A Hybrid Particle Swarm Optimization and Simulate Annealing (PSO-SA) Algorithm for Scheduling a Flowshop Manufacturing cell with Sequence Dependent Setup Times

Al-Mehdi M. Ibrahem,
College of Engineering, University of Gharayn, Libya
Email: umalmehd@myumanitoba.ca
Tarek Elmekkawy
College of Engineering, Qatar University, Qatar
Email:tmekkawy@qu.edu.qa
and
Hussein Elmehdi
College of Sciences, University of Sharjah, AUE
Email:hmelmehdi@sharjah.ac.ae

Abstract

In cellular manufacturing systems, minimization of the completion time has a great impact on production time, material flow, and productivity. An effective scheduling is crucial to attaining the advantages of cellular manufacturing systems.

In this paper, a Hybrid Particle Swarm Optimization (PSO-SA) algorithm is proposed to solve a cellular flowshop scheduling problem with family sequence-dependent setup time. The proposed PSO-SA algorithm combines Particle Swarm Optimization (PSO) algorithm with Simulated Annealing (SA) as a local search to balance between diversification and intensification. The objective is to find the best sequence of families as well as jobs in each family in order to minimize total flow time; the problem is classified as: $F_m \setminus fmls, Se_{lki}, prum \setminus \sum_{j=1}^{N} C_j$.

The research problem is shown to be an NP-hard problem. PSO-SA is developed to improve the effectiveness of the PSO algorithm and to reduce the average variation from the lower bounds.

The performance the proposed PSO-SA is evaluated based on the Relative Percentage Deviation (RPD) from lower bounds and compared with the best available algorithm.

Results showed that the hybridization of the PSO with SA improves the quality of the PSO algorithm and reduces the gap from the lower bounds especially for large problems.

keywords

Particle Swarm Optimization, Simulated Annealing, Cellular flowshop, Sequencing-dependent setup time, Total Flow Time, Hybridization, Local search.

خوارزمية PSO-SA الهجينة للتخطيط لحلقة تدفق الخلايا مع أوقات ترتيب تعتمد على تسلسل العائلة

في أنظمة التصنيع الخلوية، يكون لتقليل وقت الإكمال تأثير كبير على وقت الإنتاج وتدفق المادّة والإنتاجية. التخطيط الفعّال أمر بالغ الأهمية لبلوغ مزايا أنظمة التصنيع الخلوية. في هذه الورقة، تم اقتراح خوارزمية تحسين الجسيمات المختلطة (PSO-SA) لحل مشكلة جدولة التدفقات الخلوية مع وقت الإعداد المعتمد على التسلسل العائلي. تجمع خوارزمية PSO-SA المقترحة بين خوارزمية تحسين سرب الجسيمات (PSO) وخواص الصلب المحاكاة (SA) كبحث محلي لتحقيق التوازن بين التنويع والتكثيف. الهدف هو إيجاد أفضل تسلسل للعائلات بالإضافة إلى وظائف في كل عائلة من أجل تقليل وقت التدفق الإجمالي. تبين أن مشكلة البحث مشكلة NP- صعبة. تم تطوير PSO-SA لتحسين فعالية خوارزمية PSO ولتقليل التباين المتوسط مقارنة بالحدود الدنيا. يتم تقييم أداء PSO-SA المقترح بناءً على الفجوة النسبية (RPD) من الحدود الدنيا ومقارنته بأفضل خوارزمية متاحة. أظهرت النتائج أن تهجين PSO مع SA يحسن جودة خوارزمية PSO ويقلل الفجوة للحدود الدنيا خاصةً للمشاكل الكبيرة.

1 Introduction and Problem definition

Flowshop group scheduling problem has received much attention in the academic and practice-oriented literature Due to its practical relevance [1]. Manufacturing organizations seek productive efficiency or cost-effectiveness solutions by competing via fast time to market and low production costs [2]. Cellular Manufacturing (CM) uses Group Technology (GT) in grouping machines according to parts processed. GT is a philosophy to put the products with similar design or manufacturing characteristics or both in one group [3]. CM aims to improve the productivity by grouping the parts into part families based on their similarity such as production requirements and setup times. Further, production scheduling is a decision-making process that improves the utilization; it deals with the allocation of resources to tasks over given periods and its goal is to optimize one or more objectives [4]. An efficient job scheduling is a crucial aspect of any manufacturing environment. Thus, the next step for improving the efficiency of a manufacturing cell is to find the best sequence of processing the assigned parts (jobs). The problem is (classified as: $F_m \setminus fmls, Se_{lki}, prum \setminus \sum_{j=1}^{N} C_j$); and known as Flowshop Manufacturing Cell Scheduling Problem (FMCSP) or Cellular Flowshop Scheduling Problem [5]. Most of the existing research focuses on minimization of Makespan (MS) due to its lower computational complexity. MS computes the maximum completion time; the completion time of last job to be processed on the last machine. Total Flow Time (TFT) computes the sum of completion times of jobs over the last machine. Minimization of TFT reflects a stable utilization of resources, reduces the work-in-process inventory, and minimizes setup times costs. Therefore, TFT is more relevant to a dynamic production environment [6]. Moreover, the minimization of TFT of the jobs will reduce the production time, and decrease the number of delayed deliveries. Consequently, productivity, and profitability of the firm will be increased. For example, a company may receive several orders from different customers, and all of them have the same priority (weight) for the company. Minimization of TFT is appropriate as it would indirectly minimize the Work- In-Process inventories (WIP). Therefore, The main goal of this paper is to find the best sequence of part families as well as the jobs or parts in a part family in order to minimize total flow time.

The mentioned problem "Scheduling a flowshop of cellular manufacturing systems with family setup times" was studied for the first time by [7]. Schaller et al. developed several heuristic algorithms with minimization of the makespan as the criteria. Further, they developed a lower bounding method to evaluate the solution quality of the proposed heuristic algorithms. A Genetic algorithm (GA) and a Memetic algorithm (MA) with local search are proposed by [8] for the makespan minimization. They concluded that the solution quality of the MA was outperformed the available algorithms. Hendizadeh, presented various TS based meta-heuristics for FMCSPs with SDSTs to minimize makespan [9]. They proposed the concepts of elitism and the acceptance of worse moves from SA to improve intensification and diversification.

1.1 Problem Statement and Assumptions

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 in order to minimize the total flow time and makespan simultaneously. Using the triplet notation [4], the problem can be notated as: $F_m \setminus fmls, Se_{lki}, prum \setminus C_{max}, \sum_{j=1}^{N} C_j$. 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 [10, 11, **?**, 12, 13, 14]. The sequence of part families and the parts within each part family are the same on all machines (permutation flowshop). As shown in Figure 1, for a feasible schedule, a solution π of FMCSP takes the following structure:

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

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. Figure 1 shows solution structure of part families as well as jobs in each part family:

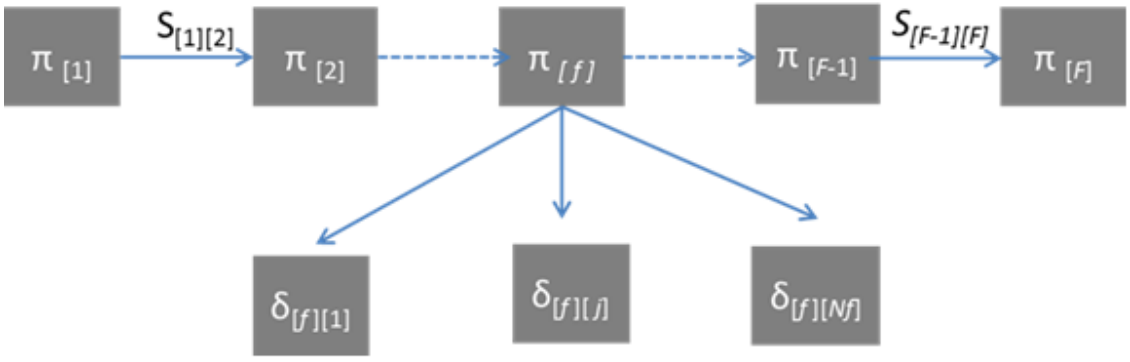

Figure 1: The solution structure of a Flowshop Manufacturing Cell Scheduling Problem [12].

The problem is classified as: $F_m \setminus fmls, Se_{lki}, prum \setminus \sum_{j=1}^{N} C_j$. The objective is to minimization of the Tolal Flow Time (TFT); The following assumptions are used in this research:

- The number of parts (jobs), their processing times, the number of part families, and their setup times are known in advance.
- The sequence of parts and part families are the same on all machines (permutation scheduling).
- Once a part starts to be processed on a machine, the process cannot be interrupted before completion (pre-emption is disallowed).
- Each machine can handle only one part (job) at a time.
- All buffers have unlimited size.
- The jobs belonging to each family should be processed without any preemption by other jobs of other families (group technology assumption).
- The ready time of all parts is zero, i.e, all parts in all part families are available for processing at the start time.
- Setup time depends on both the part to be processed and its preceding part. There is a minor setup time among parts within a family but there is a major (considerable) setup time among the part families.

The minimization of the total flow time was studied for the first time by Salmasi et al [15]. They developed two meta-heuristics based on Tabu Search (TS) and Hybrid Ant Colony Optimization (ACO) algorithm to minimize TFT. They showed that the algorithm had a superior performance compared to the TS algorithm and it has been considered as the best available metaheuristic. They developed an efficient lower bound based on the Branch- and-Price for total flow time minimization. Naderi et al [16] studied the proposed problem, and they developed two different mixed integer linear programming models and showed that the models are effective in solving small and medium sized problems and provide the optimum solution in a reasonable time. They developed a hybrid genetic and simulated annealing algorithm, called (GSA) to solve the problem heuristically and proved that the proposed mathematical models and the proposed metaheuristic algorithm outperformed the available algorithms in the literature.
As a result of the limited work done on the TFT minimization, more algorithms are needed to minimize the TFT. The contribution of the work is to develop a hybrid algorithms combining PSO and SA to balance between diversification and intensification.furthermore, the PSO-SA is developed to improve the effectiveness of the PSO algorithm and to reduce the average variation from the lower

bounds and to improve the quality of the obtained solutions by PSO algorithm. Therefore, this paper presents the a hybrid algorithm (PSO-SA) to solve the FMCSPs with SDSTs to minimize the TFT.

2 Proposed Algorithms: PSO, and PSO-SA

Particle Swarm Optimization (PSO) is a novel iterative computational evolution model that was developed by Kennedy and Eberhart [17]. An overview of important work and research directions on particle swarms as well as applications are presented by Poli et al [18]. Liu proposed an effective particle swarm optimization (PSO) based memetic algorithm (MA) for the permutation flow shop scheduling problem to minimize the makespan [19]. A novel PSO algorithm for the permutation flowshop scheduling is applied by Lian et al [20] to minimize makespan. Tseng developed a PSO algorithm for the scheduling of multiprocessor tasks in a multistage hybrid flowshop with makespan minimization [21]. Kuo proposed a hybrid PSO model named (HPSO) that combines random-key encoding scheme, individual enhancement (IE) scheme, and PSO to solve the flowshop scheduling problem to minimize makespan [22]. RameshKumar et al [23] proposed a Discrete Particle Swarm Optimization (DPSO) algorithm to solve permutation flowshop scheduling problems with the objective of minimizing the makespan . Lin et al [24] presented a hybrid algorithm combined PSO algorithm with SA technique, and multi-type individual enhancement scheme to solve the job-shop scheduling problem. Liu et al [25] proposed a hybrid PSO with estimation of distributed algorithms to solve permutation flowshop scheduling problem to minimize the makespan. Gohar and Salmasi proposed several hybrid metaheuristic algorithms based on PSO and SA to heuristically solve the flexible flow-line scheduling problem by considering constraints for the beginning and terminating times of processing the jobs. The objective again was to minimize the makespan [26].

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 new better 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 p-best, 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:

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

$$x_j(t+1) = v_j(t+1) + x_j(t) \tag{2}$$

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 g-best particles respectively. The values of these parameters are updated in each iteration based on the following equations:

$$w = w^{min} + \frac{w^{max} - w^{min}}{1 + e^{\frac{-a(maxI-I)}{maxI}}} \tag{3}$$

$$C_1 = C_1^{min} + \frac{C_1^{max} - C_1^{min}}{1 + e^{\frac{-a(maxI-I)}{maxI}}} \tag{4}$$

$$C_2 = C_2^{min} + \frac{C_2^{max} - C_2^{min}}{1 + e^{\frac{-a(I)}{maxI}}} \tag{5}$$

The maximum and minimum values of the above parameters are presented in Table 1, 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 (1), and equation (2). 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 to the new velocity and the previous position equation (1) and equation (2). Liu et al [27] 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.

Table 1: 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

2.1 Initial Swarm

The algorithm starts with generating the initial velocity and position for j^{th} particle in the swarm of size $N\{j \in \{1,2,.....,N\}$ according to the following equations:

$$X_{0i} = X_{min} + r_1(X_{max} - X_{min}), \tag{6}$$

$$V_{0i} = V_{min} + r_2(V_{max} - V_{min}), \tag{7}$$

The subscript 0 refers to the starting point, i.e., before the iterations are started. X_{min}, and X_{max} represent the minimum and maximum position values. V_{min}, and V_{max} represent the minimum and maximum velocities. Also r_1 and r_2 are random variables in the range between $[0,1]$. Therefore, the initial position of particle X_i will be as follows:

$$X_i = \begin{pmatrix} X_{i11} & X_{i12} & \cdots & X_{i1,N_1} \\ X_{i21} & X_{i22} & \cdots & X_{i2,N_2} \\ \vdots & \vdots & \ddots & \vdots \\ X_{if1} & X_{if2} & \cdots & X_{if,N_f} \\ X_{i(F+1)1} & X_{i(F+1)2} & \cdots & X_{i(F+1)F} \\ \end{pmatrix} \tag{8}$$

Particle flies in the search space with velocity matrix V_i

$$V_i = \begin{pmatrix} V_{i11} & V_{i12} & \cdots & V_{i1,N_1} \\ V_{i21} & V_{i22} & \cdots & V_{i2,N_2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{if1} & V_{if2} & \cdots & V_{if,N_f} \\ V_{i(F+1)1} & V_{i(F+1)2} & \cdots & V_{i(F+1)F} \\ \end{pmatrix} \tag{9}$$

Where F refers to number of families and N_f is maximum number of jobs of family f.

2.2 Ranked Order Value (ROV)

Ranked Order Value (ROV) is based on the random representation of permutation of jobs [22], [27]. ROV is included in the proposed PSO algorithms to convert the continuous position of particles to a permutation of families as well as jobs. Basically, ROV is applied based on Smallest Particle Value (SPV); firstly handled and assigned a smallest rank value 1. Then, the second SPV is assigned a rank value 2. Similarly, all the position values will be dealt with to convert the position information of a particle to a family and a job sequence. For example, if we have particles information about a sequence of 4 families with these values of $X_i = \{0.12, 0.22, 0.06, 1.14\}$, based on ROV method, the sequence of these families is $\{3, 2, 1, 4\}$. ROV is used to convert all random numbers generated to a sequence of families as well as the jobs in each iteration (after updating the velocity and position of particles in the swarm). The steps of the PSO algorithm are executed as follows in Algorithm 1.

In this study, two metaheuristics based on Particle Swarm Optimization (PSO) which is a population-based metaheuristic, and Simulated Annealing (SA) which is a single solution based metaheuristic are combined into two different strategies. First, PSO is used to generate the initial solution of the SA algorithm. Second, SA is combined with PSO algorithm as a local search engine. The fundamental concept of the cooperation between PSO and SA is due to the following reasons: 1) The ability of PSO to locate high performance regions of vast search spaces quickly; 2) The simplicity of SA algorithm; and 3) The successful implementation of SA for solving scheduling problems. Therefore, PSO can be applied to locate promising regions. Then, it is highly recommended to apply a single solution based algorithm (such as SA) which is a powerful optimization method in terms of exploitation [28] to be combined with PSO as a local search.

Algorithm 1 The steps of the proposed PSO algorithm

Require: Initialize parameters { swarm size n, maximum Iteration I_{max}, C_{1max}, C_{1min}, $C_{2max}, C_{2min}, V_{max}, V_{min}, W_{max}, W_{min}, X_{max}, X_{min}$}

- 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 (6, and 7)
  Else
  Generate a new swarm by updating the velocity V_i^t and position X_i^t of particles according to equations (1, and 2)
- Step 3: Apply the (ROV) on X_i^t to find the sequence of families as well as jobs in families.
- Step 4: Calculate the objective function $f\ (X_i^t)$ for each particle $i \in \{1, 2, ..., n\}$
  - o Step 4.1 For each particle in the swarm the p-best (P_i^t) is calculated as:

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

  - o Step 4.2 The g-best can be calculated as:

$$G^t = \underset{P_i^t}{\operatorname{argmin}} f(P_i^t) \qquad for \quad i \in \{1, 2, ..., n\}$$

- Step 5 Set $t = t + 1$
- Step 6 If $t = I_{max}$, STOP; else, go to step 3.

2.3 Simulated Annealing (SA) Algorithm

SA is a meta-heuristic approach that can provide optimal (or near-optimal) solutions to combinatorial optimization problems. Since its introduction by Kirkpatrick et al (1983) [29]. SA has been applied to a vast number of optimization problems. SA approach starts from an initial sequence (current solution), and then moves successively among the neighboring sequences to generate another solution. Basically, SA is a two step process: perturb (generate a new solution), and then evaluate the quality of the new solution [30]. Eglese provided an overview to implement the SA algorithm [31]. For the sake of getting an overview about simulated annealing algorithm readers are referred to references [32, 33]. A metaheuristic based on SA is proposed by Vakharia et al [34] to schedule part families as well as jobs with each part family for FMCSP with SDST. Sridhar et al [35] proposed an algorithm based on SA for scheduling the FMCSP (without considering the setup times) to minimize the total flow time.

Annealing is the process through which slow cooling of metal produces uniform, low energy-state crystallization, whereas fast cooling produces poor crystallization. The optimization procedure of SA algorithm to find a near global minimum mimics the crystallization cooling procedure. SA procedure starts with a random initial solution as the current solution. Then, the algorithm generates a new solution from the predetermined neighbourhood. The initial solution is randomly generated by using the initial position and velocity according to equations 6, and 7. The steps of SA algorithm are shown in Algorithm 2. The fitness value of the new solution is then compared with the current solution to determine if the new one is better. For minimization problems, if the fitness value of the new solution is smaller than that of the current one, the new solution is automatically accepted and becomes the current solution from which the search continues. The algorithm will then proceed with further iterations. Larger fitness values for next solution may also be accepted as the current solution under certain conditions to escape from a local minimum.

2.4 SA procedure

The procedure of the proposed SA starts with setting the levels of parameters. The current state or temperature T is set to the initial value T_0. An initial solution X is randomly generated and it is considered as the current solution. For each iteration, the next solution Y is generated from the current solution by perturbation on job sequence or on family sequence using the swapping and insertion techniques. Consider E_i as the energy state of the fitness value of current solution and E_j is the fitness value of the new solution obtained as a result of the manipulation of the job sequence within a family F. Let $\triangle E = E_j$ - E_i; which refers to the difference between $\triangle E = TFT(Y)$ - $TFT(X)$. If $\triangle E \leq 0$, the probability of replacing current solution X with the new solution Y is 1. If $\triangle E > 0$, then, probability of accepting the new solution depends on the Cauchy $\left(\frac{T}{T^2+(\triangle E)^2}\right)$ function. The new solution replaces the current solution, if $\left(\frac{T}{T^2+(\triangle E)^2}\right)$ is greater than some random number RN between 0 and 1. Then, temperature T is reduced based on cooling rate after running maximum number of iterations ($Iite$) from the previous decrease, according to the formula (the typical value of α is 0.95). The algorithm is terminated if T is lower than T_f. During the search evolutions, the best solution with the least total flow time is recorded. The algorithm is also terminated if the current solution is not improved in non-improving successive reductions in temperature. Following the termination of the SA procedure, the near- global optimal schedule is X_{best} with total flow time TFT_{best}.

2.5 Hybrid Particle Swarm Optimization algorithm

In designing metaheuristics, two conflicting criteria must be taken into account: exploration of the search space (diversification) and exploitation of the best solutions found (intensification) [**?**]. Promising regions are determined by the obtained good solutions. On one hand, in intensification,

Algorithm 2 The steps of the proposed SA algorithm

Require: Initialize SA parameters and set a solution X to be current solution { T_0, T_F, $\alpha = 0.95, I_{iter}, N_{non-imp.}$}

- Step 1: Set iteration $t = 0$
- Step 2: For F=1 to Family size
  Set $T = T_0$, No. of Moves = 1, and counter = 1
  - Step 2.1: while $T > T_F$ do:
    - Step 2.1.1: if F=1, Generate by swapping the family sequence
      Else
      Generate a new neighbour solution Y from current solution X by minor swapping on job sequence of F^{th} family, and calculate the difference in objective functions of X, and Y to be $\triangle E$= $TFT(Y)$ - $TFT(X)$
    - Step 2.1.2: IF $\triangle E \leq 0$, then replace the current solution with the new solution and go to Step 2.1.4; otherwise, go to Step 2.1.3
    - Step 2.1.3: Generate a random number RN in the range $[0, 1]$
      IF$RN < \frac{T}{T^2+(\triangle E)^2}$, then replace solution with solution Y and go to Step 2.1.4
    - Step 2.1.4: Increment No. of Moves by one. IF No. of Moves $< N_{non-imp.}$, then go to Step 2.1.1; Else, go to Step 2.1.5
    - Step 2.1.5: IF the best solution is improved, then restart the counter from zero; otherwise, increase the counter by one unit
    - Step 2.1.6 IF counter = I_{iter}, decrease the temperature using geometric schedule T= αT , make No. of Moves =1, and go to Step 2.1.
- Step 3: Return the current solution in the end as X_{best} with TFT_{best}

the promising regions are explored more thoroughly in the hope to find better solutions. On the other hand, in diversification non-explored regions have be 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. The hybrid approach implanted in this study to balance between intensification and diversification is shown in Figure 2. PSO is used to perform the global search, and SA algorithm is combined as a local search to improve quality of search.

The main goal of hybridizing the PSO algorithm is to improve the quality of the obtained solution

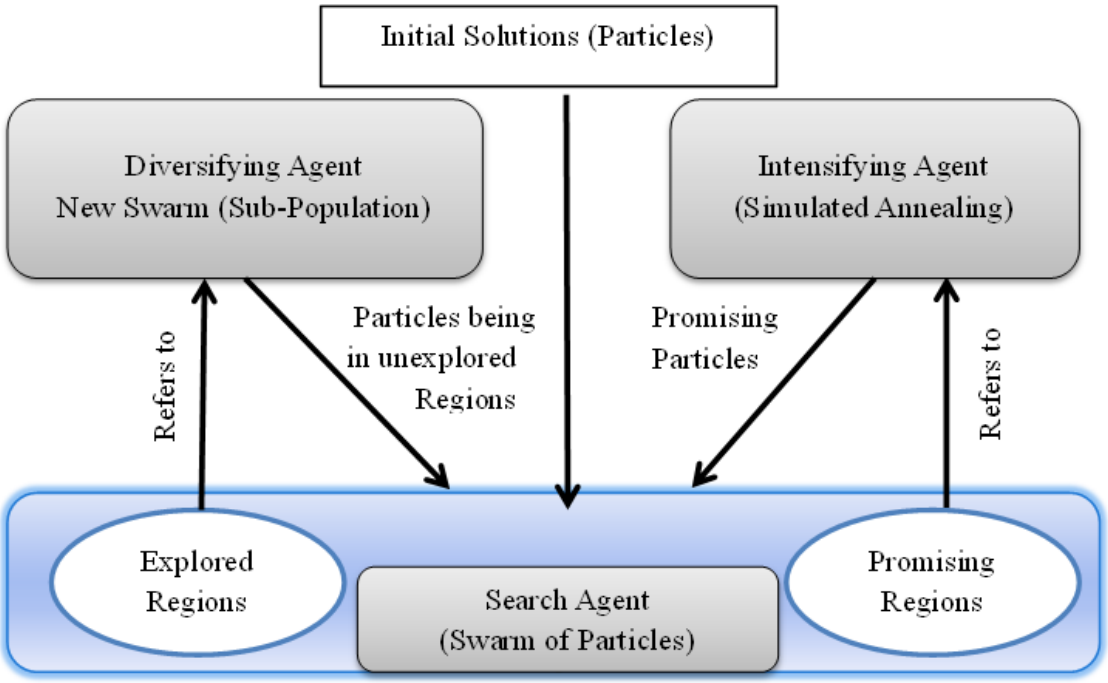

Figure 2: Hybridization of particle swarm optimization and simulated annealing

using a pure PSO metaheuristic. SA is basically combined with PSO as a local search to find better solutions. At the first stage, PSO is proposed to solve the studied problem, and it showed a better performance than other algorithms such as GA proposed by the author [36]. The proposed PSO algorithm provides very good solutions that matched the best existing algorithms for small and medium instances. For large problems, the performance of the PSO algorithm deviates from the best solutions due to the large solution space, and the higher probability to be trapped in local optima. Therefore, PSO has been combined with SA to improve the quality of solutions and reduce the variation from the lower bounds for the large problem instances. The main role of combining SA algorithm with PSO is to enhance the intensification and search for a better solution within the neighbourhoods (this combination is named PSO-SA).

In the PSO-SA, each particle flies in the solution space seeking for a better position based on its best experience (p-best), and the global experience (g-best) based on the new velocity and new position (equations 6, and 7). Therefore, particles are enhanced to move toward the global optima. A local search engine based on SA is implemented to search on the neighbourhood of the g-best particle each iteration by using swapping of the job sequences in families. Note that swapping is implemented only to the job sequence for simplicity. Local search is conducted on the g-best particle in each iteration. The procedure of the proposed PSO-SA algorithm is similar to the proposed PSO algorithm. As compared to algorithm 1, the PSO-SA algorithm shown in algorithm 3 differs only in step 6, where a local search based on SA (denoted by L.S) is incorporated with PSO to improve g-best particle that improves the searching and finding promising regions in next iterations of PSO.

Algorithm 3 The steps of the proposed PSO-SA algorithm

Require: Initialize parameters { swarm size n, maximum Iteration I_{max}, C_{1max}, C_{1min}, $C_{2max}, C_{2min}, V_{max}, V_{min}, W_{max}, W_{min}, X_{max}, X_{min}$}

- 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 (6, and 7)
  Else
  Generate a new swarm by updating the velocity V_i^t and position X_i^t of particles according to equations (1, and 2)
- Step 3: Apply the (ROV) on X_i^t to find the sequence of families as well as jobs in families.
- Step 4: Calculate the objective function $f\ (X_i^t)$ for each particle $i \in \{1, 2, ..., n\}$
  - o Step 4.1 For each particle in the swarm the p-best (P_i^t) is calculated as:

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

  - o Step 4.2 The g-best can be calculated as:

$$G^t = \underset{P_i^t}{\operatorname{argmin}} f(P_i^t) \qquad for \quad i \in \{1, 2, ..., n\}$$

- Step 5 Set $t = t + 1$
- Step 6 Set $G^t = L.S(G^t)$
- Step 7 If $t = I_{max}$, STOP; else, go to step 2.

3 Results and discussion

The proposed PSO-SA algorithm is coded using C++ language and run on a PC with an Intel ® core I7 (2.93 GHz) CPU and 4.0 GB memory. The analysis of performance of PSO-SA versus the other proposed algorithms is presented in Section 4.5. The performance of the proposed PSO-SA algorithm is compared with the best available algorithms in the literature. PSO-SA performs similar to the best available algorithm (ACO) developed by Salmasi et al [**?**] for solving most of the test problems for the six-machine test. However, there is a very small deviation from the best in solving the large size problems for two-machine test problems as shown in Figures 3 and 4. As a result of the adding of SA algorithm as a local search, the performance of the PSO algorithm has been improved and matched with the best available algorithm.

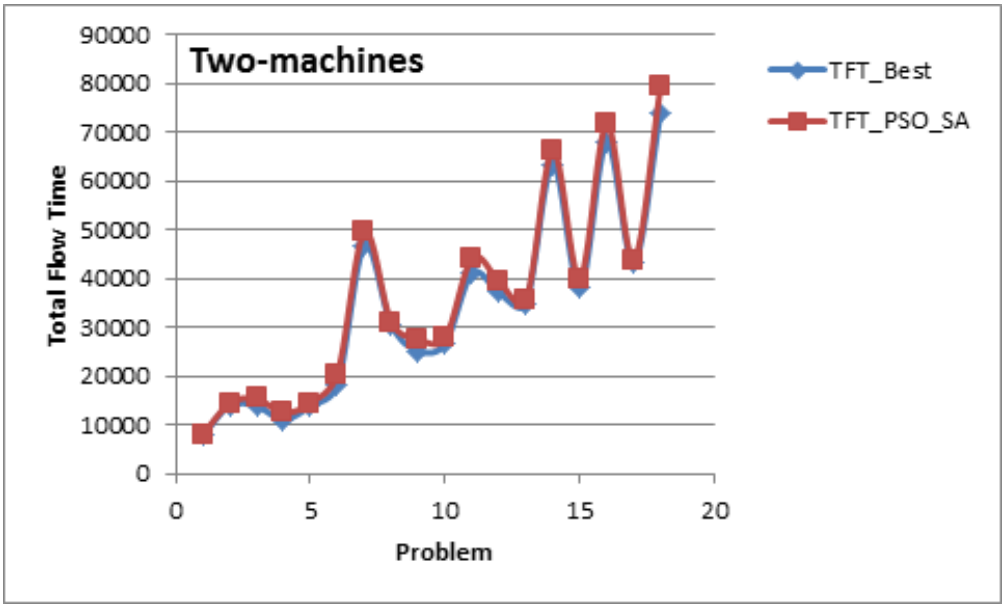

Figure 3: Performance of the proposed PSO-SA vs. ACO for 2-machine problems

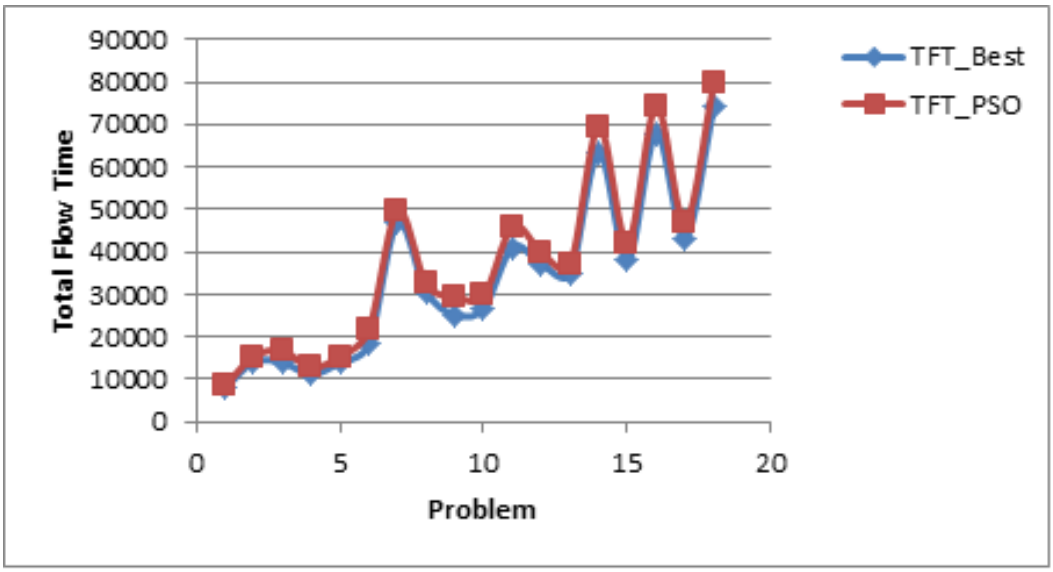

Figure 4: Performance of the proposed PSO vs. ACO for 2-machine problems

4 Conclusion

A Hybrid Particle Swarm Optimization (PSO-SA) algorithm has been developed to solve the Flow-shop Manufacturing Cell Scheduling Problem with Sequence Dependent Setup Times (FMCSP with SDSTs) to minimize the total flow time. Results showed that the PSO-SA is outperform the PSO algorithm especially for the large problems. The results of proposed algorithm was compared with best available algorithm and showed the same results in a reasonable computation time. Implementing the SA algorithm as a local search improves the quality of the obtained solutions due to the balance between diversification and intensification. As a result of the limited work on the

proposed problem, there is room for designing more algorithms based on a pure metaheuristic algorithm or combination of different metaheuristics. For instance, Ant Colony Optimization (ACO) could be combined with a single solution based algorithm such as Variable Neighborhood Search (VNS).

References

[1] J. S. Neufeld, J. N. D. Gupta, and U. Buscher, "A comprehensive review of flowshop group scheduling literature," *Computers and Operations Research*, vol. 70, pp. 56–74, 2016.

[2] F. Zhao, J. Tang, and J. Wang, "An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem," *Comput. Oper. Res.*, vol. 45, pp. 38–50, may 2014.

[3] M. Zandieh, B. Dorri, and a. R. Khamseh, "Robust metaheuristics for group scheduling with sequence-dependent setup times in hybrid flexible flow shops," *Int. J. Adv. Manuf. Technol.*, vol. 43, no. 7-8, pp. 767–778, sep 2008.

[4] M. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer, 2012.

[5] S. Lin, K. Ying, and Z. Lee, "Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1110–1121, apr 2009.

[6] L.-Y. Tseng and Y.-T. Lin, "A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem," *Int. J. Prod. Econ.*, vol. 127, no. 1, pp. 121–128, sep 2010.

[7] J. Schaller, J. N. D. Gupta, and A. J. Vakharia, "Scheduling a flowline manufacturing cell with sequence dependent family setup times," *Eur. J. Oper. Res.*, vol. 125, no. 2, pp. 324–339, sep 2000.

[8] P. Franca, J. Gupta, A. Mendes, P. Moscato, and K. Veltink, "Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups," *Comput. Ind. Eng.*, vol. 48(3), pp. 491–506, 2005.

[9] S. Hamed Hendizadeh, H. Faramarzi, S. Mansouri, J. N. Gupta, and T. Y ElMekkawy, "Metaheuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times," *Int. J. Prod. Econ.*, vol. 111, no. 2, pp. 593–605, feb 2008.

[10] S.-W. Lin, J. N. Gupta, K.-C. Ying, and Z.-J. Lee, "Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times," *Int. J. Prod. Res.*, vol. 47, no. 12, pp. 3205–3217, jun 2009.

[11] N. Salmasi, R. Logendran, and M. R. Skandari, "Makespan minimization of a flowshop sequence-dependent group scheduling problem," *Int. J. Adv. Manuf. Technol.*, feb 2011.

[12] R. Bouabda, B. Jarboui, M. Eddaly, and A. Rebai, "A branch and bound enhanced genetic algorithm for scheduling a flowline manufacturing cell with sequence dependent family setup times," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 387–393, 2011.

[13] M. Eddaly, B. Jarboui, R. Bouabda, and A. Rebai, "Hybrid Estimation of distribution algorithm for permutation flowshop scheduling with dependent family setup times," in *2009 Int. Conf. Comput. Ind. Eng.*, 2009, pp. 217–220.

[14] S. Hamed Hendizadeh, H. Faramarzi, S. Mansouri, J. N. Gupta, and T. Y ElMekkawy, "Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times," *International Journal of Production Economics*, vol. 111(2), pp. 593–605, 2008.

[15] N. Salmasi, R. Logendran, and M. R. Skandari, "Total flow time minimization in a flowshop sequence-dependent group scheduling problem," *Computers & Operations Research*, vol. 37, no. 1, pp. 199–212, jan 2010.

[16] B. Naderi and N. Salmasi, "Permutation flowshops in group scheduling with sequence- dependent setup times," *Eur. J. Ind. Eng.*, vol. 6, no. 2, pp. 177–198, 2012.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. ICNN'95 - Int. Conf. Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[18] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, aug 2007.

[19] B. Liu, L. Wang, and Y.-H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling." *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 37, no. 1, pp. 18–27, mar 2007.

[20] Z. Lian, X. Gu, and B. Jiao, "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan," *Chaos, Solitons & Fractals*, vol. 35, no. 5, pp. 851–861, mar 2008.

[21] C.-T. Tseng and C.-J. Liao, "A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *Int. J. Prod. Res.*, vol. 46, no. 17, pp. 4655–4670, sep 2008.

[22] I.-H. Kuo, S.-J. Horng, T.-W. Kao, T.-L. Lin, C.-L. Lee, T. Terano, and Y. Pan, "An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 7027–7032, apr 2009.

[23] K. Rameshkumar, R. Suresh, and K. Mohanasundaram, "Discrete particle swarm optimization (DPSO) algorithm for permutation flowshop scheduling to minimize makespan," *Adv. Nat. Comput.*, pp. 572–581, 2005.

[24] T.-L. Lin, S.-J. Horng, T.-W. Kao, Y.-H. Chen, R.-S. Run, R.-J. Chen, J.-L. Lai, and I.-H. Kuo, "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2629–2636, mar 2010.

[25] H. Liu, L. Gao, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4348–4360, apr 2011.

[26] S. Gohari and N. Salmasi, "Flexible flowline scheduling problem with constraints for the beginning and terminating time of processing of jobs at stages," *Int. J. Comput. Integr. Manuf.*, pp. 1–14, 2014.

[27] B. Liu, L. Wang, and Y.-H. Jin, "An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2791–2806, sep 2008.

[28] E.-G. Talbi, *Hybrid Metaheuristics*, ser. Studies in Computational Intelligence, E.-G. Talbi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 434.

[29] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simmulated annealing," *Science*, pp. 671–680, 1983.

[30] S. Ledesma, G. Aviña, and R. Sanchez, "Practical considerations for simulated annealing implementation," *Simulated Annealing*, no. February, pp. 401–420, 2008.

[31] R. W. Eglese, "Simulated annealing: a tool for operational research," *Eur. J. Oper. Res.*, vol. 46, pp. 271–281, 1990.

[32] R. Rutenbar, "Simulated annealing algorithms: An overview," *Circuits Devices Mag. IEEE*, no. x, 1989.

[33] F. Busetti, "Simulated annealing overview," *World Wide Web URL www. geocities. com/ DOI10.1.1.66.5018*, 2003.

[34] A. J. Vakharia and Y.-L. Chang, "A simulated annealing approach to scheduling a manufacturing cell," *Nav. Res. Logist.*, vol. 37, no. 4, pp. 559–577, aug 1990.

[35] J. Sridhar and C. Rajendran, "Scheduling in a cellular manufacturing system: a simulated annealing approach," *Int. J. Prod. Res.*, vol. 31, no. 12, pp. 2927–2945, dec 1993.

[36] A.-m. Ibrahem, T. Elmekkawy, and Q. Peng, "Robust Metaheuristics for Scheduling Cellular Flowshop with Family Sequence- Dependent Setup Times," *Procedia CIRP*, vol. 17C, pp. 428–433, 2014.