Solving Permutation Flowshop Scheduling Problem Using Improved Differential Evolutionary Algorithm

Vanita G. Tonge, Pravin Kulkarni
M-tech IV Sem.(ComputerScience)
Rajiv Gandhi College of engg. Reasech and Technology, chandrapur,India

Information Technology
Rajiv Gandhi college of engg. Reasech and Technology, chandrapur,India

Abstract

now a day's permutation flowshop scheduling problem becomes interesting research area with different types of objective functions such as minimizing completion time also called as makespan time, total flow time, total weighted tardiness. The permutation flowshop scheduling problem is normally classified as a complex combinatorial optimization problem in which a set of n jobs have to be process on set of m machines in the same order. Many exact and heuristic algorithms have been proposed over the years Tabu Search, Partical Swarm Optimization, ant colony optimization, Genetic algorithm, Differential evolution algorithm for solving permutation flowshop scheduling problem. In this paper we implement improved differential evolutionary algorithm, differential evolution algorithm using classical NEH and iterated local search with enhanced swap operator and DE with position based crossover operator with the objectives of minimizing makespan. Comparing results of these proposed techniques improves the makespan value as well as the time complexity of an algorithm.

1. Introduction

Scheduling is a decision-making process for optimally allocating resources [1]. Efficient scheduling has become essential for manufacturing firms to survive in today's intensely competitive business environment [2, 3]. As one of the best known production scheduling problems, permutation flowshop sequencing problems (PFSPs) have long been a topic of interest for the researchers and practitioners in this field [4]. The goal of this scheduling problem is to find an optimal schedule for N jobs and M machines. Evolutionary algorithms (EAs), inspired by biological evolutionary mechanism in nature, have achieved great success on many numerical and combinatorial optimizations in diverse fields [24–26]. When implementing the EAs, users need to solve several points, for example, the appropriate encoding schemes, evolutionary operators, and the suitable parameter settings, to ensure the success of the algorithms.

The earlier EAs have some disadvantages, such as complex procedure, stagnation, and poor search ability. To overcome such disadvantages, on one hand, some researchers proposed other related methods (e.g., particle swarm optimization (PSO) [27, 28], differential evolution (DE). which have better global search ability. DE is proposed by Storn and Price[23]. Differential evolutionary algorithm is one of optimization tool. It is based on population similar to genetic algorithm but there is difference between genetic algorithm and differential evolutionary algorithm. Differential evolutionary algorithm is based upon mutation factor and genetic algorithm is based on crossover operator. It soon became a popular tool for solving global optimization problems because of several attractive features like having fewer control parameters, ease in programming, efficiency etc. A DE contains the following ingredients: parameter setting, representation of chromosome, initial population and population size. Evaluation of initial population, generation of donor vector called as mutation strategy, crossover operation, selection of fittest value and a termination criterion.

2. Literature Review

The flow-shop problem with make-span (Cmax) criterion can be denoted as either n/m /f /cmax or F//cmax , where both are related to a n- jobs and m-machines problem. This notation was firstly suggested by Conway et al. [7] and until now is handy. Pinedo [8] introduced the term Permutation Flow-shop Problem (PFSP) in which the processing sequence on the first machine is maintained throughout the remaining machines. Accordingly, the make-span criterion is denoted as /prmu/. Solution methods for flow shop scheduling range from heuristics developed by Palmer [9], Campbell et al. [10], and Dannenbring [11] to more complex techniques such as branch and bound [12], tabu search [13, 14, 15], genetic algorithms [16, 17] shifting bottleneck procedure [18], and ant colony algorithm [19].

Wenbo Liu proposes an improved differential evolution (DE) for the permutation flowshop scheduling problem with the total flowtime minimization, an NP-complete problem. To enhance the exploration ability of DE, a hybrid method of simulated annealing and stochastic variable neighborhood search are incorporated. To improve the search diversification of DE, a population restart method based path relinking is applied to replace non-promising solutions. Experimental results on benchmark instances show that the proposed DE algorithm is competitive to other metaheuristics proposed for the PFSP with total flowtime minimization in the literature.

Samia kouki1, Mohamed Jemni1, Talel Ladhari2, proposed a parallel algorithm for solving the permutation flow shop problem. The algorithm is a basic parallel distributed algorithm deployed in a grid of computer (Grid'5000). The objective of this work is minimizing the total makespan of the tasks. The algorithm uses the exact Branch and Bound method to find optimal solutions of the problem through the distribution of the tasks among the available processors. Computational results of our parallel algorithm using well known Taillard's benchmarks, showed encouraging results. In particular, we succeeded to solve two new instances neither to optimality which had never been resolved before neither in sequential nor in parallel.

Quan-Ke Pana, Rubén Ruiz, proposed algorithm based on iterated local search and iterated greedy algorithm. It is simple, easy to implement and gives improved results. V. L. Huang, S. Z. Zhao, R. Mallipeddi and P. N. Suganthan, proposed Multi-objective Self-adaptive Differential Evolution algorithm to solve numerical optimization problems with multiple conflicting objectives. Optimization problems with multiple conflicting objectives. The proposed approach learns suitable crossover parameter values and mutation strategies for each objective separately in a multi-objective optimization problem. Janez Brest, Member, IEEE, Viljem ˇ Zumer, Member, IEEE, and Mirjam Sepesy Mauˇcec, proposed a self-adaptive differential evolution Algorithm where more DE strategies are used and control parameters CR and F are self-adapted[21] A. K. Qin, V. L. Huang, and P. N. Suganthan, proposed a differential evolution technique with strategy adaptation. In which both trial vector generation strategies and their associated control parameter values are gradually self-adapted by learning from their previous experiences in generating promising solutions. Consequently, a more suitable generation strategy along with its parameter settings can be determined adaptively to match different phases of the search process/evolution. [22]

3. Differential Evolutionary Algorithm

DE is a population-based stochastic search technique as well, but it is simpler and it can be implemented more easily than other EAs. Besides that, DE [29, 30] is an effective and versatile function optimizer. There are only three crucial control parameters, that is, scaling factor F, crossover rate CR, and population size NP, which are fewer than other EAs'. The appropriate settings of the three control parameters ensure successful functioning of DE. In most existing DEs, the population size remains constant over the run. However, there are biological and experimental reasoning to expect that a variable population size would work better. In a natural environment, population sizes of species change and incline to steady state due to natural resources and ecological factors. Technically, the population size in a biological system is the most flexible element. And it can be calibrated more easily than recombination. Calibrating the population size during iterative process could be more rewarding than changing the operator parameters.

The crossover constant CR is used to determine if the newly generated individual is to be recombined. Storn and Price [23] suggested that a reasonable value for NP should be between 5D and 10D, and a good initial choice of F was 0.5. The effective range of F values was suggested between 0.4 and 1. The first reasonable attempt of choosing CR value can be 0.1. However, because the large CR value can speed up convergence, the value of 0.9 for CR may also be a good initial choice if the problem is near unimodal or fast convergence is desired. Moreover, if the population converges prematurely, either F or NP can be increased. Following diagram shows main stages of DE algorithm

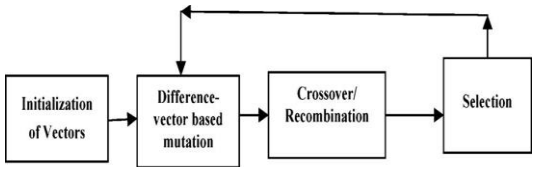


Fig.1. Differential evolution stages

Initially population NP generated randomly using following formula

$$xij = xmin + (xmax - xmin) * r1) \quad (1)$$

Where x min = lower bound and x max =upper bound and r1 is a uniform random number between 0 and 1. In this paper we have used randperms matlab function to generate population randomly.

Randperms function consist two parameters one is number of jobs and other is size of population. In DE algorithm mutation is important step. In this step donor vector is generated using DE strategies. From research DE/rand/1/bin has powerful exploitation ability and DE/best/1/bin has efficient exploration ability. Generation of donor vector is given in following formula.

$$V_{i,G+1} = X_{ri1,G+1} + F \cdot (X_{ri2,G+1} - X_{ri3,G+1}) \quad (2)$$

Where Vi,G+1 is a donor vector. Xri1,G+1, Xri2,G+1 , Xri3,G+1 are randomly selected vectors and F is a mutation factor. The range of mutation factor is in between 0 and 1. Crossover generates the trial vector. The aim of crossover operator is to generate the trial offspring by mixing the content of donor vector and target vector. Following formula shows generation of trial vector

$$U_{j,i,G+1} = \begin{cases} V_{j,i,G+1} & if\ rand \leq CR \\ X_{j,i,G} & if\ rand > CR \end{cases} \quad (3)$$

Selection is depending upon the fittest value. The offspring which having fittest value can be selected and added to the original population and thus population is updated. Updated population is send to the next generation. The process is continuing till the given stopping criterion.

4. Permutation Flowshop Scheduling Problem

Scheduling theory is concerned with the optimal allocation of resources so that time required for their execution is minimum. Consider the example of central processing unit of computer that must process a sequence of jobs that arrive at time. In what order should the jobs be processed in order to minimize completion time.

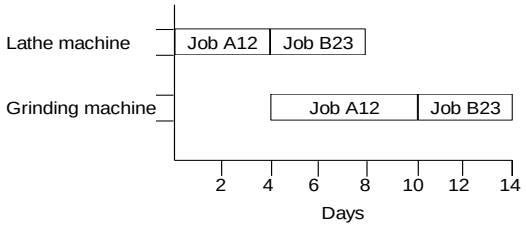


Fig. 1. Gantt chart for 2 jobs and 2 machines

The schedule shown on the Gantt chart gives a detail plan:

The lathe machine will be used by Job A12 on days 1-4, and Job B23 on days 5-8. The grinding machine will be used by Job A12 on days 5-10 and Job B23 on days 11-14.

With following assumptions

1. No job uses more than one machine simultaneously
2. No machine processes more than one job simultaneously.
3. All jobs are ready for processing at time zero.
4. The machines are continuously available from time zero onwards (no breakdowns).
5. At any time, each machine can process at most one job and each job can be processed on at most one machine.

No pre-emption is allowed (that is, once the processing of a job on a machine has started, it must be completed without interruption). Only permutation schedules are allowed (i.e. all jobs have the same ordering sequence on all machines).

Permutation flowshop scheduling problem is an optimization problem used in various industries such as production, manufacturing industry. Better scheduling system has significant impact on cost reduction, increased productivity, customer satisfaction and overall competitive advantage. Thus, the flow shop problem (FSP) is one of the most important problems in the scheduling theory. This problem can be described as follows. Each job ji (i=1, 2… n) has to be processed on m machines Mj (j = 1 . . . m), following the same order in all machines. The processing time of job ji on machine Mj is pij. In this work, we focus on the minimization of the completion time of the last job of the last machine called makespan and denoted Cmax. The problem is denoted F|prmu|Cmax. The calculation of completion time for the n-job, m-machine problem is given as follows:

$$C(\pi 1, 1) = P\pi 1, 1 \quad (4)$$

$$C(\pi j, 1) = C(\pi j-1, 1) + P\pi j,1 \quad j=2,\ldots n \quad (5)$$

$$C(\pi 1, k) = C(\pi 1, k) + P\pi 1,k \quad k=2,\ldots m \quad (6)$$

$$C(\pi j, k) = \max \{C(\pi j-1, k), C(\pi j, k-1) + P\pi j, k\} \quad j=2,\ldots n ; k=2,\ldots m \quad (7)$$

Then makespan can be defined as

$$Cmax(\pi) = C(\pi n, m).$$

So, the PFSP with the makespan criterion is to find a permutation π^* in the set of all permutations Π such that

$Cmax(\pi^*) \leq C(\pi n, m) \; \forall \; \pi \in \Pi$

5. Proposed Techniques

In this proposed work we have done three different techniques with DE

5.1 DE with Positioned based crossover operator

First proposed technique consist simple De with positioned based crossover operator. Following figure shows proposed algorithm.

Step 1: The first step is the random initialization of the parent population. Randomly generate a population of NP vectors, each of n dimensions: xi,j= xmin,j+ rand(0, 1)(xmax,j-xmin,j), where xmin,j and x max are lower and upper bounds for jth component respectively, rand(0,1) is a uniform random number between 0 and 1.

Step 2: Calculate the objective function value f (Xi) for all Xi.

Step 3: Select three points from population and generate perturbed individual Vi using equation (1a).

Step 4: Recombine the each target vector xi with perturbed individual generated in step 3 to generate a trial vector Ui using equation (2).

Step 5: Perform positioned based crossover operator on trial and Donor vector.

Step 6: Calculate the objective function value for vector Ui.

Step 7: Choose better of the two (function value at target and trial point) using equation (3) for next generation.

Step 8: Check whether convergence criterion is met if yes then stops; otherwise go to step 3

This algorithm consist modified mutation operator which dynamically set the value of mutation factor.

5.2 DE with Classical NEH_Iterated Local Search and Enhanced swap operator

The NEH heuristic algorithm made by Nawaz et al. As one of the efficient algorithm in this field. Iterated local search and enhanced swap operator are easy to implement and very effective. Following figure shows procedure on DE_NEH-ILS-ESP.

Step 1: Initialize target population randomly and initial parmeters like CR and F
Step 2:. Evaluate target population
Step3:. nehR=classical NEH
Step 4:. Search:=ILS(nehR, tareget_vector)
Step 5:. while (not termination) do
Step 6:. Obtain mutant population
Step 7:. Obtain trial population
Step 8:. Evaluate trial population
Step 9:. Apply ESP
Step 10:. Make selection
Step 11: Apply local search LS()
Step 12: End while
End

5.3 Improved DE

Following are the steps used in this algorithm
Step 1: Generate the population randomly
Step 2: Evaluate the objective function for each individual.
Step 3: Sort the objective value in ascending order.
Step 4: Select first value as target vector.
Step 5: Select first half population as a new population.
Step 6: Apply the DE algorithm
Repeat step 2-6 until the last single population with minimum makespan value.

This module gives improved makespan value in minmum time. It improves the time complexity of algorithm. Modification in differential algorithm as follows. After evaluating population we sort the population in ascending order. First value will be the minimum makespan value known as target vector. Then population is divided into subpopulation. Apply DE on subpopulation until we get minimum value. Currently, there exists several mutation strategies DE/rand/1/bin are used commonly. In this paper we will use DE/rand/1/bin.

6. Experiments

To discover the effectiveness of the presented techniques, the free available test data Car1 up to Car8 from OR library were used. This Flow shop instances were investigated by many researchers who applied a variety of techniques to solve it (and also there is a known optimal value of objective function). Firstly, the 100 simulation were carried out to determine the effective setting of control parameters f and cr. The best parameters obtained from combining these parameters during experimentation are: CR = 0, 1 and F = 0.2, 0.9. Dynamically generated value for F are -0.1,1.1. Although those setting present a relative low crossing rate and mutation, they seemed to be adequate to guarantee evolution process. After all simulation we can resume that we were able to achieve optimal solution for all instances Car. Once all the trials were done, we transformed the data and used as the response variable of the experiment
the following:

$$Relative\ Percentage\ Deviation(RPD) = \frac{Optimal\ \ Val - Obained\ \ val}{Obtained\ \ Val} \quad (8)$$

Where Optimal Value is the solution obtained by a given algorithm alternative on a given instance and Obtained value is the lowest makespan obtained in any experiment of the same instance. The results are given in Table 1:

Table1.Results for Improved DE algorithm

NP	CR	F	Optimal Value	Obtained value	RPD	CPU TIME
2500	0.8	0.7393	7038	7038	0	15.571602 sec.
400	0.6	-0.193	7166	7065	0.01429582	2.578329 sec.
2500	0.8	0.4606	7312	7032	0.03981797	15.608372 sec.
700	0.3	1.0584	8003	8003	0	4.518795 sec.
700	0.7	0.9395	7720	7680	0.00520833	4.137504 sec.
7000	0.6	1.1521	8505	8496	0.00105932	35.388487 sec.
9000	0.6	0.3126	6590	6328	0.04140329	52.126734 sec.
9000	0.6	-0.0351	8366	7889	0.06046394	53.803392 sec.

Following table shows experimental result of two different techniques which we have used in project for analysis of DE algorithm.

1. DE with positioned based crossover operator-DEPCO.

2. Comparative results of GA, QIDE and DE with classical NEH, iterated local search and enhanced swap operator.

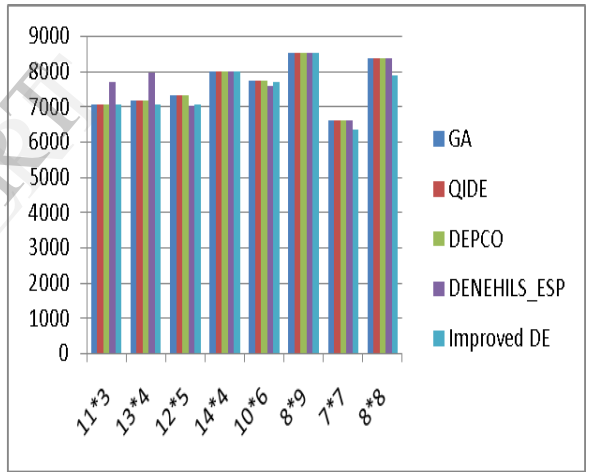
Table2.Results for DEPCO

NP	CR	F	Optimal Value	RPD	CPU TIME
2500	0.8	0.5	7038	0	30.597825 seconds.
400	0.6	0.3	7166	0	7.177144 seconds.
2500	0.8	0.7	7312	0	31.033407 seconds.
700	0.3	0.9	8003	0	11.044217 seconds.
700	0.7	0.2	7720	0	8.844516 seconds.
7000	0.6	0.4	8505	0	85.114471 seconds
9000	0.6	0.9	6590	0	111.139205 seconds.
9000	0.6	0.5	8366	0	116.937168 seconds

Table 3. Comparative result of GA and QIDE with three proposed algorithm

Problem	Optimal Value	GA	QIDE	DEPCO	RPD	DENE H_{ILS_ES}	RPD	Improved DE	RPD
11*3	7038	7038	7038	7038	0	7685	-8.419	7038	0
13*4	7166	7166	7166	7166	0	7952	-9.884	7065	0.01429582
12*5	7312	7312	7312	7312	0	6995	4.531808	7032	0.03981797
14*4	8003	8003	8003	8003	0	8003	0	8003	0
10*6	7720	7720	7720	7720	0	7557	2.156941	7680	0.00520833
8*9	8505	8505	8505	8505	0	8505	0	8496	0.00105932
7*7	6590	6590	6590	6590	0	6590	0	6328	0.04140329
8*8	8366	8366	8366	8366	0	8345	0.251648	7889	0.06046394

Fig.4 comparative graph of proposed techniques with GA



7. Conclusions

This paper consist implementation of differential evolution algorithm consisting of positioned based crossover operator, modified mutation operator, classical NEH, iterated local search, enhanced swap operator and finally some modification in algorithm. Experimental results are shown in tables. Table 3 consist comparative results of these three techniques. Comparison shows that improved algorithm is very effective. In this algorithm size of population automatically reduces. Variable population size work better compared to constant population size. Experiment shows that crossover factor greater than 0.5 gives better exploration and exploitation result. Dynamic mutation factor is used but effective choice for mutation factor is between 0.2-0.9.

10. References

[1] K. C. Ying and S. W. Lin, Multi-heuristic desirability ant colony system heuristic for nonpermutation owshop scheduling problems, International Journal of Advanced Manufacturing Technology, vol.33, no.7-8, pp.793-802, 2007.

[2] Y. Li, Y. Yang, L. Zhou and R. Zhu, Observations on using problem-speci_c genetic algorithm for multiprocessor real-time task scheduling, International Journal of Innovative Computing, Information and Control, vol.5, no.9, pp.2531-2540, 2009. 6610 S.-W. LIN, C.-Y. HUANG, C.-C. LU AND K.-C.

[3] S. W. Lin and Y. C. Ying, Applying a hybrid simulated annealing and tabu search approach to nonpermutation owshop scheduling problems, International Journal of Production Research, vol.47,no.5, pp.1411-1424, 2009.

[4] K. C. Ying, Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic, International Journal of Advanced Manufacturing Technology, vol.38, no.3-4, pp.348-354,2008.

[5] Wenbo Liu ,"An improved differential evolution for permutation flowshop scheduling problem with total flowtime criterion ",System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2012 3rd International Conference on (Volume:1)

[6] Samia kouki1, Mohamed Jemni1, Talel Ladhari2," Solving the Permutation Flow Shop Problem with Makespan Criterion using Grids", International Journal of Grid and Distributed Computing Vol. 4, No. 2, June, 2011

[7] Conway, R. W.; Maxwell, W. L.; Miller, L. W. Theory of Scheduling,Addison-Wesley: Reading,MA 1967

[8] Pinedo, M. Scheduling: Theory, Algorithms and Systems. Prentice Hall,NewJersey, second edition 2002.

[9] Palmer, D. S. Sequencing jobs through a multi-stage process in the minimum total time – a quick method of obtaining a near optimum, Operations Research. Q. 16(1965), 101-107.

[10] Campbell, H. G.; Dudek, R. A.; Smith, M. L. A Heuristic Algorithm for the n Job, m Machine Sequencing Problem, Management Science, 16 10(1970), 630-637.

[11] Dannenbring, David G. An Evaluation of Flow Shop Sequencing Heuristics, Management Science, 23, 11(1977), 1174-1182.

[12] Brucker, P.; Jurisch, B.; Sievers, B. A branch and bound algorithm for the job shop scheduling problem. Discrete Applied Mathematics, 49 1(1994), 109–127.

[13] Gendreau, M.; Laporte, G.; Semet, F. A tabu search heuristic for the undirected selective travelling salesman problem, European Journal of Operational Research, Elsevier, 106 2-3(1998), 539-545.

[14] Nowicki, E.; Smutnicki, C.Afast taboo search algorithm for the job shop problem. Management Science, 42 6(1996), 797–813.

[15] Logendran, R.; de Szoeke, P.; Barnard, F. Sequencedependent group scheduling problems in flexible flow shops. International Journal of Production Economics 102 (2006),66–86.

[16] Manikas. A.; Chang,Y. L. Multi-criteria sequence-dependent job shop scheduling using genetic algorithms Computers & Industrial Engineering 56 (2009), 179–185.

[17] Murata, T.; Ishibuchi. H.; Tanaka, H. Genetic Algorithms for Flow shop Scheduling Problems, Computers & Industrial Engineering, 30, 4 (1996), pp. 1061-1071.

[18] Balas, E. and A. Vazacopoulos. Guided Local Search with Shifting Bottleneck for Job Shop Scheduling. Management Science, 44, 2(1998), 262-275.

[19] Blum, C.; Sampels, M. An Ant Colony Optimization Algorithm for Shop Scheduling Problems. Journal of Mathematical Modelling and Algorithms, 3, 3(2004), 285-308

[20] V. L. Huang, S. Z. Zhao, R. Mallipeddi and P. N. Suganthan," Multi-objective Optimization Using Self-adaptive Differential Evolution Algorithm",

[21] Janez Brest, Member, IEEE, Viljem ˇ Zumer, Member, IEEE, and Mirjam Sepesy Mauˇcec," Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization", 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006

[22] A. K. Qin, V. L. Huang, and P. N. Suganthan," Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 13, NO. 2, APRIL 2009

[23] R.Storn, and K.Price, DE-a simple and efficient adaptive scheme for global optimization over continuous space, Technical Report TR-95-012, ICSI, March 1995. Available via the Internet: ftp.icsi.berkeley.edu/pub/techreports/ 1995/tr-95-012.ps.Z, 1995.

[24] Y. Tang, H. Gao, J. Kurths, and J. Fang, "Evolutionary pinning control and its application in UAV coordination," IEEE Transactions on Industrial Informatics, vol. 8, pp. 828–838, 2012. View at Publisher · View at Google Scholar

[25] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," Evolutionary Computation, vol. 6, no. 2, pp. 161–184, 1998. View at Scopus

[26] Y. Tang, Z. Wang, H. Gao, S. Swift, and J. Kurths, "A constrained evolutioanry computation method for detecting controlling regions of cortical networks," IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 9, pp. 1569–1581, 2012. View at Publisher · View at Google Scholar.