Application of Node Based Coincidence Algorithm for Flow Shop Scheduling Problems

Ornrumpha Srimongkolkul
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University,
Bangkok, Thailand
ornornmail@gmail.com

Prabhas Chongstitvatana
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University,
Bangkok, Thailand
prabhas@chula.ac.th

***Abstract*—This work proposes an algorithm called Node Based Coincidence algorithm (NB-COIN) focusing on total flowtime minimization in the permutation flowshop scheduling problems. Many algorithms have been proved to be effective for this problem. However, in the real situation, cost of computation becomes an important factor. NB-COIN produces reasonable solutions using a lot less computation power than other algorithms in consideration. Compared to a number of well-known algorithms, the results show that NB-COIN is an effective algorithm which generates less than 1.7% different from recently best known solutions from Taillard's benchmark instances.**

Keywords—Coincidence Algorithm; permutation flowshop scheduling; Estimation of Distribution Algorithms

I. INTRODUCTION

The flowshop scheduling problem challenged many researchers for many years. It is the arrangement a sequence of task to process on all machines in the same order. This problem consists of n jobs and m machines. The processing time for each job has to be predetermined. All jobs have to process completely. Each machine is idle and can process only one job at the time with no preemption allowance. For the permutation flowshop scheduling problem (PFSP), it is not allow passing any jobs in order to reduce the possible solutions into (n!). The two popular basic objectives for PFSP are makespan minimization or total flowtime minimization. In this paper, we aim to minimize the total flowtime which denote as $F/prmu/\sum C_j$. To calculate the total flowtime, the equation is defined as follow:

$$TFT=\sum_{k=1}^{n} C_{[k][m]} \quad (1)$$

$C_{[k][m]}$ denotes the completion time of job k on the last machine. The completion time of job k, $k \in \{1,2,\dots,n\}$ on the machine i, $i \in \{1,2,\dots,m\}$ can be denotes as $C_{[k]I}$ where $C_{[k]\,0}$ and $C_{[0]\,i} = 0$.The processing time of job j on machine i is denote to $t_{[k]\,j}$. Therefore, $C_{[k]j}$ is compute as follows:

$$C_{[k]j} = max\{ C_{[k]j-1}, C_{[k-1]\,j}\} + t_{[k]\,j} \quad (2)$$

This problem has been reported as NP-hard by Garey, Johnson, and Sethi (1976) [1]. Many algorithms were presented as a new tool for PFSP. Jarboui et al. (2009) proposed an Estimation of Distribution Algorithm (EDAj). EDAj shared the main steps from Estimation of Distribution Algorithm (EDA) by Lozano et al [2] but adopt Reeves technique [3] for the selection procedure and focus on either job's position in the sequence or parent's job for the Probabilistic model. Variable neighborhood search (VNS_j) is another algorithm introduced by Jarboui et al. (2009). It is a Hybrid EDA where local search method, VNS algorithm, was used to improve the performance.

This research applies a combinatorial optimization with coincidence algorithm (COIN) [4]. Unlike other approaches of Estimation of Distribution Algorithm (EDA) [5], COIN learns building blocks of the solutions using both negative and positive samples. Through the negative learning, COIN can contribute better quality of solutions and provide more diverse solutions. In addition, COIN is simple to implement and it requires small amount of resource.

This paper is organized as follows: Section 2 determines Node-Based Coincidence Algorithm (COIN). Section 3 describes the compared algorithms. Section 4 presents the computational result and the conclusion is shown in Section 5.

II. NODE-BASED COINCIDENCE ALGORITHM

Coincidence Algorithm (COIN) [4] is a method in Estimation of Distribution Algorithm (EDA) class. Similarity to Shigeyoshi Tsutsui method [6] that has two representations: Edge Histogram Based Sampling Algorithms (EHBSAs) and Node Histogram Based Sampling Algorithms (NHBSAs), COIN also has both edge-based and node-based versions. In edge-based COIN, a coincidence denotes an adjacent pair that is generated from a probabilistic table. In the contrary, Node-based COIN represents an incidence as a node so the probability of each element can be updated independently. However, the specific characteristic of COIN is the negative correlation learning. COIN makes use of both positive and negative samples to improve the solutions.

COIN is proved to be an effective algorithm for many combinatorial optimization problems such as Traveling Salesman Problem and Knight's Tour Problem [7]. It is also used to solve mixed-model u-shaped assembly line balancing problem (MMUALBP) [8] and it is reported that COIN outperformed well-known algorithm, NSGA II.

Generally, the procedure of NB-COIN is described as the following steps:

978-1-4799-0806-6/13/$31.00 ©2013 IEEE

1) Initialization: set the parameters and generate a joint probability matrix.

2) Population production: sample the population from a joint probability matrix.

3) Population evaluation: calculate the fitness value of each candidate.

4) Candidate selection: rank all candidates by fitness value, select top and bottom candidates.

5) The joint probability matrix update: update the matrix by giving a reward and punishment to the selected candidate.

6) Termination condition check: if the termination condition is not met, repeat step 2.

A. Initialization

The joint probability matrix $H(y_i|x_j)$ is a matrix of size $n \times n$ where n is the problem size. Each row(y_i) represents the position of job. Each column(x_j) represents the probability of each job that may put into the position. In this state, we give the probability 1/n to all element in matrix $H(y_i|x_j)$. Table I. shows the example of joint probability matrix where x_j refers to probability of job i and y_i refer to the position.

TABLE I. AN EXAMPLE OF PROBABILITY MATRIX

	X_1	X_2	X_3	X_4	X_5
Y_1	0.2	0.2	0.2	0.2	0.2
Y_2	0.2	0.2	0.2	0.2	0.2
Y_3	0.2	0.2	0.2	0.2	0.2
Y_4	0.2	0.2	0.2	0.2	0.2
Y_5	0.2	0.2	0.2	0.2	0.2

[a.] The problem size is five.

Population production

To create a population, we sample the position of jobs into a sequence. Then, generate the candidate as an order using that sequence.

Population evaluation and candidate selection

The algorithm evaluates the population using fitness function and ranks the candidate from the best to the worst solution. According to the unique characteristic of COIN, it selects two groups of sub-population, the good solutions and the poor solutions. The size of these two groups denote by the cut-off size parameter such as 5% or 25%.

The joint probability matrix updating

COIN updates the joint probability matrix by giving a reward to the good solution group and a punishment to bad solution group. To reward and punish the candidate H_{xy}, the probability weight is adjusted as the equation below:

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{n}(r_{xy} - p_{xy}) - \frac{k}{n^2}(\sum_{i=1}^{n} r_{xi} - \sum_{i=1}^{n} p_{xi}) \quad (3)$$

Where k is the learning step, n is the problem size, r_{xy} is the total number of coincidence H_{xy} which found in the group of good solutions and p_{xy} is the total number of coincidence H_{xy} which found in the group of poor solutions. Furthermore, In order to maintain the probability summation in each row to 1.0, other element which share the same row as the good solution or bad solution have to adjust the probability by $\frac{k}{n^2}(\sum_{i=1}^{n} r_{xi} - \sum_{i=1}^{n} p_{xi})$.

III. ALGORITHMS IN COMPARISON

EDAj and Hybrid EDA (Jarboui et al, 2009) are used as a comparison to the proposed algorithm. EDAj and Hybrid EDA are in the class of EDA which is the most powerful class of Evolutionary Algorithms at present. The result reported here is taken from the experiment reported in [10].

There are three steps in EDAj: Selection, Probabilistic model, Update.

1) Selection:

a) calculate fitness for an individual p,

$$f(p) = 1/TFT(p) \quad (4)$$

b) the population is sorted by their fitness (higher TFT value is at the top of the list. The selection is made with the probability:

$$prob(r) = 2r/P(P+1) \quad (5)$$

where r is the rank of the r th individual in the sorted list. P is the size of population.

2) Probabilistic model:

The individual is generated according to π_{jk}.

π_{jk} is the probability of selecting the job j in the position k^{th}.

$$\pi_{jk} = \eta_{jk} \times \mu_{j[k-1]} / \sum_{l \in \Omega_k} \eta_{lk} \times \mu_{l[k-1]} \quad (6)$$

η_{jk} = the number of times that the job j appears before or in the position k.

$\mu_{j[k-1]}$ = the number of times the job j is immediately after the job in the position k-1.

Ω_k the set of jobs not already scheduled until position k.

3) Update:

After a subset of population is selected, the new individual is compared with the worst individual in the current population. If the new individual has a higher fitness value and the sequence of this new individual is unique, it replaces the worst individual.

Local search (VNS) has been incorporated into EDAj to enhance the performance. Two local search methods were reported in the experiment: *swap_local_search* and *insert_local_search*. VNS is applied to a subset of individuals selected by the probability of improvement depends on the quality.

$$p^c = \max\{\exp(RD/\alpha), \varepsilon\} \quad (7)$$

$$RD = (f(x_{current}) - f(x_{best}))/f(x_{best})$$

$x_{current}$ is the created offspring, x_{best} is the best solution found so far.

IV. COMPUTATIONAL RESULT

The proposed algorithm is implement in C++ and run on a machine with MS Windows 7 using Intel Core i5 450M, 2.40GHz and 4GB of RAM. In order to test the proposed method, 30 instances of the Taillard (1993) benchmark [9] are used where n = 20 and $m \in \{5,10,20\}$ which represent into three groups, 20×5, 20×10 and 20×20. We compare the results in two aspects; the performance using same amount of CPU time and the best solution found.

We compare the performance of NB-COIN against two well-known algorithms EDA_j and VNS_j proposed by Jarboui et al. (2009), using the average relative percentage deviation from a reference solution (ARPD). The equation of ARPD is below.

$$ARPD = \left(\sum_{i=1}^{R} \frac{(S_i - S_{best}) \times 1000}{S_{best}} \right) / R \qquad (4)$$

In Eq(2) R refers to the number of round, S_i is the solution in each round and S_{best} is the best solution among all the comparison algorithms. In this case, we compare the algorithms in different environment. We use the result of EDA_j and VNS_j from Pan and Ruiz.(2012) [10] EDA_j and VNS_j are run on a cluster of 30 blade servers with two Intel XEON 5254 quad core processors, 2,5GHz and 16GB of RAM memory in each server. As the result in Pan and Ruiz.(2012) didn't mention the S_{best} of EDA_j and VNS_j, we decide to use the recently best known results found so far for instance. In addition, we adopt the same termination criterion, maximum CPU time at t=$90mn$ milliseconds. The result from Table II shows that the average ARPD value generated by COIN is lower than the other methods.

TABLE III. ARPD RESULTS OF THE ALGORITHM

Instances	EDA_j	VNS_j	COIN
20 × 5	1.20	2.92	1.37
20 × 10	1.56	2.90	1.34
20 × 20	0.97	2.15	0.53
average	1.24	2.66	1.08

Our best solution found by COIN is compared against recently best known result reported by Jarboui et al.(2009), Zheng and Li(2011) [11] and Pan and Ruiz (2012)[10]. The proposed method uses a lot less resource with the same maximum CPU time t=400mn milliseconds. The best result of our method has small difference to the best solution. As shown in the Table III, the result from COIN is less than 1.7% different from the best known result of the other methods.

TABLE II. BEST KNOWN SOLUTIONS

Instances	Best known	NB-COIN	Differentials (%)	Instances	Best known	NB-COIN	Differentials (%)
Ta001	14033	14043	0.07	Ta016	19245	19507	1.36
Ta002	15151	15237	0.57	Ta017	18363	18624	1.42
Ta003	13301	13413	0.84	Ta018	20241	20412	0.84
Ta004	15447	15575	0.83	Ta019	20330	20597	1.31
Ta005	13529	13600	0.52	Ta020	21320	21418	0.46
Ta006	13123	13333	1.60	Ta021	33623	33816	0.57
Ta007	13548	13760	1.56	Ta022	31587	31806	0.69
Ta008	13948	14026	0.56	Ta023	33920	34088	0.49
Ta009	14295	14346	0.35	Ta024	31661	31774	0.36
Ta010	12943	13142	1.54	Ta025	34557	34705	0.43
Ta011	20911	21007	0.46	Ta026	32564	32822	0.79
Ta012	22440	22806	1.63	Ta027	32922	33318	1.20
Ta013	19833	19930	0.49	Ta028	32412	32750	1.04
Ta014	18710	18899	1.01	Ta029	33600	33976	1.12
Ta015	18641	18821	0.96	Ta030	32262	32746	1.5

V. CONCLUSION

Many algorithms have been introduced to solve the total flowtime minimization flowshop scheduling problems. In this paper, we introduce a method called Node-based Coincidence Algorithm (NB-COIN). The experiments show that this method uses less computation power than other algorithms in consideration. In addition, the solutions found by NB-COIN are close to the best known solutions for Taillard's benchmark. Thus, NB-COIN is an outstanding method for flowshop scheduling problem with the small computational resource.

Acknowledgement

We would like to thank Warin Wattanapornprom for his untiring positive suggestion on many ideas used in this paper.

REFERENCES

[1] M. Garey, D. Johnson, and R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," Mathematics of Operations Research, 1976, pp. 117–29.

[2] B. Jarboui, M. Eddaly, P. Siarry, "An Estimation of Distribution Algorithm for Minimizing the Total Flowtime in Permutation Flowshop Scheduling Problems," Computers & Operations Research, 2009, pp. 2638‑2646.

[3] J. Liu, CR. Reeves, "Constructive and Composite Heuristic Solutions to The $P|\sum C_i$ Scheduling Problem," European Journal of Operational Research, 2001, pp. 439–52.

[4] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multi-objective Combinatorial Optimization with Coincidence Algorithm," IEEE Congress on Evolutionary Computation, Norway, May 18-21, 2009.

[5] P. Larrañaga, and J.A. Lozano, "Estimation of distribution algorithms: A new tool for evolutionary computation," Kluwer Academic Publishers, Boston, 2002.

[6] S. Tsutsui, M. Pelikan, and D.E. Goldberg, "Node Histogram vs. Edge Histogram: A Comparison of PMBGAs in Permutation Domains," MEDAL Report No. 2006009, July 2006.

[7] R. Sirovetnukul, P. Chutima, W. Wattanapornprom, and P. Chongstitvatana, "The Effectiveness of Hybrid Negative Correlation Learning in Evolutionary Algorithm for Combinatorial Optimization Problems," IEEE Int. Conf. on Industrial Engineering and Engineering Management, Singapore, 6-9 Dec, 2011.

[8] P. Chutima, and P. Olanviwatchai, "Mixed-model u-shaped assembly line balancing problem with Coincidence Memetic Algorithm," J. Software Engineering & Applications, pp.347-363, 2010.

[9] E. Taillard, "Benchmarks for Basic Scheduling Problems," European Journal of Operational Research, 1993, pp. 278–85.

[10] Q.-K. Pan, and R. Ruiz, "Local search methods for the flowshop scheduling problem with flowtime minimization," European Journal of Operational Research 222, 2012, pp.31-43.

[11] Y. Zhang, and L. Xiaoping, "Estimation of Distributed Algorithm for permutation flow shops with total flowtime minimization," Computer&Industrial Engineering, 2011, pp706-718.