Variable Neighborhood Particle Swarm Optimization for Multi-objective Flexible Job-Shop Scheduling Problems

Hongbo Liu[1,2], Ajith Abraham[3,1], Okkyung Choi[3,4], and Seong Hwan Moon[4]

[1] School of Computer Science, Dalian Maritime University, 116024, Dalian, China
[2] Department of Computer, Dalian University of Technology, 116023, Dalian, China
[3] School of Computer Science and Engineering, Chung-Ang University, Seoul 156-756, Korea
[4] Department of Science and Technology, Education for Life, Seoul National University of Education, Seocho-dong, Seocho-gu, Seoul 137-742, Korea
lhb@dlut.edu.cn, ajith.abraham@ieee.org, ok1124@freechal.com, hmoon@snue.ac.kr

Abstract. This paper introduces a hybrid metaheuristic, the Variable Neighborhood Particle Swarm Optimization (VNPSO), consisting of a combination of the Variable Neighborhood Search (VNS) and Particle Swarm Optimization(PSO). The proposed VNPSO method is used for solving the multi-objective Flexible Job-shop Scheduling Problems (FJSP). The details of implementation for the multi-objective FJSP and the corresponding computational experiments are reported. The results indicate that the proposed algorithm is an efficient approach for the multi-objective FJSP, especially for large scale problems.

1 Introduction

Flexible Job-shop Scheduling Problems (FJSP) is an extension of the classical JSP which allows an operation to be processed by any machine from a given set. It incorporates all the difficulties and complexities of its predecessor JSP and is more complex than JSP because of the additional need to determine the assignment of operations to the machines. The scheduling problem of a FJSP consists of a routing sub-problem, that is, assigning each operation to a machine out of a set of capable machines and the scheduling sub-problem, which consists of sequencing the assigned operations on all machines in order to obtain a feasible schedule minimizing a predefined objective function. It is quite difficult to achieve an optimal solution with traditional optimization approaches owing to the high computational complexity. In the literature, different approaches have been proposed to solve this problem. Mastrolilli and Gambardella [1] proposed some neighborhood functions for metaheuristics. Kacem et al. [2,3] studied on modeling genetic algorithms for FJSP. Ong et al. [4] applied the clonal selection principle of the human immune system to solve FJSP with re-circulation. By hybridizing particle swarm optimization and simulated annealing, Xia and

Wu [5] developed an hybrid approach for the multi-objective flexible job-shop scheduling problem (FJSP). Because of the intractable nature of the problem and its importance in both fields of production management and combinatorial optimization, it is desirable to explore other avenues for developing good heuristic algorithms for the problem.

Particle Swarm Optimization (PSO) incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the intelligence is emerged [6]. It has become the new focus of research recently. However, its performance deteriorates as the dimensionality of the search space increases, especially for the multi-objective FJSP involving large scale. PSO often demonstrates faster convergence speed in the first phase of the search, and then slows down or even stops as the number of generations is increased. Once the algorithm slows down, it is difficult to achieve better scheduling solutions. To avoid termination at a local minimum, we introduce a novel hybrid meta-heuristic, the Variable Neighborhood Particle Swarm Optimization (VNPSO) for the multi-objective FJSP. The basic idea is to drive those particles by a shaking strategy and get them to explore variable neighborhood spaces for the better scheduling solutions.

2 Problem Formulation

The classical FJSP considers in general the assignment of a set of machines $M = \{M_1, \cdots, M_m\}$ to a set of jobs $J = \{J_1, \cdots, J_n\}$, each of which consists of a set of operations $J_j = \{O_{j,1}, \cdots, O_{j,p}\}$. There are several constraints on the jobs and machines, such as (1) each machine can process only one operation at a time; (2) operations cannot be interrupted; (3) there are no precedence constraints among operations of different jobs; (4) setup times for the operations are sequence-independent and included in the processing times; (5) there is only one of each type of machine; (6) machines are available at any time.

To formulate the objective, define $C_{i,j,k}$ ($i \in \{1, 2, \cdots, m\}$, $j \in \{1, 2, \cdots, n\}$, $k \in \{1, 2, \cdots, p\}$) as the completion time that the machine M_i finishes the operation $O_{j,k}$; $\sum C_i$ represents the time that the machine M_i completes the processing of all the jobs. Define $C_{max} = max\{\sum C_i\}$ as the makespan, and $C_{sum} = \sum_{i=1}^{m}(\sum C_i)$ as the flowtime. The problem is thus to both determine an assignment and a sequence of the operations on all machines that minimize some criteria. Most important optimality criteria are to be minimized: (1) the maximum completion time (makespan): C_{max}; (2) the sum of the completion times (flowtime): C_{sum}.

Minimizing C_{sum} asks the average job finishes quickly, at the expense of the largest job taking a long time, whereas minimizing C_{max}, asks that no job takes too long, at the expense of most jobs taking a long time. Minimization of C_{max} would result in maximization of C_{sum}. The weighted aggregation is the most common approach to the problems. According to this approach, the objectives, $F_1 = min\{C_{max}\}$ and $F_2 = min\{C_{sum}\}$, are summed to a weighted combination:

$$F = w_1 min\{F_1\} + w_2 min\{F_2\} \tag{1}$$

where w_1 and w_2 are non-negative weights, and $w_1 + w_2 = 1$. These weights can be either fixed or adapt dynamically during the optimization. The dynamic weighted aggregation [7] was used in the paper. Alternatively, the weights can be changed gradually according to the Eqs. (2) and (3). The variation for different values of w_1 and w_2 ($R = 200$) are illustrated in Fig. 1.

$$w_1(t) = |sin(2\pi t/R)| \tag{2}$$

$$w_2(t) = 1 - w_1(t) \tag{3}$$

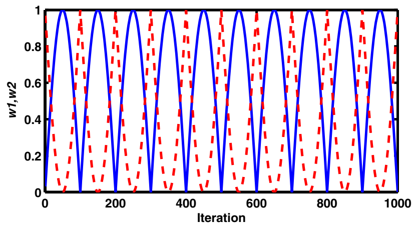

Fig. 1. Dynamic weight variation

3 The VNPSO Heuristic for FJSP

The classical PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimensional problem space to search the new solutions, where the fitness, f, can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector $\boldsymbol{x}_i$ (i is the index of the particle), and a velocity represented by a velocity-vector $\boldsymbol{v}_i$. Each particle remembers its own best position so far in a vector $\boldsymbol{x}_i^{\#}$, and its j-th dimensional value is $x_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\boldsymbol{x}^*$, and its j-th dimensional value is x_j^*. During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined by Eq.(4). The new position is then determined by the sum of the previous position and the new velocity by Eq.(5).

$$v_{ij}(t) = wv_{ij}(t-1) + c_1r_1(x_{ij}^{\#}(t-1) - x_{ij}(t-1)) + c_2r_2(x_j^*(t-1) - x_{ij}(t-1)) \tag{4}$$

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \tag{5}$$

The particle swarm algorithm can be described generally as a population of vectors whose trajectories oscillate around a region which is defined by each individual's previous best success and the success of some other particle. The best particle acts as an attractor, pulling its neighborhood particles towards it. Some previous studies has been shown that the trajectories of the particles oscillate in different sinusoidal waves and converge quickly [8]. During the iteration, the particle is attracted towards the location of the best fitness achieved so far by the particle itself and by the location of the best fitness achieved so far across the swarm. The algorithm has faster convergence. But very often for multi-modal problems involving high dimensions it tends to suffer from premature convergence.

Variable Neighborhood Search (VNS) is a relatively recent metaheuristic which relies on iteratively exploring neighborhoods of growing size to identify better local optima with shaking strategies [9,10]. More precisely, VNS escapes from the current local minimum x^* by initiating other local searches from starting points sampled from a neighborhood of x^*, which increases its size iteratively until a local minimum is better than the current one is found. These steps are repeated until a given termination condition is met. The metaheuristic method we propose, the VNPSO, was originally inspired by VNS. In PSO, if a particle's velocity decreases to a threshold v_c, a new velocity is assigned using Eq.(6):

$$v_{ij}(t) = w\hat{v} + c_1 r_1 (x^{\#}_{ij}(t-1) - x_{ij}(t-1)) + c_2 r_2 (x^*_j(t-1) - x_{ij}(t-1)) \quad (6)$$

Algorithm 1. Variable Neighborhood Particle Swarm Optimization Algorithm

```
01. Initialize the size of the particle swarm $n$, and other parameters.
02. Initialize the positions and the velocities for all the particles randomly.
03. Set the flag of iterations without improvement $Nohope = 0$.
04. While (the end criterion is not met) do
05.   $t = t + 1$;
06.   Calculate the fitness value of each particle;
07.   $\boldsymbol{x}^* = argmin_{i=1}^{n}(f(\boldsymbol{x}^*(t-1)), f(\boldsymbol{x}_1(t)), f(\boldsymbol{x}_2(t)), \cdots, f(\boldsymbol{x}_i(t)), \cdots, f(\boldsymbol{x}_n(t)))$;
08.   If $\boldsymbol{x}^*$ is improved then $Nohope = 0$, else $Nohope = Nohope + 1$.
09.       For $i$= 1 to $n$
10.       $\boldsymbol{x}^{\#}_i(t) = argmin_{i=1}^{n}(f(\boldsymbol{x}^{\#}_i(t-1)), f(\boldsymbol{x}_i(t))$;
11.       For $j = 1$ to $d$
12.           If $Nohope < 10$ then
13.               Update the $j$-th dimension value of $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$
14.               according to Eqs.(4),(5)
15.           else
16.               Update the $j$-th dimension value of $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$
17.               according to Eqs.(7),(6).
18.       Next $j$
19.   Next $i$
20. End While.
```

$$\hat{v} = \begin{cases} v_{ij} & \text{if } |v_{ij}| \geq v_c \\ u(-1,1)v_{max}/\rho & \text{if } |v_{ij}| < v_c \end{cases} \tag{7}$$

Our algorithm scheme is summarized as Algorithm 1. The performance of the algorithm is directly correlated to two parameter values, v_c and ρ. A large v_c shortens the oscillation period, and it provides a great probability for the particles to leap over local minima using the same number of iterations. But a large v_c compels the particles in the quick "flying" state, which leads them not to search the solution and forcing them not to refine the search. The value of ρ changes directly the variable search neighborhoods for the particles. It is to be noted that the algorithm is different from the multi-start technique and the turbulence strategy [11]. We also implemented the Multi-Start PSO (MSPSO) and Velocity Turbulent PSO (VTPSO) to compare their performances.

For applying PSO successfully for the FJSP problem, we setup a search space of O dimension for a $(m - Machines, n - Jobs, O - Operations)$ FJSP problem. Each dimension was limited to $[1, m+1)$. Each dimension of the particle's position maps one operation, and the value of the position indicates the machine number to which this task is assigned to during the course of PSO. The particle's position may appear real values such as 1.4, etc. We usually round off the real optimum value to its nearest integer number.

4 Experiment Settings and Results

To illustrate the effectiveness and performance of the proposed algorithm, three representative instances based on practical data have been selected. Three

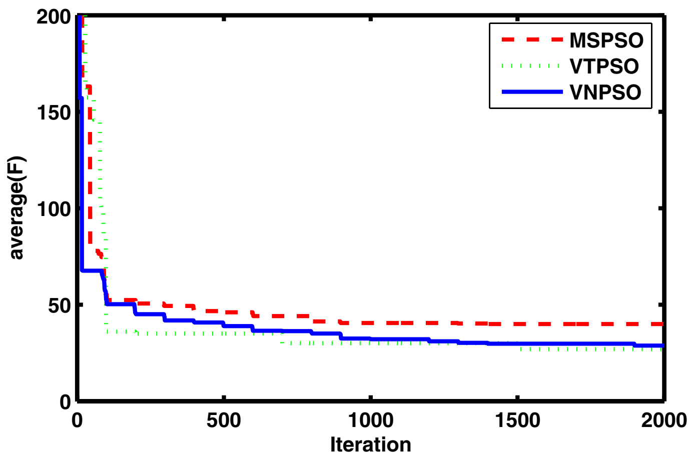

Fig. 2. The performance of the algoriths for $(J8, O27, M8)$ FJSP

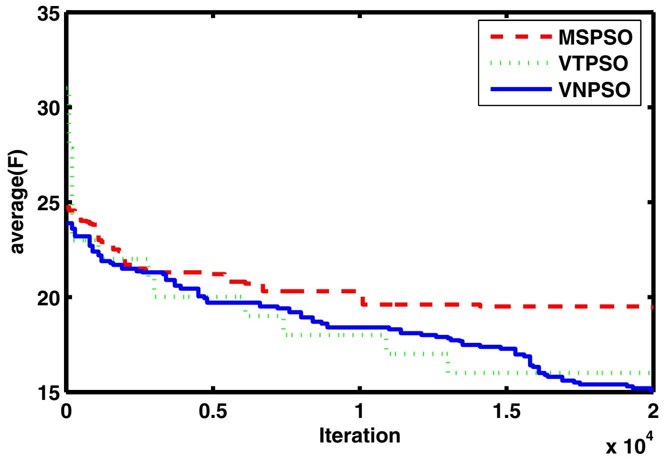

Fig. 3. The performance of the algoriths for $(J10, O30, M10)$ FJSP

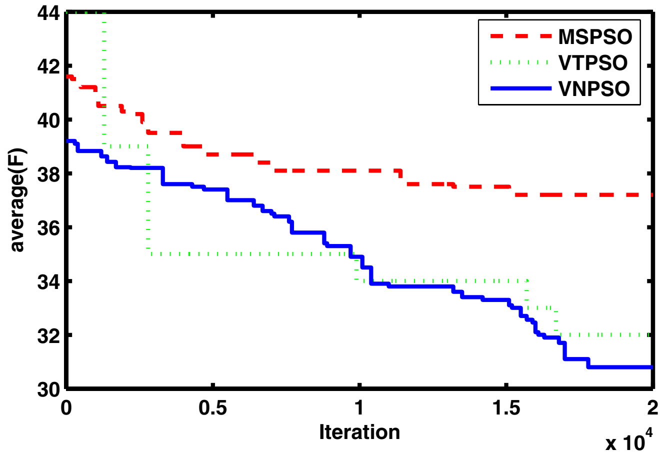

Fig. 4. The performance of the algoriths for $(J15, O56, M10)$ FJSP

problem instances ($(J8, O27, M8)$, $(J10, O30, M10)$ and $(J15, O56, M10)$ are taken from Kacem et al. [2,3]. In our experiments, the algorithms used for comparison were MSPSO (Multi-start PSO), VTPSO (Velocity Turbulent PSO) and VNPSO (Variable Neighborhood PSO). The parameters c_1 and c_2 were set to 1.49 for all the PSO algorithms. Inertia weight w was decreased linearly from 0.9 to 0.1. In VTPSO and VNPSO, ρ and v_c were set to 2 and 1e-7 before 15,000 iterations, while they were set to 5 and 1e-10 after 15,000 iterations. The swarm size in all the algorithms were set to 20. The average fitness values of the best solutions throughout the optimization run were recorded. The averages (F) and

Table 1. Comparing the results for FJSPs

Instance	*Items*	MSPSO	VTPSO	VNPSO
	Best makespan	30	26	24
	Best flowtime	168	155	152
$(8, 27, 8)$	average	39.9087	28.3853	28.8000
	std	±5.7140	±2.3146	±3.8239
	time	184.0620	181.2500	181.2970
	Best makespan	19	13	11
	Best flowtime	96	92	75
$(10, 30, 10)$	average	19.4612	15.2000	15.0000
	std	±1.8096	±1.3166	±1.8257
	time	1.7145e+003	1.5891e+003	1.5908e+003
	Best makespan	36	30	29
	Best flowtime	231	241	220
$(15, 56, 10)$	average	37.2000	31.9000	30.8000
	std	±1.0328	±1.2867	±1.7512
	time	2.0497e+003	12.0816e+003	2.0703e+003

the standard deviations (std) were calculated from the 10 different trials. The standard deviation indicates the differences in the results during the 10 different trials. Usually another emphasis will be to generate the schedules at a minimal amount of time. So the completion time for 10 trials were used as one of the criteria to improve their performance. Figs. 2, 3 and 4 illustrate the performance for the three algorithms during the search processes for the three FJSPs. Empirical results are illustrated in Table 1. In general, VNPSO performs better than the other two approaches, although its computational time is worse than VTPSO for the low dimension problem, $(J8, O27, M8)$. VNPSO could be an ideal approach for solving the large scale problems when other algorithms failed to give a better solution.

5 Conclusions

In this paper, we introduce a hybrid metaheuristic, the Variable Neighborhood Particle Swarm Optimization (VNPSO), consisting of a combination of the Variable Neighborhood Search (VNS) and Particle Swarm Optimization(PSO), and considered its application for solving the multi-objective Flexible Job-shop Scheduling Problems (FJSP). The details of implementation for the multi-objective FJSP are provided and its performance was compared using computational experiments. The empirical results have shown that the proposed algorithm is an available and effective approach for the multi-objective FJSP, especially for large scale problems.

Acknowledgments. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2006-353-B00059).

References

1. Mastrolilli M. and Gambardella L. M.: Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling, (2002) 3(1):3-20.
2. Kacem I., Hammadi S., and Borne P.: Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man and Cybernetics, (2002) 32(1):1-13.
3. Kacem I., Hammadi S. and Borne P.: Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Mathematics and Computers in Simulation, (2002) 60:245-276.
4. Ong Z. X., Tay J. C. and Kwoh C. K.: Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules. in: Jacob C. et al. (eds.), ICARIS2005, LNCS3627, 2005, 442-455.
5. Xia W. and Wu Z.: An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers and Industrial Engineering, (2005) 48:409-425.
6. Kennedy J. and Eberhart R.: Swarm Intelligence. Morgan Kaufmann (2001).
7. Parsopoulos K. E. and Vrahatis M. N.: Recent Approaches to Global Optimization Problems through Particle Swarm Optimization, Natural Computing, (2002) 1: 235-306.
8. Cristian T. I.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters, (2003) 85(6):317-325.
9. Hansen P. and Mladenović N.: Variable neighbourhood search: Principles and applications. European Journal of Operations Research, (2001) 130:449-467.
10. Hansen P. and Mladenović N.: Variable neighbourhood search. In: Glover F. W. and Kochenberger G. A. (eds.), Handbook of Metaheuristics, Dordrecht, Kluwer Academic Publishers (2003).
11. Liu H. and Abraham A.: Fuzzy Adaptive Turbulent Particle Swarm Optimization, in: Proceedings of the Fifth International conference on Hybrid Intelligent Systems, (2005) 445-450.