia-x. URL: <https://www.bmwk.de/Redaktion/EN/Artikel/Digital-World/GAIA-X-Use-Cases/shared-production.html>.
- [8] Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey, in: *Annals of discrete mathematics*. Elsevier. volume 5, pp. 287–326.
- [9] Gu, M., Gu, J., Lu, X., 2018. An algorithm for multi-agent scheduling to minimize the makespan on m parallel machines. *Journal of Scheduling* 21, 483–492.
- [10] Kung, J.Y., Chao, Y.P., Lee, K.I., Kang, C.C., Lin, W.C., et al., 2013. Two-agent single-machine scheduling of jobs with time-dependent processing times and ready times. *Mathematical Problems in Engineering* 2013.
- [11] Lee, W.C., Wang, J.Y., 2017. A three-agent scheduling problem for minimizing the makespan on a single machine. *Computers & Industrial Engineering* 106, 147–160.
- [12] Mitsubishi, R.N., 2023. Renault-nissan-mitsubishi alliance open a new chapter for their partnership. URL: <https://media.alliancernm.com/renault-nissan-mitsubishi-alliance-open-a-new-chapter-for-their-partnership/?lang=eng>. accessed: 02.05.2024.
- [13] RIHN, . Redistributed Manufacturing in Healthcare. Technical Report. URL: https://rihn.org.uk/wp-content/uploads/2020/05/RiHN_WP_Full_double_web.pdf.
- [14] Wang, X., Ren, T., Bai, D., Ezech, C., Zhang, H., Dong, Z., 2022. Minimizing the sum of makespan on multi-agent single-machine scheduling with release dates. *Swarm and Evolutionary Computation* 69, 100996.
- [15] Wu, W.H., Xu, J., Wu, W.H., Yin, Y., Cheng, I.F., Wu, C.C., 2013. A tabu method for a two-agent single-machine scheduling with deterioration jobs. *Computers & Operations Research* 40, 2116–2127.
- [16] Yu, F., Wen, P., Yi, S., 2018. A multi-agent scheduling problem for two identical parallel machines to minimize total tardiness time and makespan. *Advances in Mechanical Engineering* 10, 1687814018756103.