Solving a Bi-criteria Scheduling Problem of cellular Flowshop with Sequence Dependent Setup Times

Al-Mehdi M. Ibrahem[1], and Mohamed I. Ghoma[2]

Department of Mechanical & Industrial engineering,
University of Gharyan[1,2]

الملخص

تتناول هذه الورقة حل مسألة جدولة العمليات الإنتاجية في نظام الخلايا الصناعية Cellular flow shop بهدف التعظيم الثنائي لتقليل معايير وقت التدفق الإجمالي Total Flow time (TFT)وأقصى وقت لإستكمال العمليات الإنتاجية Makespan في وقت واحد ، أي ان عملية التعظيم تتم في آن واحد ويرمز لهذه المسألة بالرمز (FMCSP) مع .(SDSTs) وقد تم تصميم خوارزمية سرب الجسيمات متعدد الأهداف multi-objective Particle Swarm Optimization (MPSO)وخوارزمية المحاكاة المتعددة (MOSA)لحل المشكلة المقترحة. كما تم برمجة خوارزمية تهدف الى تحسين جودة الحلول تسمى) IMPSO-TA)، حيث يتم دمج MPSO مع خوارزمية قبول العتبة (TA) لتحسين تقارب الحلول والحصول على أفضل الحلول التي تم التوصل إليها باستخدام النماذج المقترحة. كما تم تقييم الخوارزميات المقترحة باستخدام عدة مقاييس لمؤشرات الجودة (QI) لمشاكل التحسين متعددة الأهداف. أظهرت النتائج أن الخوارزميات المقترحة يمكن أن الحصول على حلول ثنائية "باريتو " Pareto Front تقريبية في وقت قياسي. علاوة على ذلك ، فإن جودة حلول Pareto التي تم إنشاؤها بواسطة الخوارزمية IMPSO-TA أفضل من الحلول التي تم التوصل إليها باستخدام الخوارزمية MPSO و MOSA بناءً على الاختبارات المستخدمة في هذه الورقة . كما أثبتت الدراسة أن جودة الحلول التي تم الحصول عليها باستخدام خوارزمية-IMPSO TAالمقترحة أفضل الخوارزميات المتاحة في مسائل جدولة العمليات الإنتاجية في نظام الخلايا الإنتاجية

Abstract

This paper addresses a bi-criteria optimization problem to minimize total flow time and makespan simultaneously for a cellular flowshop with Sequence Dependent Setup Times (FMCSP with SDSTs); A multi-objective Particle Swarm Optimization (MPSO) and a Multi-objective Simulated Annealing (MOSA) Algorithm are proposed to solve the proposed problem. furthermore, an improved algorithm (named as IMPSO-TA), where MPSO is combined with Threshold Acceptance (TA) algorithm to improve the convergence of the obtained Pareto Fronts. The proposed algorithms are evaluated using several Quality Indicators (QI) measures for multi-objective optimization problems. Results showed that proposed algorithms can generate approximated Pareto Fronts in a reasonable CPU time. Furthermore, quality of Pareto fronts generated by IMPSO-TA is better than Pareto fronts found by MPSO and MOSA based on the test problems that are used in this research at the cost of CPU time. Further, the proposed IMPSO-TA performs as best available algorithms in the literature for small and medium test problems with a very minor deviation for best results for large test problems.

Keywords: Multi-objective optimization, Multi-objective Particle Swarm Optimization, Multi-objective Simulated Annealing, Pareto fronts, Cellular flowshop, Sequence dependent setup times

I. INTRODUCTION

In a cellular manufacturing environment, machines are grouped into cells. Each cell is dedicated to the production of a specific part family. A cell consists of machines or workstations, arranged in a processing sequence. The wide application of the Cellular Manufacturing Systems (CMS), the high level of competitiveness in the current market has forced manufacturers to improve the productivity and use small-lot size production systems (Bevilacqua et al., 2015). Further, the intense competition has forced manufacturing organizations to find methods to reduce cost and to deliver product on time with high customer satisfaction (Panwar et al., 2015). Cellular manufacturing helps to highlight the economic advantage of batch manufacturing.

Most of the currently available algorithms only deal with the optimization of a single criterion measure (S. W. Lin & Ying, 2012). However, considering more than a single objective compromises the advantages of the studied objectives. For instance, minimization of the makespan improves production efficiency, while minimization of the total flow time reduces the work-in-process inventory. Therefore, minimizing both at the same time leads to accomplishing the benefits of both measures (efficiency and cost reduction). This paper aims to solve a Flowshop Manufacturing Cell (cellular flowshop) Scheduling Problem with Sequence Dependent Setup Times (FMCSP with SDSTs). Scheduling a flowshop of cellular manufacturing systems with family setup times was studied for the first time by Schaller, Gupta, and

Vakharia (Schaller et al., 2000), they developed several heuristic algorithms with minimization of the makespan as the criterion. (Li et al., 2014) considered the FMCSP with SDSTs for total tardiness and mean total flow time minimization. Hendizadeh et al (Hendizadeh et al., 2007) and (S. W. Lin & Ying, 2012) studied the proposed problem to minimize makespan and total flow time as a bi-criteria scheduling problem. The former studied the problem for the first time, and they developed a Multi-Objective Genetic Algorithm (MOGA) to optimize the total flow time and makespan simultaneously. They compared their results to the lower bounded proposed by Schaller (Schaller, 2001) for makespan criterion. The latter developed a two-level multistar simulated annealing (TLMSA) for the same problem to minimize makespan and total flow time (or total tardiness). As a result of the limited work done on the proposed problem, the author has been motivated to develop two multi-objective algorithms to generate approximated Pareto fronts for the proposed problem. The convergence and the diversity of the generated Pareto fronts are evaluated based on QI measures rather than the deviation from the lower bound.

II. PROBLEM DEFINITION

In a cellular manufacturing environment, machines are grouped into cells. Each cell is dedicated to the production of a specific part family. A cell consists of machines or workstations, arranged in a processing sequence. In FMCSP, there are N_0 jobs which are grouped according to their similarity and production requirements. Therefore, there are F part

families {1,2,....,F} to be processed in a cell that has m machines {M_1,M_2,...,M_m}. The ultimate goal is to find the best sequence of processing the part families as well as jobs within each family to minimize the total flow time and makespan simultaneously. Using the triplet notation (Pinedo, 2012). The problem can be notated as:$F_m \backslash fmls, Se_{lki}, prum \backslash C_{max} \sum_{j=1}^{N} C_j$.

The solution of the studied problem is achieved in two phases or levels:

1) Sequencing of part families
2) Sequencing of jobs within each part family

The solution representation consists of F + 1 segments; the first segment F represents the sequence of part families on each machine, the other segments correspond to the sequence of jobs within each part family (Bouabda, Jarboui, Eddaly, et al., 2011), (Eddaly et al., 2009a), (Eddaly et al., 2009b), (Hamed Hendizadeh et al., 2008), (S.-W. Lin et al., 2009), (Salmasi et al., 2010), (Salmasi et al., 2011), and (Bouabda, Jarboui, & Rebai, 2011). The sequence of part families and the parts within each part family are the same on all machines (permutation flowshop). For a feasible schedule, a solution π of FMCSP takes the following structure Fig.1

[15]):

$$\Pi = \{\Pi_{[1]}, \Pi_{[2]}, ..., \Pi_{[f]}, \Pi_{[f+1]}, ..., \Pi_{[F]}$$

where

$$\Pi_{[f]} = \{\Pi_{[f][1]}, \Pi_{[f][2]}, ..., \Pi_{[f][j]}, ..., \Pi_{[F][N_f]}$$

is the sequence of the jobs in each part family.

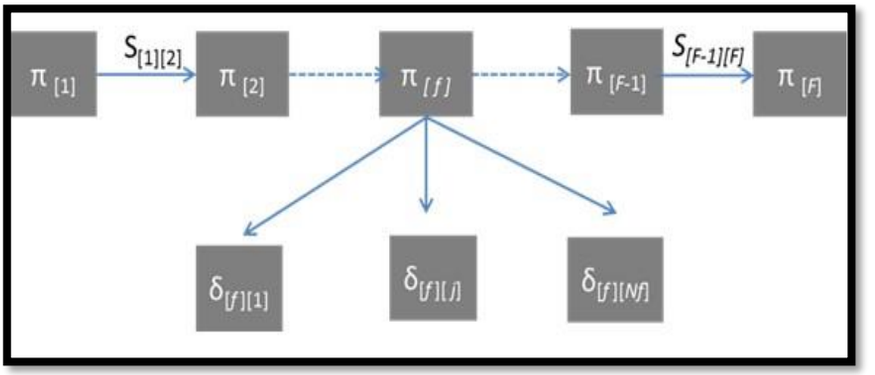

Fig. 1. The solution structure.(Bouabda, Jarboui, & Rebai, 2011)

The proposed problem is an NP-hard problem. Research effort has focused on finding the approximation algorithms that can provide a near-optimal solution at a relatively minor computational expense. Most of the currently available algorithms only deal with the optimization of a single criterion measure. Therefore, the author is motivated to develop two multi-objective algorithms to generate approximated Pareto fronts for the proposed problem. The convergence and the diversity of the generated Pareto fronts are evaluated based on QI measures rather than the deviation from the lower bound. A multi-objective Particle Swarm Optimization (MPSO) and a Multi-objective Simulated Annealing (MOSA) Algorithm are further proposed to solve the bi-criteria optimization problem to minimize the total flow time and makespan simultaneously. Furthermore, an improved PSO is combined with Threshold Acceptance (TA) algorithm to improve effectiveness of the proposed MPSO (named as IMPSO-TA) for the convergence of the obtained

Pareto Front. The proposed algorithms are evaluated using several Quality Indicators (QI) measures for multiobjective optimization problems. The proposed algorithms can generate approximated Pareto Fronts in a reasonable CPU time. The proposed IMPSO-SA outperforms MOSA algorithm in terms of CPU time and minimize the objective functions.

III. PRINCIPLES OF MULTI-OBJECTIVE OPTIMIZATION PROBLEM (MOP)

MOP is defined as a vector of decision variables which satisfies constraints within a feasible region to optimize a vector of objective functions (Mansouri et al., 2009). Unlike mono objective, there is a set of solutions called Pareto solutions or Pareto front which is found using Pareto Optimality Theory (Coello et al., 2007). A general multi-objective optimization problem includes a set of ***n*** parameters (decision variables), a set of M objectives, and a set of k constraints. Objective functions and constraints are functions of the decision variables. In general, the minimization problem of *m* objectives can be presented as follows:

$$\text{Minimize } f_m(x), m = 1,...,M.$$

$$\text{subject to}$$

$$g_j(x) \leq 0, j = 1,...,k.$$

$$h_l(x) \leq 0, l = 1,...,p. \qquad (1)$$

$f_i(x)$is the i^{th} objective function; $g_j(x)$ and $h_l(x)$ are inequality constraint and equality constraints respectively.

A. Pareto Dominance

The optimal solutions in multi-objective optimization problem can be defined based on the concept of domination. The definition of the non-dominated set or Pareto front can be defined as follows:

A solution $x^{[1]}$ is said to dominate other solution $x^{[2]}$; if both of the following conditions are true:

1) The solution $x^{[1]}$ is no worse than $x^{[2]}$ in all objectives. Thus, the solutions are compared based on their objective function values.
2) The solution $x^{[1]}$ is strictly better than $x^{[2]}$ in at least one objective.

$$f_i(x) \leq f_i(y)\ \forall i \in 1,...,M. \qquad (2)$$

and

$$\exists i \in \{1,...,M\} : f_i(x) < f_i(y) \qquad (3)$$

All decision vectors that are not dominated by any other feasible decision vector are called non-dominated sets or front (Pareto-optimal set). Achieving the exact Pareto front of an arbitrary problem is usually quite difficult. Nevertheless, reasonably good approximations of true Pareto Front are generally acceptable within a limited computational time (Coello et al., 2007). Thus, the goal for multi-objective optimization problem is to find a set of solutions which lie on the Pareto front and find

a set of solutions which are diverse enough to represent the entire range of the Pareto front.

IV.PROPOSED ALGORITHMS

In designing metaheuristics, two conflicting criteria must be considered: exploration of the search space (diversification) and exploitation of the best solutions found (intensification) (Talbi, 2009). Promising regions are determined by the obtained good solutions. On one hand, in intensification, the promising regions are explored more thoroughly in the hope to find better solutions. On the other hand, in diversification non-explored regions have been visited to be sure that all the regions of the search space are explored. Therefore, the proposed algorithm should compromise and balance between diversification and intensification criteria. Moreover, hybridization is implemented to balance between these criteria and to manage the cooperation between the operation of the search among the candidate solutions (populations or swarms), a diversifying agent, and the intensifying agent.

In fact, two major challenges must be addressed when an evolutionary algorithm is applied to multi-objective optimization: First challenge is to accomplish fitness assignment and selection to guide the search towards the Pareto front. The second challenge is to maintain a diverse population in order to prevent premature convergence and achieve a well distributed and well spread nondominated set (Zitzler et al., 2000). Metaheuristics based on MPSO, MOSA, and IMPSO are proposed to generate the approximated

Pareto front sets. First, the MPSO algorithm is developed to minimize the total flow time and makespan in a cellular flowshop scheduling problem. Then, a Multi-Objective simulated annealing algorithm is developed to solve the same problem. Lastly, a hybrid algorithm based on an Improved Multi-objective Particle Swarm Optimization (named IMPSOTA) is proposed. In this algorithm, IMPSO proposed by (Zhao et al., 2014) is combined with (TA) algorithm proposed by (Dueck & Scheuer, 1990) as local search engine to provide efficient Pareto fronts. The analysis of the performance of the proposed MPSO, MOSA, and IMPSO algorithms will be discussed in detail. Moreover, QI measures are used to evaluate the effectiveness of the proposed algorithms.

A. Particle Swarm Optimization

Simply, PSO algorithm simulates birds swarm behaviour, and makes every particle in the swarm move according to its experience and the best particles experience to find a better new position. After the evolution, the best particle in the swarm is seen as the best solution for the input problem. The population of PSO is called swarm, and each individual or particle which is a potential solution is known with its current position and current velocity. The new position of each individual particle is obtained by assigning a new position as well as a new velocity to the particle. Each particle gains a different position, and the value of each position is evaluated based on the value of the objective function. The main advantage of this approach is that every particle always remembers its best position in the experience. When a particle moves

to another position, it must refer to its best experience and the best experience of all particles in the swarm. The best position of each particle that has been gained so far during the previous steps is called the best particle (p-best). The best position gained by all particles so far is called the global best (g-best). The new position as well as the new velocity of each particle are obtained based on the previous positions, the pbest, and the g-best.

Considering an n-dimension search space, there are S particles (swarm size) cooperating to find the global optimum in the search space. In a swarm of S particles, the i^{th} particle is associated with the position vector $\{x_{i1},x_{i2},.....,x_{in}\}$ and the velocity $\{v_{i1},v_{i2},.....,v_{in}\}$. The p-best and g-best are updated each iteration based on generation of new swarms. Each particle uses its own search experience and the global experience by the swarm to update the velocity and flies to a new position based on the following equations:

$$v_j(t+1) = wv_j(t)+C_1r_1^*(P_j^t-x_j(t))+C_2r_2^*(G^t-x_j(t)) \quad (4)$$

$$x_j(t+1) = v_j(t+1) + x_j(t) \quad (5)$$

Where w is the inertia weight, it controls the influence of the previous velocity of particles, C_1 and C_2 are called acceleration coefficients that provide weight to the social influence. The parameters r_1 and r_2 are uniformly distributed random variables in the range between [0,1]. For the t^{th} iteration, P_i^t and G^t are the p-best (for i^{th} particle) and gbest particles respectively. The values of these parameters are updated in each iteration based on the following equations:

$$w = w^{min} + \frac{w^{max} - w^{min}}{1 + e^{\frac{-a(maxI-I)}{maxI}}} \quad (6)$$

$$C_1 = C_1^{min} + \frac{C_1^{max} - C_1^{min}}{1 + e^{\frac{-a(maxI-I)}{maxI}}} \quad (7)$$

$$C_2 = C_2^{min} + \frac{C_2^{max} - C_2^{min}}{1 + e^{\frac{-a(I)}{maxI}}} \quad (8)$$

TABLE I

THE MAXIMUM AND MINIMUM VALUES OF PSO PARAMETERS

Parameter.	Max. values	Min. values
W	2	0.4
C_1	2	0.4
C_2	2	0.4
Position value	0	4
Velocity	-4	4

In the area of multi-objective optimization problems, PSO is attracting much research interest because of the simple computational model introduced by PSO and exploration capabilities of the algorithm (Elhossini et al., 2010). In the PSO algorithms for mono objective optimizations, determination of local and global best particles is based on

the extreme value of the objective function. However, in multi-objective optimizations problems, each particle might have a set of different g-bests from which just one can be selected in order to update its position. This set of g-bests is usually stored in a different place from the swarm that is called external archive named as {*NDS*} set. This is an externally stored archive or elite set that was used to store the non-dominated solutions found so far. The set of solutions contained in the external archive. Global best (g-best) particle is randomly selected from an external archive. Velocities and Positions of particles are updated each iteration. Solutions are updated according to the non-domination check, and stored in the external archive. The final set of non-dominated solutions in the external archive is reported as the final output of the algorithm.

B.Initialization

The MPSO algorithm starts by randomly generating several particles $X_0 = \{x_1,.....,x_N\}$ (subscript 0 refers to initial state before starting iterations) where N is the swarm size. Each particle is considered as a potential solution that refers to the sequence of the families and jobs in each family. Initial position and velocity of particles are determined using the following equations:

$$X_{oi} = X_{min} + C_1 (X_{max} - X_{min}) \quad (9)$$

$$V_{0,I} = V_{min} + C_2 (V_{max} - V_{min}) \quad (10)$$

where $i \in \{1,2,.....N\}$, X_{min} and X_{max} represent the minimum and maximum position values respectively. V_{min} and V_{max} represent the minimum and

maximum velocities respectively. C_1 and C_2 are random variables in the range [0,1]. The values of these parameters are given in Table I.

C. Pareto Search

After initializing the first swarm, an External Archive is defined to store the Non-dominated Solution Set (*NDS*) or the Pareto front. This archive (named as NDS-Set) will be filled by initial non-dominated solution obtained from the initial swarm. In the t^{th} iteration ($t > 0$), the velocity and the position are updated based on the following equations:

$$v_j^{t+1} = wv_j^t + C_1 r_1^*(P_j^t - x_j(t)) + C_2 r_2^*(G^t - x_j(t)) \quad (11)$$

$$x_j^{t+1} = v_j^{t+1} + x_j^t \quad (12)$$

Where w is the inertia weight that controls the influence of the previous velocity of particles, r_1 and r_2 are uniformly distributed random variables in the range [0,1], C_1 and C_2

are the acceleration constants. The values of these constants are updated according to the equations 6, 7, and 8. Unlike the standard PSO for single objective problem, g-best (G^t) at the t^{th} iteration is selected randomly from *NDS*. They guide every particle toward the local best and the global best solutions during the search process. The Ranked Order Value (ROV) (Kuo et al., 2009) is applied to find the sequence of the families as well as the jobs in each family. This procedure will be repeated till the maximum number of iterations is reached. The final *NDS* archive will be reported as a final solution of the Pareto front. The algorithm is outlined in Algorithm 1. Note

that the p-best, and g-best are calculated to update the velocity of the particles in the swarm for the next iteration.

The maximum and minimum values of the above parameters are presented in Table I, where I is the current iteration, $maxI$ is the pre-set value of maximum number of iterations, and is constant equal to 10. Particles fly in the search space based on equation (4), and equation (5). Every particle always remembers its best position in the experience. When a particle moves to another position, new velocity is calculated according to the previous velocity and the distance of its position from both p-best and g-best. However, the new velocity is limited to the range to control the extreme traveling of particles outside the search space. Particles gain their new positions according Algorithm 1

Algorithm1: The steps of the proposed MPSO algorithm

Require: Initialize parameters:{swarm size, n, maximum IterationI_{max}, C_{1max},C_{1min},

$C2max$,$C2min$,$Vmax$,$Vmin$,$Wmax$,$Wmin$,$Xmax$,$Xmin$}

- Step 1: Set iteration $t = 0$
- Step 2: If $t = 0$ Generate initial positions X_i^t and V_i^t initial velocities for $i \in \{1,2,....,n\}$ according to equations (9, and 10)

Else

Generate a new swarm by updating the velocity V_i^t and position X_i^t of particles according to equations (11, and 12)

- Step 3: Set the initial Non Dominated Set *NDS* = {} (The Null Set)
- Step 4: Apply the (ROV) on X_i^t to find the sequence of families as well as jobs in families.
- Step 5: Calculate the objective function $f_{TFT}(X_i^t)$ and $f_{MS}(X_i^t)$ for each particle $i \in \{1,2,....,n\}$
- Step 6: Check whether the particle (X_i^t) qualifies to be included in *NDS* o Step 6.1 For each particle in the swarm the p-best (P_i^t) is calculated as:

  $P_i^t = \text{argmin}_j\, f(X_i^j) \quad for\, j \in \{1,2,....,t\}\ X$

$$f_i^t = \left(0.5 f_{TFT}(X_i^t) + 0.5 f_{MS}(X_i^t)\right)$$

- Step 6.2 The g-best G^t is randomly selected from

the *NDS* set.

- Step 7 Set $t = t + 1$
- Step 8 If $t = I_{max}$, STOP; and set NDS^{Imax} as the

Pareto front; else, go to step 2.

The new velocity and the previous position equation (4) and equation (5) (Liu et al., 2008). implemented Ranked Order Value (ROV) to convert the continuous position value of the particles to job sequence to solve permutation flowshop scheduling problem. In this study, ROV is implemented to convert the position of particles to part families

sequences as well the sequence of jobs in each part family (Salmasi et al., 2010).

V. MULTI-OBJECTIVE SIMULATED ANNEALING ALGORITHM

SA is often distinguished from the evolutionary algorithms that are used in solving the multi-objective optimization problems because SA does not involve candidate solutions (Simon, 2013). Yet, MOSA has been proposed to solve the multi-objective optimization problems. For instance, the two-machine flowshop scheduling problem is addressed by (Mansouri, 2005) to minimize setups and makespan. SA algorithm provides excellent solutions to single and multiple objective optimization problems with a substantial reduction in computation time (Suman & Kumar, 2005). Pareto SA was presented by (Czyżżak & Jaszkiewicz, 1998) to generate an approximation of the Pareto front for multiple objective combinatorial optimization. Furthermore, the problem of permutation flowshop scheduling is considered by (Varadharajan & Rajendran, 2005), in which they presented MOSA algorithm with the objectives of minimizing the makespan and total flow time of jobs. The proposed MOSA mainly consists of the following steps: first, initialize the search parameter, and generate the initial Non-dominating Solutions (*NDS*) or the Elite set. Second, generate neighbourhood solutions. Lastly, implement the nondomination check of the new solutions, and update *NDS* set. The algorithm is given in Algorithm 2.

A. Initial Solutions

A number of initial solutions are randomly generated after setting the search parameters are selected; the combination of the parameters were as follows: I_{iter} = 150, T_0 = 20, T_f = 1, α =0.95, $N_{non-imp.}$ = 15. Cauchy function is used in the annealing process that gives the SA more chances to escape from local minima. The non-dominated check is implemented to specify the initial Pareto set that would be updated based on the neighbourhood generation in the sequence of the families as well as the jobs in each family.

B. Neighbourhood Generation

Two common methods were used in generating the new solution: Swapping, and Insertion. For a given a sequence(π), the set of families and jobs in each family can be obtained from the current sequence π using the swapping and insertion as follows:

- Swapping: let *j*,*k* be two positions in the sequence π, which are selected randomly. For example, consider a neighbor of (π). Swapping is obtained by interchanging the jobs in the positions j and k. For instance if π = (5,2,3,4,1,6) and j = 2, k = 4, then a neighbor of (π) will be: (5,4,3,2,1,6).
- Insertion: let *j* and *k* be two positions in the sequence (π) which are selected randomly. A neighbor of (π) is obtained by inserting the job of position *j* to the position *k* pushing the jobs in between these positions backward (forward), including the job of position *j*, if *k* is greater.

(or less) than *j*. For example if π = (5,2,3,4,1,6) and *j* = 2, *k* = 4, then the neighbour of π will be: (5,3,4,2,1,6). If *j* = 4, *k*= 2, then the neighbour of π

will be: (5,4,2,3,1,6). For a given sequence of the families, and jobs in each family. A family *f* is selected randomly for swapping and insertion methods; the job sequence of the selected family is changed using swapping and insertion. Then, sequence of families is changed to pick another family for swapping and insertion for a several of iterations.

C. An Improved Multi-objective Particle Swarm Optimization (IMPSO)

The performance of classic PSO like all evolutional algorithms depends on the search parameters. Further, it often suffers from problem of premature convergence that might be because of being trapped in local optima. The probability of being trapped into the local optima can be reduced by improving the ability of particles to explore the global and Algorithm 2.

Algorithm2: The steps of the proposed MOSA algorithm

Required: Initialize the search parameters { T_0,T_F, α = 0.95,*Iiter*, *Nnon−imp.*}

- Step 1: Generate initial population (pop) solutions randomly, initialize *NDS* = {};
- Step 2: Do the non-dominated check on the population and update the *NDS* set.
- Step 3: Set Solution *X* (randomly selected from initial population) to be the current solution.
- Step 4: Set $obj_1 = TFT$; and $obj_2 = MS$ • Step 5: For *F*=1 to Family size

Set $T = T_0$, No. of Moves = 1, counter = 1 o Step 5.1: while $T > T_F$ do:

- Step 5.2: Generate a new neighbour solution Y from current solution X by minor swapping on job sequence of F^{th} family F
- Step 5.3: Generate a random number RN uniformly between 0 and 1, and calculate:

$4E_1 = obj_1(Y) - obj_1(X)$; $4E_2 = obj_2(Y) - obj_2(X)$

IF $4E_1 \leq 0$, and $4E_2 \leq 0$ then X=Y;

IF $4E_1 > 0$, and $4E_2 \leq 0$;

IF $\{RN < \frac{T}{T^2+(\triangle E_1)^2}, then X = Y\}$

ELSE No. of Moves = No. of Moves + 1;

ELSE IF $4E_1 \leq 0$, and $4E_2 > 0$;

IF $\{RN < \frac{T}{T^2+(\triangle E_2)^2}, then X = Y\}$ o Step 5.4: Do non-dominated check with the possibility of in the population and update NDS

- Step 5.5:IF (counter = I_{iter}, OR No. of Moves $= N non-imp.$)

$T = \alpha T$

I_{iter}=0;

No. of Moves =0;

ELSE counter = counter+1;

• Step 6: Return the NDS as the final solution set (Pareto front) local optimization solutions.

The Decline Disturbance Index (*DDI*) proposed by (Zhao et al., 2014) is applied to enhance the ability of particles to explore the global and local optimization solutions. The *DDI* is added to improve the updating velocity formula.

Therefore, the new formula for the velocity will be as follows:

$$v_j^{t+1} = wv_j^t + C_1 r_1^* (P_j^t - x_j(t)) + C_2 r_2^* (G^t - x_j(t)) + lr_3 \quad (13)$$

$$x_j^{t+1} = v_j^{t+1} + x_j^t \quad (14)$$

Where $j \in \{1,2,....,n\}$, $t \in \{1,2,....,I_{max}\}$, P_j^t is the best for the j^{th} particle and the i_{th} iteration, G^t is the g-best till the t^{th} iteration, $l = -d_1(x - d_2)$ is a linear decline function controlled by parameters d_1 and d_2 with $xt4(x)$. Both d_1 and d_2 are small parameters that can be set dynamically, t is the iteration index, and $4(x)$ is an interval whose length can be adjusted according to the objective functions. During the evolution process of the algorithm, the *DDI* declines at a certain rate, and has minimal impact on the evolution of the particles at last, thus, increasing the chance of convergence to an optimum solution (Zhao et al., 2014).

D. Local Search based on Threshold Acceptance (LS-TA)

The ultimate goal of any search strategy is to find an optimal or a near-optimal solution. Local search or Memetic Algorithms (*MAs*) are usually implemented due to the simple and successful application of these algorithms to various optimization problems. Ishibuchi et al presented the first *MA* algorithm named as a multiobjective genetic local search approach. They

implemented the local search algorithm for multi-objective flowshop scheduling problems and they concluded that a high performance is demonstrated by applying the local search to multi-objective flowshop scheduling problems (Ishibuchi & Murata, 1998). Knowles et al provided a simple framework that guides for designing MAs for MOPs (Knowles & Corne, 2005). A local search method based on TA is implemented to enhance the g-best after generating the initial swarm to improve the intensification. Due to the computational time advantage, TA is implemented to improve the quality of Pareto sets obtained by MPSO by guiding the g-best to the promising regions. Basically, TA escapes from local optima by accepting solutions that are not worse than the current by more than a given threshold (Talbi, 2009).

The pseudo code of the LS-TA approach is shown in Algorithm 3.

Algorithm 3: The Pseudo code for Loacl Search Threshold Acceptance algorithm LS-TA

Require: Initializethe search parameters $\{I_{iter}, N_{non-imp.}, Familysize\}$

- Step 1: Set Solution X to be the current
- Step 2: Set $obj_1 = TFT$; and $obj_2 = MS$ • Step 3: For F=1 to Family size

Set No. of Moves =1, and counter = 1

- Step 4 while counter < I_{iter}
- Step 4: Generate a new neighbour solution Y from current solution X by minor swapping on the job sequence of family F

- Step 5: Calculate: $4E_1 = obj_1(Y) - obj_1(X)$; $4E_2 = obj_2(Y) - obj_2(X)$
- Step 6: Generate a random number *RN* uniformly between 0 and 1:

IF $4E_1 \leq 0$, and $4E_2 \leq 0$; then $X = Y$

Else $4E >$ $\left[RN < \frac{T^2+(\triangle E_1)^2}{T^2+(\triangle E_1)^2}, then X = Y\right]$ 0, and $4E \leq 0$; IF

else No. of Moves = No. of Moves + 1; Else $4E \leq 0$, and $4E > 0$; $\left[RN < \frac{T^2}{T^2+(\triangle E_2)^2}, then X = Y\right]$ IF

else No. of Moves = No. of Moves + 1;

- Step 7 IF (No. of Moves $= N_{non-imp.}$)

counter = counter+1;

VI. THE PROPOSED IMPSO-TA ALGORITHM

The proposed IMPSO-TA is like MPSO (Algorithm1). The velocity and position of each particle are updated according to equations 13 and 14. Local search algorithm is inserted in each swarm to increase the chance of getting g-best in promising regions, and to enhance the quality of the g-best. TA (Algorithm 3) is combined with IMPSO to form a hybrid algorithm named as IMPSO-TA to solve FMCSP with SDSTs to minimize TFT and MS simultaneously. Therefore, the overall steps of IMPSO-TA is shown in Algorithm 4.

Algorithm 4: The steps of proposed IMPSO-TA algorithm

Require: Initialize parameters { swarm size *n*, maximum Iteration I_{max}, C_{1max},C_{1min},

C2max,C2min,Vmax,Vmin,Wmax,Wmin,Xmax,Xmin}

- Step 1: Set iteration $t = 0$
- Step 2: If $t = 0$

o Generate initial positions X_i^t and V_i^t initial velocities for $i \in \{1,2,....,n\}$ according to equations (9, and 10)

Else o Generate a new swarm by updating the velocity V_i^t and position X_i^t of particles according to equations (13, and 14)

- Step 3: Set the initial Non Dominated Set $NDS = \{\}$ (The Null Set)
- Step 4: Apply the (ROV) on X_i^t to find the sequence of families as well as jobs in families.
- Step 5: Calculate the objective function $f_{TFT}(X_i^t)$ and $f_{MS}(X_i^t)$ for each particle $i \in \{1,2,....,n\}$
- Step 6: Check whether the particle (X_i^t) qualifies to be included in *NDS*
- Step 7 Implementing the local search $LS - TA$ o Step 7.1 For each particle in the swarm the p-best (P_i^t) is calculated as:

$$P_i^t = \underset{X_i}{\operatorname{argmin}_j} \; f(X_i^j) \quad for \quad j \in \{1,2,....,t\}$$

$$f_i^t = \left(0.5 f_{TFT}(X_i^t) + 0.5 f_{MS}(X_i^t)\right)$$

- Step 7.2 For $j = \{1,2,....,NDS\}$,

$G^t = LS - TA(G^t)$

- Step 7.3 Do the non-dominated check (Section

5.2.1), and update the *NDS* o Step 7.4 The g-best G^t is randomly selected from

the *NDS* set.

- Step 8 Set $t = t + 1$
- Step 9 If $t = I_{max}$, STOP; and set NDS^{Imax} as the

Pareto front; else, go to step 2.

VII. COMPUTATIONAL RESULTS AND DISCUSSION

The proposed algorithms: MPSO, MOSA, and IMPSO-TA are coded using C++ language and run on a PC with an Intel core I7 (2.93 GHz) CPU and 4.0 GB memory. The performance of the proposed algorithms are tested using test problems proposed by Schaller (Schaller, 2001). The obtained Pareto fronts are compared and evaluated based on lower bounds. MPSO-TA showed a better performance than MPSO, and MOSA. As shown in Figure 2, the obtained Pareto fronts obtained by IMPSO-TA is much better than other algorithms especially for small a compared with Pareto fronts found by MOSA, and in Figure 3 MPSO-TA shows better results even for the large problems.

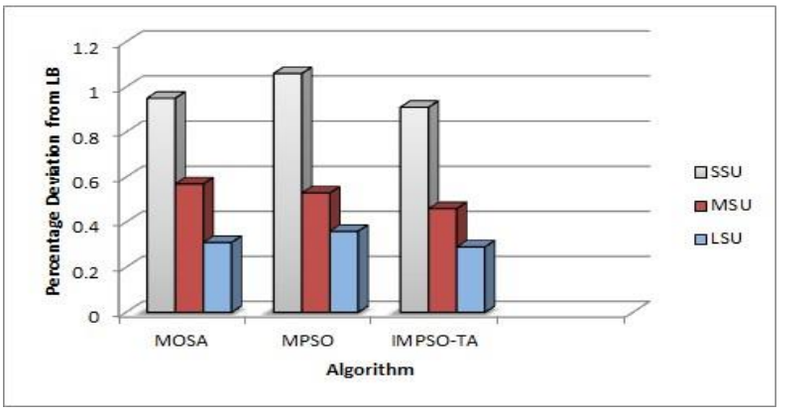

Fig. 2. The percentage deviation from makespan for small test problems

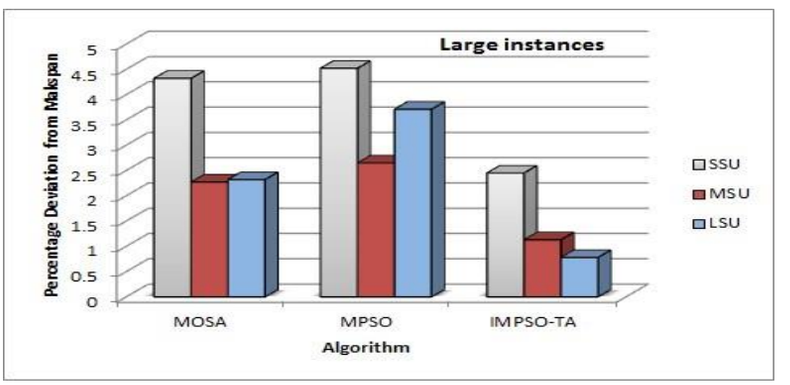

Fig. 3. The percentage deviation from makespan for large test problems

VIII. PERFORMANCE MEASURES BASED ON QUALITY INDICATORS

Unlike the single objective optimization, there are several Quality Indicators (QI) often used to evaluate the quality of NDS. These performance indicators are widely used in practice. In this work, the QIs listed below are calculated

using the C++ code; results are showed that the proposed algorithms have generated Parto fronts; yet, the IMPSO-TA has the superiority over the MPSO algorithm especially in large sized problems 2

1) Number of Non-dominated solutions

The first quality measure used in this research is how many non-dominated solutions in a Pareto front. Let NDS_1 is the Pareto set generated by IMPSO-TA algorithm. The size of this set is X_1. Similarly, X_2 is the size of Pareto set obtained by using MOSA algorithm.

2) Percentage of non-dominated solutions

The ratio between the sizes of *NDS* in each set is calculated (X_1/X_2) to indicate which algorithm generates more non-dominated solutions.

3) Coverage index (*CM*)

For the two different non-dominated solution sets X and X^0, consider a mapping $(X,X^0) \rightarrow [0,1]$ where $x \in X$, and $x^0 \in X^0$. If a solution a dominates a solution b , that means $b \prec a$, then the *CM* is defined by:

$$CM(X,X') = \frac{|\, x' \in X' \mid \exists : x \in X : x' \prec x \,|}{X'} \quad (15)$$

If all the solutions in X^0 were dominated by the solutions in X, then $CM(X,X^0) = 1$; on the contrary, if all the solutions in X were dominated by the solutions in X^0 , then $CM(X,X^0) = 0$.

4) *Dave* and *Dmax*

The performance measure D_{av} is defined as:

$$D_{av}(X) = \frac{1}{|X'|} \sum_{x' \in X'} \min_{x \in X} (d_{xx'} \mid x \in X) \quad (16)$$

D_{max} is defined as:

$$D_{max}(X) = \frac{1}{|X'|} \max_{x' \in X'} \{\min_{x \in X} \{d_{xx'} \mid x \in X\}\} \quad (17)$$

where

$$d_{xx'} = \sqrt{(obj_1(x') - obj_1(x))^2 + (obj_2(x') - obj_2(x))^2} \quad (18)$$

5) Spacing Distance measure

Multi-objective evolutionary methods aim to minimize the distance between the obtained solution and the true Pareto front (Tan et al., 2006). The Spacing Distance (*SD*) measures the uniformity spread of solutions over the approximated Pareto front. For non-dominated solutions set, *SD* metric is calculated using the following equation:

$$SD(X) = \sqrt{\frac{1}{X} \sum_{1}^{X} (d_i - d')^2} \quad (19)$$

where $d' = \sum_{i=1}^{|X|} \frac{d_i}{|X|}$, d_i is the Euclidean distance between solution x_i in X and the nearest solution in the space. *X* is the number of the non-dominated solution in the Pareto front. The smaller of the *SD*, the superior of the solution distribution is.

6) Quality Index measure

Quality index *QI* is used to compare the obtained Pareto front with the true Pareto front. Since there is no true Pareto front for the studied

problem, the Pareto fronts found by IMPSO-TA and the MPSO are compared using QI. Quality index can be calculated using the following formula:

$$QI = \sum_{x \in X} \left\{ 1 - Min \left(\frac{obj_1^x - \min_{x' \in X'} \left\{ obj_1^{x'}(X') \right\}}{\min_{x' \in X'} \left\{ obj_1^{x'}(X') \right\}} \right) \left(\frac{obj_2^x - \min_{x' \in X'} \left\{ obj_2^{x'}(X') \right\}}{\min_{x' \in X'} \left\{ obj_2^{x'}(X') \right\}} \right) \right\} / \mid X \mid \quad (20)$$

,

A. Discussion of the quality measures results

The results of the performance evaluation based on the QI for small, medium, and large test problems showed that the number of non-dominated solutions (NDS), in most cases the average number of non-dominated solutions found by IMPSOTA is greater than the obtained solutions by MOSA. Furthermore, the results of *CM* metric indicate that solutions found by MOSA are dominated by the one obtained by IMPSOTA (*CM* = 0). Therefore, quality of Pareto fronts found by IMPSO-TA is better than Pareto fronts generated by MOSA algorithm. For instance, the divergence between the proposed algorithms is shown in Figure 4 for one of the small instances, and Figure 5 for medium problem size. In addition, as shown in Figure 6, the variation between the obtained Pareto fronts is increased as the problem size is increased. Yet, the variation is less for small size instances as shown in Figure 4. Furthermore, the Pareto fronts generated by IMPSO-TA is better than that of MOSA in terms of average distance $D_{ave}$and as well as in terms of maximum distance (D_{max}). *QI* metric is introduced to evaluate the

performance between proposed algorithms for multi-objective problem. It is obvious that *QI* computed by IMPSO-TA is smaller than that of MOSA, which states that the overall performance for Pareto optimal from IMPSO is better than the ones that obtained by PSO and MOSA. We can conclude that IMPSO-TA is an effective algorithm to solve the bi-criteria scheduling problem of flowshop manufacturing cell with sequence dependent setup times than MPOS, and MOSA algorithms in terms of the quality of the solutions.

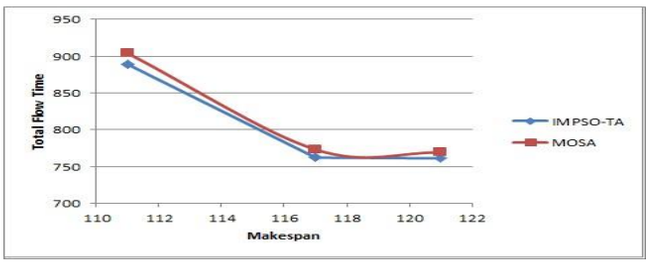

Fig. 4. Pareto fronts obtained by MOSA and IMPSO-TA (SSU34)

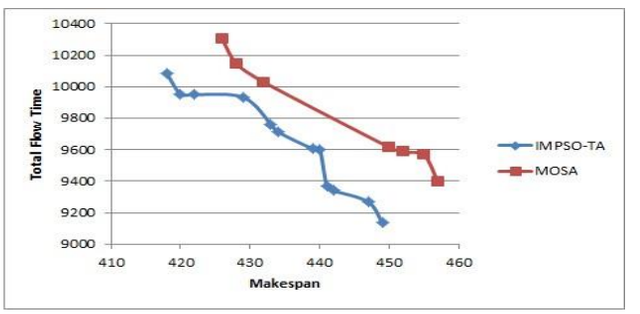

Fig. 5. Pareto fronts obtained by MOSA and IMPSO-TA (LSU66)

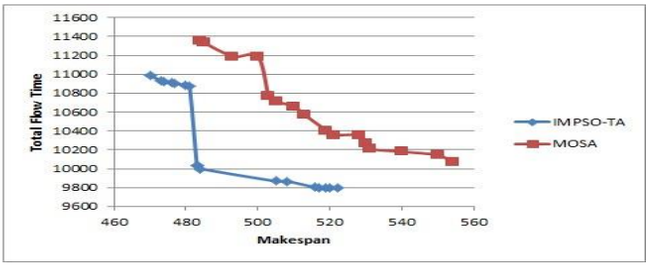

Fig. 6. Pareto fronts obtained by MOSA and IMPSO-TA (LSU88)

IX. CONCLUSIONS

Scheduling optimization of a FMCSP with SDSTs is studied to minimize the total flow time and makespan at the same time. An approximation of true Pareto Fronts are generated within a limited computational time. Multi-objectives algorithms based on MPSO, MOSA, and IMPSO-TA are proposed to find a

set of solutions which lie on the Pareto front sets; proposed algorithms are evaluated and tested using the well-known test problems developed by (Schaller, 2001). Furthermore, the performance of the proposed algorithms is evaluated using several Quality indicator that are used to assess the best convergence and the diversity of the Pareto fronts. Results indicate that quality of Pareto fronts generated by IMPSO-TA is better than Pareto fronts found by MPSO and MOSA based on the test problems that are used in this study at the cost of CPU time. Further, the proposed IMPSO-TA performs as best available algorithms in the literature for small, medium, and large test problems.

REFERENCES

Bevilacqua, M., Ciarapica, F. E., De Sanctis, I., Mazzuto, G., & Paciarotti, C. (2015). A Changeover Time Reduction through an integration of lean practices: a case study from pharmaceutical sector. *Assembly Automation*, *35*(1), 22–34. https://doi.org/10.1108/AA-05-2014-035

Bouabda, R., Jarboui, B., Eddaly, M., & Rebai, A. (2011). A branch and bound enhanced genetic algorithm for scheduling a flowline manufacturing cell with sequence dependent family setup times. *Computers & Operations Research*, *38*(1), 387–393. https://doi.org/10.1016/j.cor.2010.06.006

Bouabda, R., Jarboui, B., & Rebai, A. (2011). A nested iterated local search algorithm for scheduling a flowline manufacturing cell with sequence dependent family setup times. *(LOGISTIQUA), 2011 4th*, 526–531.

Coello, C., Lamont, G., & Veldhuizen, D. Van. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

Czyżżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, *7*(February 1997), 34–47.

Dueck, G., & Scheuer, T. (1990). Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*.

Eddaly, M., Jarboui, B., Bouabda, R., & Rebai, A. (2009a). Hybrid estimation of distribution algorithm for permutation flowshop scheduling problem with sequence dependent family setup times. *Computers & Industrial Engineering, 2009. CIE 2009. International Conference On*, 217–220.

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5223755

Eddaly, M., Jarboui, B., Bouabda, R., & Rebai, A. (2009b). Hybrid Estimation of distribution algorithm for permutation flowshop scheduling with dependent family setup times. *2009 International Conference on Computers and Industrial Engineering*, 217–220.

Elhossini, A., Areibi, S., & Dony, R. (2010). Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization. *Evolutionary Computation*, *18*(1), 127–156. https://doi.org/10.1162/evco.2010.18.1.18105

Hamed Hendizadeh, S., Faramarzi, H., Mansouri, S. A., Gupta, J. N. D., & Y ElMekkawy, T. (2008). Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. *International Journal of Production Economics*, *111*(2), 593–605. https://doi.org/10.1016/j.ijpe.2007.02.031

Hendizadeh, H., Elmekkawy, T., & Wang, G. (2007). Bi-criteria scheduling of a flowshop manufacturing cell with sequence dependent setup times. *European Journal of Industrial Engineering*, *1*(4), 1751–5254.

Ishibuchi, H., & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *28*(3), 392–403. https://doi.org/10.1109/5326.704576

Knowles, J., & Corne, D. (2005). Memetic algorithms for multiobjective optimization: issues, methods and prospects. *Recent Advances in Memetic Algorithms*.

Kuo, I.-H., Horng, S.-J., Kao, T.-W., Lin, T.-L., Lee, C.-L., Terano, T., & Pan, Y.

(2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. *Expert Systems with Applications*, *36*(3), 7027–7032. https://doi.org/10.1016/j.eswa.2008.08.054

Li, Y., Li, X., & Gupta, J. N. D. (2014). Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search. *Expert Systems with Applications*, *September*. https://doi.org/10.1016/j.eswa.2014.09.007

Lin, S.-W., Gupta, J. N. D., Ying, K.-C., & Lee, Z.-J. (2009). Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times. *International Journal of Production Research*, *47*(12), 3205–3217.

Lin, S. W., & Ying, K. C. (2012). Scheduling a bi-criteria flowshop manufacturing cell with sequence-dependent family setup times. *European J. of Industrial Engineering*, *6*(4), 474–495. https://doi.org/10.1504/EJIE.2012.047666

Liu, B., Wang, L., & Jin, Y.-H. (2008). An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research*, *35*(9), 2791–2806. https://doi.org/10.1016/j.cor.2006.12.013

Mansouri, S. A. (2005). Coordination of set-ups between two stages of a supply chain using multi-objective genetic algorithms. *International Journal of Production Research*, *43*(15), 3163–3180. https://doi.org/10.1080/00207540500103821

Mansouri, S. A., Hendizadeh, S. H., & Salmasi, N. (2009). Bicriteria scheduling of a two-machine flowshop with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, *40*(11–12), 1216–1226. https://doi.org/10.1007/s00170-008-1439-z

Panwar, A., Jain, R., & Rathore, A. (2015). Lean implementation in Indian process

industries: some empirical evidence. *Journal of Manufacturing Technology Management*, *25*(1), 131–160.

Pinedo, M. (2012). *Scheduling: theory, algorithms, and systems*. Springer.

Salmasi, N., Logendran, R., & Skandari, M. R. (2010). Total flow time minimization in a flowshop sequence-dependent group scheduling problem. *Computers & Operations Research*, *37*(1), 199–212.

Salmasi, N., Logendran, R., & Skandari, M. R. (2011). Makespan minimization of a flowshop sequence-dependent group scheduling problem. *The International Journal of Advanced Manufacturing Technology*.

Schaller, J. (2001). A new lower bound for the flow shop group scheduling problem. *Computers & Industrial Engineering*, *41*(2), 151–161. https://doi.org/10.1016/S0360-8352(01)00049-3

Schaller, J., Gupta, J. N. D., & Vakharia, A. J. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*, *125*(2), 324–339. https://doi.org/10.1016/S0377-2217(99)00387-2

Simon, D. (2013). *Evolutionary optimization algorithms*. John Wiley & Sons, Inc.

Suman, B., & Kumar, P. (2005). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, *57*(10), 1143–1160. https://doi.org/10.1057/palgrave.jors.2602068

Talbi, E.-G. (2009). Metaheuristics: from design to implementation. *Jonh Wiley and Sons Inc., Chichester*.

Tan, K. C., Goh, C. K., Yang, Y. J., & Lee, T. H. (2006). Evolving better population distribution and exploration in evolutionary multi-objective optimization.

European Journal of Operational Research, *171*(2), 463–495. https://doi.org/10.1016/j.ejor.2004.08.038

Varadharajan, T. K., & Rajendran, C. (2005). A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *European Journal of Operational Research*, *167*(3), 772–795. https://doi.org/10.1016/j.ejor.2004.07.020

Zhao, F., Tang, J., & Wang, J. (2014). An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem. *Computers & Operations Research*, *45*, 38–50. https://doi.org/10.1016/j.cor.2013.11.019

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, *8*(2), 173–195. https://doi.org/10.1162/106365600568202