Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

STOCHASTIC CUSTOMER ORDER SCHEDULING USING SIMULATION-BASED GENETIC ALGORITHM

Xiaoyun Xu
Yaping Zhao
Haidong Li
Zihuan Zhou
Yanni Liu

Department of Industrial Engineering and Management
College of Engineering
Peking University
Beijing 100084, P. R. CHINA

ABSTRACT

This study considers a dynamic customer order scheduling problem in a stochastic setting. Customer orders arrive at the service station dynamically and each consists of multiple product types with random workloads. Each order will be processed by a set of non-identical parallel servers. The objective is to determine the optimal workload assignment policy that minimizes the long-run expected order cycle time. A simulation-based genetic algorithm, named SimGA, is proposed to solve the problem, and a computable lower bound is developed for performance evaluation. Numerical experiments are reported to evaluate the performance of SimGA against two well-known simulation optimization methods.

1 INTRODUCTION

This study considers a stochastic customer order scheduling problem on non-identical parallel servers. Customer orders arrive at a service station dynamically and each order consists of T product types with independent random workloads. The service station has a total of M servers working in parallel and a separate queue is formed in front of each server. Processing speeds are predetermined and heterogeneous across all product type and server combinations. The workload of any product type can be distributed arbitrarily and processed independently on each server. Processing within a given queue is on the First-In-First-Out (FIFO) basis and preemption of any kind is forbidden. If a server is busy, the workloads assigned to this particular server will wait in line until its turn to be processed. A customer order must be delivered as a single shipment after the entire workloads have been finished. The objective is to determine the workload assignment policy that minimizes the long-run expected order cycle time. The problem described above is illustrated in Figure 1.

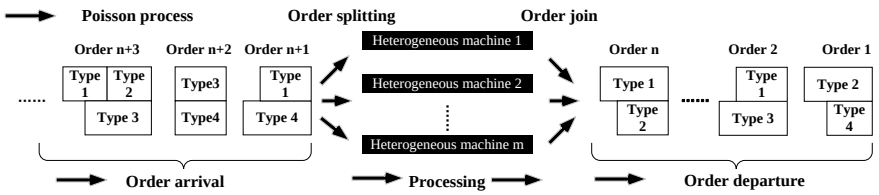

Figure 1: Dynamic customer order scheduling system: an illustration.

The order-oriented production described above is rather common in manufacturing industry. Examples include compound semiconductor wafer production (Kim et al. 2002), textile production (Serafini 1996) and printed circuit board manufacturing (Kim et al. 2004). In these applications, a customer order usually comprises products of different kinds and needs to be processed by a set of servers with different capacities. Products are then joined to form a complete order and pass on to the downstream stations. A typical example of this kind is the dicing process of compound semiconductor wafers. An incoming order usually contains thousands of wafers that belong to several different types. To process the wafers, silicon ingots are diced on machines based on the specifications of different wafers. When finished, the wafers are joined together and sent to the packing station. This wafer production example can be illustrated in Figure 2.

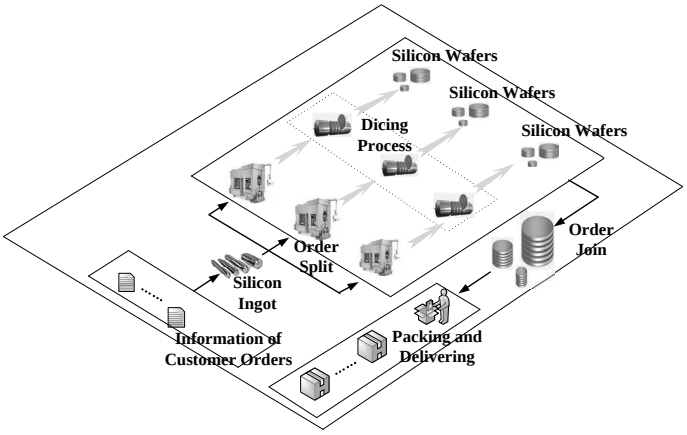

Figure 2: Compound semiconductor wafer production: an illustration.

The primary objective in this study is to reduce production cycle time of customer orders. It is well acknowledged in the literature that long cycle time reduces the accuracy of forecast, increases delayed order shipment, and eventually drives up the cost of the supply chain (Monden 2012, Stalk and Hout 1990). Shorter cycle time, on the other hand, could not only boost customer satisfaction, but also greatly facilitate the task of operational planning and production control (Stalk 1988). In modern manufacturing practice, cycle time reduction in customer order fulfilment process is widely regarded as the key element in ensuring a quick responsive supply chain (Ahmadi et al. 2005).

A large body of literature has emerged to address the problem of scheduling customer orders. However, most of the existing literature relies on the assumption of a static environment where the specifics of all incoming orders are known. The case of identical parallel server has been well explored (Blocher and Chhajed 1996, Yang and Posner 2005, Leung et al. 2006). For the non-identical server cases, Xu et al. (2013) consider the problem of scheduling customer order with type-splitting property. No additional result on the non-identical server case has been reported in the literature to the knowledge of the authors.

Much fewer works are found addressing the dynamic customer order scheduling problems. The problem is expected to be much more difficult as neither the arrivals nor the workloads of the incoming orders are known. Flatto and Hahn (1984) consider a case with two identical parallel servers and show that even performance evaluation presents significant challenges. For systems with more than two servers, the problem becomes even more difficult (Flatto 1985). While there has been simulation analysis on performance evaluation (see, e.g. Chen (2011)), the problem is only addressed for the case where assignment policy is pre-determined, and only approximation of the performance upper bound of the objective function is provided through multiple simulations.

Compared with the existing studies in the literature, the problem considered in this study has several interesting characteristics. First, unlike the problems in static settings, this study considers a dynamic problem where orders arrive according to a stochastic process. Second, the workload of each product type is random and not known until it arrives. Third, the servers in the station are non-identical, so the service

times of different product types may vary significantly from server to server. Finally, the problem seeks for optimum long-run expected cycle time, which may discourage transient improvements on certain orders.

The remainder of the paper is organized as follows: Section 2 formally defines the problem and lists the primary assumptions. A simulation-based genetic algorithm, named SimGA, is proposed in Section 3. The design of the algorithm intends to integrate the property of the proposed problem into the bio-inspired structure of genetic algorithm. A computable lower bound is developed in Section 4 to be used as the benchmark in the performance evaluation. Computational experiments are reported in Section 5 to evaluate the performance of SimGA against two other well-known simulation optimization algorithms. Section 6 concludes the paper and outlines the future research directions.

2 PROBLEM DESCRIPTION

The problem considered in this study can be formally defined as follows: Customer orders arrive at a service station according to a Poisson process with rate λ and inter-arrival times $\{a_n\}_{n=1}^{\infty}$ where n is order index. Each order consists of T product types, $\mathbb{T} = \{1,2,\ldots,T\}$, and each product type t has an independent random workload $\{q_n^t\}_{n=1}^{\infty}$ with expectation $E[q_n^t] < \infty$.

The service station consists of a set of M parallel servers, $\mathbb{M} = \{1,2,\ldots,M\}$. A separate queue with infinite buffer is formed in front of each server. Processing speeds are predetermined by speed matrix $V = [v_{mt}]_{M\times T}$ which is independent with workloads and inter-arrival times. The workload of any product type can be split arbitrarily and processed independently on each server. Upon the arrival of a customer order, the workload assignment policy $\delta = [\delta_{mt}]_{M\times T}$ is applied, where δ_{mt} is the portion of type t workload assigned to server m. The aggregation of the workloads within an order on each server forms the service time process $\{s_n^m(\delta)\}_{n=1}^{\infty}$.

Processing within a given queue is on the FIFO basis and preemption of any kind is not allowed. Therefore, the waiting times in queue, $\{w_n^m(\delta)\}_{n=1}^{\infty}$, and cycle times on each server, $\{CT_n^m(\delta)\}_{n=1}^{\infty}$, have the following relationships:

$$\begin{aligned} w_{n+1}^m(\delta) &= (w_n^m(\delta) + s_n^m(\delta) - a_{n+1})^+, \forall m \in \mathbb{M}, \\ CT_n^m(\delta) &= w_n^m(\delta) + s_n^m(\delta), \forall m \in \mathbb{M}. \end{aligned}$$

Moreover, a customer order will not leave the system until its entire workloads have been finished. That is, the cycle times of the n-th customer order under δ, $\{CT_n(\delta)\}_{n=1}^{\infty}$, is constrained by

$$CT_n(\delta) = \max_{1\le m\le M}\{CT_n^m(\delta)\}.$$

The objective is to determine the best workload assignment policy δ that will minimize the long-run expected order cycle time, namely,

$$\textbf{minimize } OBJ(\delta) = \lim_{n\to\infty} E\left[CT_n(\delta)\right] = \lim_{n\to\infty} E\left[\max_{1\le m\le M}\{CT_n^m(\delta)\}\right]. \quad (1)$$

3 DESIGN OF SIMULATION-BASED GENETIC ALGORITHM

The problem considered in this study is essentially a simulation optimization problem in a parallel machine setting. A genetic algorithm (GA) is suitable for such a problem because the number of possible workload allocations is huge, and the property of implicit parallelism in GA (Whitley 1994, Srinivasan 1998) is consistent with the underlying structure of the problem. Moreover, the cycle time reduction objective takes the form of mathematical expectation and thus needs to be evaluated through simulation. Due to the limitation of finite sampling in simulation, the potential inaccuracy in objective function evaluation may increase the probability of an algorithm being trapped in local optima. Literature (Salomon 1996) indicates that GA has an advantage of escaping from local optima. Therefore, GA is adopted in this study as the primary meta-heuristic framework.

There is a vast amount of literature on applications of GA to solve parallel server scheduling problems (Cheng et al. 1995, Chiu et al. 1999, Luu et al. 2002, Min and Cheng 2006). However, most of the existing studies require the processing speeds of the servers to be identical. A few attempts are reported in the literature in recent years to solve problems in non-identical parallel server settings (Van Hop and Nagarur 2004, Balin 2011), but these researches focus on jobs instead of customer orders, and the discussions are restricted to static domain where both the arrival and the workload are known in advance. To the knowledge of the authors, no attempt has been made in the existing literature to apply GA into stochastic customer order scheduling problems.

This study intends to propose a simulation-based genetic algorithm, named SimGA, to solve the stochastic customer order scheduling problem. The overall structure of SimGA is illustrated in Figure 3 and the detailed algorithm design is explained in the following subsections.

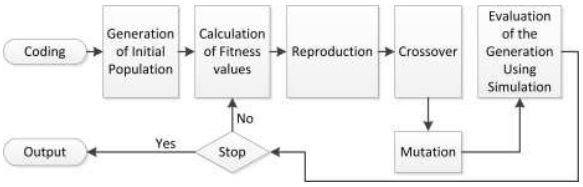

Figure 3: The structure of SimGA.

3.1 Coding

In this study, the decision variables are collected in the workload assignment policy matrix $\delta = [\delta_{mt}]_{M\times T}$. Column vectors of matrix δ are set as genes $g_1, g_2, \ldots, g_t, \ldots, g_T$, where $g_t \in R_+^M$, and each chromosome (solution) consists of T genes. Since each gene $g_t, t = 1, \ldots, T$ represents the workload assignment of type t, all genes are independent with each other. This independency contributes to the generation of feasible solutions when chromosomes are crossed.

The coding method used in this study is in direct contrast to the prevailing binary coding design. For the problem concerned, each gene is designed as a multi-dimensional vector with continuous elements denoting the allocation of a single product type. This design takes into consideration the fact that the feasible domain of decision matrix $\delta = [\delta_{mt}]_{M\times T}$ is continuous, and the variables within the same column of the decision matrix are correlated with each other. The dependencies in decision variables are kept within the gene so that the crossover operations between selected genes can be easily performed.

3.2 Initial Population

The initial population (chromosomes) constitute the first generation. In this study, as columns (genes) of the matrix δ are independent with each other, an initial chromosome can be obtained by generating a matrix (M rows, T columns) randomly and then normalizing each column vector to one. A total of N initial chromosomes $\delta_1, \delta_2, \ldots, \delta_n, \ldots, \delta_N$ can be obtained by repeating the same operation, and they constitute the initial population.

3.3 Fitness

Fitness is the performance evaluation of chromosomes (Zhou, Feng, and Han 2001). The higher the fitness value, the better the performance of the chromosome. At each iteration, members of the current population are selected for the mating pool on the basis of their fitness.

In this study, the well-known fitness function in Goldberg (1989) is adopted, that is, $F(n) = \alpha exp(-\beta \overline{OBJ(\delta_n)})$, where α and β are positive real numbers, and $\overline{OBJ(\delta_n)}$ is the mean objective function value of chromosome

δ_n. Since the problem considered in this study is a stochastic one, $\overline{OBJ(\delta_n)}$ is obtained through multiple computer simulations and the value is set as the sample mean.

3.4 Reproduction

Reproduction is the selection process in which the parent candidates are evaluated according to their fitness values. Parents with higher fitness values will have more chances to be chosen to produce the next generation.

In this study, the selection is conducted according to the roulette wheel selection method (Goldberg 1989). This method ranks the chromosomes based on their fitness function values, and then assigns these chromosomes a probability distribution in favor of good chromosomes. Selection is then made based on the distribution so as to obtain a better chance of producing good offsprings. The selection probability of chromosome δ_n is calculated as $P(n) = F(n)/\sum_{k=1}^{N} F(k), i = 1,2,\ldots,N$, where $F(n)$ is the fitness value of chromosome δ_n. Consequently, the cut points, $S(n)$ can be generated as $S(0) = 0$, $S(n) = P(1) + P(2) + \ldots + P(n), n = 1,2,\ldots,N$.

To select the parents, first generate N random real numbers ζ_s which are uniformly distributed between 0 and 1, that is, $\zeta_s \in U(0,1), s = 1,2,\ldots,N$. Then for each ζ_s, if $S(n-1) < \zeta_s < S(n)$, chromosome δ_n is selected as a parent for the next generation.

3.5 Crossover

Crossover is the process for the selected parents to generate offsprings. In the design of SimGA, the crossover operation generates a new chromosome by replacing one gene with a linear combination of two genes at the same position from a pair of parents. All the other genes remain unchanged and are inherited directly from their parents.

To be specific, assume chromosome δ_i and δ_j are crossed, and genes at position n are selected from chromosome chain randomly to be manipulated. Let g_i and g_j represent genes at position n of chromosome δ_i and δ_j respectively, and $p + q = 1$, where p is set to take the values of 0, $1/4$, $1/2$, $3/4$ and 1. Then, new genes are obtained through linear combinations of g_i and g_j, i.e. $(p \times g_i + q \times g_j)$. Replace gene g_i and g_j of chromosome δ_i and δ_j with the new genes, and ten new chromosomes (children) are generated. Then, two children with highest fitness values are selected to survive, and they constitute the new generation. The crossover operation is illustrated in Figure 4.

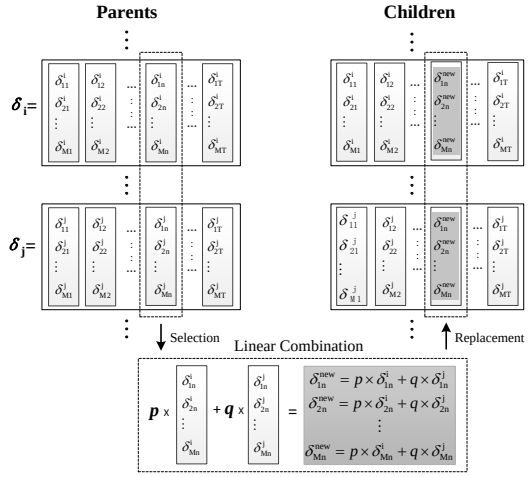

Figure 4: Crossover operation: an illustration.

3.6 Mutation

As oppose to crossover operation where new chromosomes are obtained through the combination of existing genes, mutation operation creates new chromosomes by introducing perturbations in genes, and is performed with pre-determined probability. Such operation diversifies the population and helps avoid getting stuck into local optimal solutions.

For a given customer order in the proposed problem, since cycle time is decided by the server with the latest completion time, it is apparent that the objective function value can always be improved if an appropriate amount of workload on that server is relocated to others. Therefore, in order to reduce the expected cycle time, it is preferable for servers to complete at the same time.

Inspired by the above observation, this study adopts the following mutation operation: Select a gene randomly from a chromosome that will mutate. Given the rest of the gene setting, the elements of the new gene (the workload allocation of the selected product type) are determined by solving a static parallel server makespan minimization problem, where makespan is defined as the longest total mean service time that is required to process a given order. This makespan objective will force the expected service times on all servers to be balanced and thus lead to reduction in per-order cycle time. Once the elements of the new gene are determined, replace the selected gene with the new one, and a new chromosome is obtained. The mutation step is illustrated in Figure 5.

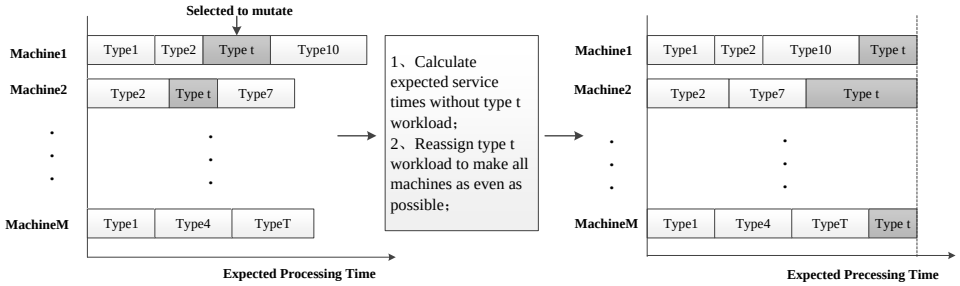

Figure 5: Mutation operation: an illustration.

4 A COMPUTABLE LOWER BOUND

To evaluate the proposed SimGA, it is preferable to compare the results of SimGA and the optimal assignment policy. However, due to the synchronization constraints on arrival and departure, evaluation of the objective function is hard even when workload assignment policy δ is pre-determined (Baccelli and Makowski 1989). In view of this difficulty, it is relevant to seek bounds to the objective. This section is devoted to the derivation of a computable lower bound (*LB*) through the adjustment of server speed matrix.

In order to establish the *LB*, it is convenient to first consider a new system that has the same arrival and service pattern as the original one, with the only difference being that the new system has a different server speed matrix. That is, for every product type, its speeds on all servers are replaced with the maximum one to process this type, namely $v^*_{mt} = v^*_t = \max_m v_{mt}$. Since every element of the new speed matrix is no less than that of the original one, for each δ, the objective function value of the new system is no greater than that of the original one, and therefore serves as a lower bound. This property holds true for every single feasible solution of the new system.

However, two major difficulties arise when the above lower bound is directly used as a performance measure. First, in order to establish the lower bound, it is required that the optimal assignment policy δ^{opt} of the original system is known. Since the optimal assignment policy δ^{opt} is in general difficult to obtain (Baccelli and Makowski 1989), this bound cannot be easily developed. Second, even when δ^{opt} of the original system is available, the numerical evaluation of the objective function (1) requires extensive

computer simulations (Baccelli et al. 1989, Chen 2011), which limits its applicability in manufacturing practices.

To remedy these difficulties, the optimal solution of the new system is found instead. The following Lemma addresses the relationship between the original and the new system:

Lemma 1 The optimal objective value of the new system is a lower bound to that of the original system.

Proof. Since the objective function value of the new system does not exceed that of the original system under any workload assignment policy δ, the optimal objective value of the new system must be no larger than any objective value (including the optimal one) of the original systems. □

Lemma 1 establishes a lower bound that is completely independent of assignment policy δ. Based on this result, it is sufficient to find the optimal assignment policy of the new system. The optimal policy of the new system is found by the following theorem:

Theorem 1 Workload assignment policy $\delta^{NEW} = [1/M]_{M\times T}$ is optimal to the new system.

Proof. Essentially, this assignment policy requires workload of each product type in every incoming order to be allocated evenly across all M servers. The proof begins by investigating an arbitrary realization of the random cycle time process $\{CT_n(\delta)\}_{n=1}^{\infty}$ in this study. Consider a finite start of order $n < \infty$ at the very beginning of a busy period and relabel that order as the first one. The following proof proceeds by induction.

For the first order, since all the servers are idle, there is no waiting time. Therefore, $CT_1^m(\delta) = s_1^m(\delta)$, $\forall m \in \mathbb{M}$ where $CT_1^m(\delta)$ and $s_1^m(\delta)$ are cycle time and total service time of the first order on server m, respectively. Given that $CT_1(\delta) = \max\limits_{1\le m\le M}\{CT_1^m(\delta)\}$, any assignment policy leading to different completion times among servers can be dominated by one with the same completion times, as this can be achieved by moving some workload on the server with longer completion time to the one with shorter. It is therefore obvious that the above workload assignment policy δ^{NEW} obtains the minimum $CT_1(\delta)$.

For the second order, if it arrives after the departure time of the first order, then it becomes the first case which has already been proved. If, however, it arrives before the first order finishes, then consider a single-order case where the workload of each product type is equal to the summation of the first and second orders. In this way, the single order is reduced to the first case whose cycle time can be minimized by schedules that evenly distribute the workload on all servers. This schedule can be identically constructed by applying it to both orders, resulting in that the assignment policy δ^{NEW} minimizes $[CT_1(\delta)+CT_2(\delta)]$.

Assume the induction hypothesis that $\sum_{i=1}^{n} CT_i(\delta)$ is minimized by the assignment policy δ^{NEW} holds for all $i \le n$. If the $(n+1)$-th order arrives after the n-th order finishes, it becomes the first case which has already been proved. Otherwise, merge the workloads of all $(n+1)$ orders as one single order. The cycle time of this single order will be minimized by a schedule that evenly distributes the workloads of that order. Without changing the completion time of the aggregated order, this schedule can be identically constructed by merging 1) the segment that includes all the workloads of the first n orders and 2) the workloads of the $(n+1)$-th order. Based on the induction hypothesis and the fact that the assignment policy δ^{NEW} also minimizes $CT_n(\delta)$, it becomes apparent that the hypothesis holds true for $n+1$.

The proof is then concluded provided the fact that $\sum_{i=1}^{n} CT_i(\delta)/n$ is an unbiased estimator of $OBJ(\delta)$ when $n \to \infty$. □

Theorem 1 finds the global optimum of the new system. Since it has been proved in Lemma 1 that the optimal objective of the new system serves as a lower bound to the original system, the explicit expression of this lower bound can be obtained, as shown in the following theorem:

Theorem 2 The objective $OBJ(\delta)$ has the following lower bound (LB):

$$LB = (\rho + \frac{\rho^2 + \lambda^2\sigma^2}{2(1-\rho)})/\lambda, \tag{2}$$

where ρ is the traffic intensity, λ is the arrival rate of the incoming customer orders, and σ is the standard deviation of service time on each server under $\delta^{NEW} = [1/M]_{M\times T}$.

Proof. This result is an immediate consequence of Theorem 1. The setting of $\delta^{NEW} = [1/M]_{M\times T}$ ensures that all servers share the identical workload and finish processing at the same time for each incoming order. Therefore, the new system can be reduced to the $M|G|1$ queue whose cycle time can be expressed as Equation (2). The rest of the proof is trivial and thus omitted for brevity. □

Theorem 2 provides an explicit and computable expression of the lower bound of the targeted objective (1). Since optimal solution of the original system is in general difficult to obtain (Baccelli and Makowski 1989), this deterministic lower bound is especially valuable as a benchmark when used to gauge the performance of the algorithms targeted to solve the proposed problem.

5 COMPUTATIONAL EXPERIMENT

5.1 Experiment Design

Computational experiments are conducted in order to evaluate the performance of the proposed simulation-based genetic algorithm (SimGA). Two other well-known simulation optimization algorithms, namely, Simultaneous Perturbation Stochastic Approximation (SPSA) and Simulated Annealing (SA), are also implemented for comparison purpose. Both SPSA and SA have been widely applied and demonstrated to be effective in solving simulation optimization problems (Spall 2005, Spall and Cristion 1994, Van Laarhoven et al. 1992, Kim et al. 2002, Melouk et al. 2004, Low 2005). In this computational study, the settings of primary parameters in SPSA and SA are based on works of Wang and Spall (2003) and Kim et al. (2002), respectively. For the proposed SimGA, the probability of mutation is set as 0.05 and the initial population is set as $1000 \times M$.

The scale of testing instances generated in this computational experiment is set according to the compound semiconductor manufacturing data used in Kim et al. (2002). With the reference of their settings, the following cases are considered in this study:

- Number of servers $M = 3, 6, 9, 12$.
- Number of product types $T = 5, 10, 15, 20, 25$.

Server speed v_{mt}'s are randomly generated from the normal distribution $N(\mu = 5, \sigma = 0.125)$ and take the absolute values. Mean workload of each type $E[q_n^t]$'s are randomly drawn from the normal distribution with $\mu = 0.5$, $\sigma = 0.025$ and take the absolute value. For each order, workload of type t is randomly generated from the exponential distribution with mean $E[q_n^t]$.

Computer simulation is performed to evaluate the long-run behavior of the system. Simulation begins with all servers being empty and idle, and runs until the 1000-th order is completed. Welch procedure (Welch 1983) is performed to eliminate the effect of the warm-up and cool-down periods, resulting in the statistics of 600 orders (200-th to 800-th) being used to approximate the long-run behavior. It also requires that all algorithms consume the same number of simulation runs so as to ensure a fair comparison.

For each of the three algorithms, there are $|\mathbb{M}| \times |\mathbb{T}| = 4 \times 5 = 20$ cases tested, and 30 independent replicates are generated randomly for each individual case. Therefore, a total of $4 \times 5 \times 30 \times 3 = 1800$ replicates are conducted in the entire computational experiment.

In view of the difficulty in obtaining the optimal assignment policy in this stochastic setting, the performances of algorithms are gauged with reference to the lower bound (LB) proposed in Theorem 2:

$$\text{Performance Gap} = \frac{OBJ(\delta) - LB}{LB} \times 100\%.$$

5.2 Numerical Results and Analysis

The numerical testing results are summarized in Table 1.

Table 1: The performance gaps of SimGA, SPSA and SA.

Problem Setting		Reference	Performance Gap (%)			
T	M		SimGA	SPSA	SA	Best Gap
5	3	*LB*	3.41	4.33	47.75	3.41
	6	*LB*	8.35	12.91	49.39	8.35
	9	*LB*	9.83	12.17	43.22	9.83
	12	*LB*	11.91	19.30	8.28	8.28
10	3	*LB*	4.33	8.12	40.33	4.33
	6	*LB*	5.80	10.90	6.93	5.80
	9	*LB*	8.29	10.60	46.11	8.29
	12	*LB*	9.99	19.23	43.12	9.99
15	3	*LB*	7.19	22.40	40.52	7.19
	6	*LB*	9.28	12.06	45.73	9.28
	9	*LB*	13.52	20.21	17.60	13.52
	12	*LB*	10.38	21.52	40.53	10.38
20	3	*LB*	8.25	26.36	24.85	8.25
	6	*LB*	11.32	34.37	17.79	11.32
	9	*LB*	10.11	18.42	65.09	10.11
	12	*LB*	14.12	17.58	59.41	14.12
25	3	*LB*	5.86	15.56	23.87	5.86
	6	*LB*	9.03	11.97	12.89	9.03
	9	*LB*	11.34	23.56	5.18	5.18
	12	*LB*	11.67	35.04	42.18	11.67
		Average Gap	9.20	17.83	34.04	8.71
		Max - Min	14.12−3.41 =10.71	35.04−4.33 =30.71	65.09−5.18 =59.91	14.12−3.41 =10.71

In this study, the proposed lower bound is used as the performance benchmark and its quality can be measured by the gap between *LB* and the best performing algorithm. This is shown in the "Best Gap" column of Table 1. Since the objective value achieved through the best performing algorithm can be regarded as an upper bound of the global optimum, the true gap between the optimal and *LB* must be less than the results shown in the "Best Gap" column. Table 1 shows that the average gap is 8.71%, and the size of the gap grows slowly as the problem instance becomes large. Given the consideration of sample mean variation caused by the randomness in simulation, it is reasonable to conclude that the *LB* proposed in Section 4 is relatively tight. Besides, as the difference of best gaps is less than 10.71%, it shows that *LB* is very robust to the change in problem scale. These evidences together suggest that the proposed *LB* is a good reference for performance evaluation. Since the gap is relevantly tight and easy to calculate, this lower bound can be used in practice as an indicator to evaluate the status of the current production.

For the three tested algorithms, it is plain to see from Table 1 that SimGA dominates the other two algorithms in nearly all the problem instances. The average performance gap of SimGA is 9.20% and the best case of 3.41% is achieved at the instance of $T = 5, M = 3$. The SPSA, on the other hand, has the mean gap of 17.83% which is nearly twice as large as that of the SimGA. The SA is the worst of the three and its mean gap is almost three times larger than the smallest one. As for the worst case analysis, the maximal gap of SimGA is bounded below 15%, while the other two exceed 35% and 65%, respectively. In

both the average and the worst case analysis of the testing instances, SimGA turns out to have the smallest performance gap and is the most robust algorithm among the three. These observations suggest that the proposed SimGA is capable of delivering sound and consistent results at different production scales, which is a highly desirable feature in manufacturing practice.

Another interesting observation is that, for almost all levels of product type T, the performance gap of SimGA becomes large as the number of servers M increases. This phenomenon may be attributed to two aspects of the SimGA design. One aspect is from the coding rules proposed in Section 3.1. Since the assignment policy of one type on all servers is coded as a "gene", an increase in number of servers will lead to more potential gene forms. Such growth of gene forms may overwhelm the linear growth of initial population applied ($1000 \times M$) and dramatically increase the difficulty in searching for the global optimum. The other aspect is the mutation operation. Since each time only one gene is selected randomly to mutate, the expected service times of all servers cannot be guaranteed to be equal. With the increase of servers, such unevenness in service time can be amplified as there is less capacity on average for the selected gene to compensate the difference. Due to these two aspects of the design, SimGA is more robust to the increase of product type T than the number of servers M. This observation suggests that SimGA is more suitable for manufacturing practices, as the number of stock keeping units (SKUs) almost always far exceeds the number of servers available on the factory floor.

6 CONCLUSION

This paper addresses a stochastic customer order scheduling problem in the non-identical parallel server system to minimize the long-run expected order cycle time. A simulation-based genetic algorithm, namely SimGA, is proposed. The design of SimGA makes use of sub-assignments of product types, and computer simulations are embedded to admit stochastic objective function evaluation. An easily computable lower bound is also obtained for performance evaluation. The proposed SimGA is tested against two popular simulation optimization algorithms, namely SPSA and SA, in the literature. Numerical studies show that the proposed lower bound is reasonably tight and SimGA outperforms both SPSA and SA in almost all testing instances.

The study makes the following contributions to both academia and industry. First, it extends the traditional static customer order scheduling problems to the stochastic domain where both arrivals and workloads of customer orders are random. This is a more realistic reflection of the order scheduling problem in practice. Second, servers considered in this study are completely heterogeneous, that is, the processing rate of each (server, product type) pair is totally independent. Very few existing studies address the scheduling problem in heterogeneous process systems even in the static environment. Third, a simulation-based genetic algorithm SimGA is proposed and shown to be effective to solve the problem in this study. It demonstrates the ability of GA type meta-heuristic in solving simulation optimization problems.

The model considered in this study admits a number of directions for future research. First, it is interesting to extend the current study to include a dynamic workload based decision scheme. Second, further effort needs to be made to improve the robustness of SimGA against the change in the server number. Third, a transformation of the solution space may enhance the effectiveness of GA encoding scheme. Finally, more advanced mathematical techniques such as entropy theory may also be considered in order to improve the search efficiency.

ACKNOWLEDGEMENT

This work is partially supported by National Science Foundation of China (NSFC) grant No. 11471023. The authors thank anonymous reviewers for their valuable suggestions and comments.

REFERENCES

Ahmadi, R., U. Bagchi, and T. A. Roemer. 2005. "Coordinated scheduling of customer orders for quick response". *Naval Research Logistics (NRL)* 52 (6): 493–512.

Baccelli, F., and A. M. Makowski. 1989. "Queueing models for systems with synchronization constraints". *Proceedings of the IEEE* 77 (1): 138–161.

Baccelli, F., A. M. Makowski, and A. Shwartz. 1989. "The fork-join queue and related systems with synchronization constraints: Stochastic ordering and computable bounds". *Advances in Applied Probability* 21 (3): 629–660.

Balin, S. 2011. "Non-identical parallel machine scheduling using genetic algorithm". *Expert Systems with Applications* 38 (6): 6814–6821.

Blocher, J. D., and D. Chhajed. 1996. "The customer order lead-time problem on parallel machines". *Naval Research Logistics (NRL)* 43 (5): 629–654.

Chen, R. J. 2011. "An upper bound solution for homogeneous fork/join queuing systems". *Parallel and Distributed Systems, IEEE Transactions on* 22 (5): 874–878.

Cheng, R., M. Gen, and T. Tozawa. 1995. "Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms". *Computers & Industrial Engineering* 29 (1): 513–517.

Chiu, N.-C., S.-C. Fang, and Y.-S. Lee. 1999. "Sequencing parallel machining operations by genetic algorithms". *Computers & Industrial Engineering* 36 (2): 259–280.

Flatto, L. 1985. "Two parallel queues created by arrivals with two demands II". *SIAM Journal on Applied Mathematics* 45 (5): 861–878.

Flatto, L., and S. Hahn. 1984. "Two parallel queues created by arrivals with two demands I". *SIAM Journal on Applied Mathematics* 44 (5): 1041–1053.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st ed. Boston, MA, USA.: Addison-Wesley Longman Publishing Co., Inc.

Kim, D.-W., K.-H. Kim, W. Jang, and F. F. Chen. 2002. "Unrelated parallel machine scheduling with setup times using simulated annealing". *Robotics and Computer-Integrated Manufacturing* 18 (3): 223–231.

Kim, Y. D., S. O. Shim, S. B. Kim, Y. C. Choi, and H. M. Yoon. 2004. "Parallel machine scheduling considering a job-splitting property". *International Journal of Production Research* 42 (21): 4531–4546.

Leung, J. Y.-T., H. Li, and M. Pinedo. 2006. "Approximation algorithms for minimizing total weighted completion time of orders on identical machines in parallel". *Naval Research Logistics (NRL)* 53 (4): 243–260.

Low, C. 2005. "Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines". *Computers & Operations Research* 32 (8): 2013–2025.

Luu, D. T., E. L. Bohez, and A. Techanitisawad. 2002. "A hybrid genetic algorithm for the batch sequencing problem on identical parallel machines". *Production Planning & Control* 13 (3): 243–252.

Melouk, S., P. Damodaran, and P.-Y. Chang. 2004. "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing". *International Journal of Production Economics* 87 (2): 141–147.

Min, L., and W. Cheng. 2006. "Genetic algorithms for the optimal common due date assignment and the optimal scheduling policy in parallel machine earliness/tardiness scheduling problems". *Robotics and computer-integrated manufacturing* 22 (4): 279–287.

Monden, Y. 2012. *Toyota production system: an integrated approach to just-in-time*. CRC Press.

Salomon, R. 1996. "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms". *BioSystems* 39 (3): 263–278.

Serafini, P. 1996. "Scheduling jobs on several machines with the job splitting property". *Operations Research* 44 (4): 617–628.

Spall, J. C. 2005. *Introduction to stochastic search and optimization: estimation, simulation, and control*, Volume 65. John Wiley & Sons.

Spall, J. C., and J. A. Cristion. 1994. "Nonlinear adaptive control using neural networks: Estimation with a smoothed form of simultaneous perturbation gradient approximation". In *American Control Conference*, Volume 3, 2560–2564. IEEE.

Srinivasan, D. 1998. "Evolving artificial neural networks for short term load forecasting". *Neurocomputing* 23 (1): 265C276.

Stalk, G. 1988. "Time–the next source of competitive advantage". *Harvard Business Review* 66 (4): 41–51.

Stalk, G., and T. M. Hout. 1990. *Competing against time: How time-based competition is reshaping global markets*. Free Press New York.

Van Hop, N., and N. N. Nagarur. 2004. "The scheduling problem of PCBs for multiple non-identical parallel machines". *European Journal of Operational Research* 158 (3): 577–594.

Van Laarhoven, P. J., E. H. Aarts, and J. K. Lenstra. 1992. "Job shop scheduling by simulated annealing". *Operations Research* 40 (1): 113–125.

Wang, I.-J., and J. C. Spall. 2003. "Stochastic optimization with inequality constraints using simultaneous perturbations and penalty functions". In *Proceedings of 42nd IEEE Conference on Decision and Control*, Volume 4, 3808–3813. IEEE.

Welch, P. D. 1983. "The statistical analysis of simulation results". *The computer performance modeling handbook* 22:268–328.

Whitley, D. 1994. "A genetic algorithm tutorial". *Statistics & Computing* 4 (2): 65–85.

Xu, X., Y. Ma, Z. Zhou, and Y. Zhao. 2013. "Customer Order Scheduling on Unrelated Parallel Machines to Minimize Total Completion Time". *IEEE Transactions on Automation Science and Engineering* 12 (1): 244–257.

Yang, J., and M. E. Posner. 2005. "Scheduling parallel machines for the customer order problem". *Journal of Scheduling* 8 (1): 49–74.

Zhou, H., Y. Feng, and L. Han. 2001. "The hybrid heuristic genetic algorithm for job shop scheduling". *Computers & Industrial Engineering* 40 (3): 191–200.

AUTHOR BIOGRAPHIES

XIAOYUN XU is an Associate Professor in Department of Industrial Engineering and Management, Peking University, P.R.China. He received both his Master and Ph.D. degree in Department of Industrial Engineering from Arizona State University, Tempe, Arizona. His research interests include simulation optimization, scheduling, and stochastic optimization. His email address is xiaoyun.xu@pku.edu.cn.

YAPING ZHAO is a Ph.D. student in the Department of Industrial Engineering and Management, Peking University, Beijing, China. She received her bachelor's degree from the Department of Industrial Engineering, Nanchang University, Jiangxi, China. Her research interests are in scheduling, stochastic modeling, and simulation. Her email address is 1301111516@pku.edu.cn.

HAIDONG LI is a Ph.D. student in the Department of Industrial Engineering and Management, Peking University, Beijing, China. He received his bachelor's degree from the Department of Engineering Structure Analysis in Peking University. His research interests are in scheduling and stochastic modeling. His email address is 1100011031@pku.edu.cn.

ZIHUAN ZHOU is a master student in Department of Industrial Engineering and Management at Peking University, P.R.China. He received his bachelor's degree from Department of Mechanics and Aerospace Engineering in Peking University. His email address is zhouzihuan@pku.edu.cn.

YANNI LIU is a master student in Department of Industrial Engineering and Management at Peking University, P.R.China. She received her bachelor's degree from Department of Energy and Source Engineering in Peking University. Her email address is liuyanni@pku.edu.cn.