Application of Node Based Estimation of Distribution Algorithms for Solving Order Acceptance with Capacity Balancing Problems by Trading Off between Over Capacity and Under Capacity Utilization

Abstract – **Order acceptance with capacity balance problems require trading off between over- and under capacity utilization in order to gain more profits. This research proposes a new over- and under capacity tradeoff order acceptance model and propose adaptations of node based estimation of distribution algorithm to solve the order acceptance decisions in multi-process environments. The results show that node based coincidence algorithm is a potential algorithm which can maximize both profit and can maximize the capacity used at the same time.**

Keywords – **order acceptance, capacity balancing, combinatorial optimization, estimation of distribution algorithm, genetic algorithm, node based coincidence algorithm, node histogram based sampling algorithm**

I. INTRODUCTION

The order acceptance and rejection decision problems have gained increasing attention over the past few decades. Manufacturers favor among the orders that they accept for processing for a variety of reasons including market focuses, competitive advantages and capacity limitations. While surplus of orders might be hailed by a manufacturing or service facility, demand that exceeds capacity brings with it some hard choices. There is an important trade-off between the profit-enhancing revenue of an order, and the costs of capacity that it may distract with the other jobs. In addition, late delivery of some orders may result in penalties such as reduction of revenues and long-term losses of trust and market share. In a competitive market, the importance of on-time delivery may make it cost- and profit-effective to reject some orders [1][2].

Order acceptance and scheduling (OAS) problem 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 of order sequencing can result in difference profit level[3][4]. This problem is divided into two research group, which are order acceptance (OA) and order rejection (OR). Whether acceptance or rejection, both group of research have one similar key issue which is for solving the selecting and sequencing order problem to meet the production capacity constraint with the best profit result. From the order rejection perspective, the main objective is to minimize penalty of rejection such as minimizing make span, completion time, machine capacity cost, etc. On the other hand, from order acceptance perspectives, the main objective is rather on the maximization of profits, such as maximize profit, capacity utilization, etc. [1]

The penalties of rejection in OR problems indirectly reflect the profit in the OA problems. For example, the make span minimizing in OR can result in increasing of the capacity utilization in OA. Accordingly, when the capacity cost is fixed, it is not necessary to calculate both penalty and profit at the same time. However, in many situations, manufacturers could not raise the capacity level to support more orders in order to gain higher profits, they need to work overtime (OT) which inevitably require higher production cost. Nevertheless, it is not necessary for the manufactures to utilize all of the OT capacity that they have in order to get the best profits, especially in multi-product with multi-process production lines, which is not easy to re-balance the production lines. Consequently, it is necessary to choose the series of orders that are not only profitable but also balance in capacity utilization.

Trading off between underutilization and overutilization is not a new concept in OAS problem, it was used in the upstream of order acceptance process by microeconomic theories for pricing, making decision in capacity reallocation and selecting the profitable orders under the time constraint [2]. However, there is no existing research that downstream applies about trading off between under capacity and over capacity utilization. This is the first research that use two conflicting capacity constraint as the objectives in OAS problem to select and sequence the orders to maximize the profits while maintaining the balance of the production capacities at the same time.

This article introduces the application of node-based estimation of distribution algorithms (EDA) [5][6] for solving the order acceptance with capacity balancing problem. The contribution of this work is to demonstrate new approaches to the order acceptance problem that compete successfully with previously purposed genetic algorithm especially in larger problems.

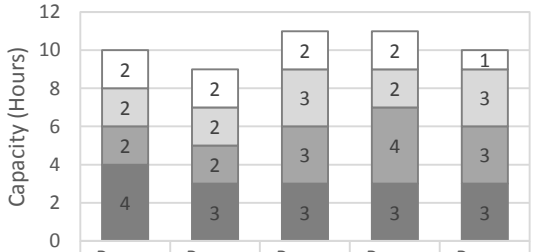
The remaining sections of this paper are organized as follows. The problem model and the procedures are introduced in Section II. The results are discussed in Section III. Finally, Section IV concludes this work.

II. METHODOLOGY

A. Order Acceptance Model

In practice, most production lines are balanced such that they are suitable with standard products that have higher demand. Each process utilize the balanced capacity such that similar kind of product can be produce smoothly. Unfortunately, these production lines usually lose their strength when facing unusual products that are not frequently ordered, it results in inefficacy use of working capacity. Therefore, the prices of lower demand products become higher in order to compensate the unusual production times and materials in the stocks. The unused leftover capacities become inevitable costs that the manufacturers have to spend. However, the manufactures can choose to use the OT capacities in some processes to accept more profitable orders and to avoid the under capacities penalty.

The order acceptance model in this research consider not only the under capacity utilizations but also allow the over capacity utilizations in multi-process environment. This model is on an assumption that the employees can work totally 8 hours per day and can have extra 2 hours OT. In addition, there is no waiting time involved in the model. Figure 1 illustrates the under and over capacity utilization of orders. Each order needs to be manufactured in 5 difference processes which required difference time capacities. The total capacity for each process is 8 hours plus addition 2 extra hours. The capacity plan accumulate the working capacities of each order and its processes. Form figure 1, the order A, B and C utilize up to 9 hours in the process 3, 4 and 5, while in process 3, the capacity is 1 hour wasted, yet the order D should be rejected as it overuse the total capacity for the process 3 and 4. It can be clearly seen that without the order C and D, the utilization of this plan would be worse as it wastes totally 10 hours capacity in five processes. The capacity plan could be better if order C is rejected as there is no leftover capacity and the capacities are not over used.



	Proces s 1	Proces s 2	Proces s 3	Proces s 4	Proces s 5
Order D	2	2	2	2	1
Order C	2	2	3	2	3
Order B	2	2	3	4	3
Order A	4	3	3	3	3

Fig. 1. The under and over capacity utilization of orders.

The order of set $i = (1,2,\cdots,i)$, where i is one of the k product type and profit per unit is P_{ik} . Each order will be processed through set $N = (1,2,\cdots,n)$, production unit. An order i is said to be early if finish period t is equal or less than due date d, t-$D_i \leqslant 0$ and overdue if more than the due date t-$D_i > 0$. Product k utilizes capacity CTP_{nt} as e_{ijknt} per unit, so the orders will occupy total production capacity $\sum_i e_k q_{ikt}$ for $\forall t$. Each order for an item consist of several processes. RT_n and OT_n is the regular working time and overtime allowed in a day. The model can be defined as follow:

Capacity Constraint

RT_n	Total capacity of workstation n
OT_n	Total overtime 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 consume 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_{1n}	Cost rate of stretching the production capacity at Workstation n
α_n	Cost Rate of not using the whole capacity at Workstation n
d_{nt}^+	Amount of over capacity production at workstation n in period t
d_{nt}^-	Amount of under capacity production at workstation n in period t

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 in time period t
	=	0, otherwise

$$\text{Maximize } Z = \sum_t \sum_i \sum_k p_{ik} q_{ikt} R_{ik} - (\sum_t \sum_n \alpha_{1n} d_{nt}^+ + \alpha_{2n} d_{nt}^-) g_n$$

Profit − Minimizing the capacity variation (1)

Subject to

Workstation-level activities Constraint

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

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

$$d_{nt}^+ = \sum_t \sum_n e_{ijk} q_{ikt} f_{iktn} - RT_n \quad \forall n \quad (4)$$

Order-level activities Constraint

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

$$f_{kij}q_{ikt} \leq e_{kij}q_{ikt}F_{iktn} \qquad \forall i,j,k,n,t \qquad (6)$$

$$\sum_n tF_{i|j|knt} \leq D_i R_{ik} \qquad \forall i,k,t \qquad (7)$$

$$\sum_n \sum_t f_{kj(i-1)}q_{ikt'} + \sum_n f_{ikt}q_{ikt} \geq \sum_n e_{kj(i-1)}q_{ikt}F_{iktn} \qquad \forall i/\{1\},k,t \qquad (8)$$

Binary and non-negativity Constraint

$$R_{ik} = 0or1 \qquad \forall i,k \qquad (9)$$

$$F_{ijknt} = 0or1 \qquad \forall i,j,k,n,t \qquad (10)$$

$$f_{ijknt} \geq 0 \qquad \forall i,j,k,n,t \qquad (11)$$

The key objective is to maximize the overall profit. The model helps to unify two decisions: which orders to accept and how much capacity is required of each resource in order to complete an accepted order. The secondary research objective is to balance the usage of capacity in production lines by tradeoff the wage penalty between over and under capacity utilization, thus this problem is considered to be a three objectives optimization problem. However, the three objectives are bind into one single objective. The objective function consists of three parts; (i) to maximize the total profit (ii) to minimize the residual working capacity and (iii) to minimize the OT capacity. Generally speaking, the objective is to choose the set and sequence of the profitable orders using as much working capacity as well as less OT as possible. According to this term the leftover available capacities have some certain penalty cost while the OT usages also cause extra cost. The first set of constraints is established to ensure that the whole capacity of plant is not violated. Constraint (3) and (4) were set to calculate the penalty of not using the whole capacity and stretching the production capacity. Constraints (5) and (6) set the F_{ijkrt} decision variables to either 1 or 0. The F_{ijkrt} variables are the indicator variable; they take a value of 1 when $f_{ijkrt} > 0$ indicating that job *j* of item *i* is being processed on resource *k* in time period *t*, otherwise they take a value of *0*. The F_{ijkrt} variables are used to ensure the precedence relationship. The constraint set (7) ensures that when an order for an item is accepted, the completion time of the last job of that order does not exceed the order due date. The constraint sets (8) impose precedence restrictions, to ensure that job *j* of item *i* can be processed in period t only after completing job *j* of item *i-1*. The constraints (9) and (10) are the binary constraints and constraints (11) to negativity constraints.

B. Solution Procedures

This work compares the result of GA with two node based EDAs including NHBSA, NB-COIN. From preliminarily study, the results of the EDAs for solving OA in single machine are too far better than GA and its benchmarks [6], therefore this paper only compare the results with GA.

1. Genetic Algorithm

The first procedure is an ordering GA with Position-based crossover (PBX) [7] which preserve not only absolute order substructures but also preserve relative order substructures from two parents. Figure 2 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 sample solutions, that is, sequenced subsets of jobs. The diversity is maintained by ancestor replacement. If 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 are 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.

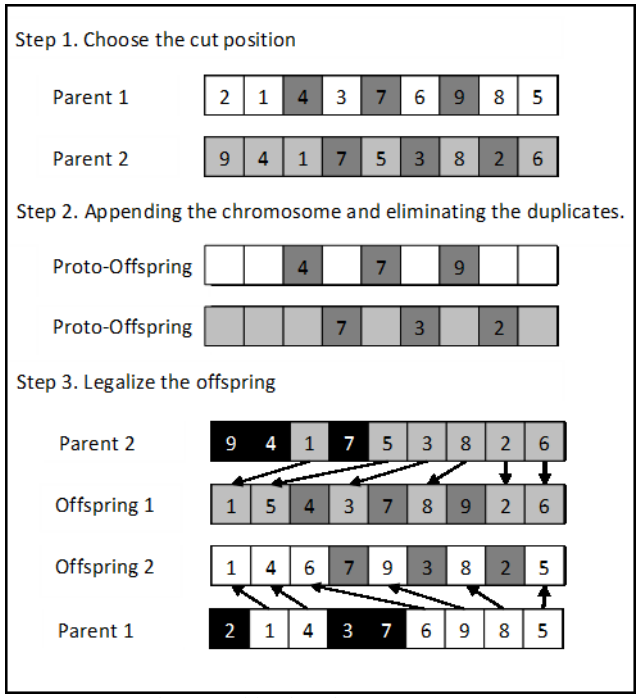


Fig. 2. Position-based crossover (PBX).

Although, the encoded solution of GA is a full set of the jobs in the pool. However, the evaluation process considers only the accepted jobs. The evaluation process

does not only evaluate the jobs sequence, but also re-sorts the jobs sequences to separate the accepted and rejected jobs as illustrated in the figure 3. The sequence of the accepted jobs are kept in the accepted pool while the remaining jobs are kept in the rejected pool. The candidate solution is re-sorted by concatenating the accepted pool with the rejected pool.

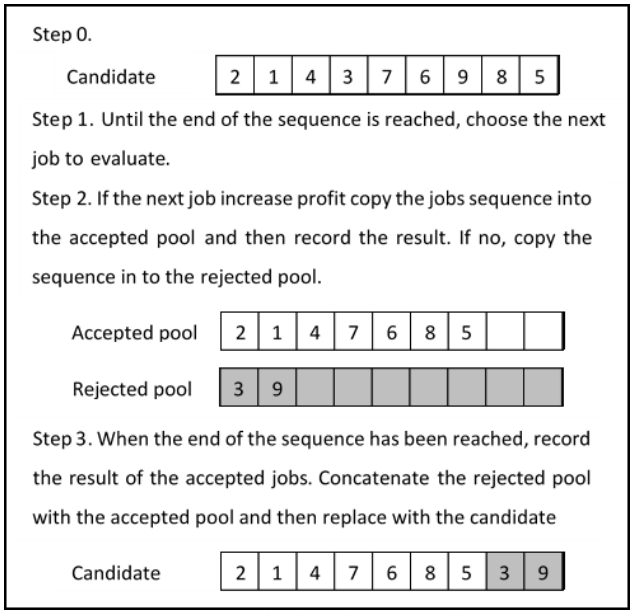


Fig. 3. Evaluation with cutting off.

2. Estimation of Distribution Algorithms

The EDAs used in this research are Node Histogram Based Sampling Algorithm (NHBSA) and Node Based Coincidence Algorithm (NB-COIN). They generate solution strings in sequences, ensuring that only valid permutations are sampled. The differences of these two node based EDAs are the learning methods. NHBSA belongs to the ad hoc learning methods, while NB-COIN belongs to the incremental learning methods. The pseudo code of EDAs are 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.

Although, GA and EDAs are in the same group of evolutionary algorithms, however, the evaluation process and the updating process of EDAs for the order acceptance are slightly different. GA needs to maintain the genetic materials, therefore the whole set of jobs need to be maintained. However, EDAs can reproduce the missing sequences by themselves, in addition, the sequences of the rejected pool are considered to be the useless information, therefore, EDAs only update the models from the accepted sequences of jobs. Consequently the evaluation process doesn't 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 EDAs.

2.1. Node Histogram Based Sampling Algorithm

NHBSA was proposed by Tsutsui in 2006.[9] It utilizes Node Histogram Matrix (NHM) to learn the mutual information of absolute position. Matrix $NHM = [h_{ij}]$, where $h_{ij} = P(\sigma_i = j)$ and $i, j \in \{1, 2, \ldots, n\}$. Hence, h_{ij} represents the probability of the index j to be in the i-th position in the selected individuals. h_{ij} is added to a ε value denoted as

$$\varepsilon = \frac{N}{n} B_{ratio} \tag{12}$$

to control the pressure in sampling and to avoid individuals with probability 0.

2.4. Node Based Coincidence Algorithm

NB-COIN [5] is a variation of coincidence algorithm (COIN) [8] proposed by Wattanapornprom and Chongstitvatana in 2013. It learns substructures from absolute positions, similar to NHBSA. The matrix H_{xy} represents the probability of y found in the absolute position x. The update equation is

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{(n)}\left(r_{xy}(t+1) - p_{xy}(t+1)\right) + \frac{k}{(n)^2}\left(\sum_{j=1}^{n} p_{xj}(t+1) - \sum_{j=1}^{n} r_{xj}(t+1)\right), \tag{13}$$

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 $j \neq y$.

NB-COIN has a special characteristic, that is, it not only learns from the better candidates but also learns from the poorer candidates as well. 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.

C. Test Problems and Experimental Design

The list of products and their profit per piece were 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 used for each product such that producing the least profitable product would utilize the most balance capacity in each working 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 orders were also randomly generated according to the product and their profits such that the less profitable products have more chance to be ordered. Each order was generated from a log-normal distribution with an underlying normal distribution with mean 0 and standard deviation 1. The quantities for each order were 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 means that there are totally 5 processes × 5 parallel machines for each processes. The maximum capacity were set to two weeks. Each working day has eight working hours plus extra two OT hours. 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 from Thailand. Therefore, the wage penalty for this problem was set to 300 baht and OT cost was set to 450 baht per worker per one production unit per day.

To compare the results, each algorithm was given the same population sizes 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 steps, k, of NB-COIN is 0.05. The bias ratio, B_{ratio}, of NHBSA is 0.005. The selection pressure of GA and NHBSA 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 Mac OS 10.4 on Intel Pentium Core i5 2.50 GHz processor with 4 GB of RAM.

The performances of GA and EDAs are compared in terms of average of the best actual profits and percentage of over and under capacity utilization.

III. RESULTS

Table I compares the performance of the benchmark algorithms. The capacity utilization is the wage penalty already deducted from the actual profit. Figure 4 and 5 compare the gained profit and wage penalty in a problem with 50 orders. Since the solutions of NB-COIN and NHBSA were generated from generation to generation, without keeping the elitists, the best solution in each generation does not necessary increasing.

From the table I, it can be clearly seen that the node based EDAs yield better results compared to GA with local search. NB-COIN can find the best solution in every benchmark as it can seek for sequences of subset order which gain the best profits. In addition, NB-COIN can utilize the full capacity of the working hours. It can also find the set of profitable orders which could utilize more OT capacity. However, from figure 4, NHBSA can find better solutions than NB-COIN in the very beginning generation. It can find competitive solutions with less number of function evaluation. Unfortunately, NHBSA was trapped in some pitfalls whereas it cannot combine the solutions with higher profits such that satisfy the orders due dates and capacity utilizations. The generated test problems were design such that the lowest profitable product utilize the most balanced capacity. On the other hand, the most

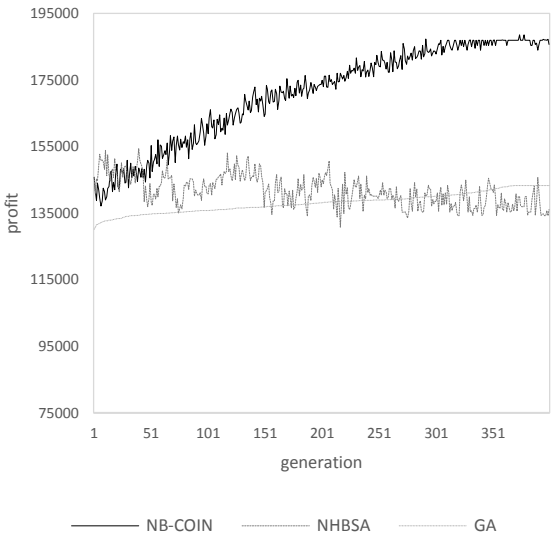


Fig. 4. Performance of NB-COIN, NHBSA and GA in maximizing the profit in the order acceptance with capacity balancing problems.

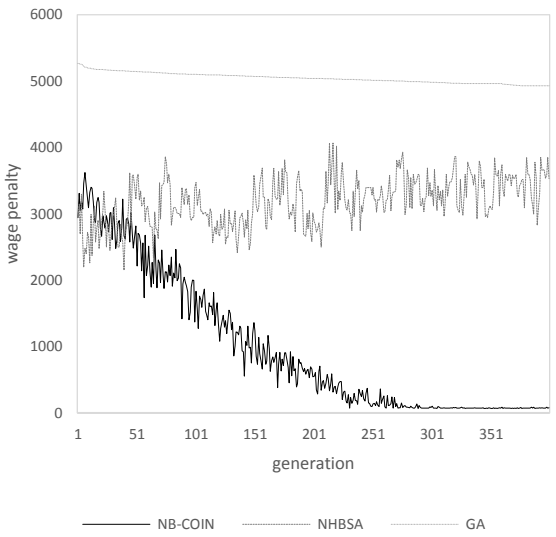


Fig. 5. Performance of NB-COIN, NHBSA and GA in minimizing the wage penalty in the order acceptance with capacity balancing problems.

profitable product leave more capacity leftover. The greedier profit maximization would results in the worse capacity utilization.

The unique characteristic of NB-COIN is that it not only learn from the good solutions, but also learn from the poor solutions. This characteristic enables NB-COIN to

find not only the good quality solutions, but also the diverse of solutions [9], which is the fundamental characteristic to solve multimodal and multi-objective problems. The incremental learning method enables NB-COIN to maintain the high potential substructure to be composed. NB-COIN simply estimated the sequence of the accepted orders in which the good sequences of orders may be conflict with each other.

IV. CONCLUSION

Node based EDAs have proved themselves as the competitive procedure in solving the combinatorial problems. This article propose an innovative adaptation of node based EDAs to solve the order acceptance problem in which the solution string are sub-sequences of the all given jobs. From the empirical study, NB-COIN, which is a node based incremental learning method, is a competitive algorithm to solve this problem.

TABLE I
PERFORMANCE OF GA, NHBSA AND NB-COIN IN ORDER ACCEPTANCE WITH CAPACITY BALANCING PROBLEMS

problem size	GA+LS			NHBSA			NB-COIN		
	profit (baht)	% under capacity	% over capacity	profit (baht)	% under capacity	% over capacity	profit (baht)	% under capacity	% over capacity
50 orders	143303	12.8	0	154378	9.56	0	188586	0	3.49
75 orders	153556	12.2	0	163649	8.43	0	195075	0	5.67
100 orders	164657	11.4	0	178939	7.34	0	213732	0	6.78

REFERENCES

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

[2] F.H. de B. Harris ,J.P. Pinder "A revenue management approach to demand management and order booking in assemble-to-order manufacturing" , *European Journal of Operations Management 13(4), pp. 299–309, 1995*

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

[4] X.L. Zhong, J.W. Ou, G.Q. Wang "Order acceptance and scheduling with machine availability constraints" *European Journal of Operational Research 232, pp.435–441,2014*

[5] K. Waiyapara, W. Wattanapornprom, P. Chongstitvatana, "Solving Sudoku Puzzles with Node Based Coincidence Algorithm" *in Proc. of International Joint Conference on Computer Science and Software Engineering (JCSSE 2013), 2013.*

[6] W .Wattanapornprom, T.K. Li, W.Wattanapornprom ,P.Chongstitvatana "Application of Estimation of Distribution Algorithms for Solving Order Acceptance with Weighted Tardiness Problems " *Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on*. IEEE, 2011.

[7] G. Syswerda, "Schedule Optimization Using Genetic Algorithms" in *A Handbook of Genetic Algorithms*, 1991.

[8] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multiobjective combinatorial optimisation with coincidence algorithm," in *Proc. of IEEE Congress on Evolutionary Computation (IEEE CEC 2009)*, pp. 1675-82, 2009.

[9] K Waiyapara, P Chongstitvatana, "Solving Multimodal Problems by Coincidence Algorithm." in *Proc. of International Joint Conference on Computer Science and Software Engineering (JCSSE 2012)*, 2012.