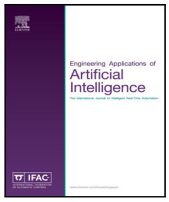
Engineering Applications of Artificial Intelligence 108 (2022) 104556

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Outranking-based multi-objective PSO for scheduling unrelated parallel machines with a freight industry-oriented application ☆

Gilberto Rivera, Raúl Porras, J. Patricia Sanchez-Solis *, Rogelio Florencia, Vicente García

Universidad Autónoma de Ciudad Juárez, División Multidisciplinaria de Ciudad Universitaria, Av. José de Jesús Macías Delgado #18100, Cd. Juárez, Chihuahua, 32000, Mexico

ARTICLE INFO

Keywords:
Swarm intelligence
Multi-objective optimisation
Fuzzy outranking
Unrelated parallel machine scheduling
Particle swarm optimisation

ABSTRACT

This paper presents Outranking-based Particle Swarm Optimisation (O-PSO) a novel metaheuristic to address the multi-objective Unrelated Parallel Machine Scheduling Problem. It is a particle swarm optimisation algorithm enriched with the preferences of the Decision Maker (DM), articulated in a fuzzy relational system based on ELECTRE III. Unlike other multi-objective metaheuristics, O-PSO searches for the Region of Interest (RoI) instead of approximating a sample of the complete Pareto frontier. The RoI is the subset consisting of those Pareto-efficient solutions that satisfy the outranking relations, that is, they are the best solutions in terms of the DM's system of preferences. Therefore, O-PSO not only approximates the Pareto solutions but also supports multicriteria decision analysis of the schedules. The efficiency of O-PSO is validated on a benchmark of synthetic instances from the scientific literature, where the Wilcoxon rank-sum test provides statistical evidence that O-PSO offers high-quality solutions when compared with two state-of-the-art metaheuristics; specifically, O-PSO is capable of generating a greater proportion of solutions (on average, ranging from 7% to 14%) dominating those of the state-of-the-art algorithms, as well as finding more solutions (from 13% to 18%) that satisfy the DM's preferences. O-PSO is also applied to a real-world case study in the transport industry to provide evidence for its applicability.

1. Introduction

There are various problems in organisations that can be addressed using Artificial Intelligence (AI) models. However, organisations prefer to address these issues through the skills they have gained from experience, designing strategies that allow them to generate solutions that will be acceptable in practice.

Clear examples are the heuristics used by companies in the cargo transport sector. The function of these companies is to deliver goods from the point of origin to the destination point. The Traffic Coordinator (TC) – called the Decision Maker (DM) in a broader sense – makes the decisions about the planning of the deliveries of goods. The most common techniques are queuing-based policies, where the TC assigns the first requested trip to the first available operator (driver). Even though this type of strategy is straightforward and can be quickly applied by the TC, they hardly evaluate the programmed schedule's overall impact on the goals of the organisation. These difficulties can be addressed if they are modelled as a Scheduling Problem (SP).

An SP is a decision-making problem focused on allocating resources to tasks in specific periods whose purpose is to optimise one or more objectives associated with productivity. These resources and tasks vary. For example, the resources may be workshop machines (e.g. Harbaoui and Khalfallah, 2020), runways (e.g. Yin et al., 2020), operating theatres (e.g. Rivera et al., 2020), or processing units (e.g. Chang et al., 2020). The tasks may be process operations (e.g. Harbaoui and Khalfallah, 2020), take-offs and landings (e.g. Yin et al., 2020), surgeries (e.g. Rivera et al., 2020), or computer programs (e.g. Chang et al., 2020).

Murakami and Morita (2010) identify the so-called 'known uncertainty' in this problem, which refers to information about events that have already happened in past schedules. Different proposals in the literature have modelled the uncertainty in scheduling optimisation given the domain's nature, including probabilistic models and fuzzy systems (cf. Wojakowski and Warżołek, 2014). Here, we must distinguish between two types of uncertainty:

- risk, which refers to events that jeopardise the schedule; consequently, the DM should select the schedule with the most advantageous trade-off between risks and benefits; this decision depends on their degree of conservatism; and
- imprecision, which means that the information to characterise the machines, jobs, and tasks may vary; the optimisation method

☆ This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

* Corresponding author.
E-mail address: julia.sanchez@uacj.mx (J.P. Sanchez-Solis).

https://doi.org/10.1016/j.engappai.2021.104556
Received 5 April 2021; Received in revised form 7 November 2021; Accepted 8 November 2021
Available online 8 December 2021

should be robust and consistent when this kind of knowledge is provided in advance by experience.

Although extensive scientific evidence (cf. Frausto-Solis et al., 2021; Ezugwu, 2019; Akbar and Irohara, 2018; Fuchigami and Rangel, 2018; Ramos-Figueroa et al., 2020) highlights the importance of solving SPs efficiently, it remains challenging to put this into practice for real-world cases. This difficulty is mainly because of two factors: (a) there are different variants of the problem, which derive from the organisation's particular needs, and (b) the number of objectives to be evaluated.

Regarding the first factor, variants of the SP may be classified into different categories, depending on the interaction between the tasks and the machines. We can mention Flow Shop, Open Shop, Identical Parallel Machines, and Unrelated Parallel Machine Scheduling Problem (UPMSP). In the latter, each machine has a different performance parameter describing how it processes a task, and all machines may work simultaneously. The present paper focusses on the UPMSP.

Regarding the second factor, optimising time-related objectives is the most popular goal in the SP literature. In this line, mono-objective approaches (e.g. Cheng et al., 2020; Kayvanfar et al., 2017; Shabtay and Zofi, 2018; Shahvari and Logendran, 2017) concentrate efforts on minimising this objective, while other concerns are modelled as hard constraints. However, in the last decade, the idea of addressing the Multi-Objective SP (MOSP) has become popular because, in practice, it is desired to optimise schedules along different dimensions. Most specialised studies address the MOSP through the concept of Pareto dominance and often provide large sets of solutions. Selecting only one final prescription – called the best compromise in the field of MultiCriteria Decision Analysis (MCDA) – from many solutions evaluated according to several criteria may be extremely hard for a DM with regular cognitive capacities (cf. Cruz-Reyes et al., 2020; Fernandez et al., 2019a). As far as we know, there are no proposals that model the preferences of the DM for supporting a multicriteria decision on the final prescription. In the present paper, we focus on addressing this issue in multi-objective UPMSP optimisation.

According to the literature, scheduling optimisation is computationally complex, being an NP-hard problem (cf. Ezugwu, 2019; Meng et al., 2019). Consequently, the application of multi-objective metaheuristic algorithms is booming. There is a wide variety of metaheuristics applied to multi-objective UPMSP, including Imperialist Competitive Algorithms (e.g. Garavito-Hernández et al., 2019; Lei et al., 2020), Artificial Immune System (e.g. Zhu and Tianyu, 2019; Manupati et al., 2017), Ant Colony Optimisation (e.g. Zhao et al., 2018; Afzalirad and Rezaeian, 2017), Tabu Search (e.g. Lin et al., 2016; Wang and Alidaee, 2019), Genetic Algorithms (e.g. Meng et al., 2019), and hybrids (e.g. Lu et al., 2018), to mention only a few.

On the other hand, only problems with a small number of objectives have been addressed in the literature: problems with two objectives (e.g. Zhu and Tianyu, 2019; Afzalirad and Rezaeian, 2017; Zhou and Gu, 2021; Kurniawan, 2020; Tirkolaee et al., 2020; Yepes-Borrero et al., 2021), three objectives (e.g. Lin et al., 2016; Bitar et al., 2021), four objectives (e.g. Manupati et al., 2017), and five objectives (e.g. Zhao et al., 2018) have been treated.

This paper introduces the Outranking-based Particle Swarm Optimisation (O-PSO) algorithm, which addresses the multi-objective UPMSP applied to the transport industry where the resources are the operators and the tasks are the delivery of the freight. Compared to other approaches, O-PSO offers better results in terms of efficiency while at the same time facilitating the decision-making process. The results obtained on synthetic instances and a real-world application in the cargo transport sector support these claims. The main contributions of this proposal are the following:

(1) O-PSO is enriched with a relational system – built on the underlying principles of fuzzy outranking (i.e. ELECTRE III) – to reflect the preferences of the DM;
(2) O-PSO is capable of generating non-dominated schedules and supporting multicriteria analysis to select the best compromise; and
(3) to the best of our knowledge, O-PSO is the first to address the multi-objective UPMSP with a freight industry-oriented application incorporating an *a priori* model of DM preferences.

The structure of this paper is as follows. In Section 2, we describe the background, including the baseline algorithm and the preference model built on ELECTRE III. Section 3 introduces our proposal, presenting the algorithm enriched with preferences to address the problem. Section 4 presents the conducted computational experiments, including both synthetic and real-world instances and a non-parametric statistical test to validate the results. Finally, in Section 5, we discuss some insights, and we make some recommendations for future research.

2. Background

A recent review of the related literature on UPMSP can be found in Ojstersek et al. (2020) and Ramos-Figueroa et al. (2020). Among the studies that address the UPMSP from an exclusively Paretian view, we must explicitly mention the Truncated Restarted Iterated Pareto Greedy (T-RIPG) by Yepes-Borrero et al. (2021) and the Evolutionary Discrete Particle Swarm Optimisation (EDPSO) by Wang et al. (2018). T-RIPG seeks to minimise the makespan and the number of resources in a bi-objective UPMSP that considers additional resources in the setup of machines. EDPSO seeks to minimise the processing time and the consumption of raw material to elaborate products in a real-world application. Both algorithms achieve a better performance than NSGA-II, a standard in the literature. Due to the lack of a qualified and widely-accepted benchmark for unrelated parallel machines, a comprehensive data generation driver was developed to reflect the real-world conditions plausibly encountered in practice. The latter is also an essential contribution of Wang et al. (2018).

O-PSO (our proposal) is an extension of EDPSO, enriched with a relational system of preferences built on the underlying principles of fuzzy outranking. The fuzzy outranking system also assembles indifference thresholds connected to the imprecision, and veto thresholds for risk-associated objectives to reflect the DM's conservatism.

Accordingly, Section 2.1 presents PSO (the basis of EDPSO and O-PSO), and Section 2.2 presents the fuzzy relational system of preferences.

2.1. Particle Swarm Optimisation

Kennedy and Eberhart (1995) proposed the Particle Swarm Optimisation (PSO) method by observing the behaviour of flocks in search of food, passing from one locality to another. During this activity, there is always a bird in the flock that has the best sense of smell for locating food. Once the source of food is found, the leading bird transmits that information so that the rest of the flock can also locate the food. Flocks are equivalent to clouds of particles, and the bird with the best sense of smell is the particle with the best-known solution, which transmits the information to the others to improve it together. PSO is still an object of scientific interest, and several research studies in the literature have recently extended PSO to propose novel optimisation algorithms (e.g. Naderi et al., 2019, 2020, 2021; Zhang et al., 2021; Gilvaei et al., 2020; Yan et al., 2020).

The basic PSO algorithm starts by generating a set of n particles at random, which are moved around in the search space according to a few simple but effective formulae. Originally, the point of each particle in an $\mathbb{N}$-dimensional space represents a possible solution for a problem with $\mathbb{N}$ variables and a single objective function. The movement of the ith particle is guided by its own best-known position ($\mathcal{P}_i$) as well as the entire swarm's best-known position ($\mathcal{G}$).

According to Bai (2010), the position update in the ιth iteration is defined as

$$X_{i,j}^{\iota} = X_{i,j}^{\iota-1} + \vec{V}_{i,j}^{\iota} \quad \forall i \in \{1,2,3,\dots,n\}, j \in \{1,2,3,\dots,\mathbb{N}\}, \tag{1}$$

where $X_{i,j}^{\iota-1}$ is the coordinate of the ith particle along the jth dimension in the previous iteration $(\iota-1)$, and $V_{i,j}^{\iota}$ is the velocity of the movement for the ith particle along the jth dimension in the ιth iteration, which is defined as

$$\vec{V}_{i,j}^{\iota} = \vec{V}_{i,j}^{\iota-1} + c_1 r_p(\mathcal{P}_{i,j} - X_{i,j}^{\iota-1}) + c_2 r_g(\mathcal{G}_d - X_{i,j}^{\iota-1}), \tag{2}$$

where $\vec{V}_{i,j}^{\iota-1}$ is the velocity along the jth dimension of the ith particle during the $(\iota-1)$th iteration; $\mathcal{P}_i$ and $\mathcal{G}$ are points (in the $\mathbb{N}$-dimensional hyperspace) representing, respectively, the best point individually known by particle i and the best one collectively known by the swarm; c_1 and c_2 are weight parameters balancing the orientation between flying to $\mathcal{P}_i$ and to $\mathcal{G}$; and r_p and r_g are pseudorandom numbers, $r_p, r_g \sim \mathrm{U}(0,1)$.

On the one hand, the criterion for selecting $\mathcal{P}_i$ and $\mathcal{G}$ is well-defined for single-objective optimisation problems. On the other hand, when there are multiple objective functions, the most relevant criterion for selecting the best solutions is the concept of Pareto dominance. There is a comprehensive discussion of this concept and its impact on evolutionary optimisation in Coello Coello et al. (2007, Chap. 1). Here, for the sake of simplicity, let $f(X_i) = (f_1(X_i), f_2(X_i), f_3(X_i), \dots, f_p(X_i))$ be the values of the multi-objective function for the ith particle in a problem with p minimising objectives. Given two particles (with indices i, j), X_i dominates X_j if X_i is not worse than X_j in any objective and X_i is strictly better than X_j at least in one objective. With more formality:

$$\begin{aligned} X_i \preceq X_j \text{ iff } \quad & \forall k \in \{1,2,3,\dots,p\},\ f_k(X_i) \le f_k(X_j) \quad \wedge \\ & \exists k \in \{1,2,3,\dots,p\} : f_k(X_i) < f_k(X_j), \end{aligned} \tag{3}$$

where $\preceq$ stands for the relation of Pareto dominance. In terms of the computational complexity theory, the objective functions have to be compared sequentially to determine the dominance relations between a pair of solutions. Hence the complexity function of a program implementing Eq. (3) belongs to $\Theta(p)$.

The Pareto set is then $PS = \{X_i : \nexists X_j \preceq X_i\}$. Because the best solution belongs to PS, the multi-objective optimisation approaches aim to approximate PS. It is assumed that the DM can analyse *a posteriori* the Pareto set and its multi-objective image to select the best compromise solution — the prescription to be implemented, the true solution to the problem.

In this paper, we present a PSO algorithm that searches for the best compromise solution. The preferences of the DM are articulated, focussing the computing power on identifying the best compromise solution. This model is not only compatible with the notion of Pareto efficiency but also contributes to support the multicriteria decision analysis of the best schedules.

2.2. The ELECTRE III-based system of preferences

ELECTRE III was proposed for management problems where the preferences are based on relations under the concept of fuzzy outranking (Roy and Vanderpooten, 1996), which can be defined as the degree of truth of "X_i is at least as good as X_j", denoted by $\sigma(X_i, X_j)$, where X_i and X_j are alternatives (in our case, schedules), with multiple criteria. The two basic measures to calculate outranking are:

(1) the concordance index, which models the strength of a coalition in favour of $\sigma(X_i, X_j)$, and is denoted by $c(X_i, X_j)$; and
(2) the discordance index, which inversely measures the strength of a coalition against $\sigma(X_i, X_j)$, denoted by $d(X_i, X_j)$.

Therefore, for "X_i is at least as good as X_j" to be true, the values of $c(X_i, X_j)$ and $d(X_i, X_j)$ must be high. More precisely,

$$\sigma(X_i, X_j) = c(X_i, X_j) \cdot d(X_i, X_j). \tag{4}$$

To estimate the concordance index, it is necessary to know how the DM perceives the values of the criteria, expressed through the following parameters:

(1) the weights of the objectives, denoted by $W = (w_1, w_2, \dots, w_p)$, represent the importance that the DM attaches to each of the objectives; here, $1 \ge w_k > 0\ \forall k \in \{1,2,3,\dots,p\}$ and $\sum_{k=1}^{p} w_k = 1$, where p is the number of objectives; and
(2) the indifference thresholds, denoted by the p-dimensional vector $U = (u_1, u_2, \dots, u_p)$, indicate how small the differences must be for the DM to take them as marginal on a practical level (because of, for example, slight inaccuracies or uncertainties).

The concordance index is defined as

$$c(X_i, X_j) = \sum_{k=1}^{p} c_k(X_i, X_j), \tag{5}$$

where

$$c_k(X_i, X_j) = \begin{cases} w_k & \text{if } X_i \mathbf{P}_k X_j \vee X_i \mathbf{I}_k X_j, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $X_i \mathbf{P}_k X_j$ and $X_i \mathbf{I}_k X_j$ are respectively the logical functions of preference and indifference when evaluating the kth objective. Preference is defined as

$$\mathbf{P}_k = \left\{ (X_i, X_j) : f_k(X_i) < f_k(X_j) \wedge \neg X_i \mathbf{I}_k X_j \right\}, \tag{7}$$

where f_k is the evaluation function for the kth objective. Indifference is defined by

$$\mathbf{I}_k = \left\{ (X_i, X_j) : |f_k(X_i) - f_k(X_j)| \le u_k \right\}, \tag{8}$$

where u_k is the indifference threshold for the kth objective. Regarding the complexity analysis of the concordance, a computer program of Eq. (5) would belong to $\Theta(p)$ because the function c_k (Eq. (6)) is called p times, and its number of comparisons is constant.

The discordance index is calculated based on the two following sets of parameters:

(a) pre-veto thresholds, denoted by the vector $S = (s_1, s_2, s_3, \dots, s_p)$, indicating the magnitude of the differences when veto conditions begin to be observed, $s_k \ge u_k\ \forall k \in \{1,2,3,\dots,p\}$; and
(b) veto thresholds, denoted by the vector $V = (v_1, v_2, v_3, \dots, v_p)$, which indicates the magnitude of the differences between alternatives for triggering a veto condition, $v_k \ge s_k \forall k \in \{1,2,3,\dots,p\}$.

Thus, $d(X_i, X_j)$ is denoted by

$$d(X_i, X_j) = \min_{k \in \{1,2,3,\dots,p\}} \left\{ 1 - d_k(X_i, X_j) \right\}, \tag{9}$$

where $d_k(X_i, X_j)$ is defined as

$$d_k(X_i, X_j) = \begin{cases} 0 & \text{if } \nabla_k(X_i, X_j) < s_k, \\ \dfrac{\nabla_k(X_i, X_j) - u_k}{v_k - u_k} & \text{if } s_k \le \nabla_k(X_i, X_j) < v_k, \\ 1 & \text{otherwise,} \end{cases} \tag{10}$$

where $\nabla_k(X_i, X_j) = f_k(X_i) - f_k(X_j)$, v_k and s_k are the veto and pre-veto thresholds respectively. The discordance produces a rejection effect when there is at least one difference against X_i that exceeds v_k regardless of the value of the concordance coalition. On the computational complexity of the discordance, a program implementing Eq. (9) implies a complexity function in $\Theta(p)$ because the function d_k (whose number of comparisons is constant, see Eq. (10)) is called p times.

Table 1
Relational system of fuzzy preferences.

Object	Definition
Strict preference $(X_i \mathbf{P} X_j)$	$\mathbf{P} = \{ (X_i, X_j) : X_i \preceq X_j \ \vee \ \sigma(X_i, X_j) \geq \lambda \ \wedge \ \big(\sigma(X_j, X_i) < 0.5 \ \vee \ 0.5 \leq \sigma(X_j, X_i) < \lambda \ \wedge \ \sigma(X_i, X_j) - \sigma(X_j, X_i) \geq \beta \big) \}$
Indifference $(X_i \mathbf{I} X_j)$	$\mathbf{I} = \{ (X_i, X_j) : \sigma(X_i, X_j) \geq \lambda \ \wedge \ \sigma(X_j, X_i) \geq \lambda \ \wedge \ \lvert \sigma(X_i, X_j) - \sigma(X_j, X_i) \rvert \leq \epsilon \}$
Weak preference $(X_i \mathbf{Q} X_j)$	$\mathbf{Q} = \{ (X_i, X_j) : \sigma(X_i, X_j) \geq \lambda \ \wedge \ \sigma(X_i, X_j) \geq \sigma(X_j, X_i) \wedge \neg X_i \mathbf{P} X_j \ \wedge \ \neg X_i \mathbf{I} X_j \}$
Incomparability $(X_i \mathbf{R} X_j)$	$\mathbf{R} = \{ (X_i, X_j) : \sigma(X_i, X_j) < 0.5 \ \wedge \ \sigma(X_j, X_i) < 0.5 \}$
k – preference $(X_i \mathbf{K} X_j)$	$\mathbf{K} = \{ (X_i, X_j) : 0.5 \leq \sigma(X_i, X_j) < \lambda \ \wedge \ \sigma(X_i, X_j) - \sigma(X_j, X_i) > \frac{\beta}{2} \}$
Solutions strictly outranking X_i	$S(O, X_i) = \{ X_j \in O : X_j \mathbf{P} X_i \}$
The non-strictly outranked frontier	$NS(O) = \{ X_i \in O : S(O, X_i) = \emptyset \}$
Solutions weakly outranking X_i	$W(O, X_i) = \{ X_j \in NS(O) : X_j \mathbf{Q} X_i \vee X_j \mathbf{K} X_i \}$
The non-weakly outranked frontier	$NW(O) = \{ X_i \in NS(O) : W(O, X_i) = \emptyset \}$
The net flow-score of X_i	$F_n(X_i) = \sum_{X_j \in O} \big[\sigma(X_i, X_j) - \sigma(X_j, X_i) \big]$
Solutions with a net flow-score greater than X_i	$F(O, X_i) = \{ X_j \in NS(O) : F_n(X_j) > F_n(X_i) \}$
The net-flow non-outranked frontier	$NF(O) = \{ X_i \in NS(O) : F(O, X_i) = \emptyset \}$
The best compromise from O	$\arg\min_{X^* \in O} \big\{ \big(\lvert S(O, X^*) \rvert, \lvert W(O, X^*) \rvert, \lvert F(O, X^*) \rvert \big) \big\}$

The complexity of σ (Eq. (4)) is the sum of the complexity functions of the concordance and discordance, and both are in $\Theta(p)$; therefore, the complexity of σ is also in $\Theta(p)$.

Fernandez et al. (2015) proposed a relational system of preferences based on fuzzy outranking to identify the best compromise in a set of solutions O. Considering the parameters λ, β and ϵ, with $0 \leq \epsilon \leq \beta \leq \lambda \leq 1$ and $\lambda \geq 0.5$, Table 1 presents the mathematical objects of this outranking-based system.

Strict preference models the situation when the DM justifiably prefers X_i over X_j. Indifference models the situation when the DM perceives a high level of equivalence between X_i and X_j; hence, there is no preferred schedule. Weak preference models the situation when the DM hesitates between $X_i \mathbf{P} X_j$ and $X_i \mathbf{I} X_j$. Incomparability models the situation when the DM perceives a high degree of heterogeneity between X_i and X_j; hence, the DM cannot (or does not want to) set a preference. The condition of k-preference occurs when the DM hesitates between $X_i \mathbf{P} X_j$ and $X_i \mathbf{R} X_j$.

Let O be the set of feasible schedules. The subset of solutions that are not strictly outranked by any other schedule, $NS(O)$, is known as the non-strictly-outranked frontier. The subset of solutions that are not weakly outranked by any other schedule, $NW(O)$, is the non-weakly-outranked frontier.

Besides the strict and weak outranking preferences, the net-flow score is used as a preferential measure. Consequently, $F_n(X_i) > F_n(X_j)$ denotes a preference of X_i over X_j. Here, $NF(O)$ is the net-flow non-outranked frontier. Note that $NW(O) \subseteq NS(O)$, $NF(O) \subseteq NS(O)$, and $NS(O)$ is a subset of the Pareto-efficient solutions of O.

Hence, the best compromise solution, X^*, is the schedule whose sets $S(O, X^*)$, $W(O, X^*)$, and $F(O, X^*)$ have the minimum cardinality, with a preemptive priority in favour of $|S(O, X^*)|$. This definition of the best compromise is in compliance with the concept of Pareto dominance, which is a relevant property.

The strongest criticism of outranking models is the difficulty of finding the values of the parameters (i.e. λ, β, ϵ, and the thresholds U, S and V) that actually reflect the DM's preferences. This difficulty was solved in our work using a Preference Disaggregation Analysis (PDA) (Doumpos and Zopounidis, 2019). PDA has received increasing attention for addressing this challenging issue. Some recent studies have satisfactorily exploited and extended PDA to support a multicriteria analysis embedded in evolutionary optimisation (e.g. Cruz-Reyes et al., 2018; Rangel-Valdez et al., 2015; Fernandez et al., 2020, 2019b; Alvarez et al., 2018a,b). For real-world problems, the DM often requires a close interaction with a Decision Analyst (DA) to apply these PDA-based approaches adequately, e.g. to be able to reach acceptable settings even when the examples are noisy (Rangel-Valdez et al., 2018).

Regarding the complexity of the fuzzy system of preferences (Table 1), the most costly operations for calculating any relation between a pair of solutions are the outranking function σ (for **I**, **W**, **R** and **K**) and the Pareto dominance (included in the definition of **P**). Consequently, a program to determine any of these relations between a pair of solutions has a complexity function on the order of $\Theta(p)$. These relations have to be computed for all pairs $(X_i, X_j) \in O \times O$. Hence, the complexity of a computer program calculating the best compromise belongs to $\Theta(p|O|^2)$, where $|O|$ is the cardinality of the explored solution set.

3. The Outranking-based Particle Swarm Optimisation algorithm (O-PSO)

O-PSO incorporates the fuzzy system of preferences of the DM into EDPSO to identify the best compromise solution in the swarm and guide the iterative process towards an approximation of the true best compromise. In the following, the modelling of the particles is presented.

Let N be the number of tasks and M the number of machines. The vector $X_i = (X_{i,1}, X_{i,2}, X_{i,3}, \dots, X_{i,\mathbb{N}})$ represents the ith particle, where $\mathbb{N} = N + M - 1$, and $X_{i,l} \in \{*, 1, 2, 3, \dots, N\} \, \forall l \in \{1, 2, 3, \dots, \mathbb{N}\}$. Accordingly, the tasks are indicated by indices (integers), and each sequence separated by asterisks represents the set of tasks on the same machine. For example, $X_i = (6, 2, 4, *, 9, 5, 1, 8, *, 3, 7)$ is a possible schedule in an instance with nine tasks and three machines. Here, tasks 6, 2 and 4 are assigned to the first machine, tasks 9, 5, 1 and 8 to the second one, and tasks 3 and 7 to the last one. Each task is assigned according to the parameters describing the performance of each machine when processing the task. The whole swarm is $X = (X_1, X_2, X_3, \dots, X_n)$, where n is the size of the swarm. In accordance with Section 2.1, the multi-objective return of X_i is $f(X_i) = \big(f_1(X_i), f_2(X_i), f_3(X_i), \dots, f_p(X_i)\big)$ considering p objective functions. Because the way to calculate $f_k(X_i) \forall k \in \{1, 2, 3, \dots, p\}$ depends on the specific application domain, these definitions are presented within the case studies. Additionally, all the functions, relations, and sets of the preference system (Table 1) can be applied to the schedules of X.

The processes of O-PSO are described in Sections 3.1–3.5, and the main algorithm is presented in Section 3.6. Although Algorithms 1–4 are structured in an original way in this paper, they are greatly inspired by EDPSO. Lines written in blue in the algorithms highlight the processes of O-PSO to enrich the solutions with the DM preferences, and lines in black are the processes that are analogous to EDPSO.

3.1. Initial positions of the swarm

Three strategies are used to improve the quality and diversity of the initial solutions. The swarm's size is denoted by n, and it is split into three sub-swarms, with sizes n_1, n_2, and n_3 ($n = n_1 + n_2 + n_3$).

The first sub-swarm contains n_1 randomly generated schedules. Here, n_1 permutations of N tasks are generated and, for each particle,

$M-1$ asterisks are randomly inserted to differentiate task assignments to machines. This strategy aims to improve the exploration of the search space by improving the diversity. The computational complexity of generating a random permutation of $\mathbb{N}$ items belongs to $\Theta(\mathbb{N})$.

Algorithm 1: Full greedy algorithm for initial population

Input: M, N, p, the objective functions f_k, and weights w_k
Output: X_i ▷*generated schedule*
1 **Initialise:** $ul \leftarrow \{1,2,3,\dots,N\}$ ▷*the set of unscheduled tasks*
2 $X \leftarrow$ `EmptySchedule()` ▷*the schedule under construction*
3 **while** $ul \neq \emptyset$ **do**
4 Select a j from ul at random
5 **foreach** $h \in \{1,2,3,\dots,M\}$ **do**
6 $X_i' \leftarrow X_i$ ▷X_i' *is a copy of* X_i *before adding the* j*th task*
7 Program task j in schedule X_i' in machine h
8 $\mathrm{H}_h \leftarrow \sum_{k=1}^{p}\left[(w_k)\dfrac{f_k(X_i')-f_k(X_i)}{f_k(X_i)}\right]$ ▷*heuristic value*
9 $\hbar \leftarrow \arg\min_{1\le h\le M}\{\mathrm{H}_h\}$
10 Program task j in schedule X_i in machine $\hbar$
11 $ul \leftarrow ul \setminus \{j\}$

The second sub-swarm is composed of n_2 full greedy schedules generated by using Algorithm 1. First, a task is randomly selected from the set of unscheduled tasks (Line 4), and its effect on the objective functions is calculated for each machine (Lines 5–8). Here, an heuristic value that considers the weights of the objectives (H_h) measures the impact of scheduling the jth task in the hth machine (Line 8). Subsequently, the task is assigned to the machine with the best heuristic value (Lines 9 and 10). The process is repeated until all tasks have been scheduled (Lines 3 and 11). This strategy aims to generate solutions with an advantageous trade-off between the objectives. According to Algorithm 1, the most complex operation is to calculate H_h; therefore, the complexity function of the full greedy algorithm belongs to $\Theta(pMN)$.

Algorithm 2: Partial greedy algorithm for initial population

Input: M, N, p, k (an objective index), and the objective function f_k
Output: X_i ▷*generated schedule*
1 **Initialise:** $ul \leftarrow \{1,2,3,\dots,N\}$ ▷*the set of unscheduled tasks*
2 $X \leftarrow$ `EmptySchedule()` ▷*the schedule under construction*
3 **while** $ul \neq \emptyset$ **do**
4 Select a j from ul at random
5 **foreach** $h \in \{1,2,3,\dots,M\}$ **do**
6 $X_i' \leftarrow X_i$ ▷X_i' *is a copy of* X_i *before adding the* j*th task*
7 Program task j in schedule X_i' in machine h
8 $\eta_h^k \leftarrow \dfrac{f_k(X_i')-f_k(X_i)}{f_k(X_i)}$ ▷*heuristic value for the* k*th objective*
9 $\hbar \leftarrow \arg\min_{1\le h\le M}\{\eta_h^k\}$
10 Program task j in schedule X_i in machine $\hbar$
11 $ul \leftarrow ul \setminus \{j\}$

Finally, n_3 partially greedy schedules – generated through Algorithm 2 – compose the third sub-swarm. Unlike the previous technique, this approach considers each objective separately. The tasks are randomly selected, one by one, from the set of unscheduled tasks (Line 4) and are assigned to the machine with the best heuristic value η_h^k, which considers one objective alone (Lines 5–10). This strategy aims to generate extreme solutions in the search space. The complexity of Algorithm 2 is in $\Theta(NM)$ because the heuristic value is calculated for each pair (task, machine).

3.2. The update of the elitist set and the best schedules

An elitist set ($\mathcal{E}$) is implemented to preserve the particles that offer the best compromise in terms of the DM's preferences during the optimisation process. At the end of each iteration (including the initial cloud), the elitist group up to the ιth iteration is redefined as

$$\mathcal{E}^{\iota} \leftarrow \arg\min_{x\in O}\{(|S(O,x)|,|W(O,x)|,|F(O,x)|)\} \tag{11}$$

where $O = \mathcal{E}^{\iota-1} \cup X$, with X the current cloud, and $\mathcal{E}^0 = \emptyset$. O-PSO updates (at the end of each iteration) the global best schedule, $\mathcal{G}$, by picking randomly a solution from the elite group.

The best schedule known by the ith particle, $\mathcal{P}_i$, is updated using the concept of outranking in the following manner:

$$\mathcal{P}_i = \begin{cases} \mathcal{P}_i & \text{if } \sigma(\mathcal{P}_i, X_i) > \sigma(X_i, \mathcal{P}_i), \\ X_i & \text{otherwise,} \end{cases} \tag{12}$$

where X_i is the current position of the ith particle.

3.3. The calculation of the velocity

Because the solutions in the search space are discrete, the strategy that O-PSO uses for updating the velocity of the ith particle in the ιth iteration is

$$\vec{\mathrm{V}}_i^{\iota} = \mu \times \vec{\mathrm{V}}_i^{\iota-1} + c_1 \times (\mathcal{P}_i - X_i) + c_2 \times (\mathcal{G} - X_i), \tag{13}$$

where μ, c_1 and c_2 are weight parameters, $\vec{\mathrm{V}}_i^{\iota-1}$ is the velocity in the previous iteration, X_i is the current particle's position, and $-$, $\times$, and $+$ represent the subtraction, selection, and the and-or operator, respectively.

The subtraction operator ($-$) is used for learning from the best particles ($\mathcal{P}_i$ and $\mathcal{G}$). This operator filters (in a binary vector) the elements of the current particles that are equal to the elements of the best ones. For example, the operation $\mathcal{G} - X_i$ can be represented as

$$\mathcal{G} - X_i = \Delta^{\mathcal{G}} = \left(\Delta_1^{\mathcal{G}}, \Delta_2^{\mathcal{G}}, \Delta_3^{\mathcal{G}}, \dots, \Delta_{\mathbb{N}}^{\mathcal{G}}\right), \tag{14}$$

where $\mathbb{N} = N + M - 1$ is the size of the vectors, and

$$\Delta_j^{\mathcal{G}} = \begin{cases} 1 & \text{if } X_{i,j} = \mathcal{G}_j, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Similarly, $\mathcal{P}_i - X_i = \Delta^{\mathcal{P}_i} = \left(\Delta_1^{\mathcal{P}_i}, \Delta_2^{\mathcal{P}_i}, \Delta_3^{\mathcal{P}_i}, \dots, \Delta_{\mathbb{N}}^{\mathcal{P}_i}\right)$.

The selection operator ($\times$) randomly takes a proportion of the binary patterns in the vectors $\vec{\mathrm{V}}_i^{\iota-1}$, $\Delta^{\mathcal{G}}$ and $\Delta^{\mathcal{P}_i}$; the weights μ, c_1 and c_2 denote these proportions. The non-selected patterns take the opposite binary value. For instance,

$$c_2 \times \Delta^{\mathcal{G}} = {}^{c_2}\Delta^{\mathcal{G}} = \left({}^{c_2}\Delta_1^{\mathcal{G}}, {}^{c_2}\Delta_2^{\mathcal{G}}, {}^{c_2}\Delta_3^{\mathcal{G}}, \dots, {}^{c_2}\Delta_{\mathbb{N}}^{\mathcal{G}}\right), \tag{16}$$

where

$${}^{c_2}\Delta_j^{\mathcal{G}} = \begin{cases} \Delta_j^{\mathcal{G}} & \text{if } r \le c_2, \\ 1 - \Delta_j^{\mathcal{G}} & \text{otherwise,} \end{cases} \tag{17}$$

where r is a pseudorandom number, $r \sim \mathrm{U}(0,1)$. Analogously, $c_1 \times \Delta^{\mathcal{P}_i} = {}^{c_1}\Delta^{\mathcal{P}_i} = \left({}^{c_1}\Delta_1^{\mathcal{P}_i}, {}^{c_1}\Delta_2^{\mathcal{P}_i}, {}^{c_1}\Delta_3^{\mathcal{P}_i}, \dots, {}^{c_2}\Delta_{\mathbb{N}}^{\mathcal{G}}\right)$ and $\mu \times \vec{\mathrm{V}}_i^{\iota-1} = {}^{\mu}\vec{\mathrm{V}}_i^{\iota-1} = \left({}^{\mu}\vec{\mathrm{V}}_{i,1}^{\iota-1}, {}^{\mu}\vec{\mathrm{V}}_{i,2}^{\iota-1}, {}^{\mu}\vec{\mathrm{V}}_{i,3}^{\iota-1}, \dots, {}^{\mu}\vec{\mathrm{V}}_{i,\mathbb{N}}^{\iota-1}\right)$.

The and-or operator ($+$) combines the three parts of Eq. (13), updating the velocities of the particles to obtain new velocities based on the learned patterns. Each component of $\vec{\mathrm{V}}_i^{\iota} = \left(\vec{\mathrm{V}}_{i,1}^{\iota}, \vec{\mathrm{V}}_{i,2}^{\iota}, \vec{\mathrm{V}}_{i,3}^{\iota}, \dots, \vec{\mathrm{V}}_{i,\mathbb{N}}^{\iota}\right)$ is defined as

$$\vec{\mathrm{V}}_{i,j}^{\iota} = \begin{cases} f_{\mathrm{AND}}(i,j,\iota) & \text{if } r < 0.5, \\ f_{\mathrm{OR}}(i,j,\iota) & \text{otherwise ,} \end{cases} \tag{18}$$

where r is a pseudorandom number, $r \sim \mathrm{U}(0,1)$, and

$$f_{\mathrm{AND}}(i,j,\iota) = \begin{cases} 1 & \text{if } {}^{\mu}\vec{\mathrm{V}}_{i,j}^{\iota-1} = 1 \ \wedge \ {}^{c_1}\Delta_j^{\mathcal{P}_i} = 1 \ \wedge \ {}^{c_2}\Delta_j^{\mathcal{G}} = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

and

$$f_{\mathrm{OR}}(i,j,\iota) = \begin{cases} 1 & \text{if } {}^{\mu}\overline{\mathrm{V}}_{i,j}^{\iota-1} = 1 \ \vee \ {}^{c_1}\Delta_j^{\mathcal{P}_i} = 1 \ \vee \ {}^{c_2}\Delta_j^{\mathcal{G}} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

The parameters μ, c_1 and c_2 are critical for keeping an appropriate balance between exploitation and exploration ($0 < \mu, c_1, c_2 \leq 1$). High values of c_2 intensify the search around the global best solution ($\mathcal{G}$), and high values of c_1 around local optima ($\mathcal{P}_i$). Low values of μ avoid revisiting solution patterns recently generated by the same particle.

The complexity of a program that computes the velocity according to Eq. (13) belongs to $\Theta(\mathbb{N})$ because it is only composed of non-nested operations ($-$, $\times$ and $+$) that also belong to $\Theta(\mathbb{N})$ (See Eqs. (14), (16) and (18)).

3.4. Updating the positions

To model the movement of the swarm in the ιth iteration, O-PSO uses the equation

$$X_i^{\iota} = X_i^{\iota-1} \oplus \overline{\mathrm{V}}_i^{\iota} \quad \forall i \in \{1,2,3,\ldots,n\} \tag{21}$$

where n is the size of the swarm, $X_i^{\iota-1}$ is the previous position of the ith particle, $\overline{\mathrm{V}}_i^{\iota}$ is its current velocity, and $\oplus$ is the map operator which splits the particle indices into two subsets, $\mathbb{C}_i^{\iota} = \left\{ j \in \{1,2,3,\ldots,\mathbb{N}\} : \overline{\mathrm{V}}_{i,j}^{\iota} = 0 \right\}$ and its complement, $\left(\mathbb{C}_i^{\iota}\right)^{\complement} = \{1,2,3,\ldots,\mathbb{N}\} \setminus \mathbb{C}_i^{\iota}$. Let $\mathbb{P}_i^{\iota}$ be a random permutation of $\mathbb{C}_i^{\iota}$, and $\mathtt{pop}\left(\mathbb{P}_i^{\iota}\right)$ be a function that takes in sequence the elements from $\mathbb{P}_i^{\iota}$, following a queue policy. The position of the ith particle in the ιth iteration is

$$X_{i,j}^{\iota} = \begin{cases} X_{i,j}^{\iota-1} & \text{if } j \in \left(\mathbb{C}_i^{\iota}\right)^{\complement}, \\ X_{i,\mathrm{pop}(\mathbb{P}_i^{\iota})}^{\iota-1} & \text{otherwise,} \end{cases} \quad \forall j \in \{1,2,3,\ldots,\mathbb{N}\}. \tag{22}$$

Analysing the computational complexity of Eq. (21), the three main sequential operations are: (a) the calculation of the velocity, with a complexity function in $\Theta(\mathbb{N})$; (b) the binary classification of the indices, whose complexity is also $\Theta(\mathbb{N})$; and, (c) the generation of a permutation of the indices in $\mathbb{C}_i^{\iota}$, whose complexity (worst case) is in $\mathrm{O}(\mathbb{N})$. Thus, a program repositioning a particle belongs to $\Theta(\mathbb{N})$.

3.5. The local search

Each iteration, O-PSO performs a local search, referred to as Λ-swap, on the elite set. This search consists of exchanging the positions between pairs of components of the elite particles. Algorithm 3 describes Λ-swap. Here, the main loop depends on the cardinality of $\mathcal{E}$. Although $|\mathcal{E}| \ll \mathbb{N}$, assume that $|\mathcal{E}| = \mathbb{N}$, as the worst case. According to Lines 6–12, six basic operations are repeated Λ times, where $\Lambda = \lfloor \log_2 \mathbb{N} \rfloor$. Thus, the complexity of the while-loop (Lines 6–12) is in $\mathrm{O}(\log_2 \mathbb{N})$, and the complexity of the for-loop (Lines 2–13) belongs to $\mathrm{O}\left(\mathbb{N} \log_2 \mathbb{N}\right)$. Another time consuming operation is to identify the best compromise (Line 15), which belongs to $\mathrm{O}(\mathbb{N}^2 p)$. Accordingly, the complexity of Λ-swap involves two sequential parts: the first belonging to $\mathrm{O}\left(\mathbb{N} \log_2 \mathbb{N}\right)$, and the second one to $\mathrm{O}(\mathbb{N}^2 p)$. Therefore, the resulting time complexity function is in $\mathrm{O}\left(\mathbb{N}^2 p\right)$, where $\mathbb{N} = M + N - 1$ (M is the number of machines, and N is the number of tasks); so, it is expressed in terms of the input size.

3.6. The outranking-based particle swarm optimisation algorithm

Algorithm 4 presents an algorithmic outline of O-PSO. Line 1 initialises the main variables, and the initial positions are computed in Lines 2–5 (n_1 random schedules, n_2 full-greedy schedules, and n_3 partial greedy schedules). In Line 5, `mod` stands for the modulus operator (the remainder after dividing one number by another), and is only used to take the indices of the objectives sequentially.

Algorithm 3: The local search Λ-swap

Input: $\mathcal{E}$ ▷*original elite set*
Output: $\mathcal{E}^*$ ▷*improved elite set*
1 **Initialise:** $\Lambda \leftarrow \lfloor \log_2 \mathbb{N} \rfloor$, $\mathcal{E}' \leftarrow \emptyset$
2 **foreach** $X_i \in \mathcal{E}$ **do**
3 $S_1 \leftarrow \emptyset$
4 $S_2 \leftarrow \emptyset$
5 $X_i' \leftarrow X_i$ ▷*X_i' is a copy of X_i*
6 **while** $|S_1 \cup S_2| \leq 2\Lambda$ **do**
7 Select a $j \in \{1,2,3,\ldots,\mathbb{N}\} \setminus (S_1 \cup S_2)$ at random
8 $S_1 \leftarrow S_1 \cup \{j\}$
9 Select a $\jmath \in \{1,2,3,\ldots,\mathbb{N}\} \setminus (S_1 \cup S_2)$ at random
10 $S_2 \leftarrow S_2 \cup \{\jmath\}$
11 $X_{i,j}' \leftarrow X_{i,j}$
12 $X_{i,\jmath}' \leftarrow X_{i,\jmath}$
13 $\mathcal{E}' \leftarrow X_i'$
14 $O \leftarrow \mathcal{E} \cup \mathcal{E}'$
15 $\mathcal{E}^* \leftarrow \arg\min_{x \in O} \left\{ \left(|S(O,x)|, |W(O,x)|, |F(O,x)|\right) \right\}$

Algorithm 4: Outranking-based Particle Swarm Optimisation

Input: M, N, p, and f_k $\forall k \in \{1,2,3,\ldots,p\}$ ▷*Data of the problem*
Output: $\mathcal{E}$ ▷*An approximation of the best compromise*
1 **Initialise:** $\iota \leftarrow 1$, $\mathcal{E} \leftarrow \emptyset$,
$\overline{\mathrm{V}}_{i,j} \leftarrow 0 \ \ \forall i \in \{1,2,3,\ldots,n\}, j \in \{1,2,3,\ldots,\mathbb{N}\}$
▷*Generating initial positions, Lines 2–5*
2 Let X be a set of n "empty" particles
3 $X_i \leftarrow \mathtt{generate\_schedule\_at\_random}(N,M) \ \ \forall i \in [1, n_1]$
4 $X_i \leftarrow \mathtt{full\_greedy}(N,M) \ \ \forall i \in (n_1, n_1+n_2]$ ▷*Algorithm 1*
5 $X_i \leftarrow \mathtt{partial\_greedy}(N,M,1+i \bmod p) \ \ \forall i \in (n_1+n_2, n]$
▷*Algorithm 2*
6 $\mathcal{E} \leftarrow \arg\min_{x \in X} \left\{ \left(|S(X,x)|, |W(X,x)|, |F(X,x)|\right) \right\}$ ▷*The elite group*
7 $\mathcal{E} \leftarrow \Lambda\_\mathtt{swap}(\mathcal{E})$ ▷*Algorithm 3, the local search*
8 $\mathcal{G} \leftarrow \mathtt{select\_at\_random}(\mathcal{E})$ ▷*Updating the global best solution*
9 $\mathcal{P}_i \leftarrow X_i \ \ \forall i \in \{1,2,3,\ldots,n\}$ ▷*Updating the local best solutions*
10 **forall** $1 \leq \iota \leq iter_{max}$ **do**
▷*Updating velocities through Eq. (13)*
11 $\overline{\mathrm{V}}_i \leftarrow \mu \times \overline{\mathrm{V}}_i + c_1 \times (\mathcal{P}_i - X_i) + c_2 \times (\mathcal{G} - X_i) \ \ \forall i \in \{1,2,3,\ldots,n\}$
12 $X_i \leftarrow X_i \oplus \overline{\mathrm{V}}_i \ \forall i \in \{1,2,3,\ldots,n\}$ ▷*Updating positions through Eq. (21)*
13 $O \leftarrow \mathcal{E} \cup X$
14 $\mathcal{E} \leftarrow \arg\min_{x \in O} \left\{ \left(|S(O,x)|, |W(O,x)|, |F(O,x)|\right) \right\}$
15 $\mathcal{E} \leftarrow \Lambda\_\mathtt{swap}(\mathcal{E})$ ▷*Algorithm 3, the local search*
16 $\mathcal{G} \leftarrow \mathtt{select\_at\_random}(\mathcal{E})$ ▷*Updating the global best solution*
▷*Updating the local best solutions through Eq. (12)*
17 $\mathcal{P}_i \leftarrow \begin{cases} \mathcal{P}_i & \text{if } \sigma(\mathcal{P}_i, X_i) > \sigma(X_i, \mathcal{P}_i), \\ X_i & \text{otherwise,} \end{cases} \quad \forall i \in \{1,2,3,\ldots,n\}$

Subsequently, Line 6 assesses the quality of the initial solutions, identifying the elite set, which is submitted to the Λ-swap algorithm (Line 7). Lines 8 and 9 identify the first leading particles, $\mathcal{P}_i$, and $\mathcal{G}$.

The main loop of O-PSO is presented in Lines 10–17. Here, the first step is to compute the velocity of each particle (Line 11) and, by this means, relocate the swarm (Line 12). Accordingly, the elite set is updated (Lines 13 and 14), and Λ-swap tries to improve the quality of this solution set (Line 15). Finally, the leading particles for the next

iteration are identified in Lines 16 and 17. These steps are repeated $iter_{\max}$ times. At the end of this process, $\mathcal{E}$ contains the best compromise solution(s).

Considering M, N, and p as the parameters defining the input size, the most complex operations are the full greedy algorithm, $O(MNp)$, and the local search, $O(\mathbb{N}^2 p)$; considering $\mathbb{N} = M+N-1$, the complexity function of O-PSO belongs to $O(\mathbb{N}^2 p)$.

4. Experimental results

We implemented O-PSO, which was developed using Python 3.7, on a computer with an AMD FX-8800P R7 processor with 8 GB of RAM, and Microsoft Windows 10 as operating system[1]. In this section, we present the numerical results structured as follows: Section 4.1 presents the application of the algorithm to a real-world case study of a freight company. Section 4.2 analyses the impact on performance of incorporating the relational system of fuzzy outranking preferences. Section 4.3 compares the efficiency of O-PSO with an relevant state-of-the-art metaheuristic for multi-objective UPMSP optimisation.

4.1. Real-world case study: Freight transport scheduling

Here, the traffic coordinator is responsible for programming the transportation schedule for a freight company every single day. This DM faces a reinterpretation of the multi-objective UPMSP. Analogously, (s)he has a set M of truck operators (drivers) instead of machines and a set N of cargo deliveries instead of tasks. Then, the question is "how should (s)he assign freights to truck operators in an optimal way?" The notion of optimality strongly depends on the impact of the multicriteria return of the schedules, which should be assessed by the DM to select the best one. On the whole, the DM considers the skillfulness of each operator at delivering each type of freight, which can possess different features: weight, fragility, dimensions of the required truck, among others. The DM takes the following three criteria ($p = 3$) into account to select the best schedule:

- Completion time: The estimated time required for the truck operators to complete all the deliveries in parallel. The completion time of a schedule X_i is

$$f_1(X_i) = \max_{1\le h\le M}\left\{\mathrm{T}_h(X_i)\right\}, \tag{23}$$

where $\mathrm{T}_h(X_i)$ is the time the hth operator requires to complete his/her assignments, hence,

$$\mathrm{T}_h(X_i) = \sum_{\ell=1}^{N} \mathrm{T}_h^{\ell}(X_i), \tag{24}$$

where

$$\mathrm{T}_h^{\ell}(X_i) = \begin{cases} \tau_{h,\ell} & \text{if } \varphi(X_i,\ell) = h, \\ 0 & \text{otherwise,} \end{cases} \tag{25}$$

where $\tau_{h,\ell}$ is the time the hth operator requires to complete the ℓth delivery of freight ($1 \le h \le M$, $1 \le \ell \le N$), and $\varphi(X_i,\ell)$ is a function that returns the index of the operator who has the ℓth freight delivery programmed in schedule X_i. In this study, the matrix τ is approximated through linear regression on the data from the latest year, provided by the company. The independent variables are the type of truck, the volume of the load, and the travel time calculated by the Google Maps API to visit the following points in order: company (departure), origin (client), destination (client), and company (return). The coefficients of the linear function are calculated ad hoc for each operator.

- Delivery reliability: This second criterion is related with the certainty that the operators will deliver the products satisfactorily. To assess this objective, the central tendency is calculated as follows:

$$f_2(X_i) = \underset{1\le\ell\le N}{\text{median}}\left\{\mathrm{F}\left(\varphi\left(X_i,\ell\right)\right)\right\}, \tag{26}$$

where $\mathrm{F}\left(\varphi\left(X_i,\ell\right)\right)$ is the number of times that the assigned operator has completed satisfactorily deliveries of the ℓth type of freight. This count is calculated by an SQL query on the database of the latest year. The type of freight is the conjunction of the client that requested the service, the type of truck, and the volume of the load.

- Risk: The likelihood that a schedule will be interrupted, which is estimated as follows:

$$f_3(X_i) = \max_{1\le\ell\le N}\left\{\mathrm{R}\left(\varphi\left(X_i,\ell\right)\right)\right\}, \tag{27}$$

where $\mathrm{R}\left(\varphi\left(X_i,\ell\right)\right)$ is the proportion of the assigned operator's deliveries of the ℓth type of freight that were not satisfactorily completed.

These cases of unsatisfactory deliveries may be the consequence of traffic offences, or clients' complaints. Again, this proportion is calculated by an SQL query on the database from the latest year.

The first and third objectives are minimising, and the second one is maximising. For the sake of simplicity, the best compromise schedule is obtained by applying the system of Table 1 to the Pareto frontier, defined as

$$\min_{X\in RF}\left\{\left(f_1(X), -f_2(X), f_3(X)\right)\right\}, \tag{28}$$

where RF is the combinatorial space of feasible solutions. The postulate of economic rationality says that the DM should choose the X^* with the best balance between f_1, f_2, and f_3, to achieve the organisation's goals. Because of its computational complexity, the task of identifying the optimum is extremely difficult, even impossible for large scale instances. So, O-PSO is an adequate algorithm to approximate X^*, supporting the decision making about schedules.

To apply O-PSO, the first step is to find the parameter settings for the outranking model. We used the PDA-based approach proposed by Fernandez et al. (2019b) on a set of 10 archived schedules. The DM expressed their preferences for each pair of these schedules. This information about the preferences was used as input in the PDA to find the parameter values. The parameter values suggested by the PDA are the following: $W = (0.41, 0.23, 0.36)$, $U = (11, 2, 1)$, $S = (95, 17, 8)$, $V = (145, 25, 12)$, $\epsilon = 0.08$, $\beta = 0.13$, and $\lambda = 0.67$.

The parameter values for O-PSO, taken from Wang et al. (2018), are $\mu = 0.5$, $c_1 = 0.5$, $c_2 = 0.5$, $n = 1000$, and $iter_{\max} = 500$. They are used for all the experiments in this paper. To illustrate how O-PSO works, we present a numerical example of a case study with 10 truck operators and 30 types of freight. Fig. 1 plots the solutions of the last iteration of O-PSO and the reference schedule.

In Fig. 1, the Region of Interest (RoI) obtained in this run is composed of the solutions satisfying the system of preferences. According to the outranking model, all the solutions in the RoI are indifferent to each other. This means that these solutions are highly equivalent (under the assumption that the model actually reflects the DM's preferences). This set of 5 solutions is presented to the DM. The best compromise solution chosen is $X^*_{\text{O-PSO}}$, with the values of the objectives $f(X^*_{\text{O-PSO}}) = (388, -474, 7)$. Then, the best compromise was compared with the schedule obtained via the current policy, X^*_{ref}, which has the image $f(X^*_{\text{ref}}) = (720, -470, 8)$.

Let us compare both solutions taking each objective at a time in order of importance (weight). The difference in completion time is (f_1) 332, which is greater than the veto threshold ($v_1 = 145$). Hence, as a consequence of Eqs. (9) and (10), $\sigma\left(X^*_{\text{ref}}, X^*_{\text{O-PSO}}\right) = 0$, which allows us to state this first remark: "X^*_{ref} is not as good as $X^*_{\text{O-PSO}}$". Regarding

[1] The source code is available at: https://github.com/GibranPorras/O-PSO.git

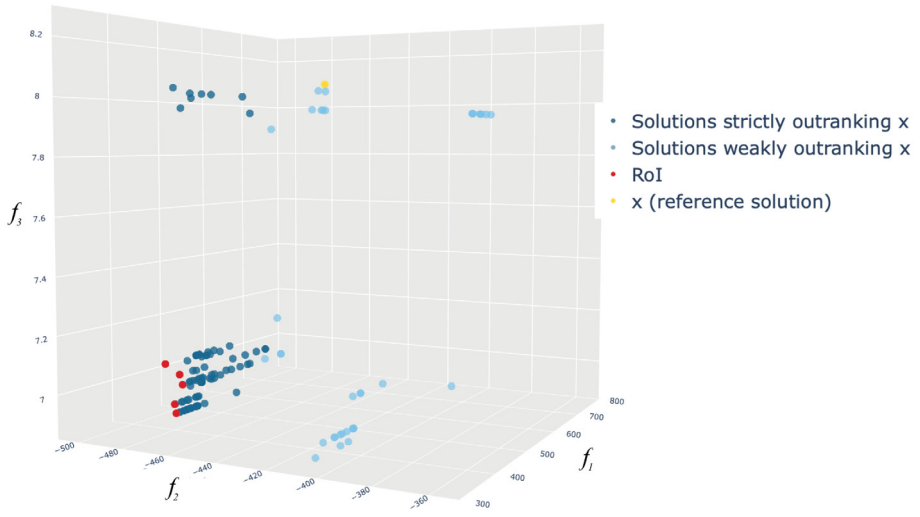


Fig. 1. 3D representation of the results for one instance of the case study, with $N = 30$ and $M = 10$.

Table 2
Results on instances of the real-world case study.

Instance	N	M	Runs with solutions in the A-RoI
1	30	10	10
2	30	8	12
3	25	7	9
4	30	8	10
5	26	7	11
6	28	6	12

the associated risk (f_3), the difference is just equal to the indifference threshold ($u_3 = 1$). Consequently, there is no important difference in risk, considering the imprecision expressed by the DM in the fuzzy outranking system. Finally, the difference in delivery reliability (f_2) is 4; this improvement is taken into account by the DM, although it is still insufficient to veto schedule X^*_{ref} (recall the related indifference and pre-veto thresholds, $u_2 = 2$ and $s_2 = 17$). According to Eqs. (5)–(8), $\sigma\left(X^*_{\text{O-PSO}}, X^*_{\text{ref}}\right) = 1$, leading to this second remark: "$X^*_{\text{O-PSO}}$ is at least as good as X^*_{ref}". In these circumstances, $X^*_{\text{O-PSO}}$ is strictly preferred to X^*_{ref} (see Table 1).

It is worth stressing that any solution from the RoI also dominates the reference schedule in the Paretian sense. This constitutes the most important argument with which the DM can justify using O-PSO instead of the current policy based on queues. Fig. 1 also presents the O-PSO solutions that are better than the reference in terms of fuzzy relations. Similar behaviour was observed in all the instances of this case study.

To validate the efficiency of O-PSO, the algorithm was run 30 times over 6 instances of the case study, whose features are described in Table 2. Because metaheuristic algorithms are stochastic approaches, the results vary from run to run. We approximated the true RoI for each instance by joining the 30 solution sets and applying the definition of the best compromise. This approximate RoI (A-RoI) is used as a measure of the O-PSO's variability in efficiency.

Table 2 shows the data for 6 real-world instances as well as the number of runs in which O-PSO generated solutions in the A-RoI. The instances consist of the number of cargo deliveries (N) and the number of truck operators (M). The data for each instance corresponds to a different day of the week. This experiment indicates that the DM could conduct a new execution of the algorithm if (s)he is not confident about a single run. On average, we suggest up to 6 runs. More runs did not contribute to making the A-RoI any denser. The run time of O-PSO ranged from 36 s to 95 s on these instances, with a mean of 52 s.

4.2. On the impact of incorporating fuzzy outranking preferences

Here, the aim is to know how O-PSO gets the edge by increasing the selective pressure towards the RoI. Unlike the majority of multi-objective metaheuristics, O-PSO is not intended to approximate uniformly distributed samples of the Pareto frontier. Hence, most metrics are not adequate to measure its performance (e.g. spread, spacing, and hypervolume). Because the RoI is a subset of the Pareto frontier, O-PSO would be competitive if it generates solutions that are close enough to the true RoI for a wide range of input instances.

Wang et al. (2018) addressed a scheduling problem very similar to this real-world case study (Section 4.1), providing a benchmark for comparison. Their proposal outperforms other multi-objective algorithms that are standard approaches for multi-objective optimisation with 2–4 objective functions. These instances have two minimising objectives: the completion time (as in Eq. (23)), and the total cost, which is the number of containers of raw material required by the schedule. This number is calculated as a function of the sum of the consumption of each hth machine to process each ℓth task. With more formality, $f_2(X_i)$ is given by

$$f_2(X_i) = \sum_{h=1}^{M} \left\lceil \frac{\sum_{\ell=1}^{N} c_h^{\ell}(X_i)}{\mathbb{C}} \right\rceil, \tag{29}$$

where $\mathbb{C}$ is the volume of raw material per container, and $c_h^{\ell}(X_i)$ is defined as

$$c_h^{\ell}(X_i) = \begin{cases} \varpi_{h,\ell} & \text{if } \varphi(X_i, \ell) = h, \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

where $\varpi_{h,\ell}$ is the quantity of raw material the hth machine consumes for processing the ℓth task, and $\varphi(X_i, \ell)$ is the function that returns the index of the machine that has the ℓth task programmed in schedule X_i.

Additionally, Wang et al. (2018) proposed 45 synthetic instances, each one having a combination of the following features:

- Number of machines M: 5, 10 and 15.
- Number of tasks N: 100, 200 and 500.
- Ranges for processing time and consumption of raw material:
  - Class 1 ($\tau_{h,\ell} \sim \text{U}(0, 25]$, $\varpi_{h,\ell} \sim \text{U}(0, 25]$),
  - Class 2 ($\tau_{h,\ell} \sim \text{U}(0, 50]$, $\varpi_{h,\ell} \sim \text{U}(0, 50]$),
  - Class 3 ($\tau_{h,\ell} \sim \text{U}(0, 100]$, $\varpi_{h,\ell} \sim \text{U}(0, 100]$),
  - Class 4 ($\tau_{h,\ell} \sim \text{U}(0, 200]$, $\varpi_{h,\ell} \sim \text{U}(0, 200]$), and
  - Class 5 ($\tau_{h,\ell} \sim \text{U}(0, 500]$, $\varpi_{h,\ell} \sim \text{U}(0, 500]$).

Table 3
Coverage and approximation to the RoI for small-size instances. Comparison with EDPSO.

Inst.	N	M	Class	Coverage		Solutions in the A-RoI	
				O-PSO over EDPSO	EDPSO over O-PSO	O-PSO	EDPSO
1	100	5	1	52%	49%	85%	90%
2	100	5	2	10%	51%	100%	85%
3	100	5	3	56%	53%	100%	100%
4	100	5	4	71%	33%	50%	0%
5	100	5	5	41%	64%	40%	0%
6	100	10	1	50%	45%	93%	85%
7	100	10	2	58%	44%	85%	73%
8	100	10	3	50%	50%	100%	100%
9	100	10	4	43%	67%	100%	95%
10	100	10	5	49%	51%	0%	45%
11	100	15	1	62%	48%	95%	100%
12	100	15	2	64%	43%	85%	50%
13	100	15	3	54%	55%	95%	100%
14	100	15	4	40%	61%	50%	73%
15	100	15	5	42%	68%	0%	30%

Table 4
Coverage and approximation to the RoI for medium-size instances. Comparison with EDPSO.

Inst.	N	M	Class	Coverage		Solutions in the A-RoI	
				O-PSO over EDPSO	EDPSO over O-PSO	O-PSO	EDPSO
16	200	5	1	53%	49%	90%	81%
17	200	5	2	55%	45%	80%	45%
18	200	5	3	61%	47%	87%	92%
19	200	5	4	47%	52%	45%	0%
20	200	5	5	76%	26%	50%	44%
21	200	10	1	65%	41%	96%	90%
22	200	10	2	63%	46%	84%	74%
23	200	10	3	50%	45%	93%	86%
24	200	10	4	57%	50%	60%	0%
25	200	10	5	76%	21%	75%	0%
26	200	15	1	59%	43%	100%	88%
27	200	15	2	67%	40%	92%	81%
28	200	15	3	51%	48%	81%	92%
29	200	15	4	38%	67%	86%	89%
30	200	15	5	61%	41%	45%	0%

Table 5
Coverage and approximation to the RoI for large-size instances. Comparison with EDPSO.

Inst.	N	M	Class	Coverage		Solutions in the A-RoI	
				O-PSO over EDPSO	EDPSO over O-PSO	O-PSO	EDPSO
31	500	5	1	42%	52%	92%	94%
32	500	5	2	61%	40%	95%	30%
33	500	5	3	59%	34%	90%	50%
34	500	5	4	86%	15%	49%	0%
35	500	5	5	88%	12%	50%	0%
36	500	10	1	57%	28%	97%	89%
37	500	10	2	58%	49%	84%	91%
38	500	10	3	50%	45%	91%	85%
39	500	10	4	69%	33%	73%	38%
40	500	10	5	83%	17%	50%	0%
41	500	15	1	62%	41%	87%	96%
42	500	15	2	53%	47%	100%	0%
43	500	15	3	53%	40%	96%	72%
44	500	15	4	77%	26%	48%	43%
45	500	15	5	73%	27%	47%	0%

We generated 45 instances following these features, classified according to the number of tasks as 'small', 'medium', and 'large'[2]. In Tables 3–5, we present the averages of 30 runs of O-PSO and EDPSO for each instance size.

Here, the first measure of quality is coverage: the coverage of an algorithm A over another B is the proportion of solutions from B dominated by those from A. High values of coverage mean being closer to the true Pareto frontier in that region. The second measure is the proportion of solutions provided by runs that are in the A-RoI. High values mean a better reaching of solutions according to the DM's preferences. We calculate the A-RoI per instance by combining the solution sets from 30 runs of EDPSO, T-RIPG, and O-PSO; and then selecting from this union those solutions satisfying the relational system of fuzzy preferences.

A comparison between EDPSO and O-PSO leads us to appreciate the impact of incorporating the DM preferences: Tables 3–5 present these results, where values that are statistically significant are in dark grey when the differences are in favour of O-PSO, and in light grey when in favour of EDPSO. From here on, the term 'significant' implies a Wilcoxon rank-sum test with a 0.95-confidence interval.

For the 100-size benchmark, the coverage measure in 6 instances (1, 3, 6, 8, 10 and 13) was similar, in 6 instances (2, 5, 7, 9, 14 and 15) EDPSO was better, and in 3 (4, 11 and 12) O-PSO was better. On the 15 small instances, the difference in coverage (in favour of EDPSO) was statistically significant. However, finding a sample of the Pareto frontier is only a step to solving this problem in practice. The DM has to implement one solution according to their preferences: so it is more important to find the best compromise solution than to find many Pareto-efficient solutions without evidence that they match the DM's preferences. According to Table 3, the approximation to the RoI was similar in 11 instances (1–3, 6–9, 11–13 and 15). In two instances (10 and 14), EDPSO provided solutions that match the DM preferences better than O-PSO; and O-PSO offered solutions that are better – in terms of outranking relations – in two instances (4 and 5). On the complete 100-size benchmark, there is no significant difference in terms of approximation to the RoI. Consequently, at this scale, both metaheuristics have the same efficiency in practice (with a plausible trend favouring EDPSO).

Table 4 presents the results on the medium size benchmark. In 8 instances (18, 20–22, 25–27 and 30), O-PSO had better coverage; in 6 instances (16, 17, 19, 23, 24 and 28), the coverage measures were similar (non-significant differences); and, in only 1 instance (29), EDPSO was better. The difference in coverage was statistically significant considering the 15 medium instances. In terms of preferences, both algorithms provided similar solutions in 10 instances (16, 18, 20–23 and 26–29); in 5 instances (17, 19, 24, 25 and 30), O-PSO provided solutions matching the DM's preferences better than EDPSO; and, on the complete 200-size benchmark, the difference in favour of O-PSO was significant. Therefore, O-PSO behaves slightly better than EDPSO at this scale (with this trend statistically tested).

Table 5 presents the results on the 500-size benchmark: here, O-PSO clearly outperforms EDPSO. In 10/15 instances (32–36, 39–41, 44 and 45), O-PSO provided better coverage than EDPSO; and, in 8/15 instances (32–35, 39, 40, 42 and 45), O-PSO offered solutions matching the DM's preferences better than EDPSO. In the rest of the cases, the measures of O-PSO were not inferior to those of EDPSO. On the 15 large instances, the differences were significant in both quality measures. Hence, we strongly suggest the outranking-based metaheuristic to treat this kind of problem at such a scale.

Additionally, Fig. 2 plots the average time of both PSO algorithms on the 45 instances: O-PSO always ran faster than EDPSO, and the difference in run time was significant. On average, O-PSO only requires 75% of the time required by EDPSO. Even though both metaheuristics have the same computational complexity in terms of big-O (cf. Wang et al., 2018), O-PSO ran faster than EDPSO as a consequence of intensifying the search towards a particular solution – the best compromise, which belongs to a relatively small set of schedules (the RoI) – instead of approximating a sample of the complete Pareto frontier. Hence,

[2] The instances can be downloaded from: https://github.com/GibranPorras/O-PSO/tree/main/instancias

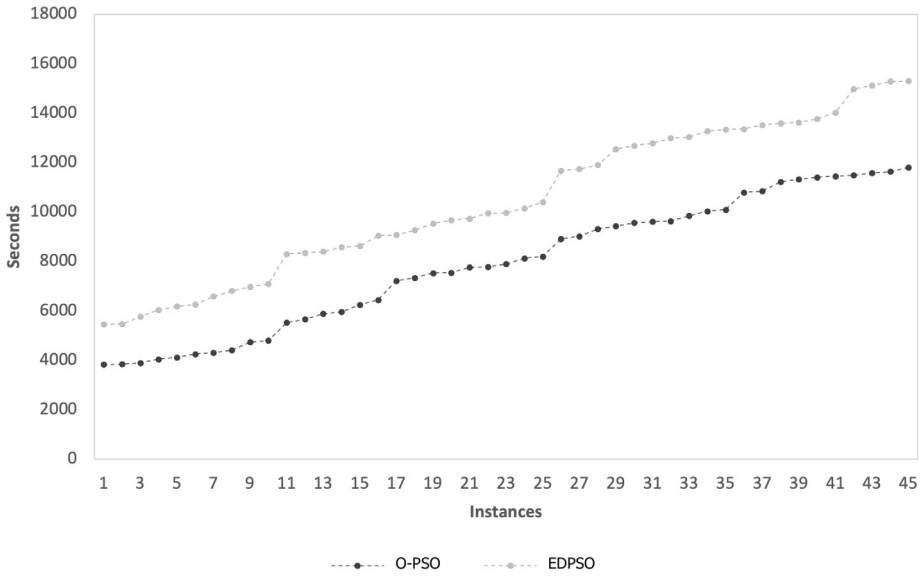


Fig. 2. Average run time on the 45 synthetic instances.

a lesser number of pair-wise comparisons per iteration are made to identify the efficient schedules.

4.3. On the efficiency of O-PSO

Section 4.1 provides evidence in favour of O-PSO at the application level. However, more experiments are needed to validate the competitiveness of this approach in terms of the state of the art. Here, O-PSO is compared with T-RIPG (Yepes-Borrero et al., 2021), a recently published metaheuristic based on the iterated Pareto greedy algorithm for treating a bi-objective UPMSP. According to Yepes-Borrero et al. (2021), T-RIPG performs better in terms of hypervolume than other algorithms facing similar multi-objective UPMSPs. Here, T-RIPG is used to solve the instances described in Section 4.2, approximating the Pareto frontier with the right level of closeness according to the specialised literature. T-RIPG was run 30 times per instance with the parameter setting reported by Yepes-Borrero et al. (2021) ($d = 4$, $p = 1$, $\ell = 50$ and $q = 50$). Both metaheuristic algorithms were set to stop when 500,000 evaluations of the multi-objective function are reached; this setting places them on an equal footing.

Table 6 presents the results for the 100-size benchmark, which can be summarised as follows:

- the coverage measure in 10 instances (1–3, 6, 8, 10, 11, and 13–15) was statistically indistinguishable; in 3 instances (5, 7, and 9) T-RIPG was better, and in 2 (4 and 12), O-PSO was better;
- on the 15 small instances, the difference in coverage was not statistically significant;
- the approximation to the RoI was similar in 10 instances (1, 3, 6–9 and 11–14), O-PSO provided solutions that match the DM preferences better than T-RIPG in 2 instances (2 and 4), and T-RIPG offered solutions that are better – in terms of outranking – in 3 instances (5, 10 and 15); and
- on the complete 100-size benchmark, there is no significant difference in terms of approximation to the RoI; consequently, at this scale, both metaheuristics have a similar degree of efficiency.

Table 7 presents the results on the medium size benchmark, which can be summarised in the following points:

- O-PSO had better coverage in 5 instances (20–22, 25, and 27), the coverage measures were similar (non-significant differences) in 8 instances (16–18, 23, 24, 26, 28, and 30), and T-RIPG was better in only 2 instances (19 and 29);

Table 6
Coverage and approximation to the RoI for small-size instances. Comparison with T-RIPG.

Inst.	N	M	Class	Coverage		Solutions in the A-RoI	
				O-PSO over T-RIPG	T-RIPG over O-PSO	O-PSO	T-RIPG
1	100	5	1	9%	12%	85%	79%
2	100	5	2	11%	8%	100%	43%
3	100	5	3	16%	10%	100%	100%
4	100	5	4	13%	0%	50%	19%
5	100	5	5	3%	9%	40%	62%
6	100	10	1	14%	10%	93%	95%
7	100	10	2	8%	16%	85%	74%
8	100	10	3	10%	10%	100%	100%
9	100	10	4	7%	20%	100%	91%
10	100	10	5	0%	0%	0%	26%
11	100	15	1	2%	5%	95%	87%
12	100	15	2	24%	7%	85%	68%
13	100	15	3	14%	10%	95%	100%
14	100	15	4	7%	3%	50%	52%
15	100	15	5	2%	0%	0%	41%

- the difference in coverage was statistically significant considering the 15 medium instances;
- in terms of outranking, both algorithms provided similar solutions in 9 instances (16, 20–24, and 27–29); O-PSO provided solutions matching the DM's preferences better than T-RIPG in 4 instances (17, 18, 25, and 26), and T-RIPG outperformed O-PSO in 2 instances (19 and 30); and
- finally, on the complete 200-size benchmark, the difference in favour of O-PSO was significant in the number of solutions belonging to the A-RoI; therefore, O-PSO behaves better than T-RIPG at this scale.

Table 8 presents the results on the 500-size benchmark, from which the following remarks can be made:

- in 8/15 instances (33–35, 38–40, 44, and 45), O-PSO provided better coverage than T-RIPG;
- in 7/15 instances (32–35, 39, 40, and 45), O-PSO offered solutions matching the DM's preferences better than T-RIPG;
- on the 15 large instances, the differences were significant in both quality measures; and

Table 7
Coverage and approximation to the RoI for medium-size instances. Comparison with T-RIPG.

Inst.	N	M	Class	Coverage		Solutions in the A-RoI	
				O-PSO over T-RIPG	T-RIPG over O-PSO	O-PSO	T-RIPG
16	200	5	1	13%	19%	90%	79%
17	200	5	2	15%	14%	80%	46%
18	200	5	3	21%	17%	87%	42%
19	200	5	4	9%	22%	45%	60%
20	200	5	5	36%	26%	50%	42%
21	200	10	1	25%	11%	96%	89%
22	200	10	2	23%	6%	84%	73%
23	200	10	3	12%	15%	93%	95%
24	200	10	4	17%	25%	60%	51%
25	200	10	5	28%	21%	75%	24%
26	200	15	1	19%	18%	100%	58%
27	200	15	2	27%	21%	92%	71%
28	200	15	3	11%	18%	81%	82%
29	200	15	4	0%	18%	86%	90%
30	200	15	5	24%	20%	45%	57%

Table 8
Coverage and approximation to the RoI for large-size instances. Comparison with T-RIPG.

Inst.	N	M	Class	Coverage		Solutions in the A-RoI	
				O-PSO over T-RIPG	T-RIPG over O-PSO	O-PSO	T-RIPG
31	500	5	1	11%	7%	92%	89%
32	500	5	2	9%	6%	95%	35%
33	500	5	3	18%	0%	90%	66%
34	500	5	4	32%	0%	49%	17%
35	500	5	5	43%	11%	50%	15%
36	500	10	1	21%	8%	97%	83%
37	500	10	2	21%	11%	84%	81%
38	500	10	3	37%	6%	91%	76%
39	500	10	4	20%	0%	73%	28%
40	500	10	5	30%	0%	50%	20%
41	500	15	1	12%	7%	87%	96%
42	500	15	2	20%	13%	100%	94%
43	500	15	3	19%	7%	96%	72%
44	500	15	4	26%	0%	48%	43%
45	500	15	5	42%	0%	47%	14%

- O-PSO was capable of approximating the RoI at least as well as the standard reported in the state-of-art literature for this kind of multi-objective UPMSP.

5. Conclusions and directions for future research

This paper has presented a metaheuristic, referred to as O-PSO, based on swarm intelligence to carry out multi-objective scheduling optimisation in unrelated parallel machines, which incorporates the preferences of the DM. As far as we know, this is the first time that PSO has been enriched with a relational system of fuzzy outranking preferences.

We presented the application of O-PSO to a real-world problem in the freight industry with three objective functions, whose discrete optimisation model is also a contribution of this paper. In this case study, O-PSO showed that it was able to generating high-quality solutions compared with a widely used policy.

Also, an experiment with a benchmark of synthetic instances classified as small, medium, and large was conducted. The results are backed by a series of statistical tests and comparisons with EDPSO and T-RIPG, which are the metaheuristics with the best results at approximating the Pareto set for this kind of scheduling problem. For large instances, O-PSO often finds schedules that outrank those from EDPSO and T-RIPG so it becomes more suitable as the input size increases, which gives evidence of its scalability. In addition, O-PSO ran faster than EDPSO because it searches for the RoI rather than approximating a sample of the entire Pareto frontier. These results show the efficiency of our proposal. The better performance of our approach can be attributed to its use of the relational system of fuzzy outranking preferences.

The following are the advantages of O-PSO: (*i*) it supports the analysis of non-dominated schedules by incorporating the underlying principles of the European School of MultiCriteria Decision Analysis. In other words, it uses fuzzy outranking to consider the DM preferences instead of exclusively using Pareto dominance; (*ii*) the system makes selective pressure towards solutions that belong to a region of the efficient frontier according to the preferences (the RoI), preventing searching the entire Pareto frontier; (*iii*) an *a posteriori* decision analysis of the schedules is no longer required; and, (*iv*) the system deals with any number of objectives because they are always mapped into three objectives when defining the best compromise solution.

The advantages of our approach are not restricted to the domain of the freight industry; therefore, O-PSO could be adapted and applied to other real-world multi-objective UPMSPs such as energy systems (e.g. Zhou and Gu, 2021; Zhu and Tianyu, 2019), health systems (e.g. Pouria et al., 2021), textile industry (e.g. Kim and Kim, 2020), plastic processing plants (e.g. Fanjul-Peyro, 2020), automotive industry (e.g. Åblad et al., 2021), cloud computing (e.g. Bhardwaj et al., 2020), and aircraft industry (e.g. Kianpour et al., 2021), to mention only a few. Contrastingly, O-PSO requires close interaction with the DM to infer the parameter values for the outranking model. We suggest that a decision analyst support the DM during this challenging task, which is still an open field for research within the area of MCDA.

As future research, we will develop a mathematical programming method based on outranking relations to reschedule parts of the planning that could be dynamically interrupted, where the aim is to minimise the changes to the overall timetable. Since programme interruptions are common in this domain, it is important to address this issue. For this purpose, the parameter values and the known data about outranking relations will also be used to reschedule. In addition, we will conduct a study to determine how the number of objective functions affects the performance of O-PSO.

CRediT authorship contribution statement

Gilberto Rivera: Conceptualization, Methodology, Validation, Formal analysis, Supervision. **Raúl Porras:** Software, Validation, Project administration, Writing - original draft. **J. Patricia Sanchez-Solis:** Conceptualization, Investigation, Validation, Writing - original draft. **Rogelio Florencia:** Conceptualization, Validation, Investigation. **Vicente García:** Software, Validation, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Åblad, E., Strömberg, A., Spensieri, D., 2021. Exact makespan minimization of unrelated parallel machines. Open J. Math. Optim. 2, 1–15. http://dx.doi.org/10.5802/ojmo.4.

Afzalirad, M., Rezaeian, J., 2017. A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches. Appl. Soft Comput. 50, 109–123. http://dx.doi.org/10.1016/j.asoc.2016.10.039.

Akbar, M., Irohara, T., 2018. Scheduling for sustainable manufacturing: A review. J. Cleaner Prod. 205, 866–883. http://dx.doi.org/10.1016/j.jclepro.2018.09.100.

Alvarez, P.A., Leyva López, J., López Parra, P., 2018a. A new disaggregation preference method for new products design. In: 13th International FLINS Conference, Belfast, Northern Ireland, UK. World Scientific, http://dx.doi.org/10.1142/9789813273238_0128.

Alvarez, P.A., Pereira, J., Arroyo, M., Leyva López, J.C., 2018b. Disaggregating preferences for a supplier development problem in the mexican aerospace industry. In: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, pp. 1–7. http://dx.doi.org/10.1109/FUZZ-IEEE.2018.8491496.

Bai, Q., 2010. Analysis of particle swarm optimization algorithm. Comput. Inf. Sci. 3 (1), 180.

Bhardwaj, K.A., Gajpal, Y., Surti, C., Gill, S.S., 2020. Heart: Unrelated parallel machines problem with precedence constraints for task scheduling in cloud computing using heuristic and meta-heuristic algorithms. Softw. - Pract. Exp. 50 (12), 2231–2251. http://dx.doi.org/10.1002/spe.2890.

Bitar, A., Dauzère-Pérès, S., Yugma, C., 2021. Unrelated parallel machine scheduling with new criteria: Complexity and models. Comput. Oper. Res. 132, 105291. http://dx.doi.org/10.1016/j.cor.2021.105291.

Chang, Y.-M., Liao, W.-C., Wang, S.-C., Yang, C.-C., Hwang, Y.-S., 2020. A framework for scheduling dependent programs on GPU architectures. J. Syst. Archit. 106, 101712. http://dx.doi.org/10.1016/j.sysarc.2020.101712.

Cheng, C.-Y., Pourhejazy, P., Ying, K.-C., Li, S.-F., Chang, C.-W., 2020. Learning-based metaheuristic for scheduling unrelated parallel machines with uncertain setup times. IEEE Access 8, 74065–74082. http://dx.doi.org/10.1109/ACCESS.2020.2988274.

Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A., et al., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, Vol. 5. Springer, http://dx.doi.org/10.1007/978-0-387-36797-2.

Cruz-Reyes, L., Fernandez, E., Sanchez-Solis, J.P., Coello Coello, C.A., Gomez, C., 2020. Hybrid evolutionary multi-objective optimisation using outranking-based ordinal classification methods. Swarm Evol. Comput. 54, 100652. http://dx.doi.org/10.1016/j.swevo.2020.100652.

Cruz-Reyes, L., Perez-Villafuerte, M., Rangel, N., Fernandez, E., Gomez, C., Sanchez-Solis, P., 2018. Performance analysis of an a priori strategy to elicitate and incorporate preferences in multi-objective optimization evolutionary algorithms. In: Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications. Springer, pp. 401–414. http://dx.doi.org/10.1007/978-3-319-71008-2_29.

Doumpos, M., Zopounidis, C., 2019. Preference disaggregation for multicriteria decision aiding: An overview and perspectives. In: New Perspectives in Multiple Criteria Decision Making. Springer, pp. 115–130. http://dx.doi.org/10.1007/978-3-030-11482-4_4.

Ezugwu, A.E., 2019. Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. Knowl.-Based Syst. 172, 15–32. http://dx.doi.org/10.1016/j.knosys.2019.02.005.

Fanjul-Peyro, L., 2020. Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources. Expert Syst. Appl.: X 5, 100022. http://dx.doi.org/10.1016/j.eswax.2020.100022.

Fernandez, E., Gomez, C., Rivera, G., Cruz-Reyes, L., 2015. Hybrid metaheuristic approach for handling many objectives and decisions on partial support in project portfolio optimisation. Inform. Sci. 315, 102–122. http://dx.doi.org/10.1016/j.ins.2015.03.064.

Fernandez, E., Gomez-Santillan, C., Rangel-Valdez, N., Cruz-Reyes, L., Balderas, F., 2019a. An interval-based evolutionary approach to portfolio optimization of new product development projects. Math. Probl. Eng. 2019, http://dx.doi.org/10.1155/2019/4065424.

Fernandez, E., Navarro, J., Solares, E., Coello Coello, C.A., 2020. Using evolutionary computation to infer the decision maker's preference model in presence of imperfect knowledge: A case study in portfolio optimization. Swarm Evol. Comput. 54, 100648. http://dx.doi.org/10.1016/j.swevo.2020.100648.

Fernandez, E., Rangel-Valdez, N., Cruz-Reyes, L., Gomez-Santillan, C., Rivera-Zarate, G., Sanchez-Solis, P., 2019b. Inferring parameters of a relational system of preferences from assignment examples using an evolutionary algorithm. Technol. Econ. Dev. Econ. 2019, 693–715. http://dx.doi.org/10.3846/tede.2019.9475.

Frausto-Solis, J., Hernández-Ramírez, L., Castilla-Valdez, G., González-Barbosa, J., Sánchez-Hernández, J., 2021. Chaotic multi-objective simulated annealing and threshold accepting for job shop scheduling problem. Math. Comput. Appl. 26, 1–34. http://dx.doi.org/10.3390/mca26010008.

Fuchigami, H.Y., Rangel, S., 2018. A survey of case studies in production scheduling: Analysis and perspectives. J. Comput. Sci. 25, 425–436. http://dx.doi.org/10.1016/j.jocs.2017.06.004.

Garavito-Hernández, E.A., Peña Tibaduiza, E., Perez-Figueredo, L.E., Moratto-Chimenty, E., 2019. A meta-heuristic based on the imperialist competitive algorithm (ICA) for solving hybrid flow shop (HFS) scheduling problem with unrelated parallel machines. J. Ind. Prod. Eng. 36 (6), 362–370. http://dx.doi.org/10.1080/21681015.2019.1647299.

Gilvaei, M.N., Jafari, H., Ghadi, M.J., Li, L., 2020. A novel hybrid optimization approach for reactive power dispatch problem considering voltage stability index. Eng. Appl. Artif. Intell. 96, 103963. http://dx.doi.org/10.1016/j.engappai.2020.103963.

Harbaoui, H., Khalfallah, S., 2020. Tabu-search optimization approach for no-wait hybrid flow-shop scheduling with dedicated machines. Procedia Comput. Sci. 176, 706–712. http://dx.doi.org/10.1016/j.procs.2020.09.043.

Kayvanfar, V., Zandieh, M., Teymourian, E., 2017. An intelligent water drop algorithm to identical parallel machine scheduling with controllable processing times: a just-in-time approach. Comput. Appl. Math. 36 (1), 159–184. http://dx.doi.org/10.1007/s40314-015-0218-3.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4. IEEE, pp. 1942–1948. http://dx.doi.org/10.1109/ICNN.1995.488968.

Kianpour, P., Gupta, D., Krishnan, K., Gopalakrishnan, B., 2021. Optimising unrelated parallel machine scheduling in job shops with maximum allowable tardiness limit. Int. J. Ind. Syst. Eng. 37 (3), 359–381. http://dx.doi.org/10.1504/IJISE.2021.113443.

Kim, Y.-H., Kim, R.-S., 2020. Insertion of new idle time for unrelated parallel machine scheduling with job splitting and machine breakdowns. Comput. Ind. Eng. 147, 106630. http://dx.doi.org/10.1016/j.cie.2020.106630.

Kurniawan, B., 2020. Mathematical models of energy-conscious bi-objective unrelated parallel machine scheduling. J. Tek. Ind. 21 (2), 115–125. http://dx.doi.org/10.22219/JTIUMM.Vol21.No2.115-125.

Lei, D., Yuan, Y., Cai, J., Bai, D., 2020. An imperialist competitive algorithm with memory for distributed unrelated parallel machines scheduling. Int. J. Prod. Res. 58 (2), 597–614. http://dx.doi.org/10.1080/00207543.2019.1598596.

Lin, S.-W., Ying, K.-C., Wu, W.-J., Chiang, Y.-I., 2016. Multi-objective unrelated parallel machine scheduling: a tabu-enhanced iterated Pareto greedy algorithm. Int. J. Prod. Res. 54 (4), 1110–1121. http://dx.doi.org/10.1080/00207543.2015.1047981.

Lu, S., Liu, X., Pei, J., Thai, M.T., Pardalos, P.M., 2018. A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity. Appl. Soft Comput. 66, 168–182. http://dx.doi.org/10.1016/j.asoc.2018.02.018.

Manupati, V., Rajyalakshmi, G., Chan, F.T., Thakkar, J., 2017. A hybrid multi-objective evolutionary algorithm approach for handling sequence-and machine-dependent set-up times in unrelated parallel machine scheduling problem. Sādhanā 42 (3), 391–403. http://dx.doi.org/10.1007/s12046-017-0611-2.

Meng, L., Zhang, C., Shao, X., Ren, Y., Ren, C., 2019. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. Int. J. Prod. Res. 57 (4), 1119–1145. http://dx.doi.org/10.1080/00207543.2018.1501166.

Murakami, K., Morita, H., 2010. A method for generating robust schedule under uncertainty in processing time. Int. J. Biomed. Soft Comput. Hum. Sci.: Off. J. Biomed. Fuzzy Syst. Assoc. 15 (1), 45–50. http://dx.doi.org/10.24466/ijbschs.15.1_45.

Naderi, E., Pourakbari-Kasmaei, M., Abdi, H., 2019. An efficient particle swarm optimization algorithm to solve optimal power flow problem integrated with FACTS devices. Appl. Soft Comput. 80, 243–262. http://dx.doi.org/10.1016/j.asoc.2019.04.012.

Naderi, E., Pourakbari-Kasmaei, M., Cerna, F., Lehtonen, M., 2021. A novel hybrid self-adaptive heuristic algorithm to handle single- and multi-objective optimal power flow problems. Int. J. Electr. Power Energy Syst. 125, 106492. http://dx.doi.org/10.1016/j.ijepes.2020.106492.

Naderi, E., Pourakbari-Kasmaei, M., Lehtonen, M., 2020. Transmission expansion planning integrated with wind farms: A review, comparative study, and a novel profound search approach. Int. J. Electr. Power Energy Syst. 115, 105460. http://dx.doi.org/10.1016/j.ijepes.2019.105460.

Ojstersek, R., Brezocnik, M., Buchmeister, B., 2020. Multi-objective optimization of production scheduling with evolutionary computation: a review. Int. J. Ind. Eng. Comput. 11 (3), 359–376. http://dx.doi.org/10.5267/j.ijiec.2020.1.003.

Pouria, K., Vahid, K., Majid, R., Frank, W., 2021. A bi-objective home health care routing and scheduling model with considering nurse downgrading costs. Int. J. Environ. Res. Public Health 18 (3), http://dx.doi.org/10.3390/ijerph18030900.

Ramos-Figueroa, O., Quiroz-Castellanos, M., Carmona-Arroyo, G., Vázquez, B., Kharel, R., 2020. Parallel-machine scheduling problem: An experimental study of instances difficulty and algorithms performance. In: Recent Advances of Hybrid Intelligent Systems Based on Soft Computing. Springer, pp. 13–49. http://dx.doi.org/10.1007/978-3-030-58728-4_2.

Rangel-Valdez, N., Fernandez, E., Cruz-Reyes, L., Gomez-Santillan, C., Rivera, G., Florencia, R., 2018. Robustness analysis of an outranking model parameters' elicitation method in the presence of noisy examples. Math. Probl. Eng. 2018, http://dx.doi.org/10.1155/2018/2157937.

Rangel-Valdez, N., Fernández, E., Cruz-Reyes, L., Santillán, C.G., Hernández-López, R.I., 2015. Multiobjective optimization approach for preference-disaggregation analysis under effects of intensity. In: Mexican International Conference on Artificial Intelligence. Springer, pp. 451–462. http://dx.doi.org/10.1007/978-3-319-27101-9_34.

Rivera, G., Cisneros, L., Sánchez-Solís, P., Rangel-Valdez, N., Rodas-Osollo, J., 2020. Genetic algorithm for scheduling optimization considering heterogeneous containers: A real-world case study. Axioms 9 (1), 27. http://dx.doi.org/10.3390/axioms9010027.

Roy, B., Vanderpooten, D., 1996. The European school of MCDA: Emergence, basic features and current works. J. Multi-Criteria Decis. Anal. 5 (1), 22–38. http://dx.doi.org/10.1002/(SICI)1099-1360(199603)5:1<22::AID-MCDA93>3.0.CO;2-F.

Shabtay, D., Zofi, M., 2018. Single machine scheduling with controllable processing times and an unavailability period to minimize the makespan. Int. J. Prod. Econ. 198, 191–200. http://dx.doi.org/10.1016/j.ijpe.2017.12.025.

Shahvari, O., Logendran, R., 2017. An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes. Comput. Oper. Res. 77, 154–176. http://dx.doi.org/10.1016/j.cor.2016.07.021.

Tirkolaee, E., Aydın, N., Ranjbar-Bourani, M., Weber, G., 2020. A robust bi-objective mathematical model for disaster rescue units allocation and scheduling with learning effect. Comput. Ind. Eng. 149, 106790. http://dx.doi.org/10.1016/j.cie.2020.106790.

Wang, H., Alidaee, B., 2019. Effective heuristic for large-scale unrelated parallel machines scheduling problems. Omega 83, 261–274. http://dx.doi.org/10.1016/j.omega.2018.07.005.

Wang, M.-Z., Zhang, L.-L., Choi, T.-M., 2018. Bi-objective optimal scheduling with raw material's shelf-life constraints in unrelated parallel machines production. IEEE Trans. Syst. Man Cybern.: Syst. 50 (11), 4598–4610. http://dx.doi.org/10.1109/TSMC.2018.2855700.

Wojakowski, P., Warżołek, D., 2014. The classification of scheduling problems under production uncertainty. Res. Logist. Prod. 4 (3), 245–255.

Yan, W., Li, M.-J., Zhong, Y.-C., Qu, C.-Y., Li, G.-X., 2020. A novel k-MPSO clustering algorithm for the construction of typical driving cycles. IEEE Access 8, 64028–64036. http://dx.doi.org/10.1109/ACCESS.2020.2985207.

Yepes-Borrero, J., Perea, F., Ruiz, R., Villa, F., 2021. Bi-objective parallel machine scheduling with additional resources during setups. European J. Oper. Res. 292 (2), 443–455. http://dx.doi.org/10.1016/j.ejor.2020.10.052.

Yin, J., Ma, Y., Hu, Y., Han, K., Yin, S., Xie, H., 2020. Delay, throughput and emission tradeoffs in airport runway scheduling with uncertainty considerations. Netw. Spat. Econ. 21, 85–122. http://dx.doi.org/10.1007/s11067-020-09508-3.

Zhang, X., Sun, W., Xue, M., Lin, A., 2021. Probability-optimal leader comprehensive learning particle swarm optimization with Bayesian iteration. Appl. Soft Comput. 103, 107132. http://dx.doi.org/10.1016/j.asoc.2021.107132.

Zhao, B., Gao, J., Chen, K., Guo, K., 2018. Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines. J. Intell. Manuf. 29 (1), 93–108. http://dx.doi.org/10.1007/s10845-015-1091-z.

Zhou, B., Gu, J., 2021. Energy-awareness scheduling of unrelated parallel machine scheduling problems with multiple resource constraints. Int. J. Oper. Res. 41 (2), 196–217. http://dx.doi.org/10.1504/IJOR.2021.115623.

Zhu, W., Tianyu, L., 2019. A novel multi-objective scheduling method for energy based unrelated parallel machines with auxiliary resource constraints. IEEE Access 7, 168688–168699. http://dx.doi.org/10.1109/ACCESS.2019.2954601.