Advanced Engineering Forum *Online: 2011-09-09*
ISSN: 2234-991X, Vol. 1, pp 110-115
doi:10.4028/www.scientific.net/AEF.1.110
© 2011 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

A Hybrid Shuffled Frog Leaping Algorithm for solving No_Idle Permutation Flow Shop Scheduling Problems

Wang Ya-min[1, a], Bao Yun[1,b], Chen Jing[1,c], Li Jun-qing[1,d]

[1]College of Computer Science, Liaocheng University, Liaocheng, 252059, China.

[a] wangyamin@163169.net, [b] baoyun@lcu.edu.cn, [c] chenjing@lcu.edu.cn, [d] lijunqing@lcu.edu.cn

Keywords: Hybrid shuffled frog leaping algorithm; No_idle permutation flow shop scheduling; Insert neighborhood search; Roulette wheel selection

Abstract: This paper presents a novel hybrid shuffled frog-leaping algorithm (HSFLA) for solving the no_idle permutation flow shop scheduling problems (NIFS) with the criterion to minimize the maximum completion time (makespan). First, the algorithm employs insert- neighborhood-based local search to enhance the searching ability. Second, it adopts roulette wheel selection operator to generate the global best frog in the early stage of the evolution which can expand the searching solution space. The experimental results show that the proposed algorithm is effective and efficient for different scale benchmarks of NIFS.

Introduction

No-idle permutation flow shop scheduling problems as a typical manufacturing challenge have gained wide attention in academic fields, which occurs in the scheduling environment where the setup time or cost of a machine is so high that shutting down the machine after its initial setup is not cost-effective. In a no-idle permutation flow shop scheduling problem, the processing of each machine has to continuously work. That is, each machine must process jobs without any interruption from the start of processing the first job to the completion of processing the last job. Therefore, it is important both in theory and in engineering applications to develop effective and efficient novel approaches for the no-idle permutation flow shop scheduling problems [1].

However, the no-idle permutation flow shop scheduling problems have not been researched enough so far [2,3]. According to the research work [4], the no-idle permutation flow shop scheduling problem to minimize makespan with three machines is strongly NP-hard. Baptiste and Lee [4] and Saadani, Baptiste and Moalla [5] developed a branch and bound algorithm for problem. Kalczynski and Kamburowski [2] developed a constructive heuristic for minimizing the makespan in NIFS problems. Baraz and Mosheiov [3] introduced an efficient improved greedy algorithm consisting of a simple greedy heuristic and an improvement step. With the rapid development of computer technology, some meta-heuristic algorithms are presented to solve NIFS problems. Such as discrete differential evolution (DDE) [6], discrete harmony search algorithms [7], discrete particle swarm optimization (DPSO) [1] and discrete shuffled frog leaping algorithm (DSFLA) [8], etc.

The SFLA has been designed as a meta-heuristic to perform an informed heuristic search using a heuristic function (any mathematical function) to seek a solution of combinatorial optimization problem [9]. In essence, it combines the benefits of the genetic-based MAs and the social behavior-based PSO algorithms. But there are a few papers that addressed the shuffled frog leaping algorithm [9-15]. This paper develops a novel hybrid shuffled frog leaping algorithm(HSFLA) for solving the no-idle permutation flow shop scheduling problems with makespan criterion. In the proposed HSFLA, the balance between global exploration and local exploitation is well stressed by employing roulette wheel selection to effectively perform exploration and by applying an effective insert neighborhood search algorithm to perform exploitation.

1 The no-idle permutation flow shop scheduling problem

The no-idle permutation flow shop scheduling problem can be described as follows: Each of n jobs, available at time zero, from set $J = \{1, 2,..., n\}$ will be sequenced through m machines ($k = 1, 2,..., m$). Job $j \in J$ has a sequence of m operations (o_{j1}, o_{j2},..., o_{jm}). Operation o_{jk} corresponds to the

This article is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0)

processing of job j on machine k during an uninterrupted processing time $p(j, k)$, where setup time is included into the job processing time. At any time, each machine can process at most one job and each job can be processed on at most one machine [16]. To follow the no-idle restriction, each machine must process jobs without any interruption from the start of processing the first job to the completion of processing the last job. In other words, there must be no idle time between the processing of any consecutive operations on each machine. The problem is then to find a schedule such that the processing order of jobs is the same on each machine and the maximum completion time (makespan) is minimized. Paper [1] proposed a new way of calculating makespan whose computational complexity is $O(mn)$ and a speed-up method to evaluate the whole insert neighborhood of a job permutation with $(n-1)^2$ neighbors in time $O(mn^2)$.

2 Shuffled frog-leaping algorithm (SFLA)

The SFLA is inspired by the behavior of a group of frogs seeking food. It is a combination of deterministic and random approaches. The deterministic strategy allows the algorithm to use response surface information effectively to guide the heuristic search as in PSO. The random elements ensure the flexibility and robustness of the search [6]. In the SFLA, the population consists of a set of frogs (solutions) that is partitioned into subsets referred to as memeplexes. The different memeplexes are considered as different cultures of frogs, each performing a local search. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of memetic evolution. The frogs with the best and the worst fitness are identified as P_b and P_w, respectively in each memeplex, Also, the frog with the global best fitness is identified as P_g, Then, a process similar to PSO is applied to improve only the frog with the worst fitness (not all frogs) in each cycle. Accordingly, the position of the frog with the worst fitness is adjusted as follows:

$$Change\ in\ frog\ position\ (D_i) = rand(\ )*(P_b - P_w) \quad (1)$$

$$New\ position\ P_w = current\ position\ P_w + D_i\ ,\ (D_{max} >= D_i >= -D_{max}) \quad (2)$$

where $rand(\ )$ is a random number between 0 and 1, and D_{max} is the maximum allowed change in a frog's position. If the new P_w is better than the old one, calculations in Eq.1 and Eq.2 are repeated but with respect to the global best frog (i.e. P_g replace P_b). If no improvement becomes possible in this case, then the spread of defective meme is stopped by randomly generating a new frog at a feasible location to replace the frog whose new position was not favorable to progress. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process [8]. The local search and the shuffling processes continue until defined convergence criteria are satisfied [7].

3 HSFLA for solving NIFS problems

3.1 Solution representation

It is important to design solution representation in a meta-heuristic algorithm. One of the most widely used representation for sequencing problems including permutation flow shop sequencing problem is job-to-position representation. In this kind of representation, a single row array of the size equal to the number of the jobs to be scheduled is considered. The value of the first element of the array shows which job is scheduled first. The second number shows which job is scheduled second, and so on. Table 1 shows how this representation is depicted.

Table 1 Individual vector and corresponding job permutations

Dimension	1	2	3	4	5	6	7
Individual Vector	6	3	2	4	1	7	5
Job Sequence	6	3	2	4	1	7	5

3.2 Initial population

In order to guarantee the initial population with certain diversity in solution space, the frogs in the initial population are generated as follows:

Step 1 i=1;

Step 2 Generate a permutation $\pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ corresponding to the *i*th frog, in which $\pi_1 = i\%n + 1$ and π_j was generated in random.

Step 3 *i* ++;

Step 4 If *i* is not more than the number of frogs in the population, go to step 2 and continue, otherwise, stop.

3.3 Update individual frogs

Eq.1 and Eq.2 show that the essence of updating the worst frog P_w is a process of P_w studying the local best frog P_b or the global best frog P_g. Where *rand*() is a random number between 0 and 1, and it represents the degree of the frog P_w inherit the genes of the best frog. So the process of above can be replaced by crossover. When *rand*() is 1, P_w is replaced by the best frog P_b or P_g. When *rand*() is 0, P_w stays the same. The crossover is implemented detailed as follows:

Step 1 Select a part of P_b randomly as crossover string.

Step 2 Combine P_b and P_w by put the crossover string before (or behind) P_b and delete the element which has appeared in crossover string.

Illustrate with examples.

Current position P_w = {1,5,6,3,2,7,4}

the local best frog P_b = {6,5,4,7,1,3,2}

if the crossover string is: {5,4,7}

then *new position* P_w = {5, 4, 7, 1, 6, 3, 2} or {1, 6, 3, 2, 5, 4, 7}

This process shows that the update of individuals is easy to implement and the new frog can inherit the effective memes from its parents.

3.4 Insert-neighborhood-based local search

The population depends extremely on the local best result and the group best result. Based on this character, the speed-up algorithm was proposed employs the simple neighborhood search [1], and it can improve the convergence rate of SFLA. In SFLA every memeplex evolves through memetic algorithm (MA). All frogs are reordered and classified, so the information can flow and is shared among all memeplexs. This can speed up the convergence rate of population. In genetic algorithm (GA) evolutional information is obtained from the entire population in probability. Otherwise frogs can get updated information from the local best solutions and the global best solutions. Compare of GA the information flows in one way, and the purpose of flow is more efficient in SFLA. The search was deeply affected by the best individuals. So strengthening the detection capacity of the local search can improve SFLA's performance. Based on this analysis, the improved algorithm was proposed employs the simple neighborhood search on the best results. But the scope of precise search is only limited to the nearest neighborhood of the best results. It is likely to make the solutions "wandering" around the old state for a long time, which make the algorithm more easily trapped into local minima. In order to enhance the algorithm's ability to jump out of local optimum, we can implement a random insert move after a neighborhood search and repeat the process several times. If a better solution is found, the old solution is replaced by the new one. Through the precise search the population can move to the optimal solution more quickly, and it can improve the convergence rate of SFLA. Insert or swap neighborhood structure can be used in simple neighborhood search, and the process depends on the problems.

3.5 Roulette wheel selection the global best solution

The best solution P_g is the one whose performance is best in the population, and the search was largely infected by the global best solution P_g. So in the early iterations, the evolution tends to be controlled by the super-frog who has the absolute advantage. If the objective function is multimodal, in the long run, the frog is perhaps not able to guide the search to the global optimum, so it is prone to premature convergence. In order to other "less good" frogs have the opportunity to guide the search, roulette wheel selection operator is used in SFLA. This measure can reduce the control function of the super-frogs and avoid the premature phenomenon to some extent. Detailed implementation method of roulette wheel selection P_g from all the local best frogs P_b is as follows:

Step 1 *for* $i = 1,2,\cdots,M$ *calculate* $F(g_{bi}) = \dfrac{1}{C_{\max}(g_{bi})}$, $\Pr(g_{bi}) = \dfrac{F(g_{bi})}{\sum_{j=1}^{M} F(g_{bi})}$

Step 2 $H(g_{b0}) = 0$;

for $i = 1,2,\cdots,M$ *compute* $H(g_{bi}) = H(g_{bi-1}) + \Pr(g_{bi})$

Step 3 *if* $H(g_{bi}) \geq rand() \geq H(g_{bi-1})$, where *rand*() is a random number between 0 and 1, then select g_{bi} as the global best frog.

3.6 Steps of the hybrid shuffled frog-leaping algorithm

The HSFLA meta-heuristic strategy is summarized in the following steps.

Global exploration

Step 0 *Initialize*. Select *M* and *N*, where *M* is the number of memeplexes and *N* is the number of frogs in each memeplex. Therefore, the total sample size *F* in the swamp is given by $F = M \times N$. Select *Tr*, where *Tr* is the time of applying roulette wheel selection operator to get the best frog $\boldsymbol{P}_g$.

Step 1 *Generate a virtual population*. Sample *F* virtual frogs $\boldsymbol{U}(1), \boldsymbol{U}(2), \ldots, \boldsymbol{U}(F)$ in the feasible space $\Omega \subset \Re^n$, where *n* is the number of jobs. The *i*th frog is represented as a solution of NIFS problems $U(i) = (U_i^1, U_i^2, \cdots, U_i^n)$. Compute the makespan as $f(i)$ for each frog $\boldsymbol{U}(i)$. So the smaller of $f(i)$, the better of $\boldsymbol{U}(i)$.

Step 2 *Rank frogs*. Sort the *F* frogs in order of decreasing performance value. Store them in an array $\boldsymbol{X} = \{\boldsymbol{U}(i), f(i), i = 1, \cdots, F\}$, so that $i = 1$ represents the frog with the best performance value.

Step 3 *Partition frogs into memeplexes*. Partition array $\boldsymbol{X}$ into *M* memeplexes : $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_M$, each containing *N* frogs, such that $Y_k = \{U(j)_k, f(j)_k \mid U(j)_k = U(k+M(j-1)), f(j)_k = f(k+M(j-1)), j=1,\cdots,N\}, k=1,2,\cdots,M$ *e.g.*, for $m = 3$, rank 1 goes to memeplex 1, rank 2 goes to memeplex 2, rank 3 goes to memeplex 3, rank 4 goes to memeplex 1, and so on. Record the local best frog's position $\boldsymbol{P}_b$ of every memeplex, such that $\boldsymbol{P}_{bk} = U(k)$, $k=1,2,\cdots,M$.

Step 4 Application insert-neighborhood-based local searches to all the local best frogs.

Step 5 *Selection the best frog* $\boldsymbol{P}_g$. If the execution time is less than *Ts*, get the best frog's position $\boldsymbol{P}_g$ by roulette wheel selection from all the local best frogs $\boldsymbol{P}_b$. Otherwise, record the best frog's position $\boldsymbol{P}_g$ in the entire population (*F* frogs) (where $\boldsymbol{P}_g = \boldsymbol{U}(1)$).

Step 6 Application insert-neighborhood-based local search to the global best frogs $\boldsymbol{P}_g$.

Step 7 *Memetic evolution within each memeplex*. Evolve each memeplex $\boldsymbol{Y}_k$, $k = 1, \ldots, M$, according to the frog-leaping algorithm outlined below.

Step 8 *Shuffle memeplexes*. After a defined number of memetic evolutionary steps within each memeplex, replace $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_M$ into $\boldsymbol{X}$, such that $\boldsymbol{X} = \{\boldsymbol{Y}_k, k=1,2,\cdots,M\}$. Sort $\boldsymbol{X}$ in order of decreasing performance value. Update the population best frog's position $\boldsymbol{P}_g$.

Step 9 *Check convergence*. If the convergence criteria are satisfied, stop. Otherwise, return to step 3. Typically, the decision on when to stop is made by a prespecified number of consecutive time loops when at least one frog carries the "best memetic pattern" without change. Alternatively, a maximum total number of function evaluations can be defined.

Local exploration: frog-leaping algorithm

In step 7 of the global search, evolution of each memeplex continues independently *iN* times. After the memeplexes have been evolved, the algorithm returns to the global exploration for shuffling. Below are details of the local search for each memeplex.

Step 0 Set $im = 0$ where *im* counts the number of memeplexes and will be changed from zero to the total number M of memeplexes. Set $iN = 0$ where *iN* counts the number of evolutionary steps and will be changed from zero to the maximum number N_{max} of steps to be completed within each memeplex.

Step 1 Set $im = im + 1$.

Step 2 Set $iN = iN + 1$.

Step 3 *Improve the worst frog's position.* A order crossover is applied to the worst frog $\boldsymbol{P}_w$ and the best local frog $\boldsymbol{P}_b$.

Step 4 If this process produces a better frog (solution), it replaces the worst frog. Otherwise, the order crossover is repeated with respect to the global best frog $\boldsymbol{P}_g$.

Step 5 If no improvement becomes possible in this latter case, then a new solution is randomly generated to replace the worst frog with another frog having any arbitrary fitness.

Step 6 If $iN < N_{ma}$, go to step 2.

Step 7 If $im < M$, go to step 1. Otherwise return to the global search to shuffle memeplexes.

4 Experimental results

To test the performance of the proposed HSFLA , computational simulations were carried out with the flow shop benchmark set of Taillard [17], which is composed of 12 sub-sets of given problems with the size ranging from 20 jobs and five machines to 500 jobs and 20 machines, and each sub-set consists of ten instances. Paper [8] proposed DSFLA and a speed-up algorithm which we referred as DSFLA*. For each instance of each sub-set, DPSO, DSFLA, DSFLA* and HSFLA algorithms independently carried out five replications with T=30*mn millisecond*, and each replication was compared to the solution produced by NEH [18] when modified for the no idle permutation flow shop scheduling problems. The parameters were set as follows: M=20, N=5, F=100, Ts=T/10 *millisecond.* The average percentage relative deviation (APRD) and the standard deviation (SD) were calculated as the statistics for the performance measures. The algorithm was coded in C++ and run on an Intel PIV 2.0 GHz PC with 2.0 GB memory. And the results are reported in Table 2.

Table 2 Comparison of different algorithms

Problem	DPSO		DSFLA		DSFLA*		HSFLA	
n×m	PRD	SD	PRD	SD	PRD	SD	PRD	SD
20×5	9.7	1.51	9.56	1.94	8.2	2.4	7.88	2.1
20×10	23.92	2.71	23.64	2.96	18.76	2.6	19.11	2.13
20×20	48.49	2.17	49.3	2.83	42.71	3.02	42.29	2.75
50×5	6.94	1.69	7.32	1.44	4.12	1.41	3.66	0.83
50×10	22.82	2.24	23.85	2.19	17.67	2.27	17.71	2.07
50×20	55.24	3.34	55.9	3.24	41.96	2.7	39.21	2.12
100×5	4.15	1.01	4.75	1.12	1.89	0.95	1.84	0.93
100×10	20.05	2.29	21.01	2.11	12.07	1.59	11.8	1.29
100×20	46.69	2.63	48.85	2.6	33.03	2.64	32.92	2.28
200×10	15.13	1.11	16.03	1.29	10.24	1.22	9.37	0.99
200×20	38.16	1.55	40.3	2.18	28.62	2.46	26.65	2.18
500×20	26.25	1.09	27.91	1.25	22.63	1.92	21.41	2.44
Average	26.462	1.946	27.368	2.096	20.158	2.098	19.487	1.842

From Table 2, it is shown that the statistical results generated by the HSFLA are significantly better than those by DSFLA and DPSO algorithm, which indicates that by insert neighborhood search the algorithm convergences more quickly and by roulette wheel selection it's searching space is more expanded. The results of APRA and SD carried by HSLA are less than DSFLA* show that HSFLA is super to DSFLA*. This indicates that roulette wheel selection the global best solution in early time can bring down the dependence on the global best individual, and can avoid trapping into local optimum to a certain extend. The experimental results indicate that the proposed HSFLA outperforms the DSFLA with respect to these measures and is able to improve the quality of the obtained solutions, especially for the large-size problems. Thus, it is concluded that HSFLA is more robust, effective and efficient than other algorithms of above.

5 Conclusion

A hybrid shuffled frog leaping algorithm (HSFLA) has been proposed in this paper for solving no_idle permutation flow shop scheduling problems with the criterion to minimize the maximum completion time(makespan). Insert-neighborhood-based local search improves the quality of the meme of the best individuals which guide the population towards a goal more quickly. Roulette wheel selection the global best individual can make the frogs with better memes(ideas) contribute more to the development of new ideas than frogs with poor ideas, which ensure the fast convergence. Also can it make the infection process to be competive and bring down the algorithm's independence on someone excessively, which can avoid the search trapping into the local optimum to a certain degree. Experimental results indicate that the algorithm proposed in this paper can balance the exploration and exploitation, and it is an effective and efficient algorithm for solving no_idle permutation flow shop scheduling problems.

*The work was supported by a grant from the Research Development Program of Shandong Education Committee (No. J10LG25).

References

[1] Q.K. Pan, L. Wang. No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm, J. The International Journal of Advanced Manufacturing Technology. (2007) doi:10.1007/s00170-007-1252-0.

[2] P.J. Kalczynski, J. Kamburowski. A heuristic for minimizing the makespan in no-idle permutation flow shop, J. Comput Ind Eng. 49 (2005) 146-154.

[3] D. Baraz, G. Mosheiov. A note on a greedy heuristic for the flow-shop makespan minimization with no machine idle-time, J. Eur J Oper Res. (2007) DOI 10.1016/j.ejor.2006.11.025.

[4] P. Baptiste, K.H. Lee. A branch and bound algorithm for the F|no-idle|Cmax, J. Proceedings of the international conference on industrial engineering and production management (IEPM’ 1997), Lyon. 1 (1997) 429-438.

[5] N.E.H. Saadani, P. Baptisete, M. Moalla. The simple F2//Cmax with forbidden tasks in first or last position: A problem more complex than it seems, J. Eur J Oper Res. 161 (2005) 21-31.

[6] Q.K. Pan, L. Wang. A novel differential evolution algorithm for the no-idle permutation flow shop scheduling problems, J. European Journal of Industrial Engineering. 2(3) (2008) 279-297.

[7] W. Lei, Q.K. Pan, etc. Harmony search algorithms for no-idle flow shop scheduling problems, J. Computer Integrated Manufacturing Systems. 15(10)(2009) 1960-1967.

[8] Y.M. Wang, J.Z Ji, Q.k. Pan. An Algorithm Based on Discrete Shuffled Frog Leaping for No_ Idle Permutation Flow Shop Scheduling Problem, J. Journal of Beijing University of Technolodgy, 1(36) (2010) 124-130.

[9] M. Eusuff, K. Lansey, F. Pasha. Shuffled frog_leaping algorithm : a memetic meta_heuristic for discrete optimization,J. Engineering Optimization, 38(3) (2005) 129-154.

[10] M.M. Eusuff, K.E. Lansey. Optimization of water distribution network design using the shuffled frog leaping algorithm,J. Water Resour Plan Manage. 129(3) (2003) 210-225.

[11] S. Y. Liong, M. Atiquzzaman. Optimal design of water distribution network using shuffled complex evolution,J. Journal of The Institution of Engineers, Singapore. 44(1)(2004) 93-107.

[12] Emad Elbeltagi,Tarek Hegazy,Donald Grierson. Comparison among five evolutionary-based optimization algorithm,J. Advanced Engineering Informatics. 19(1) (2005) 43-53.

[13] A. Rahimi-Vahed, A. H. Mirzaei. A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem,J. Computer&Industrial Engineering. (2007) doi:10.1016/j.cie.2007.06.007.

[14] B. Amiri, M. Fathian, A. Maroosi. Application of shuffled frog-leaping algorithm on clustering, J. Appl. Math. Comput. (2007) doi: 10.1016/j.amc.2007.04.091.

[15] R.V. Alireza , A.H. Mirzaei. Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm,J. soft comput (2007) DOI 10.1007/s00500-007-0210-y.

[16] L. Wang. Intelligence optimization algorithm with applications. Tsinghua Univ Press, Beijing, China. 2001,10.

[17] M.Nawaz, E.E. Enscore Jr, I. Ham. A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, J. OMEGA. 11(1983) 91-95.