Simulated Annealing for Capacity Planning of Reentrant Production Systems

Geun-Cheol Lee
College of Business Administration,
Konkuk University, Seoul, Korea
Tel: (+82) 2-450-4100, Email: gclee@konkuk.ac.kr

Jeong Man Hong
LG CNS, Seoul, Korea
Tel: (+82) 2-7684-0627, Email: jmhong@lgcns.co.kr

Jung Ug Kim
Entrue Consulting,
LG CNS, Seoul, Korea
Tel: (+82) 2-2099-3060, Email: jugkim@lgcns.co.kr

Seong-Hoon Choi †
Department of Management Engineering
Sangmyong University, Cheonan, Korea
Tel: (+82) 43-345-2345, Email: shchoi@smu.ac.kr

Abstract. In this study, we propose a simulated annealing (SA) method to solve a problem of the capacity planning of production systems where the reentrant characteristics exist. Reentrant processes are commonly shown in the recent top of the edge electric product manufacturing, such as semiconductor, TFT-LCD, and so on. In the considered capacity planning problem, the number of machines at each stage in the production system is to be decided in such a way that the throughput of the system is to be maximized. Because, the considered production systems are too complex to obtain their throughput values via analytic methods, simulation is used. The constraint of the problem is the capital budget so that we could not configure the machines as many as we want. Generally, capacity planning problems do not require fast computation times, thus, we use one of the well-known metaheuristics, i.e., SA, to solve the problem, which known as a metaheuristic giving the effective solution rather relatively short computation time. To show the SA's performance, we carried out the computational experiments and compare with the existing benchmark heuristic methods.

Keywords: Capacity Planning, Reentrant System, Simulated Annealing, Simulation

1. INTRODUCTION

In this study, we consider a capacity planning problem of reentrant hybrid flowshops, which are composed of serial stages where multiple parallel machines exist in each stage. Furthermore, some operations are required to be performed duplicated, which means some parts have to re-enter certain range of stages one more times. Such features are common in the recent electronic product manufacturing, such as semiconductor, TFT-LCD, PCB, and so on.

In the considered capacity planning problem, the number of machines at each stage is to be determined. In the state-of-the-art electronics manufacturing environments, capital investment of the machines is tremendously high so that such a decision affects the financial status of the corporates in the long run. Therefore, it could be a crucial decision which stage is to be the stage has to increase its capacity, that is, increasing the number of machines in the stage. Survey literatures on capacity planning in the high-tech industry can be found frequently. (Wu *et al.* 2005, Geng and Jiang 2009)

The rest of the paper is organized as follows. In the next section, the detail description of the problem is presented. In section 3, a simulated annealing (SA) method is proposed. In section 4, the computational experiments are carried out to evaluate the performance of the SA as well as the benchmark methods. In the last section, the conclusion and some remarks are stated.

2. PROBLEM DESCRIPTION

In this section, the considered problem is specifically described. As mentioned earlier, we decide the number of machines at each stage of the reentrant hybrid flowshop. The performance measure, i.e., the objective function, of the problem is the throughput rate, i.e., the production output volume per unit time. A throughput rate value of a production system cannot be obtained analytically, rather we need simulation of the system to measure the throughput rate of the system. We assume that the same machines are used in a certain stage, so that, the number of machine types and the number of stages in the shop is the same. Each type of the machine has a purchase cost and there are capital budget for purchasing the machines. Therefore we cannot buy the machine as much as we want.

This problem can be formulated as typical knapsack problem as follows (Lee and Choi 2011).

$$\text{Maximize} \quad T(x_1, x_2, \dots, x_K) \qquad (1)$$
$$\text{s.t.} \ \sum_{k=1}^{K} c_k x_k \le B \qquad (2)$$
$$x_k \ge 0 \ \text{ and integer, } \ k = 1, 2, \dots, K,$$

where x_k, the decision variable, is the number of machines at stage k, $T(x_1, x_2, \dots, x_K)$ is the function which returns the throughput rate value of the system with the machine configuration of x_k, c_k is the purchase cost of the machine at stage k, and B is the pre-given budget for purchasing the machines. K is the number of stages in the considered system. The problem we consider in this study is the same with the problem considered in Lee et al. (2015). In the formulation, B, c_k, K, are parameters and they are pre-given before solving the problem.

In the problem, we assume that orders arrive dynamically, so that the parts to be produced are unknown until the order arrivals. An order has information like, the product type, amount to be produced, due date and so on. Once an order arrives in the system, the order is divided in to multiple lots for proper transportation or production. Each lot is produced according to its process plan where the sequence of stages where the lot should visit is recorded. The number of the product types is pre-specified as *N*. The process plan of each product is given. In this study, it is assumed that some parts visit each stage only once, while other parts visit each stage twice. For the dispatching rule used in the system, we assume FIFO (First In First Out) is used at each stage of the system.

3. SIMULATED ANNEALING

In this study, we propose an SA to solve the considered problem. The problem of determining the number of machines does not need to be derived in a short period time because it is rather a strategic decision problem in the companies. The result would be more desirable if a better solution is obtained with longer solving times. In this paper, we use an SA, a typical meta-heuristic method to determine the number of machines, which is expected to be consuming long computation times.

3.1 Control Parameters

SA introduced by Kirkpatrick et al. (1983) has been used to solve a number of applications in combinatorial optimization. To implement an SA to the considered problem, the control parameters of SA should be adjusted, such as, searching features, and termination conditions. In this paper, we use the control method introduced the Johnson et al (1989). A degree of cooling temperature is set on TempFactor, the number of neighborhood generations at the same temperature is set on SizeFactor, and the termination is decided by MinPercent, respectively. Detailed description of the parameters are summarized in Table 1. In this study, the values of TempFactor, SizeFactor, and MinPercent are set to 0.85, 5, 2, respectively, after preliminary tests.

Table 1. SA Parameters (Johnson et al., 1989)

Parameters	Descriptions
TempFactor	The temperature is reduced by multiplying the by TempFactor
SizeFactor	The number of iterations at each temperature is (SizeFactor×neighborhood size)
MinPercent	The algorithm terminates when acceptance move ratio is less than MinPercent

3.2 Initial Solution

Every SA requires initial solution generation method. In the proposed SA, we use *Monotone Increase* (MI) method introduced in Lee and Choi (2011). In the MI method, the selection criterion is needed, which selects the most critical stage which needs to have additional capacity, and the criterion proposed in Lee et al. (2016) is used.

In the selection criterion, we could select a stage which has the largest workload of products per a machine to increase the throughput rate. The workload of products at a stage can be estimated by summation of processing times of all product types. Additionally, because some products visit some stages multiple times, the number of visits at each stage for each product should be considered. Moreover, the setup time can be also included. Thus, the selection criterion (Lee et al., 2016) can be summarized as follows:

$$k^* = argmax_k \left\{ \frac{\sum_{i=1}^{N} \{p_{ik} \cdot v_{ik} + s_{ik}\}}{x_k} \right\} \qquad (1)$$

In the equation, N is the number of product types, p_{ik}, s_{ik}, v_{ik} are the processing time, the setup time, and the number of visits of product type i at stage k, respectively. x_k is the current number of machines at stage k.

The procedure of MI method is as follows. In the procedure, B^r is the remaining budget.

Monotone Increase Procedure

Step 0. Initialize x_k for all k; $B^r = B - \sum_{k=1}^{K} c_k x_k$.

Step 1. Perform a pre-specified simulation run.

Step 2. Among k such that $B^r > c_k$, find k^*; If exists, goto Step 3, o/w STOP.

Step 3. $x_{k^*} = x_{k^*} + 1$ and $B^r = B^r - c_{k^*}$; Goto Step 1.

With the above procedure, we can get the initial solution of the problem. In the procedure, x_k can be initialized by the equation, $\lceil \lambda \cdot \bar{Q} \cdot \sum_{i=1}^{N} (v_{ik} \cdot p_{ik})/N \rceil$, which is introduced in Lee et al. (2016), where λ is the arrival rate of orders, and $\bar{Q}$ is the average order size, respectively..

3.3 Neighborhood Generation

In an SA, neighborhood generation should be done in a way that reflects the characteristics of the problem. In this study, we use three neighborhood generation schemes. First, we add one machine at arbitrarily selected stage; second, we deduct one machine at arbitrarily selected stage; third, we pick one machine at arbitrarily selected stage and move the machine at another arbitrarily selected stage. One of these three neighborhood generation schemes are selected arbitrarily and used for generating a neighborhood solution.

An infeasible solution can be generated by the first and the second neighborhood generation schemes. That is, the remaining budget can be negative after adding or swapping machines. In that case, the neighborhood generation process is repeated until a feasible solution is generated. In a certain moment of SA search, the remaining budget can be increased over an inappropriate level, which results in poor performance in terms of throughput rate. To prevent such phenomenon, we apply MI method to consume the over-remaining budget and add the capacity to the system.

The flowchart of the procedure of the proposed SA method in the study are summarized in Figure 1.

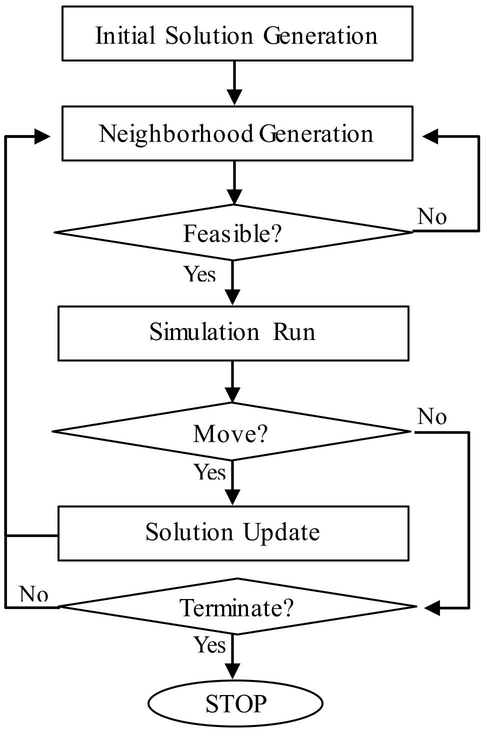

Figure 1. Flow chart of the SA

As you can see from the figure, the proposed SA in this paper starts with the initial solution generation. The initial solution is generated by the method presented in section 3.2. Then a neighborhood solution is generated by the method introduced in section 3.3. If the generated neighbor is infeasible, the neighborhood generation is repeated until the feasible one is obtained. After the feasibility check, the simulation is executed to derive the throughput rate value with the current machine configuration. Once the performance measure value, i.e., throughput rate value, is obtained, we check whether moving the solution to the newly generated neighbor. If the throughput rate is improved, the solution definitely moves to the neighborhood solution. Besides, even if the throughput rate value of the neighbor is decreased, the solution moves with a slight amount of the probability. The procedure is repeated until the termination condition is satisfied, that is, the ratio of the accepted moves among the neighborhood generation is less than the pre-specified MinPercent value.

4. COMPUTATIONAL EXPERIMENTS

To evaluate the performance of the proposed SA, the computational experiments are carried out. Particularly, simulation is needed to obtain the throughput rate of the production system with a certain combination of machines. Using simulation model to assess the performance of the complex production systems is common in the literature (Ponsignon and Mönch 2015). In this section, the description and the results of the test are presented.

4.1 Test Data

For the computational experiments, test problems are randomly generated. Among the various configurations of the problems, the number of stages and the number of product types are pre-specified. In the test, we set the number of stages as 10 and the number of product types as 10, respectively. For this configuration, ten instances are generated.

The performance of the capacity planning, i.e., the combination of the number of machines at all stages, is measured by throughput rate. In this study, the throughput rates are calculated by the number of completed lots per unit time (Kim et al, 1998), which can be obtained by dividing the number of completed lots during the valid simulation time by the corresponding simulation time, after the end of each simulation.

To validate the relative performance of the proposed SA, we introduce the existing method as the benchmark. In this study, we use the heuristic method proposed the Lee et al. (2016) as the benchmark, in which the same problem was considered.

4.2 Test Results

For ten problem instances, we solve each problem five times with the proposed SA, because every run of the SA results in different solution. Table 1 shows the overall test results. In the table, each value means the throughput rate value by the method. As you can see from the table, one result is obtained by the benchmark, whereas five results are obtained by the SA for each instance. Particularity, the best result of the SA, i.e., the result with the highest throughput rate value among five results by SA, is shown at the rightmost column.

Table 1. Overall test results

	Benchmark Heuristics	SA					Best SA
		1	2	3	4	5	
1	0.1781	0.1774	0.1723	0.1737	0.1734	0.1727	0.1774
2	0.1838	0.1874	0.1901	0.199	0.1988	0.1947	0.199
3	0.1733	0.1784	0.1745	0.1759	0.1759	0.1755	0.1784
4	0.2018	0.1962	0.1951	0.1987	0.1997	0.1966	0.1997
5	0.1639	0.1681	0.1669	0.165	0.1664	0.1653	0.1681
6	0.1716	0.1764	0.1742	0.1752	0.1721	0.1763	0.1764
7	0.1838	0.1835	0.1846	0.1792	0.1835	0.1829	0.1846
8	0.1576	0.1549	0.1545	0.1571	0.1517	0.1543	0.1571
9	0.176	0.1724	0.1691	0.1719	0.1684	0.1724	0.1724
10	0.1652	0.1648	0.1648	0.167	0.1651	0.1663	0.167
Avg.	0.1755			0.1756			0.1781

In table 1, the average throughput rate value of the benchmark (0.1755) is close to that of the SA (0.1756). Whereas, the average of the best SA (0.1781) is larger than that of the benchmark. However, the outperformance does not seem to be dominant. Next, the computation times of the tested methods are summarized in table 2.

Table 2. Computation times of the tested methods (in seconds)

	Benchmark Heuristics	SA				
		1	2	3	4	5
1	99.6	741.8	680.2	669.1	681.9	731.0
2	21.1	216.9	40.62	220.9	230.5	294.6
3	46.9	461.1	547.9	456.7	558.3	504.3
4	122.9	784.3	803.2	836.4	867.3	812.4
5	33.1	430.8	864.8	312.4	513.9	507.1
6	27.6	388.8	67.2	59.1	130.5	439.7
7	76.0	420.4	289.8	389.7	362.3	287.3
8	80.8	660.0	90.9	638.7	638.1	663.8
9	94.1	1159.4	1116.9	252.1	1183.6	1018.8
10	142.5	1125.1	1378.6	1336.3	1441.3	1326.3
Avg.	74.5			612.7		

As you can see from the table, the computation time of the SA is much larger than that of the benchmarks. The average value shows that the computation time of the SA is almost ten times longer than that of the benchmark. Considering the much longer computation time of the proposed SA, the performance of the proposed SA cannot be considered as good enough

5. CONCLUSIONS

In this study, we considered a problem of determining the number of machines in reentrant hybrid flowshops. In the problem, the proper number of machines at each stage of the system is to be determined to maximize the throughput rate of the system. To determine the appropriate number of machines of the system, we proposed an SA method. The proposed SA were compared with the existing method through the computational experiments. The results showed that the slight outperformance of the proposed method.

REFERENCES

Geng, N. and Jiang, Z., (2009) A review of strategic capacity planning for the semiconductor manufacturing industry, *International Journal of Production Research*, **47**, 3639-3655.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C., (1989) Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning, *Operations Research*, **37**, 865-892.

Kim, Y.-D., Lee, D.H., Kim, J.U., and Roh, H.K., (1998) A simulation study on lot release control, mask scheduling, and batch scheduling in semiconductor, *Journal of Manufacturing Systems*, **17**, 107-117.

Kirpatrick, S., Gelatt Jr. C.D., and Vecchi, M.P., (1983) Optimization by simulated annealing, *Science*, **220**, 671-680, 1983.

Lee, G.C., and Choi, S.H., (2011) A simulation study on capacity planning in hybrid flowshops for maximizing throughput under a budget constraint *Journal of the Korea Society for Simulation* **20**, 1-10.

Lee, G.C., Hong, J.M., Kim, J.U., and Choi, S.H., (2015) A study on capacity planning for reentrant hybrid flowshop, Proceedings of the Asia Pacific Industrial Engineering and Management Systems Conference 2015.

Lee, G.C., Hong, J.M., Kim, J.U., and Choi, S.H., (2016) A simulation study on capacity planning for reentrant hybrid flowshop, *Journal of the Korea Society for Simulation* **25**, 45-52.

Ponsignon, T. and Mönch, L. (2015) Simulation–based performance assessment of master planning approaches in semiconductor manufacturing, Omega, **46**, 21-35.

Wu, S. D., Erkoc, M., Karabuk, S. (2005) Managing capacity in the high-tech industry: a review of literature, *The Engineering Economist*, **50**, 125-158.