Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 4, pp.106－113, December 2008.

유전자 알고리즘을 이용한 유한용량 재진입 라인 성능 제어

최 진 영†

아주대학교 산업정보시스템공학부

Performance Control of the Capacitated Re-entrant Line using Genetic Approach

Jin Young Choi†

Division of Industrial and Information Engineering in Ajou University

본 논문에서는 유한용량 재진입 생산라인에서의 스케쥴링 문제에 대한 유전자적 접근 방법을 제안하였다. 알고리즘에서 사용되는 염색체의 구조는 워크스테이션의 버퍼레벨에 대한 모든 가능한 경우를 고려하여 정의되었으며, 염색체의 각 유전자에는 그에 대응되는 시스템 상태에서 우선 순위를 갖는 작업 단계의 값이 할당되도록 하였다. 또한, 제안된 알고리즘의 구현 방법으로서 워크스테이션의 버퍼와 프로세싱 자원을 할당할 때 작업 간 우선 순위를 고려하는 동시에 각 워크스테이션의 로컬 유휴 상태를 지양하는 우선순위 기반 랜덤화 정책 알고리즘을 제안하였다. 실험을 통하여 제안된 알고리즘의 성능을 평가하였으며, 기존에 무한용량 재진입 생산라인 스케쥴링 문제에 많이 이용되었던 휴리스틱과 비교하여 보다 효율적임을 보였다.

Keywords : Capacitated Re-entrant Line, Genetic Algorithm, Chromosome, Scheduling Policy, Markov-decision Process

1. Introduction

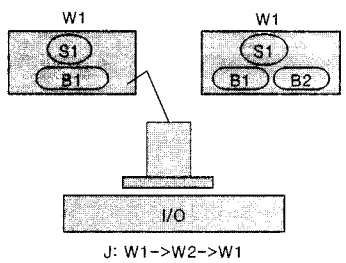
The capacitated re-entrant line (CRL) [4], considered in this work, consists of L workstations supporting the production of a single product type. Each workstation W_i, $i = 1, 2, \cdots, L$, has B_i buffer slots and S_i identical servers. The production of each unit occurs in M stages, where each job stage J_j, $j = 1, 2, \cdots, M$, is supported by one of the system workstations, which is denoted by $W(J_j)$. Then there exists at least one workstation W_k such that $|j : W(J_j) = W_k| \geq 2$. This is the re-entrant nature of the line, which can be characterized by $M > L$.

The operational behavior of the line can be described as follows: Each job instance visits the workstation for the execution of some processing stage and it is allocated one unit of buffering capacity. It holds it exclusively during staying in the station, while blocking other job instances coming into the station. In the station, the job instance competes for one of the workstation servers for the execution of the requested job stage. After having finished the processing of its current stage, the job instance waits in its allocated buffer for being transferred to the next requested station. Due to the finite buffering capacity, this should be controlled by a structural control policy (SCP) [19] which ensures that the destination workstation has free buffering capacity and it is still physically possible to process all running job instances to completion after the transfer. Under this operational framework, the CRL scheduling problem considered in this work can

논문접수일 : 2008년 08월 21일　논문수정일 : 2008년 09월 02일　게재확정일 : 2008년 09월 03일
† 교신저자 choijy@ajou.ac.kr
※ This work was partially supported by Research Grant from Ajou University.

be posed as determining how to allocate the workstation buffering and processing capacity to the competing job instances, in order to maximize the long-run system throughput, while maintaining the logical correctness of the system behavior, which is called a deadlock-free operation.

<Figure 1> shows an example CRL system consisting of two single-server workstations with buffering capacities $B_1 = 1$ and $B_2 = 2$ for workstation W_i, $i = 1, 2$, respectively. The supported production sequence is $W_1 \rightarrow W_2 \rightarrow W_1$ with $W(J_1) = W(J_3) = W_1$ and $W(J_2) = W_2$. For this configuration, the optimal deadlock avoidance policy is to maintain the total number of job instances in J_1 and J_2 less than or equal to 2.



<Figure 1> An example CRL

Based on these remarks, in this paper, we consider a genetic approach to the CRL scheduling problem, which is computationally efficient and scalable. More specifically, a chromosome including all the information needed for controlling the system behavior is defined; it considers all possible cases of the system states w.r.t. buffer levels of workstations and assigns a preferred job stage at each possible buffer level for all workstations. A set of chromosomes is randomly selected, forming a population, and the chromosomes in the selected population are subject to an evolution process in order to get more improved solution, in terms of the prespecified performance objective, specifically the system throughput. The proposed algorithm is evaluated by performing numerical experiments, resulting in that the suggested approach holds considerable promise for providing effective and computationally efficient approximations to the optimal CRL scheduling policy that consistently outperforms the typically employed heuristics.

The rest of this paper is organized as follows. Section 2 briefly summarizes the past development of the scheduling problems in the re-entrant line. In section 3, the suggested genetic algorithm is described including chromosome representation, genetic operators, and the practical implementation method. Section 4 assesses the capability of the suggested method through a numerical experiment. Finally, section 5 concludes the paper with some directions for future work.

2. Brief Literature review

During the last 15 years, there have been a lot of works dealing with the scheduling problem in the uncapacitated re-entrant line. Many of the developed results in [8-10, 14, 21] are analytically strong and a representative exposition of these results is provided in the survey paper in [11]. However, the results derived in the past works cannot be directly transferred to the CRL model, because there are many complications resulting from the blocking effect due to the finite buffering capacity. Specifically, the work in [18] demonstrated that these additional material flow dynamics negate prior analytical results based on the study of the basic re-entrant line model with infinite buffering capacity, in a strong qualitative sense, and necessitate the re-examination of the problem in the CRL model.

Moreover, results in [13] and [16] seem to suggest that this policy will be computationally intractable. Of particular interest to overcome this issue were the so-called parametric representation Neuro-Dynamic Programming (NDP) methods [1, 5, 6, 17, 20]. These methods recast the considered scheduling problem as the problem of selecting an appropriate set of values for a parametric architecture that will eventually define the adopted scheduling policy. However, they require the complete enumeration of the entire state space and the number of states is a super-polynomial function of the elements defining the CRL scheduling problem. These facts restrict the practical applicability of the results to real applications and still necessitate the development of scalable and efficient approximation methods to it.

Another computational systematic heuristic method that might provide considerable promise for such scalable and efficient approximation to the optimal solution of the considered CRL scheduling problem is a genetic approach. During last decades, there have been a significant number of works dealing with the scheduling problem using genetic approaches and a systematic classification of genetic algorithms can be found in [3]. However, they only focused on the scheduling problems with infinite buffering capacity, while not considering blocking effects due to the finite buffering capacity

in designing a chromosome and genetic operators. In the next section, we describe the suggested genetic algorithm to the CRL scheduling problem which considers the buffer occupancy of the workstation in the design of the chromosome in order to implicitly reflect the blocking effects due to the finiteness of the buffering capacity.

3. Genetic Approach to the CRL scheduling problem

In general, a genetic algorithm(GA) [15] consists of several components such as chromosome representation, population initialization, evaluation of fitness, and genetic operators. Each of these components for the suggested GA is defined as follows:

3.1 Chromosome representation scheme

First of all, a representation scheme for the genetic structure of solutions to the considered CRL scheduling problem should be defined, which is a key aspect of a genetic algorithm. More specifically, the employed representation scheme should consider how to provide the validity of solutions so that all chromosomes generated are feasible to the considered problem. Furthermore, it also should provide the optimality of solutions so that the final solution of the GA is mapped to a prominent scheduling policy with high performance for the considered CRL scheduling problem. Based on these remarks, the structure of a chromosome to the CRL scheduling problem is defined as follows:

Definition 1 : A chromosome, P, is defined as $P=(P_0^1, P_1^1, \cdots, P_{B_1}^1, \cdots, P_0^L, P_1^L, \cdots, P_{B_L}^L)$, where each $P_{n_k}^k$ represents a prioritized job stage at workstation W_k with n_k job instances that are waiting for processing or being processed, so that $P_{n_k}^k \in \sigma(W_k)$, where $\sigma(W_k)$ is the index set of job stages processed in workstation W_k.

The suggested structure of a chromosome has $\sum_{i=1}^{L}(B_i+1)$ components, and each value is translated into a prioritized job instance for allocating the workstation buffering and processing capacity at a given decision epoch. The job priority means that, at workstation W_k with n_k job instances, a job instance specified by $P_{n_k}^k$ is considered first for allocating resources. However, if it is not available, any other job instances can be allowed for allocating buffering and processing capacity. We define this policy as a priority-based randomized policy as follows:

Definition 2 : Priority-based Randomized Poilcy is a policy that first allocates resources to a prioritized job instance if it is available. Otherwise, it considers any available job instance for allocating resources.

The characteristic of the scheduling policy represented by the structure of P is to allocate available resources based on buffer level information. This idea is based on some queueing theoretic concepts and results from the past research work [2, 12] for uncapacitated re-entrant line scheduling problem. This policy satisfies the requirements of the representation scheme for the genetic structure of solutions as follows:

Property 1 : The suggested representation scheme of the chromosomes guarantees the validity and optimality of solutions of GA for the considered CRL scheduling problem.

The proof of Property 1 can be based on the nature of the priority-based randomized policy for allocating resources to the job instance represented by each component of a chromosome.

3.2 Population initialization and Evaluation of fitness

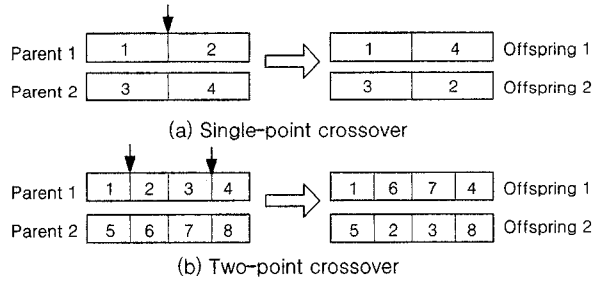
Based on the representational scheme of a chromosome suggested in section 3.1, a group of chromosomes are randomly created, becoming a population. Then elements in the population are evaluated and rated according to their fitness. More specifically, each $P_{n_k}^k$ in P is chosen randomly from the set $\sigma(W_k)$, which satisfies the feasibility of the scheduling policy translated from the considered chromosome. The size of the initial population should be even number with reasonable size.

The fitness of a chromosome is evaluated by measuring the maximal throughput of the system that can be achieved by the translated scheduling policy. In general, the throughput of the considered CRL scheduling policy can be eval-

uated by using either a well-known modeling and analysis framework such as Markov decision process (MDP), or a simulation. A chromosome with the highest value of fitness is selected as the current solution for the CRL scheduling problem.

3.3 Genetic Operators

Chromosomes are arranged in descending order of fitness and then paired with each other from the one having the highest value of fitness and the next highest one, and so on, becoming parents. Each pair of parents generates two new offsprings, where genetic operators are used to change the genetic composition of offsprings. This procedure is called reproduction. After generating new offsprings, they are evaluated and a new population of chromosomes is selected again, while finding a new improved solution. All the chromosomes are subject to this evolutionary process until an appropriate one, satisfying a pre-specified stopping rule, is obtained.



<Figure 2> Crossover operators

Two genetic operators are designed for this purpose : the crossover and the mutation operator.

3.3.1 Crossover operator

This is an operator to generate offsprings from two chromosomes by exchanging some parts of their components. In this work, we consider two kinds of crossover operators as in <Figure 2>; (i) Single-point crossover (SPC) and (ii) Two-point crossover (TPC). SPC operator divides each chromosome into two parts and exchanges the latter part between them (In <Figure 2> (a), parts 2 and 4 are exchanged between parent 1 and 2). In TPC operation, two points are selected randomly and the components between two selected points are exchanged to generate two new offsprings. By selecting one of these operators and applying it with a pre-specified rate $r_c(0 < r_c < 1)$, two offsprings are created from the parents.

3.3.2 Mutation operator

Each component of the new offsprings generated is considered for performing the mutation operation with a pre-specified rate, which is represented by $r_m(0 < r_m < 1)$. The mutation rate is the probability that an inherited feature of an offspring mutates into another value. Mutation can be thought of as a transition from a current solution to its neighborhood in a local improvement search algorithm, which is used to prevent the algorithm from falling into local optimal solution. If the operation is allowed, then the value of the considered component is changed with a randomly selected value from the set $\sigma(W_k)$.

3.4 Implementation of the Genetic Algorithm

The suggested genetic algorithm was implemented as follows:

i) Step 1 : Initialization

Start with an initial population of size n, where n is integer. Specifically, we determined the value of n by using $0.01*\left(\prod_{i=1}^{L}|\sigma(W_i)|^{B+1_i}\right)$ (However, if n < 10, we used n = 10). Evaluate the fitness of each member in the current population by using MDP framework [5]. Sort chromosomes in descending order of fitness. Identify the best solution which has maximum throughput in the current population. Go to step 2.

ii) Step 2 : Iteration

Pair up the chromosomes by selecting two chromosomes in the direction from the highest to the lowest. By selecting the values $r_c = 0.9$ and $r_m = 0.3$, apply the crossover and the mutation operations to each parent. The selection of the values of r_c and r_m was based on the observation that it would be better to have more crossover operations and less mutations since i) in each iteration the suggested GA updates the population with chromosomes with better fitness, therefore ii) there would be less benefits from mutation. Evaluate the fitness of new generated members. Form a new population with same size for next iteration by selecting n best members among the new generated offsprings and the current population. Sort them in descending order of fitness and

identify the best solution which has maximum throughput in the new population. If it is improved, then update the current best solution. Otherwise, remove it and go to step 3.

iii) Step 3 : Stopping rule

The algorithm stops when it performs three times of consecutive iterations without any improvement in the best trial solution found so far. Otherwise, go to step 2.

iv) Evaluation of a chromosome

The throughput achieved by the considered chromosome can be computed by modeling the system behavior as a continuous-time Markov decision process (CT-MDP) [5], where the corresponding policy is deterministic and the average reward $J^*(i)$ is accumulated by the control defined by the chromosome. Then we have $J^*(i) = \lambda^*$ for all states i because the structure of the underlying CT-MDP is communicating. Furthermore, there exists a function $h^*(i)$ for all states i, that satisfies the following Bellman's optimality equation:

$$h^*(i) = \max_{u \in U(i)} \left[G(i,u) - \lambda^* \overline{\tau}_i(u) + \sum_{j \in S} p_{ij}(u) h^*(j) \right] \quad (1)$$

where $G(i,u)$ is the single-stage expected reward for state i, $\overline{\tau}_i(u)$ is the expected sojourn time at state i, S is the set of states, and $p_{ij}(u)$ is the transition probability from state i to state j by taking an action u, which is defined at state i. By solving Equation (1), we can compute the throughput achieved by the considered chromosome.

4. An Experimental Investigation

We performed an experimental investigation on the potential performance of the suggested genetic algorithm by generating some prototypical example configurations and the performance of the suggested algorithm was compared with the performance by using some well-known heuristics.

4.1 Design of a numerical experiment

For an experimental investigation, we considered the same configuration and same parameter values for processing times as in [5], because, by doing that, eventually we may even compare the performance of the suggested GA with the potential performance of the parametric representation NDP method suggested in [5]. More specifically, two types of re-entrant line was considered, the first consisting of two single-server workstations and the second consisting of three single-server workstations. Both of these lines are under the operational framework stated in section 1, while the adopted SCP was the optimal -i.e., maximally permissive-policy. For each type of re-entrant line, different configurations were generated by changing buffering capacities; <Table 1> summarizes the system configurations used in this experiment. For each configuration, 30 problem instances with randomly generated processing rates were considered.

4.2 Experimental results

In order to assess the quality of the two suggested algorithms, we defined the percent error by

$$\%error = \frac{Optimal\ TH - TH by\ GA}{Optimal\ TH} X100 \quad (2)$$

Then the performance of the two suggested GA, GA-SPC and GA-TPC was tested. We also compared the percent error attained by the proposed algorithms to the percent error generated by the parametric representation NDP method suggested in [5] and some known heuristics that have been shown to perform well in the case of uncapacitated re-entrant

<Table 1> System Configurations considered in the numerical experiment

Configurations	Number of workstations	Number of job stages (JS) and job routes	Buffer capacities
Conf 1 Conf 2 Conf 3	2	3JS($W_1 \rightarrow W_2 \rightarrow W_1$)	(B_1, B_2) = (1, 2) (B_1, B_2) = (3, 2) (B_1, B_2) = (4, 4)
Conf 4 Conf 5 Conf 6	3	4JS($W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow W_1$)	(B_1, B_2, B_3) = (1, 2, 2) (B_1, B_2, B_3) = (3, 2, 2) (B_1, B_2, B_3) = (4, 3, 2)

lines, namely, the first bufer first serve (FBFS), last buffer first serve (LBFS), and first in first out (FIFO) policies [5]. The results were summarized in <Table 2>. More specifically, it lists the average, minimum, and maximum percent errors of throughput obtained by using the aforementioned methods.

4.3 Assessments

Some interesting observations regarding the results of this numerical experiment and their implications for the quality of the suggested GAs can be summarized as follows.

Overall, the percent errors generated by the suggested GAs are rather small compared to those generated by the considered heuristics.

Even more importantly, the suggested GAs are more consistent in their performance than the considered heuristics, as represented by the reported maximum percent errors.

Compared to the parametric representation NDP method [5], it seems that the suggested GAs have large percent errors. However, from the practical point of view, the parametric representation NDP method are not efficient in the sense that it should generate all state space and it is not easy to develop an efficient tuning algorithm to find the approximation architecture. Furthermore, the differences of the percent errors generated by the considered GAs are not significant that much.

More interestingly, in case of Conf3, the suggested GAs have smaller values of average, minimum, and maximum percent errors than the values of NDP method.

4.4 Statistical significance of the differences of the two algorithms

The significance of the differences of the two algorithms, GA-SPC and GA-TPC, was checked by testing the statistical significance of the mean difference of the average percent errors resulting by applying the two algorithms. This test is performed as follows:

We have 180 paired observations of the percent errors. By defining a single derived variable D as the difference between the paired values on percent errors resulting from using GA-SPC and GA-TPC, we can compute the observed sample mean difference $\overline{D}$ and sample standard deviation s_D as follows:

$$\overline{D} = \frac{1}{n}\sum_{n=1}^{n} D_i \tag{3}$$

<Table 2> Comparison of performance for the considered re-entrant lines

Config.	%error	GA -SPC	GA -TPC	NDP	FBFS	LBFS	FIFO
Config1	Avg.	0	0	0.093051	0	2.906683	0
	Min.	0	0	0.016866	0	1.419307	0
	Max.	0	0	0.173766	0	5.651860	0
Config2	Avg.	1.186665	1.179550	0.820087	2.088194	2.865571	2.088194
	Min.	0.000206	0.000206	0.003095	0.002057	0.000411	0.002057
	Max.	4.679478	4.679478	4.886354	6.013643	10.944309	6.013643
Config3	Avg.	0.532486	0.587982	0.714999	0.593519	1.322035	0.593519
	Min.	0	0	0.000104	0	0	0
	Max.	3.078559	3.951328	3.523133	4.298884	9.824424	4.298884
Config4	Avg.	0.467542	0.467542	0.525297	0.802350	2.657095	0.802350
	Min.	0	0	0.065391	0	0.308027	0
	Max.	2.506137	2.506137	1.908183	2.831750	14.249711	2.831750
Config5	Avg.	1.282840	1.258623	0.723601	3.043938	4.621490	3.043938
	Min.	0	0	0.003318	0.095917	0.822142	0.095917
	Max.	4.288475	4.288475	2.617387	8.457638	14.164897	8.457638
Config6	Avg.	0.882617	0.856900	0.727640	2.783108	1.751215	2.783108
	Min.	0	0	0.000625	0.099751	0.000172	0.099751
	Max.	4.288628	4.025460	3.502581	8.869749	5.477629	8.869749

$$s_D = \sqrt{\frac{\sum_{i=1}^{n}(D_i - \overline{D})^2}{n-1}} \quad (4)$$

where $n = 180$ and D_i is the difference of the percent error of the $i-$th paired observation. By the Central Limit Theorem [7], $\overline{D}$ has a normal distribution with unknown variance. Then, we can perform the $t-$test [7] with 179 degrees of freedom by establishing the hypothesis set

$$H_0 : \mu_D = 0,\ H_1 : \mu_D \neq 0 \quad (5)$$

The test statistic t is defined as

$$t = \frac{\sqrt{n}\,\overline{D}}{s_D} \quad (6)$$

Since $\overline{D}$ = 0.000259 and s_D = 0.149634, by Equation (6), t = 0.023195 < $t_{0.00005,179}$ = 3.346. This result shows that the hypothesis H_1 cannot be accepted with a confidence level higher than 99.99% and manifests that there is no significant difference between two suggested algorithms, at least for the considered configurations.

5. Conclusions

In this paper, we suggested a genetic approach for the capacitated re-entrant line scheduling problem. The structure of a chromosome is defined by considering all possible states of the system in terms of buffer levels of workstations and a preferred job stage is assigned to each component of the chromosome. The performance of the proposed algorithm is evaluated through a numerical experiment, showing that the suggested approach holds considerable promise for providing effective and computationally efficient approximations to the optimal scheduling policy that consistently outperforms the typically employed heuristics. Our future work intends to promote further development on the extension of the developed results to production systems more general than the CRL.

References

[1] Bertsekas, D. P. and Tsitsiklis, J. N.; *Neuro-Dynamic Programming*, Belmont, MA: Athena Scientific, 1996.

[2] Bertsimas, D., Paschalidis, I. C., and Tsitsiklis, J. N.; "Optimization of multiclass queueing networks : Polyhedral and nonlinear characterizations of achievable performance," *Ann. Appl. Probability*, 4(1) : 43-75, 1994.

[3] Cheng, R., Gen, M., and Tsujimura, Y.; "A tutorial survey of job-shop scheduling problems using genetic algorithms-I. Representation," *Computers and Industrial Engineering*, 30(4) : 983-997, 1996.

[4] Choi, J. Y. and Reveliotis, S. A.; "A generalized stochastic petri net model for performance analysis and control of capacitated re-entrant lines," *IEEE Trans. on Robotics and Automation*, 19(3) : 474-480, 2003.

[5] Choi, J. Y. and Reveliotis, S. A.; "Relative Value Function Approximation for the Capacitated Re-Entrant Line Scheduling Problem," *IEEE Trans. on Automation Science and Engineering*, 2(3) : 285-299, 2005.

[6] De Farias, D. P.; "The linear programming approach to approximate dynamic programming : Theory and application," Ph.D. dissertation, Dept. Manag. Sci. Eng., MIT, Cambridge, 2002.

[7] Hayter, A. J.; *Probability and Statistics*, International Thomson PUB, 1996.

[8] Kumar, P. R.; "Scheduling manufacturing systems of re-entrant lines," in Stochastic Modeling and Analysis of Manufacturing Systems, D. D. Yao, Ed. Berlin, Germany:Springer-Verlag, : 325-360, 1994.

[9] Kumar, P. R.; "Scheduling semiconductor manufacturing plants," *IEEE Control syst. Mag.*, 14(6) : 33-40, 1994.

[10] Kumar, S. and Kumar, P. R.; "Fluctuation smoothing policies are stable for stochastic re-entrant lines," *Discrete-Event Dynamic Systems : Theory and Application*, 6 : 361-370, 1996.

[11] Kumar, S. and Kumar, P. R.; "Queueing network models in the design and analysis of semiconductor wafer fabs," *IEEE Trans. Robotics and Automation*, 17(5) : 548-561, 2001.

[12] Kumar, J. N. and Kumar, P, R.; "Performance bounds for queueing networks and scheduling policies," *IEEE Trans. Autom. control*, 39(8) : 1600-1611, 1994.

[13] Lawley, M. A. and Reveliotis, S. A.; "Deadlock Avoidance for Sequential Resource Allocation Systems: Hard and Easy Cases," *The Intl. Jrnl. of FMS*, 13(4) : 385-404, 2001.

[14] Lu, S. H., Ramaswamy, D., and Kumar, P. R.; "Effi-

cient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Trans. on Semiconductor Manufacturing*, 7(3) : 374-385, 1994.

[15] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, 1996.

[16] Papadimitriou, C. A. and Tsitsiklis, J. N.; "The complexity of optimal queueing network control," *Math. Oper. Res.*, 24(2) : 293-305, 1999.

[17] Tsitsiklis, J. N. and Van Roy, B.; "Feature-based methods for large scale dynamic programming," *Machine Learning*, 22 : 59-94, 1996.

[18] Reveliotis, S. A.; "The Destabilizing effect of blocking due to finite buffering capacity in multi-class queueing networks," *IEEE Trans. on Automatic Control*, 45(3) : 585-588, 2000.

[19] Reveliotis, S. A., Lawley, M. A., and Ferreira, P. M.; "Structural control of large-scale flexibly automated manufacturing systems," in The Design of Manufacturing Systems, C.T. Leondes, Ed. Boca Raton, FL : CRC, : 4.1-4.34, 2001.

[20] Van Roy, B.; "Learning and value function approximation in complex decision processes," Ph.D. dissertation, Dept. Elect. Eng. Comp. Sci., MIT, Cambridge, 1998.

[21] Wein, L. M.; "Scheduling semiconductor wafer fabrication," *IEEE Trans. on Semiconductor Manufacturing*, 1(3) : 115-130, 1988.