

# Variable Neighborhood Particle Swarm Optimization Algorithm

Ajith Abraham  
School of Computer Science  
and Engineering  
Chung-Ang University  
Seoul, 156-756 Korea  
ajith.abraham@ieee.org

Hongbo Liu  
Department of Computer  
Science  
Dalian University of  
Technology  
Dalian, 116023 China  
lhb@dlut.edu.cn

Tae-Gyu Chang  
School of Electrical and  
Engineering  
Chung-Ang University  
Seoul, 156-756 Korea  
tgchang@cau.ac.kr

## ABSTRACT

In this paper, we introduce a hybrid metaheuristic, the Variable Neighborhood Particle Swarm Optimization (VNPSO) algorithm, consisting of a combination of the Variable Neighborhood Search (VNS) and Particle Swarm Optimization (PSO). The proposed VNPSO algorithm is used for solving the multi-objective Flexible Job-shop Scheduling Problems (FJSP). Flexible job-shop scheduling is very important in both fields of production management and combinatorial optimization. However, it is quite difficult to achieve an optimal solution with traditional optimization approaches owing to the high computational complexity. The details of implementation for the multi-objective FJSP are provided 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.

## Categories and Subject Descriptors

Computing Methodologies [I.2 Artificial Intelligence]: I.2.8 Problem Solving, Control Methods, and Search

## General Terms

Algorithms, Design, Performance

## Keywords

Variable Neighborhood Search, Particle Swarm Optimization, Flexible Job-shop Scheduling Problems, Multi-objective Optimization

## 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 [1]. 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. Some different approaches have been proposed to solve this problem. Brandimarte [2], Hurink and Jurisch [3] developed various tabu search algorithms for FJSP. Mastrolilli and Gambardella [4] proposed some neighborhood functions for metaheuristics. Kacem, et al. [5, 6] studied on modeling genetic algorithms for FJSP. Recently, Ong et al. [7] applied the clonal selection principle of the human immune system to solve FJSP with recirculation. By hybridizing particle swarm optimization and simulated annealing, Xia and Wu [8] 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 [9, 10, 11]. The PSO 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 metaheuristics, 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '06 Seattle, WA, USA

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

## 2. PROBLEM FORMULATION

The classical FJSP considers in general the assignment of a set of machines  $M = \{M_1, \dots, M_m\}$  to a set of jobs  $J = \{J_1, \dots, J_n\}$ , each of which consists of a set of operations  $J_j = \{O_{j,1}, \dots, O_{j,p}\}$ . There are several constraints on 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, we define  $C_{i,j,k}$  ( $i \in \{1, 2, \dots, m\}$ ,  $j \in \{1, 2, \dots, n\}$ ,  $k \in \{1, 2, \dots, p\}$ ) 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 minimizes 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:

$$F = w_1 \min\{F_1\} + w_2 \min\{F_2\} \quad (1)$$

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 [15] was used in the paper.

## 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  $d$ -dimension 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  $\vec{x}_i$  ( $i$  is the index of the particle), and a velocity represented by a velocity-vector  $\vec{v}_i$ . Each particle remembers its own best position so far in a vector  $\vec{x}_i^{\#}$ , and its  $j$ -th dimensional value is  $x_{ij}^{\#}$ . The best position-vector among the swarm so far is then stored in a vector  $\vec{x}^*$ , 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.(2). The new position is then determined by the sum of the previous position and the new velocity by Eq.(3).

$$v_{ij}(t) = wv_{ij}(t-1) + c_1r_1(x_j^{\#}(t-1) - x_{ij}(t-1)) + c_2r_2(x_j^*(t-1) - x_{ij}(t-1)) \quad (2)$$

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \quad (3)$$

In the PSO model, the particle searches the solutions in the problem space within a range  $[-s, s]$  (If the range is not symmetrical, it can be translated to the corresponding symmetrical range.) In order to guide the particles effectively in the search space, the maximum moving distance during one iteration is clamped in between the maximum velocity  $[-x_{max}, x_{max}]$  given in Eq.(4), and similarly for its moving range given in Eq.(5):

$$x_{i,j} = \text{sign}(x_{i,j}) \min(|x_{i,j}|, x_{max}) \quad (4)$$

$$v_{i,j} = \text{sign}(v_{i,j}) \min(|v_{i,j}|, v_{max}) \quad (5)$$

The value of  $v_{max}$  is  $\rho \times s$ , with  $0.1 \leq \rho \leq 1.0$  and is usually chosen to be  $s$ , i.e.  $\rho = 1$ .

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. Eberhart and Kennedy called the two basic methods as "gbest model" and "lbest model" [9]. In the lbest model, particles have information only of their own and their nearest array neighbors' best, rather than that of the entire group. Unfortunately there is a large computational cost to explore the neighborhood relation in each iteration. In the gbest model, the trajectory for each particle's search is influenced by the best point found by any member of the entire population. The best particle acts as an attractor, pulling all the particles towards it. Some previous studies has been shown that the trajectories of the particles oscillate in different sinusoidal waves and converge quickly in the "gbest model" algorithm [13, 14]. 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 whole swarm. The "gbest model" 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 [16, 17]. 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):

$$v_{ij}(t) = w\hat{v} + c_1r_1(x_j^{\#}(t-1) - x_{ij}(t-1)) + c_2r_2(x_j^*(t-1) - x_{ij}(t-1)) \quad (6)$$

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

The proposed algorithm scheme is summarized as Algorithm 1. The performance of the algorithm is directly correlated to two parameter values,  $v_c$  and  $\rho$ . 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  $\rho$  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 [18]. We also implemented the Multi-Start PSO (MSPSO) and Velocity Turbulent PSO (VTPSO) to compare their performances.

---

**Algorithm 1** VNPSO

---

01. Initialize the size of the particle swarm  $n$ , and other
  02. parameters; Initialize the positions and the velocities
  03. for all the particles randomly. Set the flag of iterations
  04. without improvement  $Nohope = 0$ .
  05. While (the end criterion is not met) do
  06.  $t = t + 1$ ;
  07. Calculate the fitness value of each particle;
  08.  $\vec{x}^* = \operatorname{argmin}_{i=1}^n (f(\vec{x}^*(t-1)), f(\vec{x}_i(t)),$
  09.  $f(\vec{x}_2(t)), \dots, f(\vec{x}_i(t)), \dots, f(\vec{x}_n(t)))$ ;
  10. If  $\vec{x}^*$  is improved then  $Nohope = 0$ ,
  11. else  $Nohope = Nohope + 1$ .
  12. For  $i = 1$  to  $n$
  13.  $\vec{x}_i^\#(t) = \operatorname{argmin}_{i=1}^n (f(\vec{x}_i^\#(t-1)), f(\vec{x}_i(t))$ ;
  14. For  $j = 1$  to  $Dimension$
  15. If  $Nohope < 10$  then
  16. Update the  $j$ -th dimension value of  $\vec{x}_i$  and  $\vec{v}_i$
  17. according to Eqs.(2),(5),(3),(4)
  18. else
  19. Update the  $j$ -th dimension value of  $\vec{x}_i$  and  $\vec{v}_i$
  20. according to Eqs.(7),(6).
  21. Next  $j$
  22. Next  $i$
  23. End While.
- 

For applying PSO successfully for the FJSP problem, one of the key issues is how to map the problem solution to the PSO particle space, which directly affects its feasibility and performance. We setup a search space of  $O$  dimension for an ( $m - Machines, n - Jobs, O - Operations$ ) FJSP problem. Each dimension was limited to  $[1, m]$ . For example, consider a little scale ( $3 - Machines, 3 - Jobs, 7 - Operations$ ) FJSP, Fig. 1 shows a mapping between a one possible assignment instance to a particle position coordinates in the PSO domain. 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. So the value of a particle’s position should be integer. But after updating the velocity and position of the particles, the particle’s position may appear real values such as 1.4, etc. It is meaningless for the assignment. Therefore, in the algorithm we usually round off the real optimum value to its nearest integer number. By this way, we

Operation	Job <sub>1</sub>		Job <sub>2</sub>			Job <sub>3</sub>	
	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$
Particle Position	2	1.4	3.7	1.8	2.6	3	1
Processing Machine	$M_2$	$M_1$	$M_3$	$M_1$	$M_2$	$M_3$	$M_1$

Figure 1: The Mapping between particle and FJSP.

convert a continuous optimization problem to a scheduling problem. The particle’s position is a series of priority levels of assigned machines according to the order of operations. The sequence of the operations will be not changed during the iteration.

## 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 problem instances (problem ( $J8, O27, M8$ ), problem ( $J10, O30, M10$ ) and problem ( $J15, O56, M10$ ) are all taken from Kacem et al. [5, 6]. In our experiments, the algorithms used for comparison were MSPSO, VTPSO and VNPSO. 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,  $\rho$  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.  $v_{max}$  was set to  $s/2$ , the range of the position domain. 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 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 curves for the three algorithms during the search processes for the three FJSPs. Other results are showed in Table 1. In general, VNPSO performs better then the other two approaches, although its time item 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, 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.

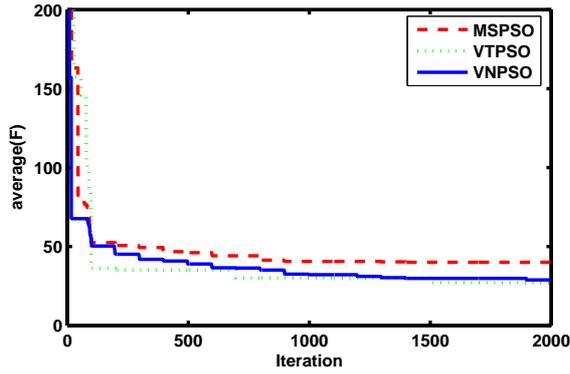


Figure 2: The performance of the algorithms for (J8, O27, M8) FJSP.

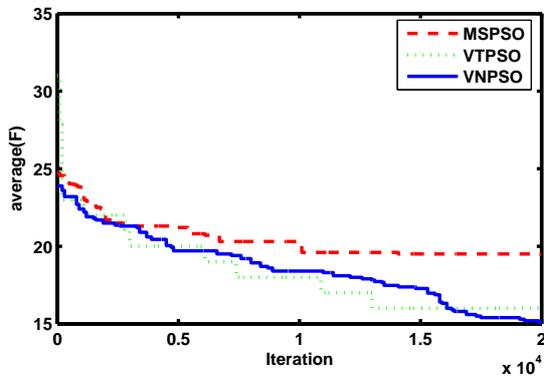


Figure 3: The performance of the algorithms for (J10, O30, M10) FJSP.

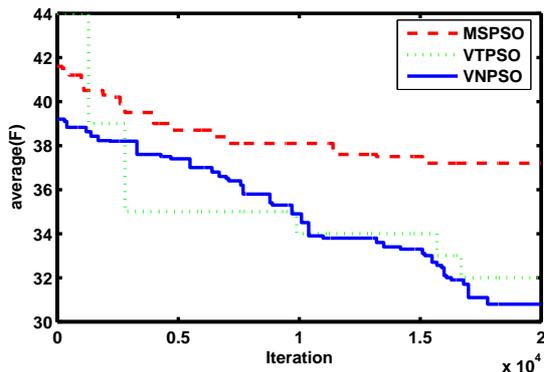


Figure 4: The performance of the algorithms for (J15, O56, M10) FJSP.

Table 1: Comparing the results for FJSPs.

FJSP	Items	MSPSO	VTPSO	VNPSO
(8, 27, 8)	makespan	30	26	24
	flowtime	168	155	152
	average	39.9087	28.3853	28.8000
	std	$\pm 5.7140$	$\pm 2.3146$	$\pm 3.8239$
(10, 30, 10)	time	184.0620	181.2500	181.2970
	makespan	19	13	11
	flowtime	96	92	75
	average	19.4612	15.2000	15.0000
(15, 56, 10)	std	$\pm 1.8096$	$\pm 1.3166$	$\pm 1.8257$
	time	1714.5	1589.1	1590.8
	makespan	36	30	29
	flowtime	231	241	220
(15, 56, 10)	average	37.2000	31.9000	30.8000
	std	$\pm 1.0328$	$\pm 1.2867$	$\pm 1.7512$
	time	2049.7	12081.6	2070.3

## 6. ACKNOWLEDGMENTS

The first author acknowledges the support received from the International Joint Research Grant of the IITA (Institute of Information Technology Assessment) foreign professor invitation program of the Ministry of Information and Communication, South Korea. The second author is supported by NSFC(60373095), MOE(KP0302).

## 7. REFERENCES

- [1] Bruker, P. and Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing, (1990) 45:369-375
- [2] Brandimarte, P.: Routing and Scheduling in a Flexible Job-Shop by Tabu Search. Annals of Operations Research, (1993) 2:158-183
- [3] Hurink, E., Jurisch, B. and Thole, M.: Tabu search for the job shop scheduling problem with multi-purpose machines. Operations Research Spektrum, (1994) 15:205-215
- [4] Mastrolilli, M. and Gambardella, L. M.: Effective neighborhood functions for the flexible job shop problem. Journal of Scheduling, (2002) 3(1):3-20
- [5] 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
- [6] 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
- [7] Ong Z. X., Tay J. C. and Kwok C. K.: Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules. in: Jacob, C. et al. (eds.), ICARIS2005, LNCS3627, 2005, 442-455
- [8] 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
- [9] Kennedy, J. and Eberhart, R.: Swarm Intelligence. Morgan Kaufmann (2001)

- [10] Clerc, M.: Particle Swarm Optimization. ISTE Publishing Company, London (2006)
- [11] Abraham A., Guo H., Liu H.: Swarm Intelligence: Foundations, Perspectives and Applications. in: Nedjah, N., Mourelle, L. (eds.), Swarm Intelligent Systems, Nova Publishers, USA (2006)
- [12] Liu, H., Li, B., Wang, X., Ji, Y., Tang, Y.: Survival density particle swarm optimization for neural network training. in: Yin, F., Wang, J., and Guo, C. (eds.), ISNN2004, LNCS3173, Springer-Verlag, (2004) 332-337
- [13] Clerc, M. and Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, (2002) 6(1):58-73
- [14] Cristian, T. I.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters, (2003) 85(6):317-325
- [15] Parsopoulos, K. E. and Vrahatis, M. N.: Recent Approaches to Global Optimization Problems through Particle Swarm Optimization, Natural Computing, (2002) 1:235-306
- [16] Hansen, P. and Mladenović, N.: Variable neighbourhood search: Principles and applications. European Journal of Operations Research, (2001) 130:449-467
- [17] 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)
- [18] 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