Application of Node Based Coincidence Algorithm for Solving Order Acceptance with Multi-process Capacity Balancing Problems

Watcharee Wattanapornprom and Tieke Li
Dongling School of Economics and Management, University of Science and Technology Beijing, Beijing, China
Warin Wattanapornprom and Prabhas Chongstitvatana
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand

ABSTRACT: Over the past decade the strategic importance of order acceptance has been widely recognized in practice. This paper presents the application of node based coincidence algorithm to solve the order acceptance problem with multi-process capacity. The results show that Node Based Coincidence Algorithm (NB-COIN) is a potential algorithm which can maximize both profit and can maximize the capacity used at the same time.
KEYWORD: Order Acceptance; limited capacity; node based coincidence algorithm; genetic algorithm

1 INTRODUCTION

In economics, excess demand refers to excess of need over supply of products being offered to the market at a given price. This leads to upper prices along with the opportunity of employment. However, excess demand in order acceptance (OA) problems is the situation that orders arrival rate greater than service level of manufacturers. So far, under the high competition conditions, the manufacturers cannot straightforwardly raise their prices or just increase the temporary working capacity, subsequently, they need to select the most profitable set of the orders with regular employment cost and capacity.

OA is classified as a multi-dimensional knapsack problem which is a well-known NP hard problem. Additionally, there also exists the necessity of order sequencing which makes it much more difficult than the general knapsack problems. For example, the difference sequence of orders can result in difference profit level. (Senju & Toyoda 1968, Kleywegt &Papastavrou 2001)

In 2011, Slotnick presented a recent overview of OA which addresses simultaneous order acceptance and scheduling decisions. From the literature it was found that most of the researches focus on accepting order in a single machine or process. However, by assuming that all the processes are grouped into a single process, the accepting consequences become inefficient in many real production situations such as tardiness or over or under capacity utilization.

This paper presents a new technique to solve order acceptance or rejection in multi-process environments using Node Based Coincidence Algorithm (NB-COIN). The method is presented in section 2. The results are compared with Genetic Algorithm in section 3. Finally, the section 4 concludes the work.

2 METHODOLOGY

2.1 *The Order Acceptance Model*

The set of order i =(1,2,⋯,i), where i is one of the k product type and profit per unit is P_{ik} . Each order must be processed through set of production unit N=(1,2,⋯,n). An order i is said to be early if finishing time t is equal or less than due date d, t-$D_i \leqslant 0$ and overdue if t is more than the due date t-$D_i > 0$. A product k consumes capacity CTP_{nt} as e_{ijknt} per unit, so the selected orders will occupy total production capacity $\sum_i e_k q_{ikt}$ for $\forall t$. Each production order consists of several jobs. The jobs have precedence (i.e., job j + 1 can start only if job j is completed). RT_n is the regular working time allowed in a day, which is assumed to be eight hours. The model can be defined as follow:

Capacity Constraint

RT_n Total capacity of workstation n

CTP_{nt} Unassigned capacity of workstation n at period t (t=1,…,T)

e_{ijknt} Consumption of CTP_t for product k in order i by job j

f_{ijknt} Time unit that workstation n utilize CTP_t for product k in order i by job j at period t
g_n Cost of unassigned capacity of workstation n CTP_n per time unit
α_n Cost rate of leftover capacity at workstation n
d_{nt} Amount of leftover capacity at workstation n

Order Constraint
p_{ik} Profit of order i
q_{ikt} Demand quantity of product k in order i due at period t

Decision Constraint
R_{ik} = 1, if the order i for product k is accepted
= 0, otherwise
F_{ijktn} = 1, if the order i for product k is produced at workstation n by job j at period t
= 0, otherwise

Model Objective

$$\text{Maximize Z} = \sum_t \sum_i \sum_k p_{ik} q_{ikt} R_{ik} - \sum_t \sum_n \alpha_n d_{nt} g_n \quad (1)$$

Subject to

Workstation-level activities Constraint

$$\sum_k \sum_i \sum_t e_{kij} q_{ikt} \times R_{iktn} \le \sum_t CTP_{nt} \quad \forall n \quad (2)$$

$$d_{nt} = RT_n - CTP_{nt} - \sum_t \sum_n e_{ijk} q_{ikt} f_{iktn} \quad \forall n \quad (3)$$

Order-level activities Constraint

$$f_{kij} q_{ikt} \ge F_{iktn} \quad \forall i,j,k,n,t \quad (4)$$

$$f_{kij} q_{ikt} \le e_{kij} q_{ikt} F_{iktn} \quad \forall i,j,k,n,t \quad (5)$$

$$\sum_n t F_{i|j|knt} \le D_i R_{ik} \quad \forall i,k,t \quad (6)$$

$$\sum_n \sum_t f_{ki(j-1)} q_{ikt'} + \sum_n f_{ikt} q_{ikt} \ge \sum_n e_{ki(j-1)} q_{ikt} F_{iktn} \quad \forall i,j/\{1\},k,t \quad (7)$$

Binary and non-negativity Constraint

$$R_{ik} = 0 or 1 \quad \forall i,k \quad (8)$$

$$F_{ijknt} = 0 or 1 \quad \forall i,j,k,n,t \quad (9)$$

$$f_{ijknt} \ge 0 \quad \forall i,j,k,n,t \quad (10)$$

This problem is considered to be a two objectives optimization problem. However, the two objectives are bind into one single objective. The objective function consists of two parts (i) to maximize the total profit and (ii) to minimize the leftover capacity. Generally speaking, the objective is to choose the set and sequence of the profitable orders using as much working capacity as possible. The leftover capacity is considered to have some certain penalty cost. The first set of constraints is established to ensure that the whole capacity of production plant is not disrupted. Constraint (2) was set to calculate the penalty of under capacity utilization. Constraints (3) and (4) sets the F_{ijkrt} decision variables to either 1 or 0. The F_{ijkrt} is the indicator variable; it becomes 1 when $f_{ijkrt} > 0$, indicating that job j of item i is being processed on resource k in period t, otherwise it becomes 0. The F_{ijkrt} variable is used to ensure the precedence relationship. The constraint set (5) ensures that when an order for an item is accepted, the completion time of the final job of that order does not exceed the order due date. The constraint set (6) imposes precedence restrictions to ensure that job j of item i can be processed in period t only after completing job j-1.

2.2 *Solution Procedures*

This work compares the result of Node Based Coincidence Algorithm (NB-COIN) (Waiyapara et al. 2013) with Genetic Algorithm (GA) (Syswerda 1991). The algorithms are modified such that they would consider only the accepted sets of orders.

2.2.1 *Node Based Coincidence Algorithm*

NB-COIN is a permutation based Estimation of Distribution Algorithm (EDA). It generates solution strings in sequences, ensuring that only valid permutations are sampled. NB-COIN is a variation of Coincidence Algorithm (COIN) proposed by Wattanapornprom and others (2013). It uses a data structure called coincidence matrix H to model substructures from absolute positions. The matrix H_{xy} represents the probability of y found in the absolute position x. The update equation of NB-COIN is

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{(n)}\left(r_{xy}(t+1) - p_{xy}(t+1)\right) + \frac{k}{(n)^2}\left(\sum_{j=1}^{n} p_{xj}(t+1) - \sum_{j=1}^{n} r_{xj}(t+1)\right) \quad (11)$$

where k denotes the learning step, n is the problem size, r_{xy} is the number of xy found in the better-group, and p_{xy} is the number of xy found in the worse-group. The incremental and detrimental step is $\frac{k}{(n-1)}$, and the term $\frac{k}{(n-1)^2}\left(\sum_{j=1}^{n} p_{xj}(t+1) - \sum_{j=1}^{n} r_{xj}(t+1)\right)$ represents the adjustment of all other H_{xj}, where $j \neq x$ and $j \neq y$.

After each population was evaluated and ranked, two groups of candidates are selected according to their fitness values: better-group and worse-group. The better-group is selected from the top $c\%$ of the rank and is used as a reward, and H_{xy} is increased for every pair of xy found in this group. The punishment is a decrease in H_{xy} for every pair of xy found in the worse group of the bottom c% of the population rank.

The pseudo code of NB-COIN is simplified as follows:

Step 1 Initialize the model
Step 2 Sample the population
Step 3 Evaluate the population
Step 4 Select candidates
Step 5 Update the model
Step 6 Repeat steps 2 to 5 until terminated.

2.2.2 *Genetic Algorithm*

The GA used in this research is the permutation based GA with Position-based crossover (PBX) (Syswerda 1991). PBX preserves not only absolute order substructures but also relative order substructures from two parents. Figure 1 illustrates the steps and the example of PBX. The proto offspring 1 mimics the absolute order substructures from the parent 1 and then imitates the relative sequence order of the remaining substructures from the parent 2 and vice versa.

For this problem, the chromosomes are sequenced subsets of jobs. The diversity is maintained by ancestor replacement. If a new candidate is better than its ancestors it is used to replace one of its own parents. In this study, the local search is also applied to the new candidates with improvement. The swapping and insertion operations are randomly applied to the candidates until the candidates are no longer improved. The pseudo code of GA is as follows:

Step 1 Randomly generate the population.
Step 2 Evaluate the population.
Step 3 Perform crossover and mutation. If the newly generated candidate is better than its ancestors, then perform the local search until the candidate is no longer improved.
Step 4 Repeat Step 3 until the maximum number of generation is reached.

Although the encoded solution of GA is a full set of the jobs in the pool, the evaluation process considers only the accepted orders. The evaluation process not only evaluates the orders sequence, but also re-sorts the orders sequences to separate the accepted and rejected orders as illustrated in the Figure 2. The sequence of the accepted orders is kept in the accepted pool while the remaining orders are kept in the rejected pool. The candidate solution is re-sorted by concatenating the accepted pool with the rejected pool.

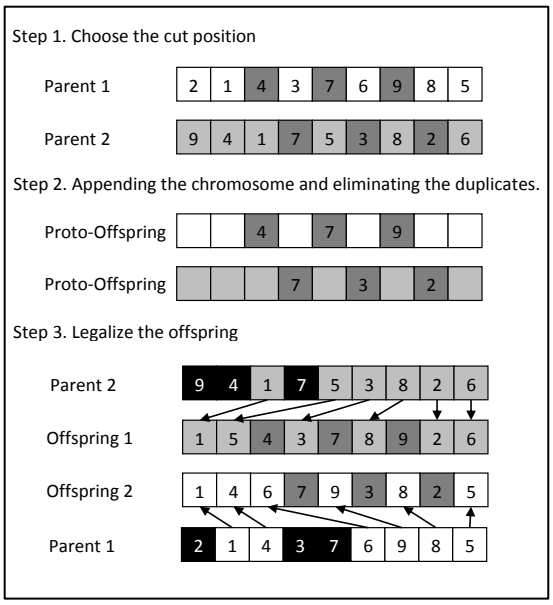

Figure 1. Position-based crossover (PBX).

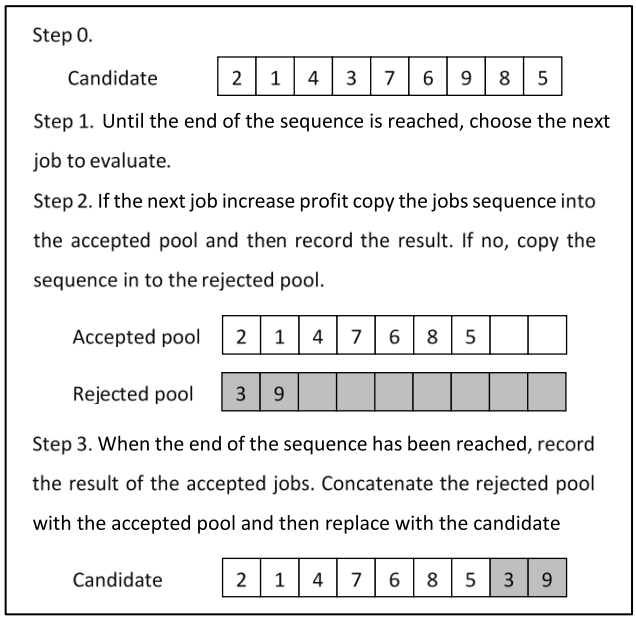

Figure 2. Evaluation with cutting off.

Even though, GA and NB-COIN are in the same group of evolutionary algorithms, however, the evaluation process and the updating process of NB-COIN for the order acceptance are slightly different. GA needs to maintain the genetic materials, therefore the whole set of orders need to be maintained. However, NB-COIN can reproduce the missing sequences by itself. In addition, the sequences of the rejected pool are considered to be the useless information, therefore, NB-COIN only updates the models from the accepted sequences of orders. Consequently the evaluation process does not need to concatenate the rejected pool with the accepted pool. The evaluation processes in the figure 2 simply use the accepted pool as the candidate for the NB-COIN.

2.3 *Test Problems and Experimental Design*

A list of products and their profit per piece was randomly generated. The generated profits are ranged between 5 to 15 currency units per piece. Then these profit attributes were used to generate the capacity utilization for each product such that producing the least profitable product would utilize the most balance capacity in each process, while the random time were added according to their profits. The capacities used by each processes are ranged between 0.1 to 1 pieces per minute.

The ten problems of size 50, 75 and 100 were also randomly generated according to the products and their profits such that the less profitable products have more chance to be demanded. Each order was generated from a log-normal distribution with an underlying normal distribution with mean 0 and standard deviation 1. The quantity for each order was randomly generated using the range between 1×1000 pieces and 12×1000 pieces. Each product has to be processed through 5 parallel production units which mean that there are totally 5 processes × 5 parallel machines for each process. The maximum capacity was set to two weeks. The due dates of each order were generated from a uniform distribution plus calculated lead-time for each of the order. These parameters were imitated from the existent manufactures in Thailand. Therefore, the wage penalty for this problem was set to 300 baht per worker per one production unit per day.

To compare the results, both NB-COIN and GA were given the same population size and maximum number of generations which are equal to the problem size × 2. The probabilities of crossover and mutation of GA are equal to 0.8 and 0.2 respectively. The learning step, *k*, of NB-COIN is 0.05. The selection pressure of GA is 50% of the whole population, while NB-COIN uses 25% of the top ranks for rewards and 25% of the bottom ranks for punishment. Test programs were coded in Lazarus and ran on OS X 10.4 on Intel Pentium Core i5 2.50 GHz processor with 4 GB of RAM.

3 RESULTS

Table 1. Performance of NB-COIN vs. GA in order acceptance with multi-process capacity balancing problem.

Problem size	NB-COIN		GA	
	Util.	Profit	Util.	Profit
50 orders	74.7%	18,5605	54.4%	14,4370
75 orders	80.7%	19,4074	57.5%	15,2562
100 orders	85.5%	21,0095	61.6%	16,1686

The performances of NB-COIN and GA are compared in terms of profit and capacity utilization. The performances are compared using the actual profit averaged from each of the best solutions out of ten runs. The capacity utilization is the wage penalty already deducted from the actual profit. The performance of NB-COIN to select from 50 orders are far better than GA that selected from 100 orders. The explanation is that the generated test problems were design such that the lowest profitable product utilizes the most balanced capacity. On the other hand, the most profitable product leaves more capacity leftover. The greedy profit maximization would results in the worse capacity utilization. NB-COIN is the algorithm that is good in solving multimodal and multi-objective problems (Waiyapara et al. 2013) as it tries to maintain the entire good substructures in order to recombine them.

4 CONCLUSION

This paper presents the application of NB-COIN to solve the order acceptance problem with multi-process capacity. The results show that NB-COIN is far better than GA for both profit and capacity utilization.

REFERENCES

Kleywegt, A.J. & Papastavrou, J.D. 2001. *The dynamic and stochastic knapsack problem with random sized items*. Operations Research 49 (1): 26 – 41.

Senju, S. & Toyoda, Y. 1968. *An approach to linear programming with 0 – 1 variables*. Management Science 15 (4), B196 – B207.

Slotnick, S.A. 2011. *Order acceptance and scheduling: A taxonomy and review*, European Journal of Operational Research 210(3): 527-536

Syswerda. 1991. A Handbook of Genetic Algorithms. *Schedule Optimization Using Genetic Algorithms*

Waiyapara, K. et al. 2013. *Solving Sudoku Puzzles with Node Based Coincidence Algorithm*; Proc. of International Joint Conference on Computer Science and Software Engineering (JCSSE 2013)

Wattanapornprom, W. et al. 2013. *Application of Estimation of Distribution Algorithms for Solving Order Acceptance with Weighted Tardiness Problems*; Proc. of IEEE International Conference on Industrial Engineering and Engineering Management (IEEM13-P-0588).