Applied Intelligence
https://doi.org/10.1007/s10489-020-02027-1

A multi objective volleyball premier league algorithm for green scheduling identical parallel machines with splitting jobs

Khodakaram Salimifard[1] · Jingpeng Li[2] · Davood Mohammadi[1] · Reza Moghdani[1]

Accepted: 16 October 2020
© The Author(s) 2020

Abstract
Parallel machine scheduling is one of the most common studied problems in recent years, however, this classic optimization problem has to achieve two conflicting objectives, i.e. minimizing the total tardiness and minimizing the total wastes, if the scheduling is done in the context of plastic injection industry where jobs are splitting and molds are important constraints. This paper proposes a mathematical model for scheduling parallel machines with splitting jobs and resource constraints. Two minimization objectives - the total tardiness and the number of waste - are considered, simultaneously. The obtained model is a bi-objective integer linear programming model that is shown to be of NP-hard class optimization problems. In this paper, a novel Multi-Objective Volleyball Premier League (MOVPL) algorithm is presented for solving the aforementioned problem. This algorithm uses the crowding distance concept used in NSGA-II as an extension of the Volleyball Premier League (VPL) that we recently introduced. Furthermore, the results are compared with six multi-objective metaheuristic algorithms of MOPSO, NSGA-II, MOGWO, MOALO, MOEA/D, and SPEA2. Using five standard metrics and ten test problems, the performance of the Pareto-based algorithms was investigated. The results demonstrate that in general, the proposed algorithm has supremacy than the other four algorithms.

Keywords Parallel machine scheduling · Splitting jobs · Wastes · Total tardiness · Multi-objective optimisation · Volleyball premier league

1 Introduction

Studying parallel machines are of high importance both theoretically and practically. In terms of theory, it is an extension of the single machine and a particular instance of the flexible flow shop. In terms of practice, it is vitally important for the sake of common parallel resources in the real world, but very difficult to find an optimum solution for the problem. Likewise, decomposition methods used in multi-stage systems sometimes utilize parallel machine techniques.

Sometimes the parallel machine scheduling (PMS) is considered as a two-step procedure. First, assigning jobs to available machines. Second, when allocating jobs to machines has been done, the sequence of the jobs needs to be allocated to each machine [1]. Since we cannot consistently find an absolute solution for parallel machines and for most of the criteria, especially those based on tardiness - that don't have a linear relationship with completion time - the problem is non-polynomial (NP). Based on the division problem, Lenstra et al. [2] proved that a parallel machine problem of minimizing the total tardiness is a binary non-polynomial problem even for two machines. Koulamas [3], based on the single machine model in Du and Leung [4], indicated that the parallel machine problem is at least a binary non-polynomial. About the minimization of weighted tardiness, Lawler et al. [5] proved that when jobs have different weights, the problem would be NP-hard.

One of the deterministic methods for solving such a category of problems is dynamic programming [6], although the extent of the application of this method is very limited due to

✉ Jingpeng Li
Jingpeng.Li@stir.ac.uk

Khodakaram Salimifard
salimifard@pgu.ac.ir

Davood Mohammadi
Mohammady1366@yahoo.com

Reza Moghdani
reza.moghdani@gamil.com

[1] CIIORG, Persian Gulf University, Bushehr 75168, Iran

[2] Division of Computer Science and Mathematics, University of Stirling, Stirling, UK

Published online: 08 December 2020

the diversity and multiplicity of options whether in job allocation or in determining their sequence on each machine. Branch and bound approach has also been used in many studies to solve parallel machine problems, for example, Elmaghraby and Park's study [7] for jobs with identical processing time and due date, Barnes and Brennans's study [8] for mostly 20 jobs and 4 machines, and Yalaoui and Chu's study [9] for minimizing total tardiness.

Moreover, some papers investigate solutions to parallel machine problems in particular situations, for example, Lawler's study [10] with respect to transportation model and jobs with an identical processing time, Root's study [11] with identical due date and Pritsker et al's study [12] in which binary linear programming is used. Ranjbar et al. [13] also used a branch and bound approach for PMS with stochastic processing time. Figielska [14] proposed a problem of scheduling pre-emptive jobs on parallel unrelated machines in the presence of renewable resource constraints and sequence-dependent setup costs. The problem was solved using the column generation method and ant colony optimization (ACO) algorithm. Chen [15] developed a column generation based branch and bound method to solve simultaneous job scheduling and resource allocation problems.

Reviewing parallel machine flexible resource scheduling (PMFRS) problems and an unspecified version of it (UPMFRS), Edis and Oguz [16] gave some extensions to the model of dynamic PMFRS and proposed integer programming (IP) models for static and dynamic UPMFRS problems with the aim of minimizing the makespan. Lee et al. [17] proposed a uniform parallel machine problem in which the objective is to jointly find an optimal assignment of operators to machines and an optimal schedule to minimize the makespan. The problem was solved using a genetic algorithm (GA) and particle swarm optimization (PSO).

Due to the complexity of PMS problems and the size of real-world problems, deterministic methods cannot be effective. Thus, heuristic algorithms are often used. These methods are often to determine a sequence based on a priority criterion and then to assign the jobs to machines based on the time of availability and also they are single-objective. Various methods are proposed in this field, among which montage's ratio method [18], cost over time [19], apparent urgency [20], traffic congestion [21], DS [22], hybrid simulated annealing [3], PDEC [9], high priority job first [23] and modified due date [24] can be indicated. Besides, in terms of the multi-objective algorithms we can count recently introduced algorithms by [25] in the field of inventory control, and [26, 27] in the field of transportation science.

Correa et al. [28] suggested a strong model for unrelated PMS. Tao [29] proposed an online algorithm for unrelated PMS with a minimization of weighted total completion time. Ouazenea et al. [30] proposed a mixed-integer mathematical model for Job loading balance on parallel machines. Also, Hsu et al. [31] studied rate-modifying operations with the aim of minimization of the total completion time on unrelated PMS.

Regarding the complexity of real-world problems, it is necessity to study scheduling problems with multi-objectives. Some scholars take the advantages of recently multi-objective algorithms extended to make contributions for solving scheduling problems to reach better solutions. Therefore, in this study, we extend a multi-objective version of the VPL algorithm using crowding distance in comparison with the original ones which is archive based.

In this study, we concentrate on identical PMS considering splitting jobs together with sequence-dependent setup waste and time. The remaining structure of the paper is shown as follows. Section 2 gives a brief literature review on PMS and then determines a theoretical gap in this research area. Section 3 is devoted to the mathematical modeling of the problem. Section 4 comes to extensively explain multi-objective Pareto-based meta-heuristic algorithms and solution representation of our approach. Sections 5 and 6 introduce our proposed algorithm. Section 7 investigates the performance of our proposed algorithm using different metrics and a set of numerical examples. Finally, Section 8 will draw the conclusion and future researches.

2 Literature review

The allocation of assets to a certain number of jobs in a predetermined time period defines scheduling while optimizing one or more objectives [1]. The scientific approach to operation planning originated from the industrial revolution and Henry Laurence Gantt's studies [32], and since then it has been widely studied. In this section, a review of PMS and the approached environment in this paper (i.e. machine scheduling with spilling jobs) is outlined. PMS problems are among the most complex machine scheduling problems. Based on the operating velocity of each machine for various jobs, there are three parallel machine scheduling (PMS) environments according to [33]: identical machines run at the same speed; heterogeneous machines run at various velocities but its operating rate/velocity is compatible for every machine operating various jobs. In the sense of unrelated machines, defining a generalization of a heterogeneous environment as an unrelated machine collection can consist of a non-identical machine collection. Some studies have been done on PMS with and without splitting jobs as follows.

Serafini [34] studied a real problem of scheduling looms in the textile industry. Two environments of uniform machines and unrelated machines have been considered in this study, while the first one was modeled as network flow and the latter one was modeled using linear programming. The objective is defined as minimizing maximum tardiness and weighted

tardiness. Xing and Zhang [35] considered the PMS problem with splitting job property and independent times of setup. Some simple polynomial solvable cases and a heuristic maximum completion time estimation procedure and set-up-time list scheduling (ML) including worst-case analysis with the aim of minimization makespan were presented as well.

Sarıçiçek and Çelik [36] presented a mathematical model using mixed-integer variables for a parallel machine scheduling problem with job split, with the aim of minimizing total tardiness. They applied two metaheuristics (i.e. Tabu search and simulated annealing) to solve three groups of test problems. Results suggest that the performance of the simulated annealing algorithm is far better than the tabu search algorithm in terms of optimal solution deviation and computational time. Park et al. [37] addressed the same problem considering sequence-dependent major/minor times of setting up, with the aim of minimizing total tardiness. They developed three heuristics: the slack-based heuristic, the dynamic scheduling window-based heuristic, and the estimated latest starting time-based heuristic. The results illustrate that the former and the latter heuristics have the same performance and they have supremacy in terms of performance than the dynamic scheduling window-based heuristic.

Shim and Kim [38] proposed a branch and bound algorithm to schedule jobs composed of some unit jobs considering independent setup times on parallel machines. The objective has been stated to be the minimization of total tardiness. Kim et al. [39] also considered the same objective. They proposed a heuristic algorithm which is consists of two-phase to find a solution for the problem. In phase one, an initial sequence is suggested for the PMS problem. In phase two, the initial scheduling plan is rescheduled on machines considering the jobs and sub-jobs.

Yalaoui and Chu [40] studied the scheduling problem of splitting jobs on parallel machines considering sequence-dependent setup times with the aim of minimization of the maximum makespan. They provided a heuristic method to solve the problem. Firstly the problem is simplified into a subject called single machine scheduling and is then metamorphosed into a well-known traveling salesman problem and solved utilizing Little's method. Using the results of the first part, the initial solution is improved in a stepwise manner taking into account the setup times and job splitting property. Zhu and Heady [41] used a mixed-integer programming approach for minimizing job earliness and tardiness in unrelated PMS problem with sequence-dependent setup time. They stated that the proposed model is beneficial to researchers for testing the performance of earliness and tardiness heuristics.

The unrelated PMS problem with the aim of minimization total tardiness has also been studied by Shim and Kim [42]. Using several dominance properties and lower bounds, they suggested a branch and bound algorithm to solve the problem. Logendran and Subur [43] Considered the PMS problem with the aim of minimization of total weighted tardy jobs in the field of dynamic job releases and machine availability. They proposed a mixed-integer linear programming model containing constraints including hard operations to guarantee that Just In Time (JIT) production system is running. Four methods are implemented to generate initial solutions that are inputs for performing a Tabu search algorithm for reaching an optimal solution. Logendran et al. [44] investigated research with the same method but without splitting job property and sequence dependent setup times. Six various Tabu search-based algorithms and four various primary solutions finding mechanisms have been extended for generating a finest scheduling plan that minimizes weighted tardiness. Based on computational results, a fixed Tabu list size and a short-term memory algorithm were suggested for small instances whereas, for medium and large instances minimum iteration and long-term memory algorithm were used.

Liaw et al. [45] addressed the issue of unrelated PMS considering the minimization of the total weighted tardiness as an objective function. A branch and bound algorithm is proposed to find a solution using supremacy rules to reduce hopeless solutions. The results show that the proposed algorithm is appropriate for up to 18 jobs and 4 machines. Wang et al. [46] investigated the unrelated PMS problem considering splitting job property with the objective of minimization of makespan. Differential evolution is applied as the solution method and a new method for crossover and mutation are presented for global search procedure according to the job splitting constraints. Experimental results show the efficiency and feasibility of the proposed hybrid differential evolution.

Chen and Wu [47] studied the unrelated PMS problem with auxiliary constraints and total tardiness minimization objective. In this study it is assumed that each job requires one operation and has a due date, and to switch from one type of job to another type, setup time for die is needed. To solve this problem they proposed a heuristic-based threshold-accepting methods, Tabu lists, and improvement procedures. The experiment results show that the proposed algorithm can obtain optimal solutions for small size problems and performs much better than SA and apparent-tardiness cost-with-setup for problems of larger sizes.

Vallada and Ruiz [48] developed a GA with a new crossover operator and a very fast local search for unrelated PMS problems with sequence-dependent setup times. They suggested a MIP model with the objective of minimization of makespan. The experiment results show that the proposed method significantly outperforms the best methods known from the literature. An unrelated PMS problem with sequence and machine-dependent setup times and weighted jobs was studied by [49]. They applied a branch and bound algorithm in which the upper bound is obtained from the solution provided by GRASP. Two MIP models are solved using CPLEX and the results are compared with the proposed algorithm.

Outcomes prove the superiority of the performance of the algorithm on instances with about 30 jobs. Lin et al. [50] compared the performance of different heuristic and metaheuristic methods for a variety of unrelated PMS environments with three performance measures. The objective function is minimization of makespan, total weighted completion time and total weighted tardiness, and the test is made using least significant difference method.

Fanjul-Peyro and Ruiz [51] studied two situations of unrelated PMS problem under makespan minimization: one situation in which not all available parallel machines should be utilized, and another situation with no obligation to do all the jobs. MIP models are proposed for both situations and it is shown that the latter situation could be solved with commercial solvers efficiently. The PMS with splitting jobs and learning effects to minimize total completion time in a labor-intensive industry was presented by Wang et al. al. [52], solved by branch and bound algorithm and greedy search in small and large scale, respectively. A Tabu-enhanced iterated Pareto greedy algorithm was proposed by Lin et al. [53] to solve the unrelated PMS of completion times. Lara et al. [54] studied identical PMS problems with a different problem. Correa et al. [28] present a scheduling problem with splitting jobs which each part needs to be set up to minimize the weighted sum release date to minimize total tardiness. In this study preemption and splitting are not allowed. Shahvari and Logendran [55] developed a Tabu search heuristic for unrelated PMS problem to minimize the total weighted tardiness and total weighted completion time.

Yin et al. [56] investigated the PMS of deteriorating jobs in a disruptive environment where some of the machines are out of reach at a specific time. To minimize the total completion time, they applied different algorithms in polynomial time. Mensendiek et al. [57] studied identical PMS problems with a fixed due date which is prevalent in the industry. This problem is solved by two exact and approximate methods, branch and bound and Tabu search and genetic algorithms; respectively. Lee and Kim [58] studied two identical PMS problems in the case of machines are not uninterrupted available to minimize total tardiness and solved it by branch and bound algorithm.

Mora and Mosheiov [59] studied scheduling problems of batch processing machines with controllable processing times both on single and identical PMS. In scheduling problems with controllable processing times, the processing times are controlled by additional resources. The aim of this research is to study two exemplars of the presented model. They minimized the total flow time and the compression cost, and in the next step minimized the total flowtime subject to an upper bound on the maximum compression. Lia et al. [60] addressed the unrelated parallel batch processing machine problems. In this study, jobs are not identical and each machine is able to process several jobs as a batch. The size of the jobs does not exceed the capacity of each machine. The objective of the problem is the minimization of the makespan. The model is solved by two groups of heuristics. In the first group, at the outset, jobs scheduled in each batch, and then each batch is assigned to each machine. By contrast, in the second group, jobs are assigned to each machine firstly and then scheduling is done in each batch.

Zhoua et al. [61] studied the parallel machine subject with the aim of minimization of the makespan of arbitrary job size. In this model, machines have different capacity and processing speed. They presented an effective differential evolution-based hybrid algorithm for the large scale problems, in which batches are generated and assigned to machines. Li et al. [62] studied the integrated production scheduling and delivery on identical parallel batch processing machines with the aim of maximization of total profit company. Multiple jobs can be loaded on each machine as a batch provided that the size of the jobs does not exceed to machine capacity. The company may earn a profit if the jobs are done on due dates. They demonstrated that, in the event that the sizes of the jobs are identical, the problem could be solved in polynomial time. Otherwise, it is known as NP-hard problem. Thus, they proposed and compared heuristics to solve NP-hard cases.

Identical PMS is studied by [63] in the field of JIT (Just-In-Time) environment in the presence of WIP (Work In Process) and preemptive jobs. Two objectives are considered in this paper: the former is the minimization of total weighted earliness and tardiness, holding cost of all jobs which are waiting to be processed as WIP costs, and the latter is related to the number of interrupted jobs. Moreover, this study utilizes two multi-objective algorithms known as the NSGAII and the NRGA.

In a research done by [64], the unrelated PMS with sequence-dependent setup time and splitting jobs is combined with the heterogeneous vehicle routing problem to tackle a real case study in the field of the metal packaging industry. The objectives of the problems are minimization of setup costs and distribution costs. Related to solving the problem they introduced a two-stage iterative heuristic. Two mathematical models are represented by [65] for identical PMS with specified operating times and tool requirements with the objective of makespan minimization. The problem is solved by an adaptive large neighborhood search metaheuristic they introduced. A memetic differential evolutionary algorithm is represented in [66] for minimizing energy-efficient and the makespan in the field of PMS.

A study done by [67] a heuristic called SLMR is presented for solving the scheduling problems of splitting jobs on parallel machines with learning effects and the vital-few law with the objective of minimization of makespan. The uniform PMS problem with splitting job and setup resources in the field of dedicated machines was studied by [68] in which the setup times are sequence-independent with the minimization of

makespan. To solve the problem they introduced a heuristic which is enough good for a real case study in the field of manufacturing of automotive pistons in Korea. The uniform PMS problem considering the green approach to minimize emissions of pollutions with the objective of minimization of makespan simultaneously was studied by [69]. To solve the problem they improved H1 and H2 heuristics. A memetic differential evolution algorithm was developed by [66] for solving unrelated PMS problems with the aim of minimization makespan and total energy consumption consequently.

The PMS problem considering job splitting and hazardous wastes in the field of fuzzy environment is formulated for the injection molding industry by [70] with the aim of minimization of the total tardiness and the total hazardous waste. For reaching a good solution for this industry, a genetic algorithm is implemented.

Identical parallel batch processing machine with the two objectives of minimizing makespan and maximum tardiness has been studied [71]. To find out the solution, a multi-objective ant colony optimization approach called Pareto-based ant colony system (PACS) is used. Wang and Leung [72] addressed parallel batch processing with identical processing time jobs on machines with different capacities to minimize the makespan. In this model, the size of the jobs does not exceed the capacity of each machine. Finally, a polynomial-time approximation algorithm was proposed and showed to be a very good algorithm in practice. Lopes and de Carvalho [73] considered the unrelated PMS problem with setup times associated with the sequence and availability dates for machines and release date for jobs with the aim of minimization of whole weighted tardiness. They developed a branch and price method to solve the problem and a new column generation acceleration method was developed in this paper. Results indicate the reasonable computational time for large size problems. In addition to above studies, it is worth mentioning that some excellent algorithms have been presented for solving combinatorial optimization problems such as [74–77], and some of them are specialized in the field of scheduling, for instance [78–81], and which all of them give us valuable insight to extend our algorithm.

The literature we have reviewed are selected from globally well-known scholarly data centers, for instance, Elsevier, Springer, etc. We search for articles published between 1996 and 2020. Since we aim to investigate recent researches in the field of PMS, we will be focused on published papers between the years 2010 and 2020. We concentrate on two aspects in the seeking procedure: mathematical modeling and solving methodologies. In terms of mathematical modeling, the type of jobs is considered (splitting or not) and the makespan, total weighted completion time, lateness, total weighted tardiness, total number of tardy jobs, maximum tardiness, maximum weighted tardiness, total tardiness, maximum makespan, earliness and tardiness, total completion time, the weighted sum of completion times, total flow time and the completion cost, total profit, and total production costs form the objective functions. In terms of solving methodologies, a variety of techniques for instance meta-heuristics methods (e.g. GA, PSO, TS, SA) and exact methods (e.g. constraint programming, branch and bond) have been used. Due to the aforementioned review, the research gap is identified as follows: first and foremost, the objective of minimization of setup wastes together with minimization of total tardiness in the form of a bi-objective problem is not studied so far. Furthermore, another restriction that increases the complexity of the problem is the limited number of molds for each job rather than a machine; in which we consider it as a constraint in our model. It is worth noting that another main contribution is presenting a Multi-Objective version of the Volleyball Premier League (MOVPL) algorithm using crowding distance borrowed from NSGA-II while the original MOVPL is archive based. Consequently, this modification of MOVPL makes it a more proper and accurate algorithm for solving identical unrelated PMS with divisible jobs with the aim of reducing total tardiness and hazardous waste simultaneously. This supremacy will prove in the following sections.

3 Problem description and mathematical model

In the plastic injection industries, the granule shaped raw plastic materials need to be melted and subsequently injected with pressure to the specified molds to making goods. In the continuing, plastic goods have been chilled to become hard, then the mold will open to deliver the goods. This procedure is performed successively. Each one of the goods needs a specific mold to be produced, and each mold could be set up on a number of specified machines. For those goods with more than one mold, we can take advantage of this opportunity by setting up molds on a number of machines concurrently to reduce the total tardiness and the total amount of wastes per machine.

One of the most common situations in the plastic injection molding industry that is an extension of the classical parallel machines model is investigated in this study. A sequence dependent wastes and mold resources constraints are added to the PMS model with splitting jobs, in addition to the sequence dependent setup times. These modifications boost the complexity of the problem. Therefore, this complexity need to be considered. Firstly, when setting up a job on a machine, it is vitally important which kind of job has been done before, since the color and material type have direct effect on the waste produced. For instance, suppose we have a sequence of two jobs on a machine where the first one has black color and the second one has white color. When a product with black color has done, the cylinder is full of black color and

we need to waste material to clean the cylinder for setting up the white color product. Therefore, this process takes time and waste. Secondly, mold is the important source of manufacturing in addition to machines. In the molding industry, commonly we have more than one mold for a specific product. We can take the advantages of this resource for reducing the completion time of each job by allocating one specified job to more than one machine at same time considering the number of mold resources of the job. So, the mold resource constraint plays a special role in the molding industry as a machine and it makes the PMS problem more complex. Therefore, we need to consider these situations to generate a good sequence of jobs on each machine.

This research suggests an MIP model for identical PMS problem. Jobs can be split on machines, but they have dependent waste and it is worth noting that there are precedence constraints amongst jobs in terms of their material and color. To clear the waste dependent problem on parallel machines we assume 4 jobs with one machine. Let job 1 be the first one to process, then all remaining jobs could be processed after job one but with different amount of setup waste which we call waste dependent. Due to the nature of injection molding industry, it takes times to shift from one job to another job, and consequently it makes different waste. So, the job with lower setup waste should be allocated after job one. The resource constraints and sequence dependent wastes were added regarding the essence of the injection industry. Minimizing wastes output and the total tardiness constitute the objective function. All machines are available throughout the time horizon. A job is defined as a single operation job which has a volume, process time, setup time and waste dependent, due date and also a limited source for proceeding called mould. Some jobs has more than one mould for proceeding which means these moulds can setup on identical parallel machines and run simultaneously to deliver the total amount of a job more quickly. Therefore, a job can be split and each part can be processed on at least two machines at a time. Jobs can be done as various lots, independently and asynchronously, but the completion time is calculated based on the biggest part of each job. Jobs are classified into different groups regarding their features. For each new group, all machines need to be installed and installation time is sequence-dependent.

For reaching the optimum solution for the splitting jobs on the parallel machine scheduling with sequence-dependent wastes, a MIL mathematical programming model is presented. The model including N jobs. Processing time of any given job i on machine K illustrates by $P(i)$ is. Due date of every job represents by $d(i)$. The parameters, decision variables and mathematical model are given as follows.

Indices:

M Set of machines indexed by $k = 1, \ldots, m$

N Set of jobs indexed by i and $j = 1, \ldots, n$

Parameters:

MO_i Total number of machines which can be used to perform job i .

P_i Processing time of job i.

Q_i Amount of job i.

$S_{i,\,j}$ Setup time in case job j coms after job i.

d_i Due date of job i.

c_{ik} completion time of job i on the machine k.

c_{0k} completion time of first job on the machine k.

$W_{i,\,j}$ Sequence dependent setup defective output of job j if it coms after job i.

a_k Amount of defective output of machine k.

Decision variables:

$X_{0,\,j,\,k}$ 1 if job j is the first job to process on machine k, 0 otherwise.

$y_{i,\,k}$ 1 if job i is processed on machine k, 0 otherwise.

$X_{i,\,0,\,k}$ 1 if job i is the last job to process on machine k, 0 otherwise.

$q_{i,\,k}$ Amount of job i on machine k.

$X_{i,\,j,\,k}$ 1 if job j comes after job i on machine k, 0 otherwise.

Z_i Tardiness for job i.

The mathematical model consists of the following equations.

$$F(1) : Min \sum_i Z_i \tag{1}$$

$$F(2) : Min \sum_k \sum_i \sum_j X_{i,j,k}\, W_{i,j} \tag{2}$$

$$\sum_k q_{ik} = Q_i \qquad i = 1 \ldots N \tag{3}$$

$$y_{ik} \geq \frac{q_{ik}}{Q_i} \qquad , i = 1 \ldots N;\ \ k = 1 \ldots M \tag{4}$$

$$y_{ik} \leq q_{ik} \quad , \quad i = 1 \ldots N;\ \ k = 1 \ldots M \tag{5}$$

$$\sum_j X_{ijk} = y_{ik} \qquad i = 1 \ldots N;\ \ k = 1 \ldots M \tag{6}$$

$$\sum_i X_{ijk} = y_{ik} \qquad i = 1 \ldots N;\ \ k = 1 \ldots M \tag{7}$$

$$\sum_{i=1}^{N} y_{ik} \leq 1 \qquad , \quad k = 1 \ldots M \tag{8}$$

$$X_{ijk} \times S_{ij} + C_{ik} + q_{jk} \times P_j \leq C_{jk} \qquad i = 0 \ldots N;\ \ j = 1 \ldots N; k = 1 \ldots M \tag{9}$$

$$C_{ik} - d_i \leq Z_i \qquad i = 1 \ldots N; \quad k = 1 \ldots M \tag{10}$$

$$C_{ok} = 0 \qquad k = 1 \ldots M \tag{11}$$

$$X_{ijk} \in \{0, 1\} \qquad i, j = 0 \ldots N;\ \ k = 1 \ldots M \tag{12}$$

$$q_{ik} \geq 0; \quad C_{ik} \geq 0; \quad Z_i \geq 0; \quad y_{ik} \geq 0 \quad X_{iik} = 0; \quad i = 1 \ldots N; \quad k = 1 \ldots M \tag{13}$$

$$\sum_i \sum_j X_{ijk} \times W_{ij} \leq a_k \quad k = 1 \ldots M \tag{14}$$

$$\sum_k y_{ik} \leq M o_i, \quad i = 0 \ldots N \tag{15}$$

The objective function is presented by Eqs. (1) and (2) for minimizing total tardiness and total wastes, respectively. Processing all parts of each job is guaranteed by means of Eq. (3). Equations (4) and (5) guarantee the assigning all parts of each job to machines. Equations (6)–(8) represent that each part of the job will not process more than once. Equation (9) calculates the completion time. Calculating tardiness for each job is done using Eq. (10). Equations (11)–(13) define variables of the model. Waste of each machine is shown in Eq. (14). Finally, Eq. (15) is used for defining the resource limitation of molds.

4 Optimization by volleyball premier league (VPL) algorithm

VPL algorithm is a newly proposed meta-heuristic algorithm founded on some realistic metaphors of a volleyball tournament [82]. To show validity of the proposed algorithm, a comprehensive analyse based on 23 test instances is organized in which nine eminent metaheuristic comprising GA, Harmony Search algorithm (HS), Fire Fly Algorithm (FA), Soccer League Competition (SLC), Differential Evolution (DE), Sin Cosine Algorithm (SCA), PSO, League Championship Algorithm (LCA) and Artificial Bee Colony (ABC) were compared to the suggested VPL. The results of the VPL show its superiority in comparison to its rivals. To show the main structure of this algorithm, we have to consider the coach, formation of team and substitute bench. These parts play important role in designing the algorithm which has heightened the contribution of VPL in exploration and exploitation. The solution representation of this algorithm contains two parts: firstly, active part contains formation ( ) used for evaluating fitness function, and secondly, passive part stores the information of substitution (**S**) which is used for specific strategy in this part. In this paper, we use $\mathbb{Z}$ and $\mathbb{Z}$**S** for evaluating the active and passive parts, respectively. Typically, there are more than two teams in a division, in which all the teams must play to each other in a specific period. To reach this goal, an organized schedule is proposed by authority of division about the time and place of each game that are exactly determined. In VPL, an specefic procedure, Single Round Robin (SRR) method, is used to generate league schedule. Logically, winner and loser teams are determined after the match. In VPL, the loser team follows some specific process, i.e. knowledge sharing strategy, repositioning strategy and finally substitution strategy to enhance its wretched performance, and accordingly, the winner team tracks the other process to keep its encouraging performance. In the next stage of knowledge sharing strategy, the valuable information is shared among all team members to reach a comprehensive insight throughout the competition, and then, some players are swapped with ones who are positioned in substitutions bench with respect to repositioning strategy in which the information of and **S** property is exchanged to each other. Another process used in this algorithm which is followed by winner teams is winner strategy, in which the winner teams in competition reconstruct their position according to the position of the best team in the league. The most important stage of this algorithm is implemented as learning strategy in which all position of teams are updated for and **S** property based on top 3 teams participating in the current tournament. In the season transfer strategy, players can transfer to other teams. And finally, like a regular sport, the best teams of lower division are replaced to the worst teams of upper division by using promotion and relegation process. So that, the worst teams are removed from the league, and the new solutions generated randomly are added to the league as promoted teams from the lower division, respectively.

5 Multi-objective Pareto-based Metaheuristic algorithm

In this section, the basic assumptions of constrained multi-objective optimization are considered. A common continuous and unconstrained multi-objective optimization problem can be written as follows [83].

$$\min_{x \in \mathcal{F}} = (f_1(u), f_2(u), \ldots, f_m(u))^T, \tag{16}$$

where $\mathcal{F}$ is defined as the decision space, $u = (u_1, u_2, \ldots, u_n)$ is denoted as a decision vector defined in $\mathcal{F}$, and m is the number of objective functions.

The following mapping function defines the objective function bounded in the decision space:

$$F : \mathcal{F} \rightarrow R^m, \tag{17}$$

where R^m is denoted as the objective space of the problem. In any multi-objective problem, it is inevitable to sacrifice one goal while the decision maker is trying to improve the other goal.

Definition 1. (Pareto-dominance): Let $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n)$ are defined as decision vectors, u is said to dominate vector v (shown as $u \succ v$) if and only if

$$(\forall i\in\{1,2,\ldots,n\} : f_i(u_i)\leq f_i(v_i))\wedge\Big(\exists j\{1,2,\ldots,n\} : f_i(u_i) < f_i(v_i) \tag{18}$$

Definition 2. (Pareto-optimal): A decision vector u is defined as a Pareto optimal if and only if

$$\neg\exists v\in\mathcal{F} : v\succ u \tag{19}$$

Definition 3. (Pareto-optimal set): The set *PS* includes all Pareto optimal decision vectors, defined as

$$PS = \{u|\neg\exists v\in\mathcal{F} : v\succ u\}` \tag{20}$$

Definition 4. (Pareto-optimal front): The set *PF* comprises the values of all the objective functions point out the Pareto-optimal decision vectors in *PS*.

$$PF = \Big\{F(u) = (f_1(u), f_2(u), \ldots, f_m(u))^T | u\in PS\Big\} \tag{21}$$

5.1 Multi-objective volleyball premier league

This part devoted to elucidating the main structure of Multi-Objective Volleyball Premier League (MOVPL). To date various methods have been developed and introduced to cope with multi-objective function in MOEAs. To manage the problem, in this algorithm we use crowding distance concept used in NSGA-II. Generally speaking, the execution of MOVPL follows the main structure of basic VPL algorithm which performs the crowding distance concept (Table 1).

As can be seen in the above Psoudeo code, the MOVPL begins with defining parameters and then letting the set *PF* equal null, the next step is setting first population as *Pt*. Evaluating *Pt* and generating fronts based on non-domination sorting method and crowding distance form the next step. After that set the non-dominated solutions as set of *PF*.

Afterward, the main loop will be started, and the loop is repeated until the specified number of iteration is done. At the beginning in the main loop, set the $Rt = Pt$ and then do the following steps. In the first step in the main loop, winner and loser teams should be determined for *Rt* and follow the knowledge sharing and winner strategy process. In the following step of learning phase, the season transfer process and repositioning strategy must be done for *Rt*. The *Qt* will be generated by the non-dominated sorting method. Consequently, according to each level of non-dominated sorting method, the solution ranks are estimated where the former stage consists of resolutions which have the top rates. Subsequently, the crowding distance among individuals is calculated for every stage with the purpose of using the selection procedure. For calculating the crowding distance, total resolutions for the first stage are denoted as $D_l = 0$, afterwards per any objective function, the set is sorted in the worst-to-best order. Therefore, the vector of sorted indices is presented as follows:

$$I_m = sort(f_m >) \tag{22}$$

Let m represent the number of objectives, crowding distance of lth resolution per any level is calculated using a linear distance criterion mention below:

$$D_{I_l^m} = D_{I_l^m} + \frac{f_m^{(I_{l+1}^m)} - f_m^{(I_{l-1}^m)}}{f_m^{max} - f_m^{min}} \tag{23}$$

Where f_m^{max} and f_m^{min} denote the maximum and minimum values of the m-th objective function, respectively. Moreover, lowest and highest amounts of function are assigned by unbounded amounts of distance for each objective function.

An operator called *binary tournament* is used for selection. Calculating the rating of the crowding distance is required for each member, therefore the first two members are chosen in the population. For the sake of selecting a member, the crowding distance and the rating for each solution are considered. Afterwards the solution is chosen based on lower rating. In case of having an equal rank, the one with the larger crowding distance will be chosen. Then, the selection, crossover and mutation operators are used to generate new crowd of offspring (Q_t) with a size of N. The algorithm introduces R_t as an integration of parents (P_t) and offspring, therefore $|R_t| = 2N$. Next, R_t the above method is used for sorting, and N best members are chosen for the subsequent descendant (P_{t+1}). The number of iteration of this procedure relies on the stopping condition. In the last step of MOVPL, the best solutions for the multi-objective optimization are calculated and illustrated in Pareto fronts set. Solution representation relies on the type of optimization problem.

This section defines the solution representation for solving the problem using in MOEAs algorithm. This special case of the scheduling problem is to deal with job sequencing and

Table 1. Psoudeo code for MOVPL.

```
Input: t (Generation)=0, parameters, cost function
Output: P_f (Nondominated set)
Initialization
Evaluate the initial population as P_t
Generate the fronts usig non-dominated sorting and crowding distance
While t < T
      Generate a league schedule
      Determine winner and loser teams for R_t
      Apply different strategies for winner and loser teams
      Apply the Learning phase for R_t
      Apply the season transfer process for R_t
      Apply the Repositoning strategy for R_t
      Calculate  Q_t = Q_t − R_t + P_t
      Evaluate Q_t, Generate fronts and Redefine P_t = P_t − Q_t
      Set the non-dominated solutions as P_f
      t = t + 1
 End
End While
```

limitation of machine availability regarding related molds. Therefore, we suggest a three-part encoding scheme, namely, $\alpha = [x_1; x_2; x_3]$ where the size of all three parts is same. The following formula is used to determine the size of each part.

$$n = \sum_{i=1}^{N} Mo(i), \tag{24}$$

where $Mo(i)$ represents the total number of eligible injection machines to carry out the operations O_i, and n defines the size of each section for any given solutions. On sequent, the size of any solution is equal to $3 \times n$. Principally, n represents the length of any part of the solution representation. The outline of each solution is illustrated in Figs. 1, 2 and 3

$$\alpha = \{x_1 = \{x_{11}, x_{12}, \ldots x_{1n}\} \;\; x_2 = \{x_{21}, x_{22}, \ldots, x_{2n}\} \;\; x_3 = \{x_{31}, x_{32} \ldots x_{3n}\}\}$$

$$A = \{X_1 = \{X_{11}, X_{12}, \ldots X_{1n}\} \;\; X_2 = \{X_{21}, X_{22}, \ldots, X_{2n}\} \;\; X_3 = \{X_{31}, X_{32} \ldots X_{3n}\}\}$$

Based on Fig. 3, set α would be changed to set A throughout solution procedure. Consequently, set A has three parts as the same as set α. The former section of the set α illustrates weight of any given solution to determine the final solution. Moreover, to transform x_1 to X_1, a method has been considered below:

No table of figures entries found. **Psoudeo code for calculating X_1**

Input: Mo, Q, x_1
Output: X_1
$index_2 = 0$
For $i = 1{:}n$
  $index_2 = index_1 + Mo(i) - 1$
  $weight = x_1(index_1{:}index_2)$
  $cumsum = \frac{weight}{\sum_{i=1}^{m} weight}$
  $f = round(Q(i) \times cumsum)$
  $X_1(index_1{:}index_2) = f$
End

For illustrating this encoding related to the former section of the solution, in the following the solution representation for 4 machines and 5 jobs is given for set x, Mo and Q:

$$x = \begin{cases} x_1 = \{0.82\ 0.98\ 0.73\ 0.34\ 0.58\ 0.11\ 0.91\ 0.88\ 0.81\ 0.26\ 0.59\ 0.02\} \\ x_2 = \{0.43\ 0.31\ 0.16\ 0.18\ 0.42\ 0.09\ 0.60\ 0.47\ 0.70\ 0.71\ 0.64\ 0.03\} \\ x_3 = \{0.07\ 0.32\ 0.53\ 0.65\ 0.41\ 0.82\ 0.72\ 0.97\ 0.53\ 0.33\ 0.11\ 0.61\} \end{cases}$$

$$Mo = [2\ \ 2\ \ 3\ \ 3\ \ 2]$$

$$Q = [10000\ \ 12000\ \ 14000\ \ 8000\ \ 25000]$$

In this example, and using Eq. (24), the size of solution is $n = 12$. So, due to the aforementioned method, the former section of X is presented in the following:

$$X_1 = [4562\ \ 5438\ \ 8158\ \ 3842\ \ 5117\ \ 944\ \ 7939\ \ 3594\ \ 3341\ \ 1065\ \ 24088\ \ 912]$$

The latter section related to machine assignment. For determining X_2, the following function has been considered.

No table of figures entries found. **Psoudeo code for calculating X_2**

Input: Mo, x_2
Output: X_2
$M = size(Mo)$ /* number of operations.*/
$X_2 = \phi$
$index_2 = 0$
For $i = 1{:}n$
  $index_2 = index_1 + Mo(i) - 1$
  $p = x_2(index_1{:}index_2)$
  $v = 0$
  Establish matrix V with size M
  For $j = 1{:}M$
    $v = (1/M) + v$
    $V(j) = v$
  End
  Establish matrix U with size x_2
  For $k = 1{:}(size\ x_2)$
    $\xi =$ find the first j where $p(k) < V$
    $U(k) = \xi$
  End
  $X_2 = [X_1\ U]$
End

Using the aforementioned method, the following vector will be generated as the second part of set X. The first operation is assigned to machine 1, the second to machine 2, and so on.

$$X_2 = [2\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 2\ \ 2\ \ 3\ \ 3\ \ 2\ \ 1]$$

The final part of the solution representation is related to the sequencing of jobs. On that account, X_3 is computed

$$\alpha = \{x_1 = \{x_{11}, x_{12}, \ldots x_{1n}\} \;\; x_2 = \{x_{21}, x_{22}, \ldots, x_{2n}\} \;\; x_3 = \{x_{31}, x_{32} \ldots x_{3n}\}\}$$

$$A = \{X_1 = \{X_{11}, X_{12}, \ldots X_{1n}\} \;\; X_2 = \{X_{21}, X_{22}, \ldots, X_{2n}\} \;\; X_3 = \{X_{31}, X_{32} \ldots X_{3n}\}\}$$

Fig. 1 General scheme of a solution

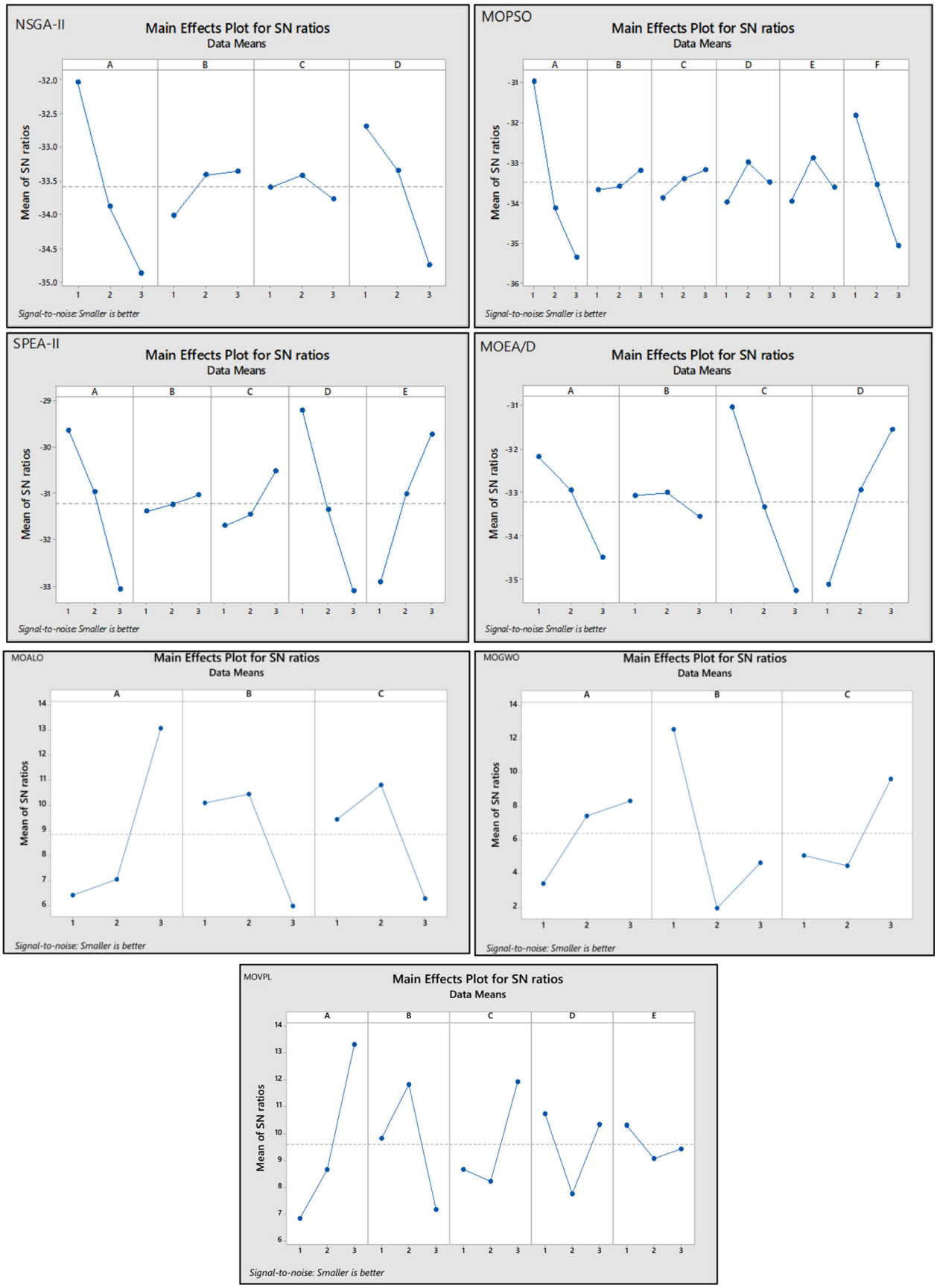

Fig. 2 The output of the Taguchi method

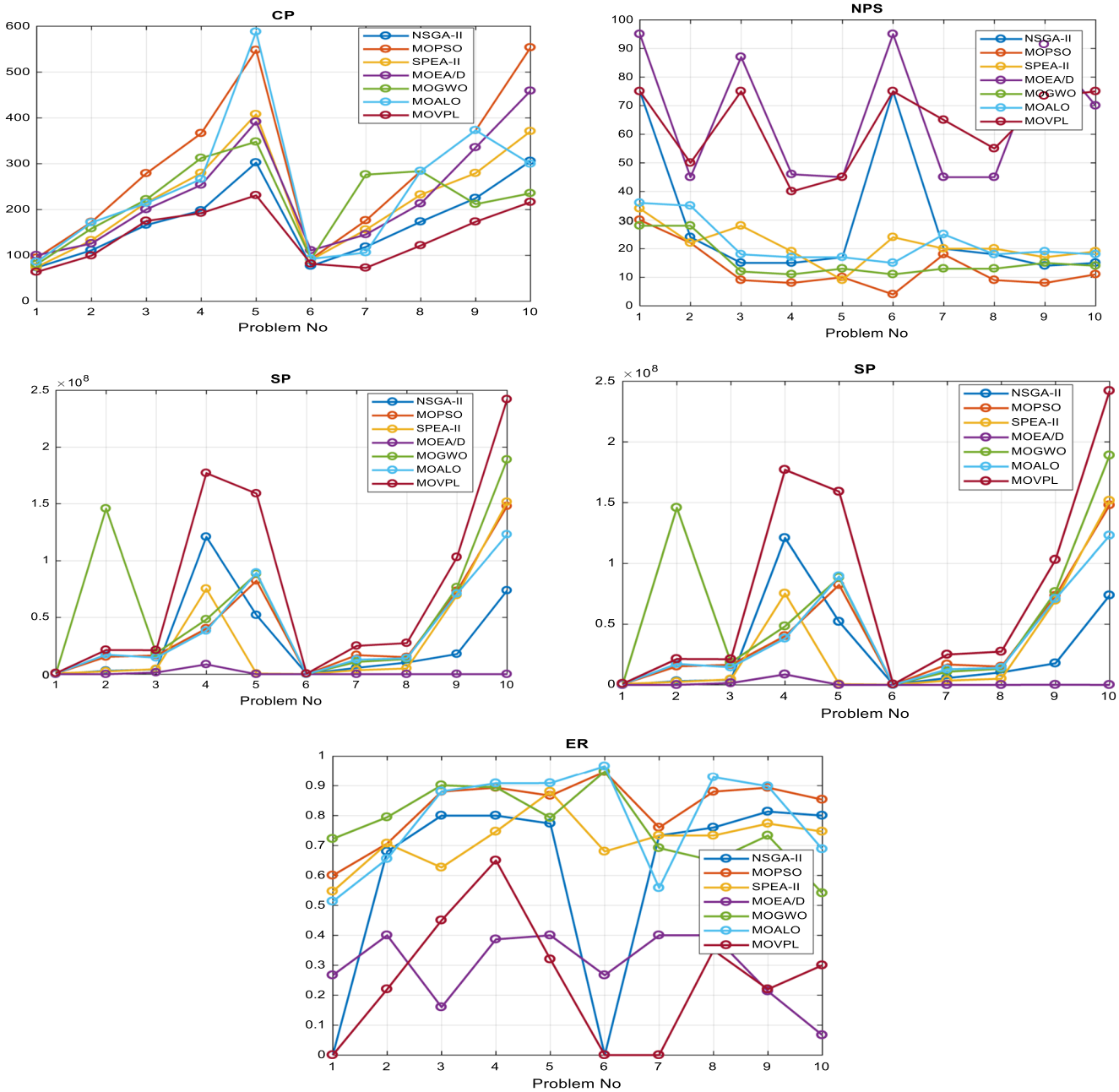

Fig. 3 Graphical comparisons of all MOEAs in terms of different metrics

according to sorting ranks of the second part of set X members in ascending order. So, X_3 is given as follows:

$$X_3 = [1\ \ 11\ \ 2\ \ 10\ \ 5\ \ 3\ \ 9\ \ 12\ \ 4\ \ 7\ \ 6\ \ 8]$$

6 Experiments and output analysis

The following five standard metrics are represented with the aim of comparing the performances of the proposed Pareto-based multi-objective algorithms.

- **Number of Pareto Solutions (NPS):** this metric measures the number of the Pareto solutions in Pareto optimal front [84]:

$$NPS = |PF| \tag{25}$$

- **The Spacing (SP):** this metric [85] is used to quantify and measure the convergence of the algorithm:

$$SP = \sqrt{\left(\frac{1}{|PF|}\right)\sum_{i=1}^{n}\left(\overline{d}-d_i\right)^2} \tag{26}$$

The following metric measures the standard deviation of the distances between members, where $d_i = min_{\overrightarrow{y} \in PF} \sum_{k=1}^{K} \left(|o_k(\overrightarrow{x}) - o_k(\overrightarrow{y})| \right)$ represents the minimum distance of resolution $\overrightarrow{x}$ from the rest of resolutions and $\overline{d}$ represents the median amount for d_i.

- **CPU Time (CT):** this metric represents the computational (CPU) time of running the algorithm [84].
- **Maximum Spread (MS):** Eq. (27) illustrates the spread metric of Pareto solutions [84]:

$$MS = \sqrt{\sum_{i=1}^{I} (\min f_i - \max f_i)^2} \tag{27}$$

Where $minf_i$ and $maxf_i$ for each fitness function are the lowest and highest values which are calculated for all non-dominated members by various manners [86].

- **Error Ratio (ER):** the ER is computed using the following formula:

$$ER \frac{\sum_{h=1}^{n} e_h}{n} \tag{28}$$

Where e_h is equal to one if a solution is not related to Pareto front otherwise zero, and n denotes the population size.

Ten test problems are utilized for experimentation. In addition, each problem is run thirty times with the aim of reducing uncertainties. Afterward, the last stage is done by calculating the averages of thirty runs. The Taguchi method [87] is utilized to detect the optimum level for any parameters in various algorithms. To calculating the amount of response variations, the signal to noise ratio (S/N) is used [88]:

$$S/N = -10 \times log\left(S\left(Y^2/N\right)\right), \tag{29}$$

Where N denotes the number of orthogonal arrays and Y idicates the response. To accomplish the procedure, identifying the levels of parameters needs to be done as in Table 2.

To make comparison among different algorithms, Table 2. presents the multi-objective metrics amounts on the 10 test problems. It should be noted that Matlab software (Version 8.3.0.532, R2014a) is used to code the proposed meta-heuristic algorithms, and the programs are executed on a 2.50 GHz core i5 CPU with 4.00 GB of RAM under iOS operating system.

Table 2 Spans and stages of MOEA parameters

MOEA	Parameter	levels of Parameters		
		Lowest	Median	Highest
NSGA-II	*nPop*	25	50	75
	P_c	0.30	0.60	0.90
	P_m	0.30	0.60	0.90
	MaxIt	50	100	150
MOPSO	*nPop*	25	50	75
	c_1	1	1.2	1.4
	c_2	1	1.2	1.4
	ω	0.2	0.4	0.6
	nRep	30	50	70
	MaxIt	50	100	150
MOEA/D	*nPop*	25	50	75
	T	5	15	25
	MaxIt	50	100	150
	nArchive	20	40	60
SPEA-II	*nPop*	25	50	100
	P_c	0.30	0.60	0.90
	P_m	0.30	0.60	0.90
	MaxIt	50	100	75
	nArchive	20	40	60
MOGWO	*nPop*	25	50	75
	nRep	30	50	70
	MaxIt	50	100	150
MOALO	*nPop*	25	50	75
	nRep	30	50	70
	MaxIt	50	100	150
MOVPL	*nPop*	25	50	75
	δ_ks	0.2	0.5	0.7
	δ_st	0.2	0.5	0.7
	δ_rs	0.2	0.5	0.7
	MaxIt	50	100	150

Orthogonal arrays are used in Taguchi method with the aim of studying all factors concurrently. Afterwards, the L9 design is used for NSGA-II, MOEA/D, MOGWO, and MOALO at the same time, L27 is performed for MOPSO, SPEA-II, and MOVPL up to now. This approach has been executed in various researches, for instance [89, 90]. Accordingly, the schemes of consequence for S/N ratios of all four compaed algorithms are illustrated in Fig. 2. Note that WOGWO and MOALO are two relatively new bio-inpired metaheuristics, where MOGWO stands for multi-objective grey wolf optimization [91], and MOALO stands for multi-objective ant lion optimization [92].

According to Fig. 2, the best levels of each algorithm are determined. To show compatibility of obtained results, we design 10 problems in this way. Results of different metrics for these problems are shown in Table 3.

Table 3 MOEA metrics for proposed Pareto-based metaheuristics on 10 problems

Problem No	Metrics	MOEAs						
		NSGA-II	MOPSO	SPEA-II	MOEA/D	MOGWO	MOALO	MOVPL
1	CP	73.537302	94.502409	74.648648	100.37028	80.3866573	85.6768985	63.87125
	NPS	75	30	34	95	28	36	75
	SP	248,742.87	620,395.39	825,574.2	0	732,545.796	637,895.371	902,556.4
	MS	4.784E+09	1.41E+10	1.38E+10	0	1,460,000,000	1.399E+10	2.22E+10
	ER	0	0.6	0.5466667	0.2666667	0.72235673	0.5132071	0
2	CP	111.21738	172.75733	132.83757	125.9493	158.558691	171.688529	99.50976
	NPS	24	22	22	45	28	35	50
	SP	2,994,755.2	15,270,180	2,395,063.4	0	145,702,180	17,270,179.7	21,345,125
	MS	2.40E+12	4.92E+13	3.88E+11	0	4.2157E+13	4.9157E+13	5.37E+13
	ER	0.68	0.7066667	0.7066667	0.4	0.79458557	0.65548796	0.22
3	CP	166.37319	278.98956	215.47749	200.33855	222.003127	214.013782	175.2004
	NPS	15	9	28	87	12	18	75
	SP	4,180,978.3	16,546,397	4,502,145.4	1,484,485.4	18,546,397	14,546,397	21,155,195
	MS	8.48E+12	2.13E+14	3.21E+12	6.674E+09	2.1311E+14	2.1311E+14	2.33E+14
	ER	0.8	0.88	0.6266667	0.16	0.90131402	0.88090556	0.45
4	CP	197.82993	366.43384	279.65779	253.76222	312.466887	266.454759	192.33
	NPS	15	8	19	46	11	17	40
	SP	120,801,223	40,265,231	75,249,957	8,640,030.2	48,265,231.4	38,265,231.4	1.77E+08
	MS	1.01E+15	1.74E+15	7.72E+14	1.77E+12	2.74E+15	1.55E+15	2.93E+15
	ER	0.8	0.8933333	0.7466667	0.3866667	0.89460615	0.908085	0.65
5	CP	302.49011	547.87928	407.87187	391.50943	347.885671	587.896792	231.0731
	NPS	17	10	9	45	13	17	45
	SP	52,073,201	82,418,611	580,604.25	0	88,418,611.1	89,418,611.1	1.59E+08
	MS	7.33E+15	2.29E+16	1.99E+12	0	1.58E+16	1.95E+16	3.36E+16
	ER	0.7733333	0.8666667	0.88	0.4	0.79279437	0.90852025	0.32
6	CP	76.92058	92.262313	90.500047	110.97324	92.2903702	92.3112111	81.7327
	NPS	75	4	24	95	11	15	75
	SP	383,637.57	21,327.735	107,747.99	0	29,327.7628	33,327.7614	442,572.5
	MS	3.87E+09	4.348E+09	1.727E+09	0	2,547,951,928	5,547,951,928	9.58E+09
	ER	0	0.9466667	0.68	0.2666667	0.94803447	0.96614026	0
7	CP	118.61654	176.22721	156.07945	145.73646	276.257557	106.275478	72.84181
	NPS	20	18	20	45	13	25	65
	SP	5,623,971.5	16,810,025	3,544,420	0	10,810,025	12,810,025	24,981,904
	MS	4.41E+12	7.06E+13	1.04E+12	0	7.15E+13	7.06E+13	7.63E+13
	ER	0.7333333	0.76	0.7333333	0.4	0.69165571	0.55829532	0
8	CP	173.58775	283.87946	232.42208	213.05613	283.914592	283.917705	121.6601
	NPS	18	9	20	45	13	18	55
	SP	10,303,577	15,004,174	5,075,675.9	0	13,304,173.6	14,084,173.6	27,415,337
	MS	1.46E+13	3.35E+14	3.85E+12	0	3.135E+14	3.346E+14	3.50E+14
	ER	0.76	0.88	0.7333333	0.4	0.64888602	0.92894085	0.35
9	CP	224.79828	372.89482	279.39663	335.47336	211.897851	372.935298	173.4679
	NPS	14	8	17	91	15	19	73
	SP	17,774,666	73,322,303	69,547,014	92,816.184	76,522,303.1	70,922,303.1	1.03E+08
	MS	1.27E+14	1.58E+15	5.16E+14	113,681,048	1.48E+15	1.68E+15	1.78E+15
	ER	0.8133333	0.8933333	0.7733333	0.2133333	0.73320199	0.89807743	0.22

Table 3 (continued)

Problem No	Metrics	MOEAs						
		NSGA-II	MOPSO	SPEA-II	MOEA/D	MOGWO	MOALO	MOVPL
10	CP	305.76538	553.53399	370.81719	458.98984	235.551918	299.563896	216.2568
	NPS	15	11	19	70	14	18	75
	SP	73,746,628	147,847,102	151,523,741	4.50E-08	188,847,102	122,847,102	2.42E+08
	MS	2.14E+15	2.05E+16	2.02E+15	3.91E+12	2.45E+16	1.95E+16	2.54E+16
	ER	0.8	0.8533333	0.7466667	0.0666667	0.54096271	0.68798617	0.3

Additionally, graphical comparisons of 10 problems solved using the proposed algorithms are presented in Table 2 for five multi-objective metrics, i.e. CPU time consumed (CP), number of pareto solutions (NPS), spacing (SP), maximum spread (MS), and error ratio (ER).

As can be seen in Table 2 and Figs. 4 and 5, in terms of computation time (i.e. CP metric) our proposed algorithm consumes significantly less time for almost all problems in comparison with the other six metaheuristic algorithms. Only for problems 3 and 6 the NSGA-II algorithm takes less time in comparison with the MOVPL (166 vs. 175 and 77 vs. 82, respectively). In terms of the NPS metric, just the MOEA/D algorithm gains a little bit better Pareto front solutions for problems 1, 3, 4, 6 and 9 compared to the MOVPL. In spite of the fact that the MOGWO algorithm defeats the MOVPL algorithm just for problem 2 in terms of the SP metric, in all other cases, the situations are vise versa. In the MS metric, the MOVPL algorithm defeats all algorithms except the NSGA-II algorithm for problem 3. For the last ER metric, the MOVPL wins all algorithms for all problems, except the MOEA/D algorithm for problems 3, 4, 9 and 10. Therefore, it can be inferred that the MOVPL is superior to the other six well-known metaheuristic algorithms based on the five aforementioned metrics.

The Pareto fronts of 10 problems obtained using proposed algorithms are illustrated in Figs. 4 and 5. As can be seen, each algorithm has a different Pareto front for its test function. The Pareto front of the tardiness objective function (F1) and the defective objective function (F2) are illustrated on all 10 problems. The solutions obtained by MOVPL and MOGWO indicate that both F1 and F2 are equally preferred, while F1 objective function for problems 2–8 is less important and dominated by F2 objective function from solutions obtained by MOPSO, MOALO and MOEA/D. By contrast, the F2 objective function is dominated by these problems by other algorithms. It is worth noting that for all problems the MOVPL equally prefers both F1 and F2 and gains minimal solutions in comparison with other metaheuristic algorithms.

In the above figure, for the purpose of comparing all algorithms, all obtained Pareto fronts are presented simply. To gain the best solution, it needs to cumulate Pareto front solutions offered by all. Let $\mathcal{L}$ is a set containing every individual of *PF* offered by all MOEAs, defined as follows:

$$\mathcal{L} = \left\{ x \in PF_i | i \in \{MOEAs\} \right\} \tag{30}$$

Because there is no general agreement between researchers about which MOEA has supreme performance in comparison with other algorithms, the Pareto fronts

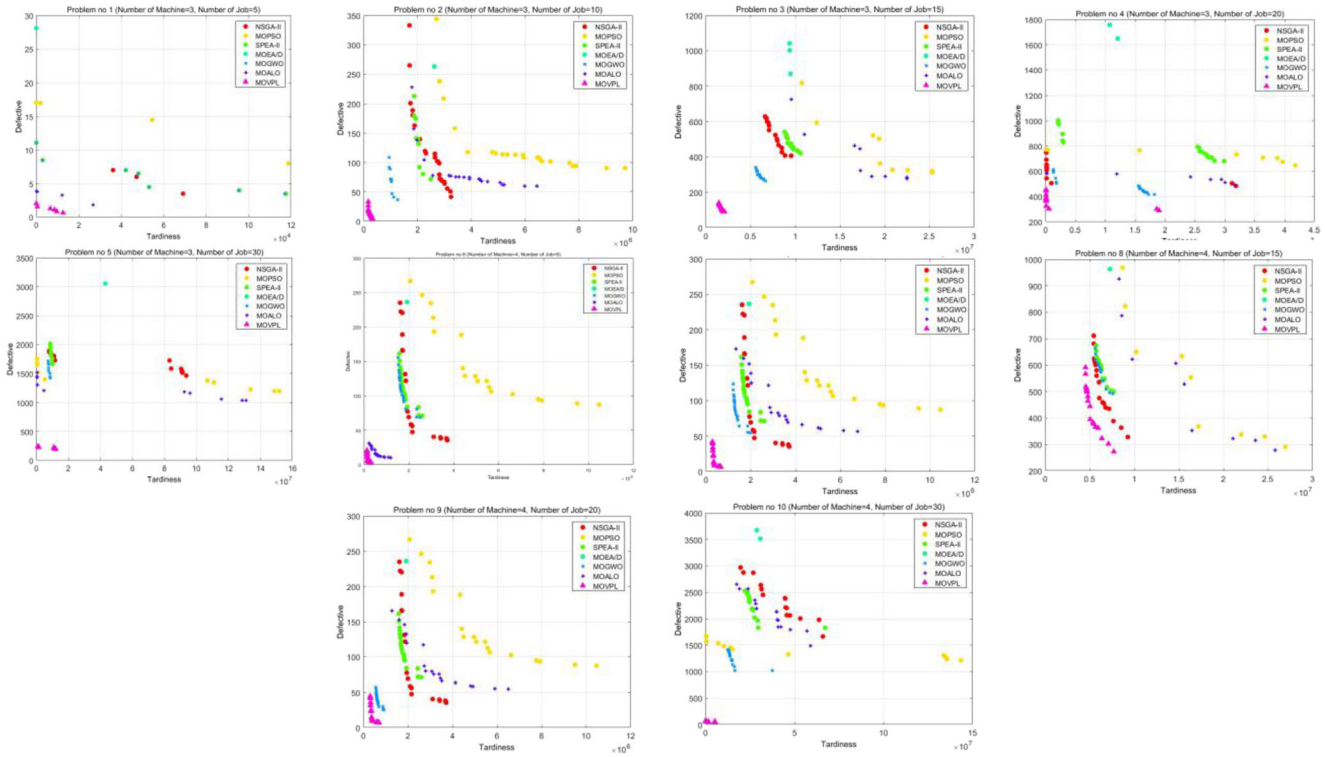

Fig. 4 Pareto fronts for 10 test problems

discover process has been accomplished for set $\mathcal{L}$ to specify ultimate *PF* for all test problems. Based on the feasibility of solutions and in comparison to the other algorithms, one (or maybe two) algorithm(s) will be better solution technique. Fig. 5 illustrates the digit of solutions in the ultimate Pareto front based on all MOEAs for total test problems.

As can be seen in Fig. 5, MOVPL has the best performance for all problems. One thing which is equally important that in problems 4 and 5, the NSGA-II, the MOPSO, and the MOALO have better performance among other problems; though the competition is between MOVPL and the NSGA-II, and the former has superiority on the latter. It can be also argued that in problem 5 the NSGA-II loses its performance and the MOVPL shows a huge superiority on the MOPSO and MOALO algorithms.

The suggested MOEAs are actuarially compared from various metrics with no major difference. Hence, the ANOVA experiments (analysis of variance) are guaranteed at at 95% confidence level. The outcomes of ANOVA test are shown in Table 4.

The seven algorithms have been ranked according to the five comparison metrics. As shown in Table 5, in terms of CP, SP, MS and ER, the MOVPL is able to defeat the other rival algorithms. The NPS is the only metric in which the MOEA/D has pushed down the MOVPL to the second rank. Overall, the MOVPL has won four out of five metrics against the other six well-known multi-objective evolutionary algorithms.

7 Conclusions

This research focused on the parallel machine scheduling with splitting jobs on a set of identical machines minimizing wastes and total tardiness. The mixed integer programming model is formulated using data from a real plastic injection industry. The nature of the problem in this industry is that two conflicting objectives, minimizing total tardiness and minimizing total wastes, should be taken into account. To make the model more realistic, it has been developed under a multi-objective framework, making the model difficult to solve in optimality using classical solution methods.

Regarding the fact that the problem at hand is NP-hard, metaheuristic algorithms have been applied. Since the mathematical model of the problem is bi-objective within the multi-objective framework, MOPSO, NSGA-II, MOEA/D MOGWO, MOALO and SPEA2 are compared with the proposed new algorithm called MOVPL. The MOVPL uses the crowding distance concept applied in NSGA-II as an extension of the Volleyball Premier League (VPL). For comparing the efficiency of all rival algorithms, the five most common standard metrics in the field of multi-objective algorithms (i.e.

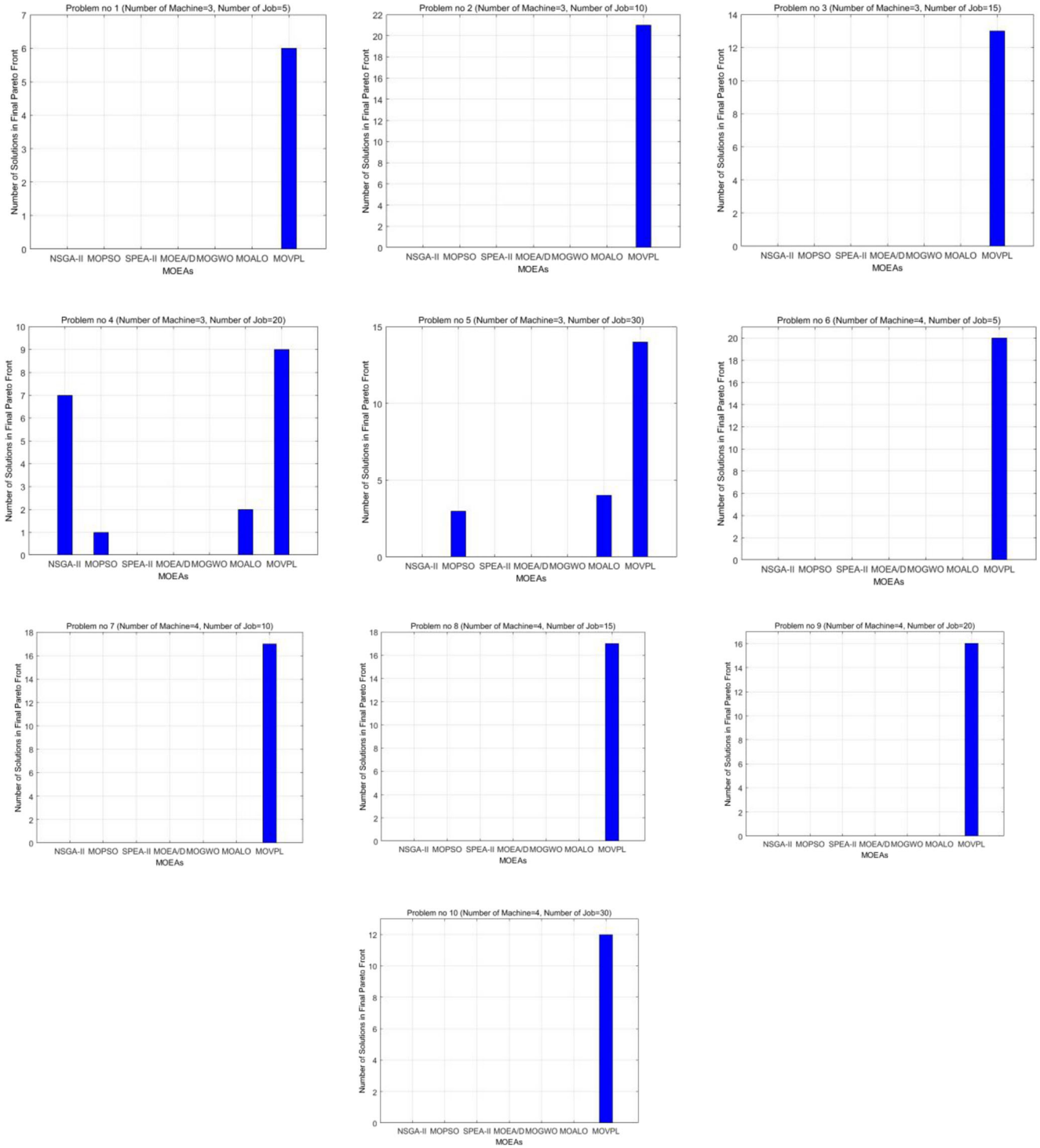

Fig. 5 Solutions in the final Pareto front of MOEAs

NPS, SP, CT, MS and ER) are used. As can be seen from the computational results, the proposed MOVPL algorithm has reached the first rank in four standard metrics except for the Number of Pareto Solutions (NPS) in which the MOVPL has been ranked the second position.

Table 4 Results of ANOVA test at a 95% of confidence interval

Metric	*p* value	The outcome
NPS	0	The null hypothesis is rejected
ER	0	The null hypothesis is rejected
SP	0.151	The null hypothesis cannot be rejected
MS	0.621	The null hypothesis cannot be rejected
CP	0.312	The null hypothesis cannot be rejected

Table 5 Rankings of the MOEAs according to the five metrics

Metrics	ranks of MOEAs						
	NSGA-II	MOPSO	SPEA-II	MOEA/D	MOGWO	MOALO	MOVPL
CP	2	7	3	4	5	6	**1**
NPS	4	7	5	**1**	6	3	2
SP	6	4	5	7	2	3	**1**
MS	5	2	6	7	4	3	**1**
ER	3	7	4	2	5	6	**1**

The originality of the paper is twofold. Firstly, we build a mathematical model for a challenging multi-objective scheduling problem in real-world plastic injection molding industry. Secondly, based on the original VPL algorithm which was proposed for the optimization on single objective and was evaluated on some simple test functions, in this paper we redesign the algorithm to enable global optimization on multiple objectives and evaluate it on the NP-hard real scheduling problem against some state-of-the-art algorithms.

As our future work, we will continue to refine the proposed MOVPL algorithm by taking into account new constraints to the problem such as preemption, maintenance times, and uncertainty in the volume of jobs.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Pinedo ML (2008) Scheduling: theory, algorithms, and systems, Third edn. Original edition published by Prentice Hall, New York, p 671
2. Lenstra JK, Kan AR, Brucker P (1977) Complexity of machine scheduling problems. Ann Discrete Math 1:343–362
3. Koulamas C (1994) The total tardiness problem: review and extensions. Oper Res 42(6):1025–1041
4. Du J, Leung JY-T (1990) Minimizing total tardiness on one machine is NP-hard. Math Oper Res 15(3):483–495
5. Lawler EL, Lenstra JK, Kan AR (1982) Recent developments in deterministic sequencing and scheduling: a survey, In Deterministic and stochastic scheduling. Springer, pp 35–73
6. Koulamas C (1997) Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. Nav Res Logist (NRL) 44(1):109–125
7. Elmaghraby SE, Park SH (1974) Scheduling jobs on a number of identical machines. AIIE Transac 6(1):1–13
8. Barnes J, Brennan J (1977) An improved algorithm for independent jobs to reduce the mean finishing time. AIIE Transac 17(1977): 382–387
9. Yalaoui F, Chu C (2002) Parallel machine scheduling to minimize total tardiness. Int J Prod Econ 76(3):265–279
10. Lawler EL (1964) On scheduling problems with deferral costs. Manag Sci 11(2):280–288
11. Root JG (1965) Scheduling with deadlines and loss functions on k parallel machines. Manag Sci 11(3):460–475
12. Pritsker AAB, Waiters LJ, Wolfe PM (1969) Multiproject scheduling with limited resources: a zero-one programming approach. Manag Sci 16(1):93–108
13. Ranjbar M, Davari M, Leus R (2012) Two branch-and-bound algorithms for the robust parallel machine scheduling problem. Comput Oper Res 39(7):1652–1660
14. Figielska E (2013) An Ant Colony Optimization Algorithm for Scheduling Parallel Machines with Sequence-Dependent Setup Costs, Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki, no. 9, pp 15–26
15. Chen Z-L (2004) Simultaneous job scheduling and resource allocation on parallel machines. Ann Oper Res 129(1–4):135–153
16. Edis EB, Oguz C (2012) Parallel machine scheduling with flexible resources. Comput Ind Eng 63(2):433–447
17. Lee W-C, Chuang M-C, Yeh W-C (2012) Uniform parallel-machine scheduling to minimize makespan with position-based learning curves. Comput Ind Eng 63(4):813–818
18. Montagne E (1969) Sequencing with time delay costs, Industrial Engineering Research Bulletin. Arizona State Univ 5:20–31
19. Wilkerson L, Irwin J (1971) An improved method for scheduling independent tasks. AIIE Transac 3(3):239–245
20. Morton TE, Rachamadugu RM, Vopsalainen A (1984) Accurate myopic heuristics for tardiness scheduling
21. Ho JC, Chang YL (1991) Heuristics for minimizing mean tardiness for m parallel machines. Nav Res Logist (NRL) 38(3):367–381
22. Dogramaci A, Surkis J (1979) Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. Manag Sci 25(12):1208–1216
23. Chen K, Wong J, Ho J (1997) A heuristic algorithm to minimize tardiness for parallel machines, In Proceedings of ISMM International Conference, pp 118–121
24. Baker KR, Bertrand JWM (1982) A dynamic priority rule for scheduling against due-dates. J Oper Manag 3(1):37–42

25. Garai T, Chakraborty D, Roy TK (2019) A fuzzy rough multi-objective multi-item inventory model with both stock-dependent demand and holding cost rate. Granular Comput 4(1):71–88
26. Maity S, Roy A, Maiti M (2019) A rough multi-objective genetic algorithm for uncertain constrained multi-objective solid travelling salesman problem. Granular Comput 4(1):125–142
27. Gupta S, Ali I, Chaudhary S (2020) Multi-objective capacitated transportation: a problem of parameters estimation, goodness of fit and optimization. Granular Comput 5(1):119–134
28. Correa J et al (2015) Strong LP formulations for scheduling splittable jobs on unrelated machines. Math Program 154(1):305–328
29. Tao J (2014) A better online algorithm for the parallel machine scheduling to minimize the total weighted completion time. Comput Oper Res 43:215–224
30. Ouazenea Y, Yalaouia F, Chehadea H, Yalaoui A (2014) Workload balancing in identical parallel machine scheduling using a mathematical programming method. Int J Compu Int Syst 7(sup1):58–67
31. Hsu C-J, Cheng TCE, Yang D-L (2011) Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time. Inf Sci 181(20):4799–4803
32. Baker RK, Trietsch D (2009) Principles of sequencing and scheduling. John Wiley & Sons, Inc
33. Unlu Y, Mason S (2010) Evaluation of mixed integer programming formulations for nonpreemptive parallel machine scheduling problems. Comput Ind Eng 58(4):785–800
34. Serafini P (1996) Scheduling jobs on several machines with the job splitting property. Oper Res 44(4):617–628
35. Xing W, Zhang J (2000) Parallel machine scheduling with splitting jobs. Discret Appl Math 103(1–3):259–269
36. Sarıçiçek İ, Çelik C (2011) Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness. Appl Math Model 35(8):4117–4126
37. Park T, Lee T, Ouk Kim C (2012) Due-date scheduling on parallel machines with job splitting and sequence-dependent major/minor setup times. Int J Adv Manuf Technol 59(1):325–333
38. Shim S-O, Kim Y-D (2008) A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. Comput Oper Res 35(3):863–875
39. Kim Y, Shim S, Kim S, Choi Y, Yoon H (2004) Parallel machine scheduling considering a job-splitting property. Int J Prod Res 42(21):4531–4546
40. Yalaoui F, Chu C (2003) An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent set-up times. IIE Trans 35(2):183–190
41. Zhu Z, Heady RB (2000) Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. Comput Ind Eng 38(2):297–305
42. Shim S-O, Kim Y-D (2007) Minimizing Total tardiness in an unrelated parallel-machine scheduling problem. J Oper Res Soc 58(3):346–354
43. Logendran R, Subur F (2004) Unrelated parallel machine scheduling with job splitting. IIE Trans 36(4):359–372
44. Logendran R, McDonell B, Smucker B (2007) Scheduling unrelated parallel machines with sequence-dependent setups. Comput Oper Res 34(11):3420–3438
45. Liaw C-F, Lin Y-K, Cheng C-Y (2003) Scheduling unrelated parallel machines to minimize total weighted tardiness. Comput Oper Res 30(12):1777–1789
46. Wang W-L, Wang H-Y, Zhao Y-W, Zhang L-P, Xu X-L (2013) Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm. Comput Oper Res 40(5):1196–1206
47. Chen J-F, Wu T-H (2006) Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. Omega 34(1):81–89
48. Vallada E, Ruiz R (2011) A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent set-up times. Eur J Oper Res 211(3):612–622
49. Rocha PL, Ravetti MG, Mateus GR, Pardalos MP (2008) Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. Comput Oper Res 35(4):1250–1264
50. Lin YK, Pfund ME, Fowler JW (2011) Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. Comput Oper Res 38(6):901–916
51. Fanjul-Peyro L, Ruiz R (2012) Scheduling unrelated parallel machines with optional machines and jobs selection. Comput Oper Res 39(7):1745–1753
52. Wang C, Liu C, Zhang Z-h, Zheng L (2016) Minimizing the total completion time for parallel machine scheduling with job splitting and learning. Comput Ind Eng 97:170–182
53. 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
54. Lara AFB, Yalaoui F, Dugardin F, Entzmann F (2016) An efficient heuristic to minimize the Total tardiness in the parallel machines scheduling problem. In: Talbi E-G, Yalaoui F, Amodeo L (eds) Metaheuristics for production systems, vol 60. Springer International Publishing, Cham, pp 241–262
55. Shahvari O, Logendran R (2016) 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:154–176
56. Yin Y, Wang Y, Cheng TCE, Liu W, Li J (2016) Parallel-machine scheduling of deteriorating jobs with potential machine disruptions, Omega, vol. In Press
57. Mensendiek A, Gupta JND, Herrmann J (2015) Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. Eur J Oper Res 243(2):514–522
58. Lee J-Y, Kim Y-D (2015) A branch and bound algorithm to minimize total tardiness of jobs in a two identical-parallel-machine scheduling problem with a machine availability constraint. J Oper Res Soc 66(4):1542–1554
59. Mora B, Mosheiov G (2014) Batch scheduling of identical jobs with controllable processing times. Comput Oper Res 41:115–124
60. Lia X, Huanga Y, Tanb Q, Chenc H (2013) Scheduling unrelated parallel batch processing machines with non-identical job sizes. Comput Oper Res 40(12):2983–2990
61. Zhoua S, Liub M, Chenc H, Li X (2016) An effective discrete differential evolution algorithm for scheduling uniform parallel batch processing machines with non-identical capacities and arbitrary job sizes. Int J Prod Econ 179:1–11
62. Li K, Jia Z-h, Leung JYT (2015) Integrated production and delivery on parallel batching machines. Eur J Oper Res 247(3):755–763
63. Aalaei A, Kayvanfar V, Davoudpour H (2017) A multi-objective optimization for preemptive identical parallel machines scheduling problem. Comput Appl Math 36(3):1367–1387
64. Fu L-L, Aloulou MA, Triki C (2017) Integrated production scheduling and vehicle routing problem with job splitting and delivery time windows. Int J Prod Res 55(20):5942–5957
65. Beezão AC, Cordeau J-F, Laporte G, Yanasse HH (2017) Scheduling identical parallel machines with tooling constraints. Eur J Oper Res 257(3):834–844
66. Wu X, Che A (2019) A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. Omega 82:155–165
67. Liu C, Wang C, Zhang Z-h, Zheng L (2018) Scheduling with job-splitting considering learning and the vital-few law. Comput Oper Res 90:264–274
68. Kim H, Lee J (2018) Uniform Parallel Machine Scheduling with Dedicated Machines, Job Splitting and Setup Resources, In 2018

IEEE 14th International Conference on Automation Science and Engineering (CASE), pp 661–663
69. Safarzadeh H, Niaki STA (2019) Bi-objective green scheduling in uniform parallel machine environments. J Clean Prod 217:559–572
70. Salimifard AAK, Mohammadi D, Moghdani R (2019) Green Fuzzy Parallel Machine Scheduling with Sequence-Dependent Setup in the Plastic Molding Industry. Asian J Manag Sci Appl, in press
71. Xu R, Chen H, Li X (2013) A bi-objective scheduling problem on batch machines via a Pareto-based ant colony system. Int J Prod Econ 145(1):371–386
72. Wang J-Q, Leung J (2014) Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan. Int J Prod Econ 156:325–331
73. Lopes MJP, de Carvalho JMV (2007) A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. Eur J Oper Res 176(3):1508–1527
74. Anand R, Kumar V (2017) Reliable Back-up Facility in Distribution Network. Prog Comput Sci 115:312–321
75. Anand R, Kumar V (2017) Firefly algorithm for reliable protection in distribution networks, In 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), pp 1164–1172
76. Aggarwal D, Chahar V, Girdhar A (2017) Firefly algorithm for the vehicle routing problem with time windows
77. Aggarwal D, Kumar V (2019) Performance evaluation of distance metrics on firefly algorithm for VRP with time windows, Int. J Inf Technol, 11/29
78. Zhu H, Qi X, Chen F, He X, Chen L, Zhang Z (2019) Quantum-inspired cuckoo co-search algorithm for no-wait flow shop scheduling. Appl Intell 49(2):791–803
79. Gaham M, Bouzouia B, Achour N (2018) An effective operations permutation-based discrete harmony search approach for the flexible job shop scheduling problem with makespan criterion. Appl Intell 48(6):1423–1441
80. Ombuki BM, Ventresca M (2004) Local Search Genetic Algorithms for the Job Shop Scheduling Problem. Appl Intell 21(1):99–109
81. Nouri HE, Driss OB, Ghédira K (2016) Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment. Appl Intell 45(3):808–828
82. Moghdani R, Salimifard K (2018) Volleyball Premier League Algorithm. Appl Soft Comput 64:161–185
83. Lin Q, Li J, Du Z, Chen J, Ming Z (2015) A novel multi-objective particle swarm optimization with multiple search strategies. Eur J Oper Res 247(3):732–744
84. Hajipour V, Fattahi P, Tavana M, Di Caprio D (2016) Multi-objective multi-layer congested facility location-allocation problem optimization with Pareto-based meta-heuristics. Appl Math Model 40(7):4948–4969
85. Coello CAC, Van Veldhuizen DA, Lamont GB (2007) Evolutionary algorithms for solving multi-objective problems. Springer
86. Maghsoudlou H, Kahag MR, Niaki STA, Pourvaziri H (2016) Bi-objective optimization of a three-echelon multi-server supply-chain problem in congested systems: Modeling and solution. Comput Ind Eng 99:41–62
87. Li J, Kwan RS (2004) A meta-heuristic with orthogonal experiment for the set covering problem. J Math Model Algorithms 3(3):263–283
88. Fattahi P, Hajipour V, Nobari A (2015) A bi-objective continuous review inventory control model: Pareto-based meta-heuristic algorithms. Appl Soft Comput 32:211–223
89. Mousavi SM, Sadeghi J, Niaki STA, Tavana M (2016) A bi-objective inventory optimization model under inflation and discount using tuned Pareto-based algorithms: NSGA-II, NRGA, and MOPSO. Appl Soft Comput 43:57–72
90. Mousavi SM, Hajipour V, Niaki STA, Alikar N (2013) Optimizing multi-item multi-period inventory control system with discounted cash flow and inflation: two calibrated meta-heuristic algorithms. Appl Math Model 37(4):2241–2256
91. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. Adv Eng Softw 69:46–61
92. Mirjalili S, Jangir P, Saremi S (2017) Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl Intell 46(1):79–95

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Khodakaram Salimifard Innovative industrial engineer offering seven years of experience in the field of industrial companies ranging from assembling to continuous production lines and also job-shop systems.