Int. J. Simul. Multisci. Des. Optim. 2014, **5**, A20
© T. Radha Ramanan et al., Published by EDP Sciences, 2014
DOI: 10.1051/smdo/2013006

IJSMDO

Available online at:
www.ijsmdo.org

ARTICLE OPEN ACCESS

A particle swarm optimization approach for permutation flow shop scheduling problem

T. Radha Ramanan[1,*], Muhammed Iqbal[2] and K. Umarali[3]

[1] Assistant Professor, Department of Mechanical Engineering, National Institute of Technology Calicut, Kozhikode 673601, Kerala, India
[2] Senior Lecturer, Department of Mechanical Engineering, MEA Engineering College, Perinthalmanna 679325, Kerala, India
[3] Research Scholar, Department of Mechanical Engineering, National Institute of Technology Calicut, Kozhikode 673601, Kerala, India

Received 3 October 2011 / Accepted 5 November 2013 / Published online 10 March 2014

Abstract – Flow shop scheduling problem (FSSP) is a combinatorial optimization problem. This work, with the objective of optimizing the makespan of an FSSP uses a particle swarm optimization (PSO) approach. The problems are tested on Taillard's benchmark problems. The results of Nawaz, Encore and Ham (NEH) heuristic are initialized to the PSO to direct the search into a quality space. Variable neighbourhood search (VNS) is employed to overcome the early convergence of the PSO and helps in global search. The results are compared with standalone PSO, traditional heuristics and the Taillard's upper bounds. Five problem set are taken from Taillard's benchmark problems and are solved for various problem sizes. Thus a total of 35 problems are solved. The experimental results show that the solution quality of FSSP can be improved if the search is directed in a quality space based on the proposed PSO approach (PSO-NEH-VNS).

Key words: Makespan, Flow shop scheduling, Particle swarm optimization, Variable neighbourhood search.

1 Introduction

Scheduling is the allocation of resources (e.g., machines) to tasks (e.g., jobs) in order to ensure the completion these tasks in a reasonable amount of time. In a flow shop, the technological constraints demand that the jobs pass between the machines in the same order; i.e., If J_1 must be processed on machine M_k before machine M_i then the same is true for all jobs. A permutation schedule is one on which each machine processes the jobs in the same order; i.e., if on machine M_1 job J_i is processed before J_k, then the same is true for all machines [1]. The objective is to find a sequence of jobs that minimizes the makespan. i.e., time required to complete all the jobs. This problem is found to be NP-hard [2].

The complexity of the flow shop scheduling problem renders exact solution methods impractical for instances of more than a few jobs and/or machines. This is the main reason for the various heuristic methods proposed in the literature.

The heuristics can be divided into either constructive heuristics or improvement heuristics, the former are heuristics that build a feasible schedule from scratch and the latter are heuristics that try to improve a previously generated schedule by normally applying some form of specific problem knowledge.

The objective of this work is to find a permutation of jobs that minimizes the makespan. The present study explores the further improvement in the solution quality, if the population in the improvement phase is provided with better initial solutions. The work describes the percentage improvement that the proposed method is able to obtain when compared with some of existing heuristics due to the improvisation in the search space of PSO. This approach can be considered as the significant contribution of this study. This work aims at directing the search space of PSO into certain quality. Then, VNS is employed to make the solution jump out of the local optima. This approach is termed as PSO-NEH-VNS heuristic. The solution thus obtained is compared with the results obtained using the PSO and the upper bounds of Taillard's [3] benchmark problems. The approach proves that results approach the optimum faster when the search space is nearer to the optimum.

The rest of the paper is organized as follows. Section 2 discusses the literature survey. The methodology is presented in Section 3. The results and discussions are made in Section 4. Conclusions and future scope is given in the Section 5.

2 Literature survey

2.1 Heuristics

Many constructive heuristic approaches have been proposed in the literature. Johnson's algorithm is the earliest known heuristic for the permutation flow shop problems.

*e-mail: radha_ramanan@nitc.ac.in

The two machine flow shop problem with the objective of minimizing makespan is also known as Johnson's [4] problems. An optimal sequence is found by following a heuristics of finding the minimum machining time and allotting the job to the machine in a preferential order is adopted.

Literature is replete with traditional heuristics [5–7] and nontraditional heuristics (Tabu search [8–10]); Simulated annealing [11–14]. Genetic algorithm [15–17], Ant colony optimization [18] to name a few.

Contrary to constructive heuristics, improvement heuristics start from an already built schedule and try to improve it by some given procedure. Many improvement heuristic approaches also have been proposed in the research [19–21].

2.2 Particle swarm optimization

Duda [22] evaluates a particle swarm optimization with variable neighbourhood search against a standalone VNS algorithm. The results of the conducted experiments indicate that hybridization of a meta-heuristic with a local search algorithms may not always bring additional performance benefit. Ali and Fawaz [23] address the flow shop scheduling problem with respect a due date-based performance measure, i.e., maximum lateness. Lian et al. [24] propose a novel particle swarm optimization algorithm to minimize the makespan of a flow shop.

Kuo et al. [25] propose a flow shop scheduling algorithm based on hybrid particle swarm optimization model. Liu et al. [26] propose a hybrid particle swarm optimization for flow shop scheduling with stochastic processing time. Chen [27] proposes an improved particle swarm optimization technique for flow shop scheduling. Sha and Lin [28] propose a PSO model for multi-objective flow shop scheduling. Zhang et al. [29] provide a hybrid alternate two phases particle swarm algorithm for flow shop scheduling problems. Deb [30] discusses about the evolutionary approaches used in solving the multi-objective optimization problems.

3 Methodology

The objective of the work is to minimize the makespan of the flow shop scheduling problems. This is attempted by developing a PSO approach. Five problem sets are taken from "Taillard" [2] benchmark problem and are solved for various sizes. A total of 35 problems are solved.

The results of PSO-NEH-VNS is compared with the upper bound (UB) of Taillard's benchmark and also compared with the results of CDS, NEH, PSO, PSO-NEH, and PSO-VNS.

The Taillards benchmark problem is first solved using NEH heuristics and a sequence is obtained. The PSO requires a population of sequences to be initialized. In this problem, the PSO is initialized with fifteen randomly generated populations. The sequence that is obtained by NEH heuristic is initialized as one of the initial population. This PSO is referred to as PSO initialized with NEH (PSO-NEH). This initialization is made with the hope that the search space is nearer to optimal solution. The results obtained by the PSO-NEH are further improved with the VNS. The VNS prevents the early convergence of PSO and jump out of local optima. This approach is termed as PSO-NEH-VNS. Hansen et al. [31] state that Variable neighbourhood search (VNS) is a meta-heuristic, or a framework for building heuristics, based upon systematic changes of neighbourhoods both in descent phase, to find a local minimum, and in perturbation phase to emerge from the corresponding valley.

3.1 PSO algorithm for FSSP

Particle swarm optimization (PSO), inspired by the motion of a flock of birds searching for food, was developed by Kennedy and Eberhart [32] for optimization of continuous nonlinear functions. To find the optimal solution each bird called a particle adjusts its searching direction according to two factors, its own best previous experience (pbest) and the best experience of all other members (gbest). The system is initialized with a population of random solutions and searches for optima by updating generations. In PSO potential solutions called particles, fly through the problem space by following the current optimum particles. PSO simulates the behaviors of bird flocking. In PSO each single solution is a "bird" in the search space. We call it "particles". All the particles have fitness value and velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. Each particle is updated by following two "best" values. The first value is the best solution it has achieved so far known as "pbest". Second value is the best value obtained so far by any particle is the population called "gbest". After finding the two best values the particle updates its velocity and position with the following equations

$$V[] = w \times v[] - c1 \times \text{rand}() \times (\text{pbest}[] - \text{present}[]) - c2 \times \text{rand}() \times (\text{gbest}[] - \text{present}[]) \quad (1)$$

where, $V[]$ = particle velocity, present [] = current particle solution, pbest [] = best solution among each particle, gbest [] = best among defined as before, w = inertia weights 0.8, $c1$ and $c2$ are learning factors. Usually $c1 = c2 = 2$.

This equation specifies that the velocity of a particle is determined by the previous velocity of the particle. Each particle moves to a new position according to the following equation.

$$\text{present}[] = \text{present}[] - v[]. \quad (2)$$

The out line of a PSO algorithm is as follows:

Step 1: (Initialization): randomly generate initial particles,
Step 2: (Fitness): evaluate the fitness of each particle in the population,
Step 3: (Update): calculate the velocity of each particle by the equation,
Step 4: (Construction): for each particle, moves to the next position according to the equation (2),
Step 5: (Termination): stop the algorithm if a specified stopping criterion is satisfied; return to Step 2 otherwise.

Table 1. Makespan of 20 jobs, 5, 10 and 20 machines.

SL No.	Problem size	Makespan						
		CDS	NEH	PSO	PSO NEH	PSO VNS	PSONEHVNS	UB*
1	20 × 5	1390	1286	1297	1286	1320	1278	1278
2	20 × 5	1424	1365	1373	1365	1368	1365	1359
3	20 × 5	1249	1159	1125	1132	1164	1100	1081
4	20 × 5	1418	1325	1364	1325	1349	1309	1293
5	20 × 5	1323	1305	1250	1263	1277	1250	1236
6	20 × 10	1757	1680	1660	1635	1673	1586	1582
7	20 × 10	1854	1729	1727	1722	1747	1684	1659
8	20 × 10	1645	1557	1517	1556	1588	1521	1496
9	20 × 10	1547	1439	1434	1419	1514	1399	1378
10	20 × 10	1558	1502	1492	1502	1538	1450	1419
11	20 × 20	2559	2410	2365	2386	2446	2330	2297
12	20 × 20	2285	2150	2177	2148	2195	2111	2100
13	20 × 20	2565	2411	2387	2399	2509	2342	2326
14	20 × 20	2434	2262	2304	2251	2410	2248	2223
15	20 × 20	2506	2397	2358	2388	2471	2302	2291

* Refers to upper bound of Taillard's benchmark problems.

Table 2. Makespan of 50 jobs, 5, 10 and 20 machines.

SL No.	Problem size	Makespan					
		CDS	NEH	PSO	PSO-NEH	PSO-NEH-VNS	UB*
1	50 × 5	2816	2733	2729	2729	2724	2724
2	50 × 5	3032	2843	2906	2843	2843	2834
3	50 × 5	2703	2640	2676	2621	2631	2621
4	50 × 5	2884	2782	2824	2782	2762	2751
5	50 × 5	3038	2868	2873	2864	2864	2863
6	50 × 10	3421	3135	3240	3134	3059	3025
7	50 × 10	3246	3032	3093	3025	2934	2892
8	50 × 10	3280	2986	3139	2965	2932	2864
9	50 × 10	3393	3198	3236	3172	3115	3064
10	50 × 10	3375	3160	3186	3115	3052	2986
11	50 × 20	4328	4082	4192	4061	4010	3875
12	50 × 20	4216	3921	4067	3918	3864	3715
13	50 × 20	4189	3927	3981	3907	3808	3668
14	50 × 20	4280	3969	4067	3957	3844	3752
15	50 × 20	4122	3835	3999	3832	3815	3635

* Refers to upper bound of Taillard's benchmark problems.

4 Results and discussions

The objective of this work is to build a PSO model to solve the flow shop scheduling problems for minimizing the makespan; Five problem set are taken from Taillard benchmark problem and are solved. The problems are 20 jobs, 5, 10 and 20 machines; 50 jobs, 5, 10 and and 20 machines and 100 jobs, 5 machines.

Table 1 shows the results of the five instances of 20 × 5, 20 × 10 and 20 × 20 problems that are solved using the various methods.

The proposed PSO-NEH-VNS clearly gives better results than other methods such as CDS, NEH, PSO, PSO-NEH, and PSO-VNS. This is possible because the search space of PSO-NEH-VNS is in the area of superior solution. The search space of PSO is initialized with the results of NEH heuristic; this ensures that the solution space is of certain quality. The VNS helps in the solution of PSO jumping out of the local optima. Thus, the interchange of VNS has helped the PSO to come out of the pre convergence.

In smaller size problems the average difference in makespan between Upper bound and PSO-NEH-VNS is less and this difference increases with increasing number of machines. Average difference in makespan between PSO-NEH-VNS and UB for 20 Job 5 machines is 12.6, 20 Job 10 machines is 21 and 20 Job 20 machines is 32.

Table 2 shows the makespan of 50 jobs and 5, 10 and 20 machines problem size. The proposed PSO-NEH-VNS clearly give better results than other methods. The makespan obtained by PSO-NEH-VNS is nearly equal (within 5%) to the upper bound of the Taillard's benchmark problems and in some cases it is equal to the upper bound. (For 50 × 5, SL No. 1). Similar

Table 3. Makespan of 100 jobs, 5 machine.

SL No.	Problem size	Makespan					
		CDS	NEH	PSO	PSO NEH	PSO NEH VNS	UB*
1	100 × 5	5592	5519	5527	5519	5493	5493
2	100 × 5	5563	5348	5327	5300	5290	5268
3	100 × 5	5493	5219	5253	5213	5213	5175
4	100 × 5	5273	5023	5078	5023	5023	5014
5	100 × 5	5461	5266	5323	5266	5253	5250

* Refers to Upper bound of Taillard's benchmark problems.

Table 4. Computational time.

SL No.	Problem size	CPU time in seconds					
		CDS	NEH	PSO	PSO NEH	PSO VNS	PSO NEH VNS
1	20 × 5	0.0156	0.0156	1.29	1.32	7.96	7.96
2	20 × 10	0.0156	0.0156	1.59	1.50	35.82	36.68
3	20 × 20	0.0156	0.0313	1.95	1.92	206.92	197.84
4	50 × 5	0.0156	0.0156	1.96	1.96	13.07	13.06
5	50 × 10	0.0156	0.0156	2.26	2.26	60.62	60.45
6	50 × 20	0.0156	0.0781	2.95	2.95	332.14	330.84
7	100 × 5	0.0313	0.0156	3.09	3.07	22.50	22.10

to the previous problem set, the average difference between upper bound and PSO-NEH-VNS increases with increasing number of machines. (For 50 Job 5 machines is 6.2; 50 job 10 machines is 52.2; 50 Job 20 machines is 139.2.)

Makespan values of 100 jobs and 5, 10 and 20 machines are shown in Table 3. For the 100 jobs problem as well the proposed PSO-NEH-VNS method gives better results when compared with other methods. In certain instances (for 100 × 5, SL No. 1), the PSO-NEH-VNS is able to give the makespan values equivalent to the upper bound of the Taillard's benchmark problems.

Consistently, the PSO-NEH-VNS is giving better results than the other methods taken for comparison. Hence, it can be stated that with a better search space provided for the PSO, the PSO with the help of VNS is able to avoid early convergence. It can also be inferred that the search is able to converge faster because of the initial quality of the search space.

In order to compare computational time required, various approaches are coded in matlab and run on a 3 G B RAM, 2.19 GHz, Dual Core system with Windows XP operating system. The CPU time for various size problems are shown in Table 4.

It can be observed that computational time for PSO NEH VNS approach is slightly lesser than PSO VNS for larger problem sizes.

Table 5 shows the percentage improvement of results of PSO VNS NEH for a 20 job problem size, when compared with other methods used in this work.

It can be observed that PSO do better than PSO-VNS. PSO when initialized with the results of NEH does better than both PSO and PSO-VNS algorithms. The VNS takes the PSO to early convergence. But when the same solution is initialized with quality solution space, obtained from the constructive heuristic (NEH), it is able to perform better.

Table 5. Improvement of makespan using PSO NEH VNS.

SL No.	Problem size	% Improvement of makespan over				
		CDS	NEH	PSO	PSO NEH	PSO VNS
1	20 × 5	8.08	0.00	0.85	0.00	2.64
2	20 × 5	4.32	0.00	0.58	0.00	0.21
3	20 × 5	13.54	5.36	2.27	2.90	5.81
4	20 × 5	8.32	1.22	4.20	1.22	3.05
5	20 × 5	5.84	4.40	0.00	1.04	2.16
6	20 × 10	10.78	5.92	4.66	3.08	5.48
7	20 × 10	10.09	2.67	2.55	2.25	3.74
8	20 × 10	8.15	2.36	0.26	2.30	4.40
9	20 × 10	10.57	2.85	2.50	1.42	8.22
10	20 × 10	7.44	3.58	2.89	3.58	6.06
11	20 × 20	9.26	2.90	0.98	1.87	4.44
12	20 × 20	87.73	1.36	2.64	1.27	3.48
13	20 × 20	8.82	2.29	1.27	1.78	6.44
14	20 × 20	8.27	0.62	2.49	0.13	7.20
15	20 × 20	7.73	3.05	1.37	2.66	6.23

Table 6. Comparison of results.

Problem size	Makespan		
	UB*	Kuo et al. HPSO	NEH-PSO-VNS
20 × 5	1278	1278	1278
20 × 10	1582	1587.3	1586
20 × 20	2297	2307	2310
50 × 5	2724	2724	2724

* Refers to Upper bound of Taillard's benchmark problems.

Table 6 shows the comparison of experimental results from literature The results show the PSO-NEH-VNS is providing results that are comparable to the Kuo et al. [25] HPSO and

in one instance (20×10) it is able to provide better result. This indicates the PSO-NEH-VNS model is worth exploring.

Though the results show the proposed heuristic performs better than certain PSO models, the percentage deviation from the optimum still persists. VNS has an inherent characteristic that is not taking the results to the optimum. But, the objective of the work is to prove that if the initial solution space is provided with quality solutions, the PSO can be used for obtaining faster results.

5 Conclusions

A particle swarm optimization algorithm named PSO-NEH-VNS is used to solve the FSSP with the objective of minimizing makespan. The research shows that the PSO initialized with a better solution space obtained from the constructive heuristic (NEH), has the ability to provide solutions that are better than some of the existing heuristics and takes the solution to early convergence of results. The results show that the percentage deviation of the makespan increases with the increasing machine, proving that the complexity of the flow shop problems depends on the increasing machine size. The results also show that the percentage deviation of the makespan with respect to the increasing job is not significant.

Acknowledgements. The authors wish to thank the anonymous reviewers for giving their valuable suggestions in improving the quality of the paper.

References

1. French S. 1982. Sequencing and scheduling: an introduction to the mathematics of the job-shop. Ellis Horwood Limited: Chichester, West Sussex, England, UK.
2. Garey MR, Johnson DS, Sethi R. 1976. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research, 1(2), 117–129.
3. Taillard E. 1993. Benchmarks for basic scheduling problems. European Journal of Operation Research, 64, 278–285.
4. Johnson SM. 1954. Optimal two-and three-stage production schedules with set up times include. Naval Research Logistics Quarterly, 1, 61–68.
5. Palmer DS. 1965. Sequencing jobs through a multistage process in the minimum total time – a quick method of obtaining near optimum. Operational Research Quarterly, 16, 101–107.
6. Campbell HG, Dudek RA, Smith ML. 1970. A heuristic algorithm for the n job m machine sequencing problem. Management Science, 16(10), B630–B637.
7. Navaz M, Enscore EE, Ham I. 1983. A heuristic algorithm for the m-machine n-job flow shop sequencing problem. Omega, 11, 91–95.
8. Widmer M, Hertz A. 1989. A new heuristic for the flow shop sequencing problem. European Journal of Operational Research, 41, 186–193.
9. Nowicki E, Smutnicki C. 1996. A fast Tabu search algorithm for the permutation flow shop problem. European Journal of Operations Research, 91, 160–175.
10. Moccellin JV, Santos MO. 2000. An adaptive hybrid metaheuristic for permutation flow shop scheduling. Control and Cybernetics, 29(3), 761–771.
11. Osman I, Potts C. 1989. Simulated annealing for permutation flow shop scheduling. OMEGA, 17(6), 551–557.
12. Chen CL, Vempati VS, Aljaber N. 1995. An application of genetic algorithms for flow shop problems. European Journal of Operational Research, 80, 389–396.
13. Low C. 2005. Simulated annealing heuristics for flow shop scheduling problems with unrelated parallel machines. Computers & Operations Research, 32(8), 2013–2025.
14. Ishibuchi H, Misaki S, Tanaka H. 1995. Modified simulated annealing algorithms for the flow shop sequencing problem. European Journal of Operational Research, 81(2), 388–398.
15. Reeves C. 1995. A genetic algorithm for flow shop sequencing. Computers and Operations Research, 22(1), 5–13.
16. Chen CL, Vempati VS, Aljaber N. 1995. An application of genetic algorithms for flow shop problems. European Journal of Operational Research, 80, 389–396.
17. Zobolas GI, Tarantilis CD, Ioannou G. 2009. Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. Computers & Operations Research, 36(4), 1249–1267.
18. Yagmahan B, Yenisey MM. 2008. Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering, 54, 411–420.
19. Rajendran C. 1995. Theory and methodology heuristics for scheduling in flow shop with multiple objectives. European Journal of Operational Research, 82, 540–555.
20. Koulamas C. 1998. A new constructive heuristic for the flow shop scheduling problem. European Journal of Operational Research Society, 105, 66–71.
21. Suliman S. 2000. A two-phase heuristic approach to the permutation flow-shop scheduling problem. International Journal of Production Economics, 64, 143–152.
22. Duda J. 2006. Local search and nature based metaheuristics: a case of flow shop scheduling problem, in Proceedings of the International multiconference on Computer Science and Information Technology, PIPS. p. 17–24, ISSN 1896-7094.
23. Allahverdi A, Fawaz S, Anzi A. 2006. A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. Computers & Operations Research, 33(4), 1056–1080.
24. Lian Z, Gu X, Jiao B. 2008. A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos, Solitons & Fractals, 35(5), 851–861.
25. Kuo IH, Horng SJ, Kao TW, Lin TL, Fan P. 2007. An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model, in IEA/AIE 2007, LNAI 4570, Okuno HG, Ali M, Editors. Springer-Verlag: Berlin, Heidelberg. p. 03–312.
26. Liu B, Wang L, Jin Y. 2005. Hybrid particle swarm optimization for flow shop scheduling with stochastic processing time, in CIS 2005, Part I, LNAI 3801, Hao Y, et al., Editor. Springer-Verlag: Berlin, Heidelberg. p. 630–637.
27. Chen Q. 2010. Flow shop scheduling using an improved PSO, vol. 2, in International Conference on Measuring Technology and Mechatronics Automation. p. 296–299.
28. Sha DY, Lin HH. 2009. A particle swarm optimization for multi-objective flow shop scheduling. International Journal of Advanced Manufacturing Technology, 45, 749–758.
29. Zhang C, Ning J, Ouyang D. 2010. A hybrid alternative two phases particle swarm optimization algorithm for flow shop scheduling problem. Computers & Industrial Engineering, 58(1), 1–11.

30. Deb K. 2007. Current trends in evolutionary multi-objective optimization. International Journal for Simulation and Multi-disciplinary Design Optimization, 1, 1–8.
31. Hansen P, Mladenovíc N. 1997. Variable neighborhood search for the p-median. Location Science, 5, 207–226.
32. Kennedy J, Eberhart RC. 1995. Particle swam optimization, in Roc. IEEE International Conference on Neural Networks (Path, Australia). IEEE Service Center: Piscataway, NJ. p. 1942–1948,

Cite this article as: Ramanan Radha T, Iqbal M & Umarali K: A particle swarm optimization approach for permutation flow shop scheduling problem. Int. J. Simul. Multisci. Des. Optim., 2014, **5**, A20.