Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management
Detroit, Michigan, USA, September 23-25, 2016

Scheduling multi-objective unrelated parallel machines using hybrid reference-point based NSGA-II algorithm

Mohammad Komaki and Behnam Malakooti
Electrical Engineering and Computer Science Department
Case Western Reserve University
Cleveland, OH 44106, USA
gxk152@case.edu, bxm4@case.edu

Abstract

This paper addresses the multi-objective unrelated parallel machines to minimize makespan, total weighted completion times, and total weighted tardiness simultaneously. To tackle this problem, a hybrid algorithm based on reference-based many objective NSGA-II is proposed that the initial population is generated based on an exact algorithm and multi-directional local search algorithm. Minimizing makespan of the problem using relaxed mathematical model can be solved in an affordable computational time. Then, the multi-directional local search algorithm uses the high quality solution of the exact method to generate the non-dominate solutions by performing greedy search algorithms which is specifically designed for each objective of the addressed problem. The generated high quality solutions are used as the initial solution of the reference-based many objective NSGA-II. The performance of the algorithm is tested based on the well-known benchmark test problems and performance of the proposed algorithm is compared to the-state-of-the-art.

Keywords

Reference-based many objective NSGA-II, multi-directional local search, unrelated parallel machines, total weighted tardiness, makespan, total weighted completion time

1. Introduction

In most of the real world optimization problems, there are multiple conflicting objectives that need to be mathematically modeled as objective functions, and then, optimized through a multi-objective optimization problem. There is much research in the optimization literature that mathematically modeled a real industry problem as a multi-objective problem. For example, since there are multiple objectives in designing a control chart, Mobin et al. (2015) and Tavana et al. (2016) proposed a multi-objective model for this problem and solved it using enhanced evolutionary algorithms. As another multi-objective case problem, Li et al. (2016) proposed a multi-objective optimization of reliability growth planning that considered all objectives of reliability growth including reliability, time, and cost. Similar to the mentioned real case multi-objective problems, the multiple conflicting objectives in scheduling problems necessitate a model that considers all objective functions at the same time optimize them concurrently. These objectives represent concerns of manufacturer as well as customers. The concerns are mainly *makespan*, equivalent to maximizing machine utilization, *total tardiness* which is equivalent to customer satisfaction and delivering order(s) on time, and *total completion time*, maximizing throughput of the factory. This paper addresses multi-objective unrelated parallel machines scheduling problem to minimize the above mentioned objectives. In this problem, the speeds of the machines don't have fixed relationship with the processing time of jobs. This problem using three-field notation of Graham et al. (1979) 0can be presented as $R_m||C_{max},\sum w_jT_j, \sum w_jC_j$. It is assumed that all jobs are available at time zero, and all machines are available thorough scheduling period. Preemption is not allowed, and each job has its own weight, processing time, and due date which are known in advance. Processing times of the jobs depend on the processing machine. Several real-world problems can be modeled as the problem at hand. For instance, semiconductor manufacturing, pharmaceutical, and so on.

Since most problems associated with unrelated parallel machines are NP-Hard (Pfund et al., 2004), exact methods such as dynamic programming and Branch and Bound algorithms are ineffective in term of computational time. Therefore, heuristic and metaheuristic algorithms are gaining much attention. For detailed survey of application of evolutionary algorithms on the scheduling problems, readers are referred to Mokotoff (2001) and Pfund et al. (2004)0.

Single objective unrelated parallel machines have been investigated considerably but surprisingly, multi-objective unrelated parallel machines have been studied by few researchers. For instance, single objective unrelated parallel machines, Li and Yang (2009) investigated different mathematical modeling of non-identical parallel machines and provided detailed review and solutions for different objective functions.

To tackle multi-objective problems there are several methods. For instance, one may consider additive function of objective such Rashidi et al. (2010), however, finding weight of each objective is extremely difficult. Therefore, most researchers use methods that directly can find solution of the problem. For the multi-objective unrelated parallel machines problem, Cochran et al. (2003) proposed a multi-population genetic algorithm. Lin et al. (2013) addressed $R_m||C_{max},\sum w_jT_j, \sum w_jC_j$ and proposed heuristic algorithms and Genetic algorithm. Lately, Lin and Ying (2015) addressed the same problem and proposed a multi-point simulated annealing (MOMSA). 0 Recently, Lin et al. (2015) for the same problem proposed an algorithm based on iterated Pareto greedy algorithm using a tabu list (TIPG). Based on extensive experiments, they showed that TIPG clearly outperforms the other algorithms.

In this paper, we address the multiple objective unrelated parallel machines scheduling problem. For this problem, we propose a hybrid NSGA-III algorithm. NSGA-III, developed by Deb and Jain (2014), is almost the same as NSGA-II (Deb et al., 2002) except in the selection mechanism where NSGA-II utilities crowding distance but NSGA-III uses the predefined reference points in the selection mechanism. NSGA-III has been successfully applied to several problems including multi-objective X-bar control chart (Tavana et al., 2016). In the developed hybrid NSGA-III algorithm, the initial solution is generated using hybrid of exact and metaheuristic algorithm called Multi-directional local search algorithm (MDLS) (Tricoire, 2012). We solve the simplified version of the problem using exact model for makespan, then we use MDLS to generate solutions by applying its operators, *ruin* and *recreate* operators. After applying the operators of MDLS, the obtained solutions are compared, if they are efficient (non-dominated), they are kept, otherwise they are discarded. Indeed, the output of MDLS which is non-dominated solutions is considered as the initial population of NSGA-III. Also, to enhance the search ability of NSGA-III, we apply limited version of MDLS algorithm.

The reminder of the paper is organized as follows. Section 2 is devoted to the problem description, and in section 3, the proposed NSGA-III and its operators as well as multi-directional local search algorithm are discussed. In Section 4, the experimental results are presented and Section 5 is devoted to the conclusion.

2. Problem Description

In the addressed problem, $R_m||C_{max},\sum w_jT_j, \sum w_jC_j$, there is a set of jobs $N = \{1,\ldots, n\}$, and a set of unrelated parallel machines $M = \{R_1,\ldots,R_m\}$. Jobs have to be processed on one of the parallel machines where each job $j \in N$ has a distinct due date d_j and an importance weight w_j. Also, the processing time of job j on each machine $k \in M$ is p_{kj}. All machines are available through scheduling horizon and never break down. Preemption is not allowed, that is, after starting processing a job, its process cannot be interrupted before its completion. All jobs are available at time zero and setup times are included in the processing time of jobs. Each machine can process a job at a time and each job can be processed by only one machine at a time.

Table 1. Processing time, due date, and weight of jobs

Job	1	2	3	4	5	6	7	8	9	10
$p_{1,j}$	66	6	69	36	85	32	81	59	68	51
$p_{2,j}$	42	97	72	18	86	11	18	47	25	37
w_j	2	4	1	9	8	2	5	3	1	4
d_j	117	68	0	135	0	75	92	77	48	72

Let $\Pi=(\pi_1, \pi_2,\ldots,\pi_k,\ldots,\pi_m)$ be a complete schedule where π_k represents the sequence of jobs allocated to the machine k, and $\pi_k(1)$ represents the first job in π_k, and n_k is the total number of jobs allocated to the machine k. Then, each of the objectives can be computed as follows.

$$C_{\pi_k(i)} = \sum_{r=1}^{i} p_{k,\pi_k(r)} \quad \text{for } k = 1,\ldots,m \text{ and } i = 1,\ldots, n_k \tag{1}$$

$$T_{\pi_k(i)} = \max\{C_{\pi_k(i)} - d_{\pi_k(i)}, 0\} \quad \text{for } k = 1,\ldots,m \text{ and } i = 1,\ldots, n_k \tag{2}$$

$$C_{max} = \max\{C_{\pi_k(n_k)}\} \quad \text{for } k = 1,\ldots,m \tag{3}$$

Eq. (1) and (2) compute the completion time and tardiness of each job and Eq. (3) represents the makespan of the schedule Π. Total weighted tardiness and total weight completion times can be present as $TWT = \sum_{k=1}^{m}\sum_{i=1}^{n_k} w_{\pi_k(i)}T_{\pi_k(i)}$ and $TWC = \sum_{k=1}^{m}\sum_{i=1}^{n_k} w_{\pi_k(i)}C_{\pi_k(i)}$, respectively.

The goal is finding Π, i.e., allocating the jobs to the machines and finding their sequences, such that makespan (C_{max}), total weighted tardiness (TWT), and total weighted completion times (TWC) simultaneously are minimized.
In the following, a numerical example for 10 jobs and two machines is presented where their processing times, weight and due date of jobs are presented in Table 1.
Assume π_1={2,5,6,3} and π_2 ={4,7,10,8,1} as presented in Fig. 1, therefore, C_{max}= 192, TWT=1378, and TWC=2695.

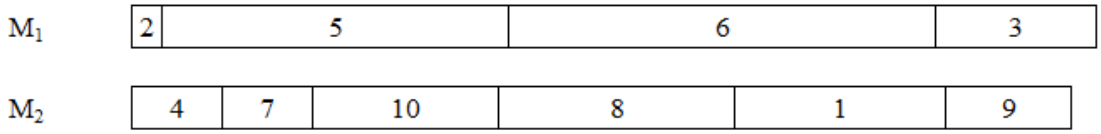


Figure 1. Solution of the numerical example

3. Proposed NSGA-III

As pointed out in Sections 1 and 2, the addressed problem is NP-Hard, therefore, heuristic and metaheuristic algorithms are effective tools to tackle the problem at hand. However, they may not be able to find the optimal solution, but they are able to find relatively high quality solutions within affordable computational time. Several algorithms have been proposed to tackle multi-objective algorithms. For detailed survey, see Konak et al. (2006) and Giagkiozis et al. (2015).

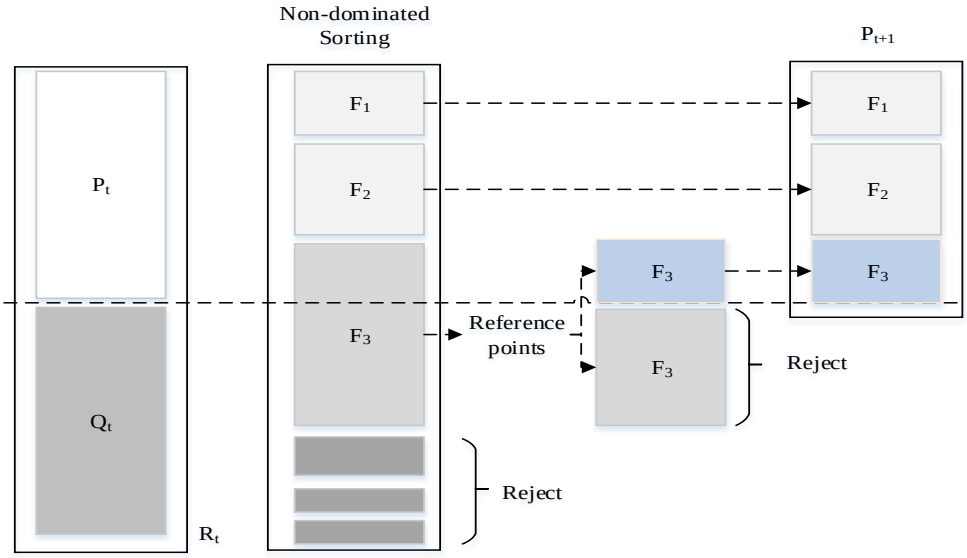


Figure 1. NSGA-III procedure (P_t: population at iteration t and Q_t: offspring at iteration t)

In this section, we develop a metaheuristic algorithm based on reference-point based many-objective NSGA-II (called NSGA-III, Deb and Jain, 2014). Also, in order to increase the efficiency of the proposed NSGA-III, a local search algorithm based on multi-directional local search (MDLS) (Tricoire, 2012**Error! Reference source not found.**) is embedded in the proposed NSGA-III.
In next subsections, first the basic structure of NSGA-III and its difference from NSGA-II are discussed in detail. Then, the proposed NSGA-III and its operators, as well as MDLS for the addressed problem, are proposed.

3.1 Introduction to NSGA-III

Reference-point based multi-objective NSGA-II (NSGA-III) is the evolved version of NSGA-II. All steps of NSGA-III are the same as NSGA-II except the selection mechanism, as in Figure 1. The purpose of the selection mechanism is keeping the diversity of the population by adding the dominated solution to the next generation. In NSGA-II, the selection mechanism is based on crowding distance which identifying the neighbors is computationally expensive and has been criticized for uneven distribution of population convergence and having poor performance in its global search power. Due to these critiques, in the evolved version of NSGA-II, NSGA-III, the selection mechanism is defined based on reference points where they are either known in advance or can be generated using a predefined method.

In the following, the brief summary of NSGA-III is presented. Like NSGA-II, NSGA-III starts with initial population P_0 with the size of N_{Pop}, and generates offspring using crossover operators and then applies mutation operator on them. The offspring at iteration t, Q_t, and population at iteration t, P_t, are combined as R_t and non-dominated sorting procedure (Deb et al., 2002) is applied and grouped into different levels where individuals in the first level (or front), F_1, is not dominated by any other individuals and individuals in the second level, F_2, are dominated by only by solutions in F_1 and so on. To create the next generation, P_{t+1}, the individuals in the first level, F_1, are copied to the P_{t+1}, and then F_2, and so on until the size of P_{t+1} reaches to N_{Pop} for the first time, let say at F_l. That is, F_1 to F_{l-1} are already in the population of the next generation, if their size, total number of individuals in F_1 to F_{l-1}, is equal to N_{Pop}, then algorithm starts the next iteration, otherwise the remaining individuals should be selected from F_l until the size of the population of the next generation reaches to N_{Pop}. In order to select individuals from F_l, objectives are normalized as will be discussed later in section 3.2.**Error! Reference source not found.**, and each individual in F_1 to F_l will be associated with one of the reference points which has the closest perpendicular distance from the line connecting the origin to the reference point, see Figure 2 where a sample of reference points distributed on the normalized hyperplane and reference lines are presented. Section 3.2.4 addresses how the reference points are generated. The number of associated individuals in P_{t+1} for each reference point j is counted and represented by ρ_j where $\rho_j = 0$ indicates that the reference point j does not have any associated member in P_{t+1}, and $\rho_j > 0$ indicates that one or more individuals in P_{t+1} are associated with the reference point j. The strategy of NSGA-III to have diverse population is based on selecting individuals from F_l associated with reference points with $\rho_j=0$, that is, it selects individuals from F_l associated with reference point j that no individual in P_{t+1} is associated with that reference point. If such individual in F_l does not exist, then a random individual(s) is added to P_{t+1}. After adding the new individual to P_{t+1}, ρ_j is increased by one unit. Then, the algorithm repeats this step with another reference point with $\min_j\{\rho_j\}$ until size of P_{t+1} reaches to N_{Pop}. Then, the algorithm restarts with the new population, P_{t+1}, and repeats the above processes until the predetermined stopping criterion such as computational time limit or the number of iterations is met.

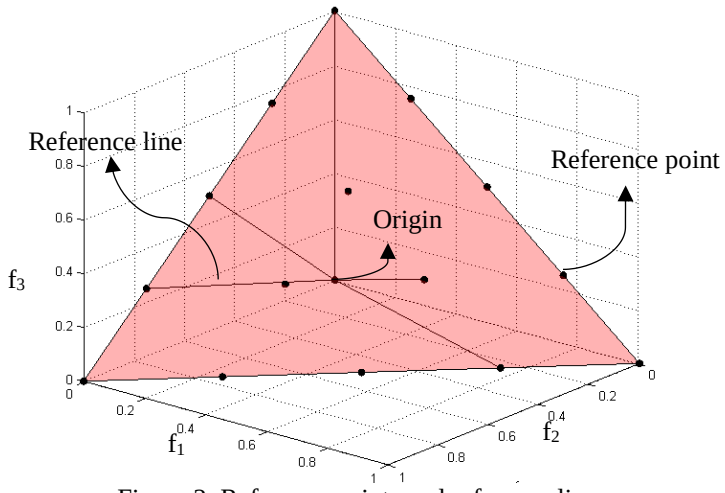


Figure 2. Reference points and reference lines

2	5	6	3		
4	7	10	8	1	9

Figure 3. Solution representation of the numerical example

3.2 Proposed Hybrid NSGA-III (HNSGA-III)

The schematic procedure of NSGA-III is presented in Figure 1, now, in the following subsections, detail of the proposed NSGA-III for the addressed problem as well as the incorporated improvements including initial solutions and multi-directional local search are discussed.

3.2.1 *Solution representation:*

Solution representation has the key role on the performance of metaheuristic algorithms. In this article, we present each solution as m vector which each vector represents the jobs allocated to each machine and the sequence of jobs

assigned to the machine. The objectives of each solution are computed as described in section 2. As an example, consider the numerical example presented in section 2 which the present solution in Figure 3 can be presented as follows.

3.2.2 *Initial population:*
It is well-established that quality of final solution of metaheuristic algorithms highly depends on the quality of the initial solution, the higher the quality of the initial solution, the higher the quality of the final solution. Generating the initial solution randomly which leads to poor performance of the algorithm, however, many studies for example, Friedrich and Wagner (2015), suggested using sophisticated methods to generate the initial population. In this study, to have diverse and high quality initial solution, we use a hybrid of an exact method and a metaheuristic algorithm. That is, the simplified version of the problem using the exact method to minimize the makespan based on the assignment of the jobs to each machine is solved. Then, the problem reduces to the m single-machine problems which two well-known heuristic algorithms are applied to generate two relatively high quality solution. Then, we apply Multi-Directional Local Search (MDLS) algorithm (Tricoire, 2012) on the relatively high quality solutions to generate non-dominated solutions.
Finding the optimal makespan of the unrelated parallel machine can be simply presented as following mathematical model (Potts, 1985).

P1. min C_{max} (4)

s.t.

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i \tag{5}$$

$$\sum_{i=1}^{n} p_{ij} x_{ij} \le C_{max} \qquad \forall j \tag{6}$$

$$x_{ij} \in \{0,1\}$$

where x_{ij} =1, if job j is assigned to machine i, 0 otherwise.

Lin et al. (2011) used the above model by relaxing the binary assignment variables, x_{ij}, and refining (rounding) them to find the assignment of the jobs to the machines. However, the above model for medium size instances, and to some extend large size instances, can be solved by a commercial software in affordable computational time, therefore, one can use this model to have a high quality solution.

Steps of the MDLS algorithm

1. Pop_1←Solve the Problem P1, and apply WSPT on each m single-machine problem
2. Pop_2←Solve the Problem P1, and apply modified Lawler's algorithm on each m single-machine problem
3. Set NDS ← { Pop_1 , Pop_2}
4. Set runtime t← 0
5. **While** $t < run_{Max}$
6. π ← Select a solution from NDS
7. π' ← Apply the *ruin operator* on π, and set J as the set of selected jobs from π
8. k←1
9. **While** $k \le 3$
10. **If** $k = 1$ % improve π' only based on C_{max}
11. S_1←Apply Procedure 1 on π'
12. Find $TWT(S_1)$ and $TWC(S_1)$
13. **Elseif** $k = 2$ % improve π' only based on *TWT*
14. S_2←Apply greedy insert procedure based on TWT on π'
15. Find $C_{max}(S_2)$ and $TWC(S_2)$
16. **Elseif** $k = 3$ % improve π' only based on *TWC*
17. S_3←Apply greedy insert procedure based on TWC on π'
18. Find $TWT(S_3)$ and $C_{max}(S_3)$
19. **End if**
20. k← k +1
21. **EndWhile**

22. Add S_1, S_2, and S_3 to NDS and then, update NDS using fast non-dominated sorting procedure
23. **EndWhile**

Figure 4. Steps of multi-directional local search (MDLS) algorithm

Having the solution of the problem P1, the problem reduces to *m* single-machine problems with given set of assigned jobs to each single machine. It is well-known that WSPT can optimally solve $1||\sum w_i C_i$, therefore, we apply WSPT to find the sequence of the assigned jobs to each machine and label this solution as Pop_1. Similarly, we apply the proposed modified Lawler's algorithm proposed by Cheng et al. (2005) on the solution of the problem P1 to find the sequence of jobs on each single machine to minimize TWT. We label this solution as Pop_2. Now, we apply MDLS algorithm on Pop_1 and Pop_2 which both have the same C_{max} and Pop_1 is better in term of TWC and Pop_2 is better in term of TWT.

MDLS algorithm (Tricoire, 2012) is a simple but effective metaheuristic algorithm which is based on Pareto local search (PLS) and uses different neighborhood structure for each objective of the problem. The algorithm, MDLS, has three simple steps; (1) select a solution (2) perform a local search on each objective of the problem (3) reject or accept the solutions. Similar to all metaheuristic algorithms, each of these steps should be defined problem specific. The local search of the algorithm is based on *ruin* and *recreate* operators which are similar to *destruction* and *construction* phases of iterated greedy (IG) algorithm, respectively. The ruin operator converts a complete feasible solution of the problem into a partial solution while the recreate operator finds a complete feasible solution of the problem from the partial solution based on local search algorithm which is specifically defined for each objective of the problem. Note that the destruction phase of IG and the ruin operator of the MDLS are conceptually similar, but the construction phase and the recreate operator completely differ from each other since the recreate operator is defined based on each objective of the problem while usually the construction phase usually is defined regardless of the objective of the problem. Furthermore, MDLS creates *k* solutions (*k* is the number of objectives of the problem) while IG generates only one solution at each iteration. After generating *k* solutions, they are compared to the current non-dominated solutions. If they are efficient, then they will be added to the Pareto set, otherwise, they will be discarded. The process continues until the stopping condition is met.

In this study, the MDLS algorithm starts with Pop_1 and Pop_2 as the set of non-dominated solutions (NDS), and at each iteration of the algorithm, it selects a solution from the NDS and generates *k* solutions (*k* is the number of the objective functions of the problem, here $k=3$) by applying the ruin and recreate operators as will be discussed later. The obtained solutions are compared to the solutions in the NDS and if they are efficient, then will be added to the NDS, otherwise the dominated solutions will be discarded. Also, after each iteration, the NDS will be updated by removing the dominated solutions using fast non-dominated sorting procedure (Deb et al., 2002). These processes continue until stopping condition is met. The steps of the proposed MDLS is presented in Figure 4.

The ruin and recreate operators are as following. The ruin process is defined based on removing *x* jobs randomly one at a time from a randomly chosen machine. Let *J* be the set of the selected jobs. Now, the set of jobs in *J* needs to be inserted in the partial sequence of jobs. To do so, we define three procedures each based on one objective of the problem at hand. These procedures are greedy based, that is, for each job $i \in J$, all positions are tested and the job is inserted in the position with the minimum increment of the objective function. Since the position of the jobs for C_{max} does not matter, it can be simplified by inserting the job to the last position of the machine which results in the minimum increment of the makespan and then WSPT rule can be applied on the selected machine, see Procedure 1 in Figure 5. In other words, Procedure 1 generates a solution with respect to makespan as well as TWC.

For the TWC and TWT objectives, we apply greedy recreate operators that evaluate every possible position to insert a job in the current partial sequence and select the best one.

Procedure 1 (C_{max})

1. For each machine j find $r_j = \sum_{i \in N \setminus J} p_{ij} x_{ij}$
2. **While** $J \neq \{\}$
3. Select job $i \in J$
4. Append job j to the current sequence of jobs of machine k^*
   where $j^* = \underset{j \in \{1,..,m\}}{\operatorname{argmin}} \{r_j + p_{ij}\}$
5. Apply WSPT rule on machine j^* to reorder the sequence of jobs
6. Update $J \leftarrow J \setminus i$
7. **EndWhile**

Figure 5. Procedure 1

Computational time of evolutionary algorithms are the main drawback of the algorithms. Therefore, incorporating the time-saving strategies can improve the performance of the algorithm. The time-saving strategies highly depend on the structure of the addressed problem. For flowshop scheduling problems, usually graph representation and its properties are used to develop the time-saving strategies. For instance, Komaki et al. (2015) have used the properties of the graph representation of the distributed permutation flowshop to reduce the computational time of their developed algorithm and they concluded that time-saving strategy tremendously improves the performance of the algorithm.
In order to save the computational time, we use the following timesaving scheme. Inserting a job to the current partial sequence of one machine does not have any effect on the sequence of jobs on other machines, and consequently one can focus on the machine where the job has been inserted. Assume $\Pi=(\pi_1, \pi_2,\ldots,\pi_j,\ldots,\pi_m)$ is the current partial sequence of jobs, and r_j, TWT_j and TWC_j represent makespan, total weighted tardiness, and total weighted completion times of machine j, respectively, and $|\pi_j|$ represents the number of jobs assigned to machine j. Note that $C_{max}(\Pi)=\max\{r_j\}$, $TWT(\Pi) = \sum_{j=1}^{m} TWT_j$, and $TWC(\Pi) = \sum_{j=1}^{m} TWC_j$. Let $\xi(\pi_j,i,l)$ represent inserting job i into position l of π_j as presented in the following figure, and the obtained sequence after applying $\xi(\pi_j,i,l)$ is π_j^{ξ}. In the following, we consider the change (increment) of the objectives after applying $\xi(\pi_j,i,l)$. Consider Inserting job i into position l of π_j results in $\boldsymbol{\pi_j^{\xi}}$ Figure 6 where the sequence of jobs on machine j before inserting job i, π_j, and after inserting job i, π_j^{ξ}, are presented.

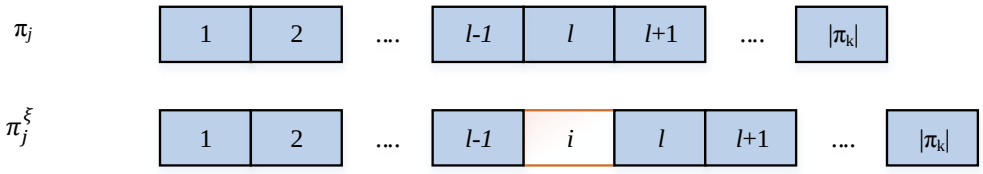


Figure 6. Inserting job i into position l of π_j results in $\boldsymbol{\pi_j^{\xi}}$

The change of makespan can be easily represented as $r_j(\pi_j^{\xi}) = r_j(\pi_j) + p_{ij}$, that is, inserting job i to any position of π_j increases the makespan by processing time of the inserted job i.
In π_j^{ξ}, the completion time and tardiness of jobs in positions 1 to l-1 remains unchanged while the completion time of the jobs in positions l to $|\pi_j|$ increases by p_{ij}, therefore, TWC of π_j^{ξ} can be obtained as following.

$$TWC_j(\pi_j^{\xi})=TWC_j(\pi_j) + w_j\left(\sum_{t=1}^{l-1} p_{tj} + p_{ij}\right)+p_{ij}\sum_{t=l}^{|\pi_j|} w_t \quad (7)$$

where $TWC_j(\pi_j)$ is total weighted completion times of π_j before inserting job i, $w_j\left(\sum_{t=1}^{l-1} p_{tj} + p_{ij}\right)$ represents the weighted completion time of job i and $p_{ij}\sum_{t=l}^{|\pi_j|} w_t$ represents the weighted increase of the completion times of jobs in positions l to $|\pi_j|$.
The change of TWT is not straightforward, but it can be derived as followsg. Let $v_i=C_i-d_i$ represent the deviation of completion time of job i from its due date and $T_i=\max(v_i,0)$ is tardiness of the job i. Since the completion time of the jobs in positions 1 to l-1 of π_j^{ξ} remains unchanged, their tardiness remains unchanged. Therefore, to find the TWT of π_j^{ξ}, one needs to focus on job i and the jobs in positions l to $|\pi_j|$.

$$TWC_j(\pi_j^{\xi})=TWC_j(\pi_j)+ w_i\, max\left(\sum_{t=1}^{l-1} p_{tj} + p_{ij} - d_i,0\right)+\sum_{t=l}^{|\pi_j|} w_t\ \left(max\left(v_t + p_{ij}, 0\right) - \max(v_t, 0)\right) \quad (8)$$

where $TWC_j(\pi_j)$ is total weighted tardiness of π_j before inserting job i, $w_i \max\left(\sum_{t=1}^{l-1} p_{tj} + p_{ij} - d_i,0\right)$ represents the weighted tardiness of job i, and $\sum_{t=l}^{|\pi_j|} w_t\ \left(\max\left(v_t + p_{ij}, 0\right) - \max(v_t, 0)\right)$ represents the weighted increase of tardiness of jobs in positions l to $|\pi_j|$.
As indicated earlier, the purpose of the MDLS algorithm is to generate initial population where all of the individuals are non-dominated. Since the executing the algorithm is limited to iterate run_{Max} times, it may not be able to find N_{Pop} non-dominated solution. In this case, that is, the number of the solution in the NDS is less than size of the initial solution, N_{Pop}, the rest of the solutions is generated by Apparent Tardiness Cost (ATC)-bi algorithm (Lin et al., 2013).

In this algorithm, the job with maximum index $\frac{w_j}{p_{i^*j}}\exp(-\frac{\max(d_j-p_{i^*j}-t_{i^*},0)}{Kp_{i^*j}})$ is assigned to the machine i^* with the minimum total processing time where K is scaling parameter, for more detail see Lin et al. (2013). Note that usually ATC-bi generates repeated (identical) solutions, so, after generating all solutions, the repeated ones are removed and only unique ones are kept in the initial solutions, then the rest of the solutions are generated randomly until the size of the initial solutions is reached to the determined size, N_{Pop}.

3.2.3 *Genetic operators:*

So far the initial population of the algorithm is generated. The next step is applying genetic operators, crossover and mutation, on the current population. The crossover operator combines two individuals and by exchanging their segments creates two new solutions which are called offspring. In this study, we use one-point crossover operator which simple and effective crossover operator and it has been used by many researchers, for example, Vallada and Ruiz (2011). In this operator, the crossing point for each machine is randomly chosen and jobs before the crossing points are directly copied to offspring. Then, for offspring 1, the jobs after the crossing point of the parent 2 which are not available in the current partial solution of the offspring 1 is inserted on the corresponding machine. A Similar process is repeated for the offspring 2.

Another operator of GA is mutation operator that improve the diversification of the algorithm. In this study, we use multiple reinsertion where several jobs are randomly are selected from the current offspring and they are inserted in random position.

3.2.4 *Fast non-dominated sorting procedure:*

Deb et al. (2002) 0 developed fast non-dominated sorting procedure with complexity $O(k.N_{\text{Pop}}^2)$ where k is the number of objectives of the problem. In this procedure, for each solution two entities are calculated: (1) n_i which is domination counter is the number of solutions that dominate solution i and (2) S_i the set of solutions that the solution i dominates them. Solutions with n_i =0 create the first non-dominated frontier. Then, for each solution i in the first non-dominated frontier, the domination counter of the solutions in the set S_i is reduced by one unit. Then, solutions with n_i =0 create the second non-dominated frontier. Now, for each solution i in the second non-dominated frontier, the domination counter of solution in the set S_i is reduced by one unit and the solutions with n_i =0 create the third non-dominated frontier. This process continues until all frontiers are identified.

Also, in order to increase the search ability of NSGA-III, we apply MDLS for a limited number of iterations, run_{FS} on the first frontier.

3.2.5 *Selection mechanism*

The main difference of NSGA-II and NSGA-III is the selection mechanism which in the former one is based on crowding distance and the latter one is based on reference points. The reference points can be known in advance or can be generated by a systematic method. Since the structure of optimal Pareto front of the problem is not known, we use the proposed method by Das and Dennis (1998) to generate the reference points where points are located on the normalized hyper-plane, see Figure 2, where points are equally distributed. The number of the reference points depends on the number of objectives (i.e., the dimension of the hyper-plane is determined by the k, the number of objectives), and the distance of them from each other, s. Das and Dennis (1998) proposed a procedure to create equally distributed points on the hyper-plane which the number of points is $\binom{k+s-1}{k-1}$. In Fig. 5, k =3 and s = 4, therefore, there are 15 reference points.

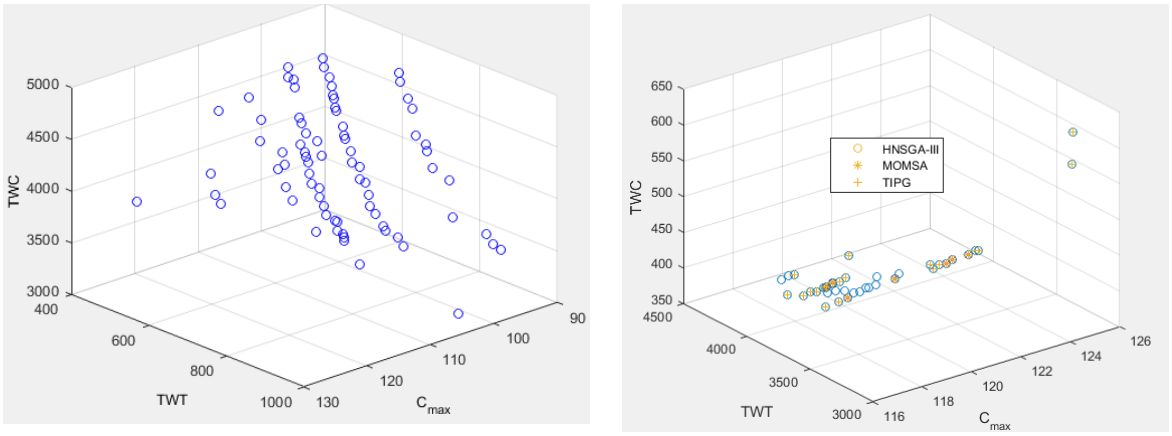
Because the scale of the objectives is different, converting them to the normalized value eases the process of assigning solutions to the reference points. Each objective is normalized as follows.

$$f_i' = \frac{f_i - z_i^{\min}}{z_i^{\max} - z_i^{\min}} \tag{9}$$

where f_i represents i-th objective function, z_i^{max} and z_i^{min} represent the max and min of the objective f_i, respectively. After normalization of the objectives, each individual (solution) is assigned to a reference point which has the least distance form reference line, the line that connects the origin to the reference point. Then, for each reference point j, ρ_j the number of associated points from $F_1, F_2, \ldots, F_{l-1}$ is counted. As indicated earlier, ρ_j=0 indicates that the reference point j does not have any associated member in the next population of the algorithm, P_{t+1}. Therefore, such reference points with no associated members have the higher priority. That is, if any solution in F_l is associated with such a point, then one of them should be in P_{t+1}. In a case that the reference point j has several associated members in F_l one of them should be selected randomly. If no point is associated with the reference point j from F_l, then the closet

solution to the reference point should be selected. After adding a solution to the P_{t+1} based on the reference point j, then ρ_j is increased by one unit. The process is restarted with the next reference point with the lowest ρ_j until the number of solutions in P_{t+1} reaches to N_{Pop}.

3.2.6 *Stopping criterion*
Steps 3.2.3-5 are repeated until the stopping condition is met. We consider the number of iterations of the algorithm as the stopping criterion, It_{max}.



a. Result of the benchmark problem 2M12N_01 b. Comparison of the algorithms for benchmark 4M20N_03

Figure 8. A sample of result of the proposed hybrid NSGA-III for benchmark problems

Table 2. Number of non-dominated solutions of each algorithm

Instance	HNSGA-III	TIPG	MOMSA	Instance	HNSGA-III	TIPG	MOMSA	Instance	HNSGA-III	TIPG	MOMSA
2M12N_01	24	24	22	4M20N_01	61	79	2	10M100N_01	87	51	9
2M12N_02	12	12	12	4M20N_02	38	26	7	10M100N_02	107	59	28
2M12N_03	4	4	4	4M20N_03	58	62	27	10M100N_03	77	24	12
2M12N_04	18	18	18	4M20N_04	29	17	7	10M100N_04	93	2	23
2M12N_05	26	26	26	4M20N_05	125	120	1	10M100N_05	65	48	21
2M12N_06	17	17	17	4M20N_06	42	42	25	10M100N_06	57	43	47
2M12N_07	12	12	12	4M20N_07	94	119	26	10M100N_07	79	7	21
2M12N_08	24	24	20	4M20N_08	75	63	14	10M100N_08	81	92	9
2M12N_09	24	24	20	4M20N_09	84	46	15	10M100N_09	25	8	42
2M12N_10	19	19	18	4M20N_10	93	163	32	10M100N_10	92	120	59
2M12N_11	75	77	48	4M20N_11	89	82	16	10M100N_11	77	63	13
2M12N_12	13	13	11	4M20N_12	81	72	18	10M100N_12	67	4	13
2M12N_13	16	16	16	4M20N_13	88	96	20	10M100N_13	102	110	9
2M12N_14	16	17	14	4M20N_14	120	111	18	10M100N_14	59	30	43
2M12N_15	19	19	18	4M20N_15	71	64	12	10M100N_15	85	81	11
2M12N_16	16	16	10	4M20N_16	133	128	20	10M100N_16	48	15	54
2M12N_17	13	13	5	4M20N_17	75	48	19	10M100N_17	32	0	31
2M12N_18	22	22	18	4M20N_18	102	133	40	10M100N_18	27	14	3
2M12N_19	13	13	12	4M20N_19	127	115	21	10M100N_19	38	22	42
2M12N_20	34	34	27	4M20N_20	98	82	26	10M100N_20	61	0	36
Average	20.85	21	17.4	Average	84.15	83.4	18.3	Average	67.95	39.65	26.3

4. Experimental results

The developed algorithm is coded in Matlab 2014Ra and run on a PC with an Intel® Core i5 CPU with 3.20GHz and 4GB RAM. The parameters of the proposed hybrid NSGA-III is set as following; Number of population N_{Pop}= 150; Number of iterations of the NSGA-III; It_{max}= 150; Number of iterations of MDLS: run_{Max}= 100; Number of iterations of MDLS on the Pareto frontier run_{FS}= 20, mutation probability p_m = 0.1; and crossover probability p_c= 0.2, distance of points on the hyperplane s = 13.

The performance of the proposed hybrid NSGA-III is compared to TIPG (Lin et al., 2015) and MOMSA (Lin and Ying, 2015) which are the state-of-the-art. The parameters values are based on suggestion of the authors. The testbed is based on the well-known benchmark test problems of Lin and Yang (2014) and its expansion by Lin et al. (2013) which is widely used in the literature. This benchmark set has problem sets; two machines with 12 jobs (2M12N), four machines with 20 jobs (4M20N), and ten machines with 100 jobs (10M100N), and each set has 20 problems. The processing times are uniformly distributed over [1,100] and the jobs weights are uniformly distributed over [1,10]. The due dates are also uniformly distributed over [P(1-T-R/2), P(1-T+R/2)] where P is average processing times and T is tardiness factor and R is relative range of due dates where $T = 0.8$ and $R = \{0.4, 0.8\}$.
Fig. 8a represents the result of the HNSGA-III for the benchmark problem with two machines and 12 jobs, 2M12N_01 and Fig. 8b represents the Pareto frontier of the algorithms for benchmark 4M20N_03.
In this section, we report the result of the experiment on the set of 4M20N and 10M100N. The non-dominated solutions obtained from all algorithms are combined and then the dominated solutions are discarded. Then, the number of remaining non-dominated solution of each algorithm is reported in Table 2. The last row of the table represents the average number of the non-dominated solutions of each algorithm. The average number of the non-dominated solutions is an indication of the performance of the algorithms. For the small benchmark instance, two machines with 12 jobs, TIPG has the best performance following by HNSGA_III, and MOMSA has the worst performance. As size of the problems increases, HNSGA_III outperforms TIPG. The average number of the non-dominated solutions of the HNSGA_III is 84.15 comparing to the average number of non-dominated solutions of TIPG which is 83.4. For the large size instances, 10M100N, HNSGA_III clearly outperforms the other two algorithms.
The final Pareto frontier can be enhanced with the multi-criteria decision making (MCDM) techniques to provide the manageable number of optimal solutions for the decision-maker. MCDM techniques have been used in the literature to rank a set of multiple alternatives considering some specific criteria. Different applications of MCDM tools are provided in Mobin et al. (2015), Saeedpoor et al. (2015), and Skeete et al. (2015). For example, the application of the MCDM tools in ranking the Pareto fronier can be mentioned as follows: Mobin et al. (2015) and Li et al. (2016) applied data envelopment analysis (DEA) to reduce the optimal solutions obtained by NSGA-II to a workable size of the efficient optimal solutions. Tavana et al. (2016) combined the optimal solutions obtained by NSGA-III and MOPSO algorithm, obtained the efficient optimal solutions using the DEA, and then ranked the efficient optimal solutions using TOPSIS technique.

5. Conclusion

This paper addressed multi-objective unrelated parallel machines to minimize makespan, total weighted completion times and total weighted tardiness simultaneously. To tackle this problem, a hybrid algorithm based on reference-point based NSGA-II developed where the initial population is generated by a hybrid of an exact algorithm and multi-directional local search algorithm. The exact method is based on relaxation of the problem to minimize makespan of the problem, then, the problem reduces to m single machines whose sequence of jobs is obtained by applying heuristic algorithms to minimize total weighted completion times and total weighted tardiness. Having two high quality solutions, a multi-directional local search algorithm is applied on these solutions. The mechanism of the multi-directional local search algorithm is based on destroying and recreating a solution. In the destroying process, jobs are randomly selected and removed from current solution, and in the recreation process, jobs are inserted in the partial sequence based on greedy search specifically designed for each objective function. That is, for each destroying process, three new solutions are generated. The multi-directional local search algorithm provides high quality initial population. Then, reference-points based NSGA-II (NSGA-III) is applied on the initial population. The NSGA-III is the evolved version of NSGA-II that its selection process is defined based on the reference points. The experiments showed that the proposed algorithm outperformed other state-of-the-art algorithms in term of quality.

References

Cheng, T. E., Ng, C. T., Yuan, J. J., and Liu, Z. H. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, vol. 165, no. 2, pp. 423-443, 2005.

Cochran, J. K., Horng, S. M., and Fowler, J. W. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, vol. 30, no. 7, pp. 1087-1102, 2003.

Das, I., and Dennis, J. E. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631-657, 1998.

Deb, K., and Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577-601, 2014.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A. M. T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182-197, 2002.

Friedrich, T., and Wagner, M. Seeding the initial population of multi-objective evolutionary algorithms: A computational study. *Applied Soft Computing*, vol. 33, pp. 223-230, 2015.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, vol. 5, pp. 287-326.1979.

Komaki, G., Mobin M., S., Teymourian, E., and Sheikh, S. A general variable neighborhood search algorithm to minimize makespan of the distributed Permutation flowshop scheduling problem. *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, vol. 9, no. 8, pp. 2576-2583. 2015.

Li, K., and Yang, S. L. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied mathematical modelling*, vol. 33, no. 4, pp. 2145-2158, 2009.

Li, Z., Mobin, M., and Keyser, T., Multi-objective and multi-stage reliability growth planning in early product-development stage, *IEEE Transaction on Reliability*, vol. 65, no. 2, pp. 769-781, 2016.

Lin, C. W., Lin, Y. K., and Hsieh, H. T., Ant colony optimization for unrelated parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1-4, pp. 35-45, 2013.

Lin, S. W., and Ying, K. C. A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems. *International Journal of Production Research*, vol. 53, no. 4, pp. 1065-1076, 2015.

Lin, S. W., and Ying, K. C. ABC-based Manufacturing Scheduling for Unrelated Parallel Machines with Machine-Dependent and Job Sequence-dependent Setup times. *Computers & Operations Research*, vol. 51, no. 1, pp. 172–181, 2014.

Lin, S. W., Ying, K. C., Wu, W. J., and Chiang, Y. I. Multi-objective unrelated parallel machine scheduling: a Tabu-enhanced iterated Pareto greedy algorithm. *International Journal of Production Research*, DOI:10.1080/00207543.2015.1047981, 2015.

Mobin M. Roshani A., Saeedpoor M., Mozaffari M., *Integrating FAHP with COPRAS-G Method for Supplier Selection (Case Study: an Iranian Manufacturing Company)*, Proceedings of the 2015 American Society of Engineering Management Conference (ASEM2015), Indiana, USA.

Mobin, M., Li, Z., and Massahikhoraskani M. Multi-objective X-bar control chart design by integrating NSGA-II and data envelopment analysis. In Proceedings of the 2015 Industrial and Systems Engineering Research Conference, Tennessee, USA, 2015.

Mokotoff, E. Parallel machine scheduling problems: a survey. *Asia-Pacific Journal of Operational Research*, vol. 18, no. 2, pp. 193-242, 2001.

Pfund, M., Fowler, J. W., and Gupta, J. N. A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. *Journal of the Chinese Institute of Industrial Engineers*, vol. 21, no. 3, pp. 230-241, 2004.

Potts, C. N. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics*, vol. 10, no. 2, pp. 155-164, 1985.

Rashidi, E., Jahandar, M., and Zandieh, M. An improved hybrid multi-objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines. *The International Journal of Advanced Manufacturing Technology*, vol. 49, no. 9-12, pp. 1129-1139, 2010.

Saeedpoor M., Vafadarnikjoo A., Mobin M., Rastegari A., *A SERVQUAL Model Approach Integrated with Fuzzy AHP and Fuzzy TOPSIS Methodologies to Rank Insurance Firms*, Proceedings of the 2015 American Society of Engineering Management Conference (ASEM2015), Indiana, USA.

Skeete A., Mobin M., *Aviation Technical Publication Content Management System Selection Using Integrated Fuzzy-Grey MCDM Method*, Proceedings of the 2015 Industrial and Systems Engineering Research Conference (ISERC2015), Tennessee, USA.

Tavana, M., Li, Z., Mobin, M., Komaki, M., and Teymourian, E. Multi-objective control chart design optimization using NSGA-III and MOPSO enhanced with DEA and TOPSIS. *Expert Systems with Applications*, vol. 50, pp. 17-39, 2016

Tricoire, F. Multi-directional local search. *Computers & operations research*, vol. 39, no. 12, pp. 3089-3101, 2012.

Vallada, E., and Ruiz, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, vol. 211, no. 3, pp. 612-622, 2011.

Biography

Mohammad Komaki is accomplished B.Sc. in Industrial Engineering at Sharif University of Technology, Tehran, Iran (2001–2006) and M.Sc. in Industrial Engineering at Mazandaran University of Science & Technology, Babol, Iran (2007–2010). He is currently studying System Engineering at Case Western Reserve University, Cleveland, USA. His research interests include optimization and multi-criteria decision making.

Behnam Malakooti obtained his PhD in 1982 from Purdue University. He has consulted for numerous industries and corporations, including General Electric, Parker Hannifin, and B.F. Goodrich. He has published over 100 papers in technical journals. In his work, systems architectures, space networks, optimization, multiple criteria & intelligent decision making, trait analysis of biological systems, adaptive artificial neural networks, and artificial intelligence theories and techniques are developed and applied to solve a variety of problems. His current research is on design and protocols analysis for NASA space-based networks. Recently Professor Malakooti developed a four-dimensional approach for decision-making process typology and risk analysis. Decision-making typology accurately identifies the four types of decision makers' behavior: Information processing, creativity, risk, and decisiveness approach. It also provides a basis for developing the next generation of intelligent robots. See http://car.cwru.edu/decision/ for computerized survey. He has made contributions to manufacturing systems developing computer aided approaches for manufacturing/production design, planning, operations, facility layout, assembly systems, scheduling, MEMS, and machine set-up, tool design, and machinability.