Optimising a Manufacturing-as-a-Service platform through mathematical modeling

Angelos Ioannis Lagos[1], Ioannis Avgerinos[1], Georgios Zois[1], Ioannis Mourtos[1], and Patricia Casla[2]

[1] ELTRUN Research Lab, Department of Management Science and Technology, Athens University of Economics and Business, 104 34 Athens, Greece
[2] Fundación Tekniker, 20600, Eibar, Spain
t8210079@aueb.gr, iavgerinos@aueb.gr, georzois@aueb.gr, mourtos@aueb.gr, patricia.casla@tekniker.es

Abstract. This paper explores the integration of exact optimisation approaches into a Manufacturing-as-a-Service (MaaS) platform. Specifically, it addresses the scheduling of production requests within a flexible and distributed real-wold manufacturing network in the plastic injection molding domain. While traditional Mixed-Integer Linear Programming (MILP) formulations struggle with scalability in realistic-sized instances, a Constraint Programming (CP) model demonstrates higher efficiency, providing near-optimal solutions within short computational times. Formally, we examine the flowshop scheduling problem under multiple objectives that reflect the diverse interests of MaaS stakeholders, i.e., the minimisation of makespan and total so-called 'locality' weights. To balance these conflicting objectives, Pareto frontiers are constructed to provide insights for managerial decision-making. Experimental results show that the CP model outperforms the MILP formulation in both solution quality and scalability. Moreover, the findings suggest that greater emphasis on makespan ensures better objective balance without significantly impacting locality weights. Overall, our work contributes to the ongoing discourse on MaaS operations, showcasing how exact optimisation methods can enhance scheduling efficiency in distributed manufacturing ecosystems.

Keywords: Manufacturing-as-a-Service · Flow shop scheduling · Mixed Integer Linear Programming · Constraint Programming.

1 Introduction

In modern manufacturing, flexibility and efficiency are critical to maintaining competitiveness. The concept of Manufacturing-as-a-Service (MaaS) [7, 16] has emerged as a paradigm shift, enabling companies to dynamically assume the roles of provider or consumer of manufacturing services, as part of a collaborative production network. MaaS leverages distributed manufacturing operations to provide on-demand production capabilities, thus improving resource utilisation and reducing costs [6, 14]. Unlike traditional manufacturing models, MaaS fosters a decentralised approach, where manufacturing services are sourced dynamically from a network of providers to support resilience against demand fluctuations and supply chain disruptions [8].

In their comprehensive review, Shen et al. [15] examine the evolution of digital servitisation, highlighting its transformative impact on manufacturing firms. The authors explore how digital technologies such as IoT, AI, and cloud computing enable the shift from product-centric to service-centric business models. While their study does not explicitly address MaaS, its insights into digital platforms, business model innovation, and value co-creation provide a conceptual foundation highly relevant to MaaS ecosystems; it also underscores the importance of robust frameworks that support modular, service-oriented, and digitally-integrated manufacturing. Additionally, Pulkkinen et al. [11] analyse modelling approaches for additive manufacturing within MaaS frameworks, demonstrating how digital twins and cloud-based optimisation can enhance scheduling flexibility and production efficiency. That work underscores the importance of real-time and adaptive planning mechanisms in MaaS environments, particularly when jobs are scheduled across multiple service providers and with varying constraints.

Our work presents an application from the EU-funded project Tec4MaaSEs[3], which studies an ecosystem of three organisations: *Moldes URA*, *ERREKA*, and *Tekniker*, operating in the plastic injection molding domain. This reflects a dynamic MaaS network where jobs are distributed among variable service providers. Recent studies have explored multi-objective approaches for dynamic task allocation and service matching in MaaS, such as game-theoretic formulations for platform-servicer coordination [3] and machine brokering in failure-tolerant production networks [9]. However, these approaches do not address production-level scheduling with constraints such as job precedences, machine eligibility, machine time windows, and transportation delays. Our contribution bridges this gap by integrating exact MILP and CP-based formulations that incorporate dynamic provider-consumer roles, distributed machine availability, inter-provider transportation, and multi-objective trade-offs tailored to MaaS.

The jobs in our MaaS case study follow fixed processing sequences across multiple services, such as additive manufacturing, drilling, and plastic injection molding, making flowshop modeling a suitable categorization. The system includes eleven manufacturing services shared among three partners. *Moldes URA* primarily consumes additive manufacturing while providing machining; *ERREKA* offers injection moulding while consuming both additive and machining; and *Tekniker* mainly offers additive and machining services, occasionally outsourcing tasks. This decentralised, role-flexible structure forms an interconnected manufacturing environment where jobs must be scheduled across providers with dynamic availability. While recent surveys, such as Duan et al. [4], indicate that heuristic and metaheuristic methods dominate the solution landscape for distributed flowshop scheduling problems, our work advances the field by introducing exact MILP and CP-based formulations. These enable precise

[3] http://tec4maases.eu

modeling of these MaaS specific complex constraints, and offer optimal solutions where heuristic methods may fall short.

Consumers provide data regarding their requests and the providers register their machines. In return, the consumers receive the schedule of their requests and the participating providers monitor the utilisation of their infrastructure. A concise illustration of the flow of information is shown in Figure 1. The optimisation service of the platform produces a set of alternative schedules; then, the platform facilitates a negotiation phase between customers and providers to select a schedule that balances acceptable costs for the customers with a profit distribution scheme that benefits the providers - this post-optimisation phase is out of scope of this paper.

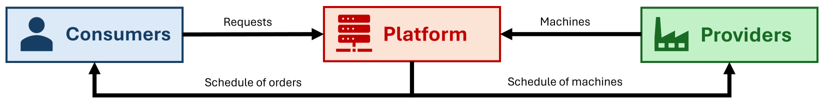

Fig. 1: Exchange of information between the stakeholders and the platform

Given the diverse partners using the platform, their potentially conflicting interests necessitate addressing multiple objectives. For instance, MaaS consumers seek quick services, platform operators aim for fair job distribution among providers, and all manufacturers prefer efficient, minimal-cost schedules. Here, we focus on minimising makespan and total locality weights, i.e., a MaaS-specific metric related to the distance between consumers and providers. These objectives can be in conflict: minimising makespan encourages job distribution across parallel machines, while minimising locality weights favours concentrating requests within local providers and avoid trips to distant ones. To determine the appropriate weighting for each objective, we construct a Pareto frontier, following an approach similar to [5].

Overall, our work integrates exact mathematical modeling approaches into the designed MaaS operations, addressing specific constraints such as dynamic role adaptation, multi-provider scheduling, and transportation-based delays. Traditional MILP formulations are limited by low scalability, making them unsuitable for such complex scenarios. Therefore, we adopt an equivalent CP model, proven efficient within the short computational times required for platform decision-making. By framing the problem within a flowshop or job shop context, we demonstrate how optimised scheduling techniques can enhance the efficiency of distributed manufacturing ecosystems.

The remainder of this paper is structured as follows: Section 2 provides a formal problem definition, the mathematical model, and the methodology. Section 3 describes the computational experiments: a comparison between the mathematical models and the construction of Pareto frontiers to incorporate multiple objectives. Finally, Section 4 concludes with insights and future research directions.

2 Problem Formulation and Methodology

2.1 Problem Description

Our motivation emanates a dynamic manufacturing ecosystem within the plastic injection moulding domain, where three companies collaborate flexibly as providers or consumers. This network enables an agile and adaptive production environment through three primary service categories: Additive Manufacturing, Machining, and Plastic Injection Moulding. *ERREKA Plastics* specialises in the design and manufacturing of plastic injection components for a wide range of industries, positioning itself primarily as a consumer of additive manufacturing and machining services while offering plastic injection moulding as a service. *Moldes URA*, a small-medium enterpise focused on the design and manufacturing of moulds, acts as a consumer of additive manufacturing services while providing machining capabilities. In contrast, *Tekniker*, a research center specialising in advanced manufacturing, operates mainly as a provider of both additive manufacturing and machining services, though it occasionally consumes machining services when necessary.

Additive manufacturing offers significant advantages over traditional methods in the injection moulding sector, such as lowering production costs, accelerating time-to-market and reducing the need for expensive spare part inventories. However, it also introduces challenges, including high investment costs and the need for additional finishing processes. The idea is to address these challenges by enabling a rapid, ad-hoc reconfiguration of supply chains based on real-time data, such as current production capacity and demand forecasts. Unlike conventional additive manufacturing marketplaces, this approach allows for seamless integration of manufacturing services, transportation logistics, and supply chain flexibility, fostering resilience and responsiveness within the network.

Focusing on production scheduling, the challenge is to efficiently assign jobs - each corresponding to specific parts and operations - to a set of machines, while considering eligibility constraints, processing order requirements and transportation times. We assume that, since the companies are geographically close, each part can be transferred individually from one provider to another, accounting for its own transportation time, without the need for batch-based transport. By formulating this problem both as a MILP model and as a CP model, we aim to explore effective scheduling strategies that maximise resource utilisation while ensuring timely delivery of manufacturing requests.

2.2 Mathematical annotation

The dynamic manufacturing ecosystem assigns requests R that arrive over time to machines K. Each request has a number of identical products, namely *parts*. Set I contains all parts to be manufactured; subsets I_r refer to the parts connected with request $r \in R$. Each part must go through a sequence of operations. The set of the operations for all parts is denoted as J, thus subsets J_i contain the operations (i.e., jobs) to be applied to part $i \in I$. All jobs within a subset J_i are ordered to define the flowshop sequence, hence let $j_i^1, j_i^2, ..., j_i^n$ be the sequence of the n jobs of a subset J_i, i.e., $n = |J_i|$. Each job $j \in J$, dedicated to a part of a request and an operation, has a machine-dependent processing time $p_{j,m}$, $m \in \mathcal{M}$ being the assigned machine. Set J contains a 'dummy' job, denoted by 0, which is used for modeling purposes. The set of all jobs is therefore denoted by $J^* = J \setminus \{0\}$.

The available machines K operate in parallel and each machine can process one job at a time. The transportation times $c_{n,k}$ represent the duration required to transfer materials or products between machines n and k. Times range in a few hours, as the participating industries are located within the same region. Additionally, transportation times $f_{r,k}$ capture the movement of raw materials from customers of requests r to machines k. Each machine operates within a defined planning horizon, which may be disrupted by pre-scheduled operations outside the ecosystem or by maintenance breaks. To account for these constraints, we introduce set T, representing the available time windows. Each machine $k \in K$ has a corresponding subset T_k that contains its specific availability periods. Each time window $t \in T$ is defined by a lower bound l_t and an upper bound u_t, indicating the start and the end of the available period, respectively.

All jobs must be assigned to exactly one machine, subject to eligibility constraints. Also, since the processing of each job must not be interrupted, each job is assigned to exactly one available time window of the corresponding machine k. A binary matrix $\beta_{j,k}$ determines whether job j can be processed on machine k, based on factors such as operational compatibility, material constraints, and machine capacity (e.g., maximum dimensions or weight limits). This scheduling problem can be optimised for multiple objectives. In this paper, we focus on minimising the makespan and the total *locality* weight, a case-specific goal which captures the preference of each customer to collaboration with a local industry. All mathematical annotations used for the models construction are consolidated in Table 1.

Sets	
R	Requests
I	Parts, $I_r \subset I$ parts of request $r \in R$
J	Jobs, $J_i \subset J$ jobs of part $i \in I$
J^*	Real jobs, $J^* = J \setminus \{0\}$
K	Machines
T	Time windows, $T_k \subset T$ time windows of machine $k \in K$
Parameters	
$p_{j,k}$	Processing time of job $j \in J$ on machine $k \in K$
l_t	Lower bound of time window $t \in T$
u_t	Upper bound of time window $t \in T$
$c_{n,k}$	Transportation time between machines n, k
$f_{r,k}$	Transportation time between the recipient of request r and machine k
$q_{j,k}$	Locality weight for job r on machine k
$\beta_{j,k}$	Binary values; 1 if job $j \in J$ is eligible for machine $k \in K$, 0 otherwise
α	Bi-objective coefficient
M	Big-M; a large number

Table 1: Mathematical annotations

2.3 Mathematical models

Mixed-Integer Linear Program. The first mathematical model is a MILP, which integrates aspects of parallel machine scheduling and flowshop scheduling problems. Binary variables $x_{j,k,t}$ are set to 1 if job $j \in J$ is assigned to time window $t \in T_k$ of machine $k \in K$, or 0 otherwise. Binary variables $y_{i,j,k}$ are set to 1 if job $j \in J$ succeeds job $i \in J$ on machine $k \in K$ or 0 otherwise. The completion times of jobs $j \in J$ on machines $k \in K$ are indicated by continuous variables $C_{j,k}$. It is noted that, if $\beta_{j,k} = 0$ for a pair of job j-machine k, then all related variables $x_{j,k,t}$ and $y_{i,j,k}$ or $y_{j,i,k}$ are set to 0. Variable C_{max} indicates makespan (maximum completion time).

MILP:

$$\min \alpha \cdot C_{max} + (1-\alpha) \cdot \sum_{j \in J} \sum_{k \in K} \sum_{t \in T_k} q_{j,k} \cdot x_{j,k,t} \tag{1}$$

$$\sum_{k \in K} \sum_{t \in T_k} x_{j,k,t} = 1 \qquad \forall j \in J^* \tag{2}$$

$$C_{j,k} - u_t \leq M \cdot (1 - x_{j,k,t}) \qquad \forall j \in J, k \in K, t \in T_k \tag{3}$$

$$C_{j,k} - p_{j,k} \geq l_t \cdot x_{j,k,t} \qquad \forall j \in J, k \in K, t \in T_k \tag{4}$$

$$C_{i,k} + p_{j,k} - C_{j,k} \leq M \cdot (1 - y_{i,j,k}) \qquad \forall i \in J, j \in J^*, k \in K \tag{5}$$

$$\sum_{i \in J} y_{i,j,k} = \sum_{t \in T_k} x_{j,k,t} \qquad \forall j \in J^*, k \in K \tag{6}$$

$$\sum_{i \in J} y_{i,j,k} = \sum_{i \in J} y_{j,i,k} \qquad \forall j \in J, k \in K \tag{7}$$

$$C_{0,k} \leq C_{j,k} - p_{j,k} \qquad \forall j \in J^*, k \in K \tag{8}$$

$$\sum_{j \in J^*} y_{0,j,k} \leq 1 \qquad \forall k \in K \tag{9}$$

$$\sum_{i \in J^*} y_{0,i,k} \geq \sum_{i \in J} y_{i,j,k} \qquad \forall j \in J^*, k \in K \tag{10}$$

$$C_{j_i^1,k} \geq \sum_{t \in T_k} (f_{r,k} + p_{j_i^1,k}) \cdot x_{j_i^1,k,t} \qquad \forall k \in K, r \in R, i \in I_r \tag{11}$$

$$C_{max} \geq C_{j,k} \qquad \forall j \in J^*, k \in K \tag{12}$$

$$C_{j_i^{n\text{-}1},k} + p_{j_i^n,k'} + c_{k,k'} - C_{j_i^n,k'} \leq M \cdot (2 - \sum_{t \in T_k} x_{j_i^{n\text{-}1},k,t} - \sum_{t \in T_{k'}} x_{j_i^n,k',t})$$
$$\forall i \in I, k \in K, k' \in K, n = 2, ..., |J_i| \tag{13}$$

$$x_{j,k,t} \in \{0, \beta_{j,k}\} \qquad \forall j \in J, k \in K, t \in T_k$$
$$y_{i,j,k} \in \{0, \min\{\beta_{j,k}, \beta_{i,k}\}\} \qquad \forall i \in J, j \in J, i \neq j, k \in K$$
$$C_{j,k} \geq 0 \qquad \forall r \in R, i \in I_r, j \in J_i, k \in K$$
$$C_{max} \geq 0 \qquad \forall r \in R$$

Constraints (2) ensure that each real job is assigned to exactly one time window, which is dedicated to one machine by construction. Constraints (3) and (4) satisfy the bounds of the assigned time window: the completion time of each job must not exceed the upper bound, and the start time of each job must exceed the lower bound respectively. Constraints (5) - (7) ensure that each machine can process one job at a time. Constraints (5) are commonly used in scheduling problems to ensure that the completion times of successive jobs on the same machine do not overlap. Constraints (6) ensure that each real job which is assigned to a time window succeeds another job on the same machine - this job may be the imaginary 'dummy' job - and (7) preserve flow. The 'dummy' job is scheduled first at each machine by constraints (8). Constraints (9) and (10) ensure that the 'dummy' job is scheduled on a machine only if at least one real job is assigned to an associated time window. The first operation of any part requires transferring material to the assigned machine, thus constraints (11) introduce the corresponding transportation times. Constraints (12) define the value of makespan. Last, constraints (13) impose precedences between operations of the same part: for each pair of successive operations, the subsequent operation starts at the completion time of the precedent one, and ends after its processing and its transportation to the associated machine are completed.

Regarding the domains of variables, we note that variables $x_{j,k,t}$ are bounded by the value of $\beta_{j,k}$: if $\beta_{j,k} = 0$, the job j is not eligible for machine k, thus all corresponding variables are set to 0. Similarly, variables $y_{i,j,k}$ are set to 0 if at least one of jobs i or j are not eligible for machine k. The objective function (1) is the weighted sum of makespan and total locality weight. The sum of weights is equal with 1, thus a coefficient value $\alpha \in [0, 1]$ is introduced.

Constraint Programming. The second mathematical formulation leverages the flexibility of Constraint Programming (CP). At its core, the CP model employs optional interval variables, $\texttt{jobIntA}_{j,k,t}$, which define the start and end times of job $j \in J$ when assigned to time window $t \in T_k$ on machine $k \in K$. Interval variables $\texttt{jobIntB}_{j,k}$ are restricted to defining the processing of job j on machine k, regardless of the assigned time window. Additionally, sequence variables $\texttt{machineSeq}_k$ establish the processing order of jobs on each machine $k \in K$.

CP:

$$\min\ \alpha \cdot C_{max} + (1-\alpha) \cdot \sum_{j \in J^*} \sum_{k \in K} q_{j,k} \cdot \texttt{presenceOf}(\texttt{jobIntB}_{j,k}) \tag{14}$$

$$\sum_{k \in K} \sum_{t \in T_k} \texttt{presenceOf}(\texttt{jobIntB}_{j,k}) = 1 \qquad \forall j \in J^* \tag{15}$$

$$\texttt{alternative}(\texttt{jobIntB}_{j,k}, [\texttt{jobIntA}_{j,k,t} | t \in T_k]) \qquad \forall j \in J^*, k \in K \tag{16}$$

$$\texttt{presenceOf}(\texttt{jobIntB}_{j,k}) \le \beta_{j,k} \qquad \forall j \in J^*, k \in K \tag{17}$$

$$\texttt{noOverlap}(\texttt{machineSeq}_k) \qquad \forall k \in K \tag{18}$$

$$\texttt{endBeforeStart}(\texttt{jobIntB}_{j_i^{n\text{-}1},k}, \texttt{jobIntB}_{j_i^n,m}, c_{k,m}) \qquad \forall i \in I, n = 2, ..., |J_i|, k \in K, m \in K \tag{19}$$

$$\texttt{startOf}(\texttt{jobIntB}_{j_i^1,k}) \ge f_{r,k} \cdot \texttt{presenceOf}(\texttt{jobIntB}_{j_i^1,k}) \qquad \forall r \in R, i \in I_r, k \in K \tag{20}$$

$$C_{max} \ge \texttt{endOf}(\texttt{jobIntB}_{j,k}) \qquad \forall j \in J^*, k \in K \tag{21}$$

$$\texttt{jobIntA}_{j,k,t} : \texttt{interval}, \texttt{optional}, [l_t, u_t] \qquad \forall j \in J^*, k \in K, t \in T_k$$

$$\texttt{jobIntB}_{j,k} : \texttt{interval}, \texttt{optional} \qquad \forall j \in J^*, k \in K$$

$$\texttt{machineSeq}_k : [\texttt{jobIntB}_{j,k} | j \in J^*] \qquad \forall k \in K$$

$$C_{max} \ge 0$$

Optional interval variables are either present ($\texttt{presenceOf}(\texttt{jobIntB}_{j,k}) = 1$), meaning that job j is assigned to machine k, or absent ($\texttt{presenceOf}(\texttt{jobIntB}_{j,k}) = 0$). Constraints (15) ensure that each job is assigned to exactly one machine. The `alternative` constraint ensures that, if an interval variable is present, then it is synchronised to exactly one (or any larger cardinality which is indicated by the user) interval variable of a given list. Constraints (16) allocate each job to exactly one time window, dedicated to the assigned machine. Ineligible assignments are prohibited by constraints (17). Since each machine can only process one job at a time, a `noOverlap` constraint is enforced on the sequence variable of each machine (18). For consecutive operations of the same part, the transportation times of the corresponding machines must be considered, as shown in Constraints (19): the `endBeforeStart` constraint indicates the minimum delay between the end time and the start time of a pair of interval variables, if both of them are present. By constraints (20), the first operation of each part should start after the required materials are transferred from the customer's location to the assigned machine. The makespan is determined as the maximum end time of all jobs (21). As for the variables, optional interval variables $\texttt{jobIntA}_{j,k,t}$ are bounded within the limits of time window t, and sequences $\texttt{machineSeq}_{k}$ consist of all present interval variables $\texttt{jobIntB}_{j,k}$.

3 Computational Experiments

3.1 Experimental setting

The use-case ecosystem consists of a set of 24 parallel machines across three providers. Each machine has a set of eligible operations it can perform (e.g., Additive Manufacturing, machining, Plastic Injection Moulding). For each operation, a rate-per-minute (rpm) weight value determines the duration of scheduled processes. Also, each machine has a set of time windows indicating when the machine can be utilised - denoted by T_k for machine $k \in K$. Each pair of machines (n, k) has a transportation time $c_{n,k}$. Each machine can only process one job at a time within its availability constraints. Machines can belong to different companies that dynamically act as either providers or consumers of manufacturing services.

In the MaaS ecosystem, production requests are submitted by various companies, specifying the required manufacturing services and associated constraints. All request parts must be scheduled within the available machine time intervals. Each request consists of multiple parts, each with an ordered list of operations that must be scheduled sequentially. These operations correspond to known manufacturing services (e.g. Additive Manufacturing, machining, Plastic Injection Moulding). Each part-operation combination is represented as a job $j \in J^*$. For each operation of a job j, the processing time $p_{j,k}$ is calculated by multiplying the quantity with the rpm weight of a compatible machine k. Additionally, each request includes a transportation time from the company's location to each machine - a locality weight is derived of this time. Additional parameters, such as dates of submission, delivery windows, types of materials and dimensions which may affect the compatibility with the machines, and user-specified criteria which should be added in the multi-objective function, may be involved - this paper is focused on parameters which are involved in the minimisation of makespan and locality weights.

To evaluate model performance, we use a request generator guided by the use case pilot's directions and specifications. All datasets share the same machine configuration. The number of requests is set to either 5 or 10, with each request containing a randomly selected number of 5 to 20 parts. Additionally, for each request, a random subset of services is chosen from those available on the machines. Since the execution order of services is strict for each part, the number of services per request significantly impacts problem complexity. Therefore, we consider three levels for the maximum number of services per request: 2, 3, or 4 (i.e., for each new request, a random selection determines whether 2, 3, or 4 services must be performed per part). In practice, this number indicates the cardinality of subsets J_i for each part $i \in I$.

The nominal processing time of each part is drawn from an integer distribution between 20 and 100 minutes. The processing time of each job j is then adjusted by multiplying it with the rpm weight of each eligible machine k, yielding $p_{j,k}$. Transportation times vary depending on provider relationships: transfers between machines of different providers range from 1 to 5 hours (converted to minutes), while transfers between machines of the same provider range from 5 to 15 minutes. Finally, transportation times from a request's location to machines also range from 1 to 5 hours (60 to 300 minutes), and these values are used as locality weights.

Table 2: Request Generation Rules

Parameter	**Rule**
$\|R\|$	$\{5, 10\}$
$\|I_r\|$, $r \in R$	$\texttt{integer}(5, 20)$
$\|J_i\|$, $r \in R, i \in I_r$	$\{2, 3, 4\}$
Processing time	$\texttt{integer}(20, 100)$
$f_{r,k}$, $r \in R, k \in K$	$\texttt{integer}(1, 5) \times 60$

Table 2 summarises the rules used for generating requests. A total of 24 instances are created by replicating each combination of request count (5 or 10) and maximum services per request (2, 3, or 4) four times. Instances 1-12 have 5 requests and instances 13-24 have 10 requests. Instances 1-4 and 13-16 have 2 services, instances 5-8 and 17-20 have 3 services and instances 9-12 and 21-24 have 4 services at most.

All experiments have been performed on a server with 4 Intel(R) Xeon(R) E-2126G @ 3.30GHz processors and 11 GB RAM, running CentOS/Linux 7.0. The experiments involve the solution of the MILP and the CP model. For the first one, we use the *Gurobi 10.0.0* optimizer, run by the open-source optimisation library *Pyomo 5.7.3* on *Python 3.8.5.* The CP model is solved by the *CP Optimizer* of the *CPLEX 20.1* optimizer - integrated into the DOCplex module.

3.2 Comparison of mathematical models

The first group of experiments compares the performance of the two mathematical models (MILP and CP), presented in Section 2. Each model is solved for the makespan objective, i.e., $\alpha = 1$ in (1) and (14) and a time limit of 1,800 seconds is imposed. The short time limit serves the operational requirements

Table 3: Comparison of the mathematical models

Instance #	MILP				CP			
	LB	UB	Gap (%)	Time (s)	LB	UB	Gap (%)	Time (s)
1	1,534	1,616	5.07	1,800	1,616	1,616	0.00	988
2	1,537	1,607	4.37	1,800	1,607	1,607	0.00	580
3	855	959	10.84	1,800	919	919	0.00	657
4	1,506	1,600	5.88	1,800	1,600	1,600	0.00	236
5	746	1,659	55.03	1,800	1,653	1,653	0.00	178
6	1,804	1,810	0.33	1,800	1,810	1,810	0.00	800
7	1,870	1,938	3.51	1,800	1,938	1,938	0.00	839
8	1,625	1,958	17.01	1,800	1,688	1,729	2.37	1,800
9	1,503	2,395	37.24	1,800	1,503	1,894	20.64	1,800
10	1,518	2,265	32.98	1,800	1,639	1,699	3.53	1,800
11	1,811	2,060	12.09	1,800	1,870	1,882	0.64	1,800
12	1,632	1,849	11.74	1,800	1,680	1,782	5.72	1,800
13	1,819	1,875	2.99	1,800	1,875	1,875	0.00	1,017
14	795	1,830	56.56	1,800	1,211	1,421	14.78	1,800
15	815	2,284	64.32	1,800	1,811	2,190	17.31	1,800
16	1,521	2,084	27.02	1,800	1,577	1,617	2.47	1,800
17	1,505	2,276	33.88	1,800	1,607	1,607	0.00	921
18	1,513	2,258	32.99	1,800	1,811	2,224	18.57	1,800
19	1,808	2,422	25.35	1,800	1,808	1,848	2.16	1,800
20	797	5,773	86.19	1,800	1,721	1,721	0.00	1,148
21	616	-	-	1,800	1,802	1,941	7.16	1,800
22	839	-	-	1,800	1,822	2,079	12.36	1,800
23	880	-	-	1,800	1,988	2,280	12.81	1,800
24	736	-	-	1,800	1,843	2,383	22.66	1,800

of the ecosystem: the platform should be able to update the schedules of the available machines in reasonable time.

Several complex constraints are more easily handled in a CP formulation. This is evident in the results of the two models shown in Table 3. The CP model outperforms the MILP in terms of upper bounds for all instances, except for 6 out of 24 instances (1, 2, 3, 6, 7 and 13), where both models yield the same objective value. Surprisingly, the CP model also provides notably better lower bounds across all instances. As expected, the MILP model is less scalable, with the largest dataset group (21 - 24) yielding no feasible solutions within the time limit.

We also mention further insights to verify the superiority of the CP model. (Near-optimal) objective values are typically found quickly within 300 seconds, while the remaining time is spent tightening the lower bound to converge to the optimal solution. A feasible solution is obtained within seconds, whereas the MILP model often requires several minutes to find an initial feasible solution.

3.3 Construction of a Pareto Frontier

The ecosystem involves multiple stakeholders with varying interests, reflected in the consideration of multiple objectives. In the described problem, we focused on minimising total locality weights and makespan. Minimising locality weights promotes faster service for customers and strengthens connections between industries and local communities. Minimising makespan optimises platform operations and ensures a fair distribution of requests across industries: jobs are preferably assigned to parallel machines to reduce maximum completion time, rather than accumulating on the same machines.

In bi-objective use cases, selecting an appropriate value for α is critical, as the objectives may vary significantly in scale. For instance, makespan values range from 1,000 to 2,000 minutes, while total locality weights are measured in tens of thousands of units. To estimate an acceptable range for α, the second group of experiments constructs a Pareto frontier for each of the 24 generated instances. We employ the CP model to solve each instance across varying α values, incrementing by 0.05 from $\alpha = 0$ to $\alpha = 0.95$, resulting in 20 variations per instance - the objective values for $\alpha = 1$ have already been obtained by the results of the first group of experiments (Table 3). Using standard step sizes for α values is a common approach in constructing Pareto frontiers. Moreover, experimental results indicate that $\alpha > 0.5$ yields minimal deviations, suggesting that reducing the step size in this range offers little practical benefit. A 300-second time limit is imposed, as prior experiments indicated that near-optimal objective values are typically reached quickly.

The impact of α on the objectives is denoted by Err_α (%), a metric which indicates the distance of each objective value from its optimum. For each value of α, $\text{Err}_\alpha = \frac{Z^\alpha - \min Z}{\min Z}$, Z^α being an objective (makespan or locality weights), given the value of α, and $\min Z$ being the value of Z as a single objective. By (14), it is implied that min Makespan = Makespan^1 and min Locality weights = Locality weights^0.

Table 4 presents the extreme values of makespan and locality weights. The 'Err (%)' column shows the value of Err_α for the α that results in the maximum value of the corresponding objective: 0 for makespan and 1 for locality weights. Notably, α variation has a greater impact on makespan, as indicated by the large gap between its extreme values. This suggests that a higher α should be chosen to maintain a low makespan. In contrast, 'Err (%)' values for locality weights are generally low - mostly below 10% - implying that higher α values have a less significant effect on locality weights.

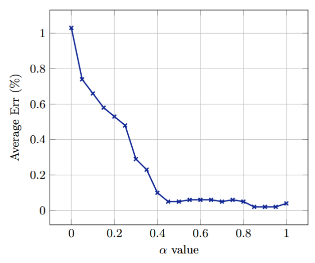

Fig. 2: Average Err (%) per value of α

This is also evident by the average values of Err_α for all objectives and all values of α, displayed in Figure 2. Each point on the graph is associated to a value of α and the respective average value of Err_α (i.e., $\frac{\sum_{i=1}^{24} \text{makespan Err}_\alpha^i}{24}$ $+ \frac{\sum_{i=1}^{24} \text{locality Err}_\alpha^i}{24}$ for each value of α, i being the number of instance). Consistent with the results in Table 4, a higher makespan coefficient ensures a better balance between the two objectives: the minimum average 'Err (%)' - 0.02% - is observed for $\alpha \in [0.85, 0.90, 0.95]$. Constructing Pareto frontiers for multiple objectives can offer valuable managerial insights, as selecting an appropriate α value ensures balanced schedules that meet the requirements of all involved partners.

Table 4: Extreme values of makespan and locality weights in the Pareto Frontier

Instance #	Makespan			Locality weights		
	min	max	Err (%)	min	max	Err (%)
1	1,616	1,814	12.25	20,040	21,180	5.69
2	1,607	5,905	267.45	21,480	24,180	12.57
3	919	2,102	128.73	32,760	32,760	0.00
4	1,600	5,742	258.88	17,280	18,840	9.03
5	1,653	5,903	257.11	20,400	24,660	20.88
6	1,810	5,898	225.11	21,420	26,700	24.65
7	1,938	5,708	194.53	32,820	35,940	9.51
8	1,730	5,897	240.87	38,280	42,660	11.44
9	1894	5,215	205.17	52,020	52,860	1.61
10	1,699	5,215	206.95	36,480	37,860	3.78
11	1,882	5,809	208.66	33,000	33,480	1.45
12	1,782	5,861	228.90	38,400	42,600	10.94
13	1,875	5,781	208.32	27,600	28,560	3.48
14	1,421	2,093	47.29	45,780	48,660	6.29
15	2,190	2,980	36.07	49,620	50,100	0.97
16	1,617	5,369	232.03	42,360	43,560	2.83
17	1,607	5,788	260.17	46,560	51,120	9.79
18	2,224	5,719	157.15	71,100	76,080	7.00
19	1,848	8,758	373.92	45,660	50,160	9.86
20	1,721	6,005	248.93	53,220	59,040	10.94
21	1,941	6,441	231.84	93,000	100,740	8.32
22	2,079	5,740	176.09	87,060	97,560	12.06
23	2,280	6,441	182.50	110,940	117,840	6.22
24	1,843	8,065	337.60	76,560	81,360	6.27

3.4 Demonstration of dynamic scheduling

This section examines an example of dynamic scheduling on the MaaS platform. Specifically, ‘Instance 5’ introduces a set of five requests at ‘01-12-2024 08:00’, while ‘Instance 6’ adds new demand at ‘02-12-2024 00:00’. Each instance assumes a different configuration of machine availability. Scheduling ‘Instance 5’ results in certain time intervals being occupied, and thus the subsequent demand from ‘Instance 6’ must be scheduled within the remaining available time windows. Figure 3 illustrates the results of this example.

The scheduling outcomes are displayed on separate screens for each stakeholder. Figure 3 shows the overview of the entire platform and a detailed view of an individual provider’s schedule (e.g., URA). Also, each customer monitors the schedule of their requests - an example for request 4 of ‘Instance 5’ is presented in Table 5. The newly assigned requests from the earlier instance are treated as occupied time slots in any subsequent scheduling.

Experimentation on the above instances highlights the need for effective resource management during each execution of the platform’s optimisation service. In addition to the existing objectives, incorporating the minimisation of unutilised enclosed time intervals - defined by lower and upper bounds - can enhance machine utilisation and create more scheduling flexibility for future demands.

A key metric for evaluating the sustainability of the platform is the equitable distribution of assignments among participating providers. While balancing machine usage across providers was not explicitly included as an optimisation objec-

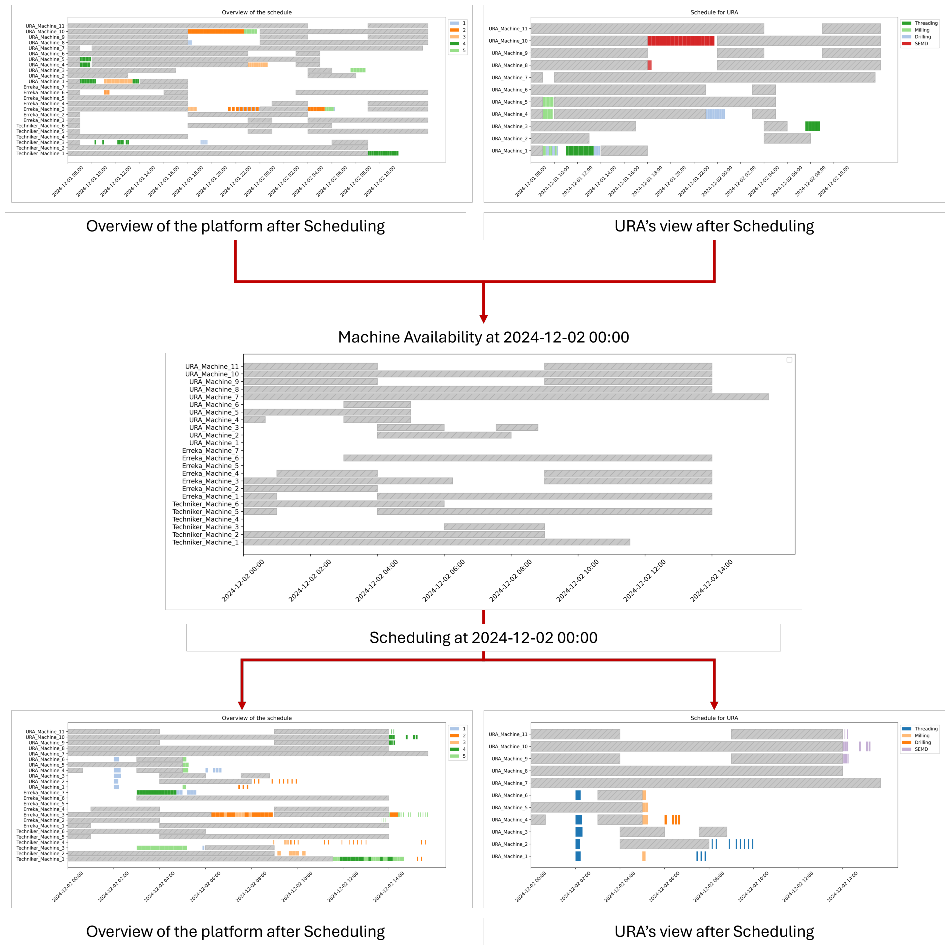

Fig. 3: Dynamic scheduling for Instances 5 and 6

tive, the results indicate a relatively fair distribution in practice. Table 6 reports the percentage of total processing time allocated to each provider across all instances. On average, Tekniker, Erreka, and URA account for 25.61%, 41.80%, and 32.59% of the total processing time, respectively. Although these results suggest a balanced utilisation under the current setup, the introduction of additional providers may amplify disparities. As such, fairness in assignment should be formally quantified and potentially incorporated into future optimisation criteria.

Table 5: Schedule from the perspective of the consumer - request 4 of instance 5

# Part	Milling			Drilling			Turning		
1	URA_1	01/12 09:00	01/12 09:08	URA_1	01/12 09:24	01/12 09:32	Tekniker_1	02/12 09:27	02/12 09:36
2	URA_1	01/12 09:40	01/12 09:48	URA_1	01/12 09:56	01/12 10:04	Tekniker_1	02/12 09:18	02/12 09:27
3	URA_1	01/12 09:32	01/12 09:40	URA_1	01/12 09:48	01/12 09:56	Tekniker_1	02/12 10:39	02/12 10:48
4	URA_5	01/12 09:00	01/12 09:07	Tekniker_3	01/12 12:07	01/12 12:15	Tekniker_1	02/12 09:09	02/12 09:18
5	URA_5	01/12 09:14	01/12 09:21	Tekniker_3	01/12 12:23	01/12 12:31	Tekniker_1	02/12 11:15	02/12 11:24

Table 6: Distribution of assignments to providers

Instance	1	2	3	4	5	6	7	8	9	10	11	12
Tekniker	20.96	15.56	19.54	61.64	16.69	40.04	43.58	14.13	21.96	19.97	18.93	18.26
Erreka	35.33	39.71	43.16	8.69	23.51	34.16	32.04	44.60	45.86	65.45	44.36	49.28
URA	43.71	44.73	37.31	29.67	59.80	25.80	24.38	41.26	32.17	14.58	36.71	32.46

Instance	13	14	15	16	17	18	19	20	21	22	23	24
Tekniker	27.16	27.03	18.11	16.73	17.18	19.68	34.38	29.22	27.57	27.04	30.18	29.18
Erreka	52.64	47.54	53.03	51.95	58.71	53.32	30.46	34.17	33.79	32.12	46.07	43.17
URA	20.20	25.43	28.86	31.32	24.11	27.00	35.17	36.60	38.65	40.84	23.75	27.65

4 Conclusions - Further Research

This paper explores the integration of exact mathematical models into the dynamic ecosystem of a Manufacturing-as-a-Service (MaaS) platform. While state-of-the-art MILP formulations are known for low scalability, limiting their applicability to large-scale practical problems, the designed equivalent CP model demonstrates high efficiency in rapidly generating near-optimal solutions. The construction of Pareto frontiers enables the examination of multiple objectives, helping to balance the often-conflicting requirements of participating partners.

Future work could involve incorporating additional objectives. For instance, minimising delivery times - an important but unexplored objective in this study - would be critical for platform customers. Similarly, addressing earliness and tardiness objectives could better integrate the platform with inventory management for both industries and customers. Minimising idle times would also benefit industries by improving facility utilisation rates.

The CP model was compared against a MILP formulation comprising common components (e.g., t-indexed variables, big-M constraints for precedence enforcement) and showed evident superiority in computational time, objective value and optimality gap. However, alternative formulations known for their efficiency in flowshop scheduling - such as position-based formulations [2] - might yield improved bounds, thus an extensive exploration of alternative formulations could follow. Beyond exact methods, metaheuristic algorithms in scheduling literature could offer faster solutions, particularly for handling larger volumes of requests. The evolving development of the MaaS platform is expected to reveal new requirements, guiding future research towards more focused areas.

Acknowledgments. This research has been supported by the EU through the Horizon Europe projects Tec4MaaSEs and MODAPTO, grant numbers 101138517 and 101091996, respectively.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Allahverdi A., Ng, C.T., Cheng T.C.E., Kovalyov M.Y.: A survey of scheduling problems with setup times or costs. European Journal of Operational Research **187**(3), 985–1032 (2008).
2. Bektaş T., Hamzadayi A., Ruiz R.: Benders decomposition for the mixed no-idle permutation flowshop scheduling problem. Journal of Scheduling **23**, 513–523 (2020).
3. Chen W., Feng P., Luo X., Nie L.: Task-service matching problem for platform-driven manufacturing-as-a-service: A one-leader and multi-follower Stackelberg game with multiple objectives. Omega **129**, 103157 (2024).
4. Duan J., Wang M., Zhang Q., Qin J.: Distributed shop scheduling: A comprehensive review on classifications, models, and algorithms. *Mathematical Biosciences and Engineering* **20**(4), 6835–6875 (2023).
5. Hamdan S., Cheaitou A., Shikhli A., Alsyouf I.: Comprehensive quantity discount model for dynamic green supplier selection and order allocation. Computers & Operations Research **160**, 106372 (2023).
6. Karamanli A., Xanthopoulos A., Gasteratos A., Koulouriotis D.: Manufacturing-as-a-Service: A Systematic Review of the Literature. In: *Proceedings of the International Conference on Smart Cities and Green ICT Systems*. Communications in Computer and Information Science, vol 2110, pp. 269–281 (2025).
7. Lee J., Kao H.A., Yang S.: Service innovation and smart analytics for Industry 4.0 and big data environment. Procedia CIRP **16**, 3–8 (2016).
8. Leng J., Zhong Y., Lin Z., Xu K., Mourtzis D., Zhou X., Zheng P., Liu Q., Zhao J.L., Shen W.: Towards resilience in Industry 5.0: A decentralized autonomous manufacturing paradigm. *Journal of Manufacturing Systems* **71**, 95–114 (2023).
9. Medeiros G.H.A., Cao Q., Zanni-Merk C., Samet A.: Manufacturing as a Service in Industry 4.0: A Multi-Objective Optimization Approach. In: Czarnowski, I. et al. (eds.) Intelligent Decision Technologies, Smart Innovation, Systems and Technologies, vol. 193, pp. 37–46. Springer, Singapore (2020).
10. Pinedo M.: Scheduling: Theory, Algorithms, and Systems. Springer, (2012).
11. Pulkkinen A., Nagarajan H.P.N., Heilala J.: A Review of Modelling Methods for Manufacturing as a Service – Case Additive Manufacturing. Advances in Transdisciplinary Engineering **52**, 313–321 (2024).
12. Reeves C.R.: Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, (1993).
13. Ruiz R., Stützle T.: An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. European Journal of Operational Research **187**(3), 1143–1159 (2008).
14. Schöppenthau F., Patzer F., Schnebel B., Watson K., Baryschnikov N., Obst B., Chauhan Y., Kaever D., Usländer T., Kulkarni P.: Building a Digital Manufacturing as a Service Ecosystem for Catena-X. *Sensors* **23**(17), 7396 (2023).
15. Shen L., Sun W., Parida V.: Consolidating digital servitization research: A systematic review, integrative framework, and future research directions. Technological Forecasting and Social Change **191**, 122478 (2023).
16. Zhang L., Luo Y., Tao F., Guo B., Cheng C., Yin Q, Zhang L.: Cloud manufacturing: a new manufacturing paradigm. Enterprise Information Systems **8**(2), 167–187 (2014).