USING A BEE COLONY ALGORITHM FOR NEIGHBORHOOD SEARCH IN JOB SHOP SCHEDULING PROBLEMS

Chin Soon Chong
Planning and Operations Management
Singapore Institute of Manufacturing Technology
71 Nanyang Drive, 638075, Singapore
Email: cschong@simtech.a-star.edu.sg

Malcolm Yoke Hean Low
School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, 639798, Singapore
Email: yhlow@ntu.edu.sg

Appa Iyer Sivakumar
School of Mechanical and Production Engineering
Nanyang Technological University
Nanyang Avenue, 639798, Singapore
Email: msiva@ntu.edu.sg

Kheng Leng Gay
School of Electrical and Electronics Engineering
Nanyang Technological University
Nanyang Avenue, 639798, Singapore
Email: eklgay@ntu.edu.sg

KEYWORDS
Scheduling, honey bee colony, neighborhood search.

ABSTRACT

This paper describes a population-based approach that uses a honey bees foraging model to solve job shop scheduling problems. The algorithm applies an efficient neighborhood structure to search for feasible solutions and iteratively improve on prior solutions. The initial solutions are generated using a set of priority dispatching rules. Experimental results comparing the proposed honey bee colony approach with existing approaches such as ant colony, tabu search and shifting bottleneck procedure on a set of job shop problems are presented. The results indicate the performance of the proposed approach is comparable to other efficient scheduling approaches.

1 INTRODUCTION

Rapid expansion of global economy is leading to intense competition in global market which has further resulted in challenging manufacturing environment with lower product costs, shorter product life cycles and more product variety. An effective scheduling system can play a significant role in reducing inventory level and cycle times while improving on-time delivery and the utilization of critical resources. The importance of manufacturing scheduling can be noted from the extensive studies of scheduling algorithms for job shop problems in the past four decades by researchers and practitioners (Blackstone et al. 1982, Rajendran and Holthaus 1999, Jain and Meeran 1999).

Scheduling is defined as the allocation of limited resources to tasks over time to meet an objective (Pinedo 1995). In a manufacturing environment, resources are usually machines while tasks are operations relating to jobs. A scheduling problem can thus be characterized by a set of jobs and each job consists of a set of operations that must satisfy specific constraints such as machine capacity and precedence constraints. The objective of scheduling is to determine the job schedules that optimize a measure of performance (Rajendran and Holthaus 1999). Commonly known performance measures are utilization, tardiness, cycle time and throughput. For our work, we focus on makespan, which is closely related to the utilization. Improving makespan will lead to better throughput rate and subsequently lower product cost.

Due to factorial explosion of possible solutions, job shop scheduling (JSP) problems are a member of a large class of intractable numerical problems known as NP-hard (Pinedo 1995, Jain and Meeran 1999). Solution techniques for shop scheduling problems range from simple priority dispatching rules such as SPT (shortest processing time) to more elaborate techniques such as branch and bound (Brucker et al. 1994), tabu search (Nowicki and Smutnicki 1996), shifting bottleneck procedure (Balas and Vazacopoulos 1998), simulated annealing (Van Laarhoven et al. 1992) and ant colony (Blum and Sampels 2004, Campos et al. 2001). Priority dispatching rules are commonly deployed in industry due to their ease of implementation, speed and flexibility. However, the performance of a dispatching rule is highly dependent on various shop factors (Baker 1984), and no single rule can distinctively dominates others (Park et al. 1997).

Branch and bound algorithms are exact methods for finding global optimal solutions, but these methods often require excessive computation time. In comparison, approximate methods based on meta-heuristics such as tabu search (TS) and shifting bottleneck procedure (SBP) have been more successfully applied to shop scheduling problems. A meta-heuristic is an iterative solution procedure, which incorporates secondary heuristics into a more sophisticated framework (Glower 1986). The family of meta-heuristics includes deterministic or probabilistic learning methods. TS and SBP fall in the first category,

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

while simulated annealing and ant colony belong to the second category.

This work aims to explore a probabilistic learning approach, which is based on nectar collection in honey bee colonies, to job shop scheduling problems. This research is inspired by the work done by Nakrani and Tovey (2004), on using a honey bee algorithm for dynamic allocation of Internet servers. In their algorithm, servers and HTTP request queues in an Internet server colony are modeled as foraging bees and flower patches respectively. In another bee colony related paper by Teodorovic and Dell'orco (2005), bee colony optimization (BCO) approaches with fuzzy sets are used to solve ride-matching problem in the transportation domain. It is concluded that models based on swarm intelligence principles could contribute to the solution of complex engineering and management problems.

Our proposed algorithm is based on local search procedure by using an efficient neighborhood structure, which is adapted from work done by Nowicki and Smutnicki (1996). The scheduling approach is an iterative improvement method that starts from a given initial solution, and continuously looks for better solutions. This work is a further activity of our earlier work (Chong et al. 2006) on the honey bee foraging model. The ealier algorithm always constructs shop schedules from scratch based on preferred operation sequence. To compare the performance of the honey bee algorithm, a computational study is done on a set of benchmark problems.

This paper first describes how honey bee colonies deploy forager bees to collect nectar amongst diverse flower patches in Section 2. The concept of neighborhood search is then described in Section 3. In Section 4, the mappings of job shop scheduling meta-heuristics to honey bees forager deployment is given. Subsequently, the implementation details are discussed in Section 5. This is followed by a comparative study on the performance of the honey bee approaches on the benchmark problems in Section 6. The paper ends with conclusions and future work in Section 7.

2 HONEY BEE COLONY

Colonies of social insects such as bees have instinct ability known as swarm intelligence (Nakrani and Tovey 2004, Teodorovic and Dell'orco 2005). This highly organized behavior enables the colonies of insects to solve problems beyond the capability of individual members by functioning collectively and interacting primitively amongst members of the group. In a honey bee colony for example, this behavior allows bees to explore the environment in search of flower patches (food sources) and then indicate the food source to the other bees of the colony when they return to the hive. Such a colony is characterized by self-organization, adaptiveness and robustness.

Seeley (1995) proposed a behavioral model of self-organization for a colony of honey bees. In the model, forager bees visiting flower patches return to the hive with nectar as well as a profitability rating of respective patches. The collected nectar provides feedback on the current status of nectar flow into the hive. The profitability rating is a function of nectar quality, nectar bounty and distance from the hive. The feedback sets a response threshold for an enlisting signal which is known as waggle dance, the length of which is dependent on both the response threshold and the profitability rating. The waggle dance is performed on the dance floor where individual foragers can observe. The foragers can randomly select a dance to observe and follow from which they can learn the location of the flower patch and leave the hive to forage. This self-organized model enables proportionate feedback on goodness of food sources.

3 NEIGHBORHOOD SEARCH

Neighborhood or local search moves from an initial solution by a sequence of neighborhood changes, which improve each time the value of the objective function until a local optimum is found (Hansen and Mladenović 2001). The connectivity property pertaining to local search states that starting with any feasible solution, there exists some sequence of moves that will reach an optimal solution.

Neighborhood structures play a very important role in local search as the time complexity of a search depends on the size of the neighborhood and the computational cost of the moves (Ten Eikelder et al. 1997). In the history of JSP problems, many neighborhood structures have been identified. The most general neighborhood definition consists of swapping any adjacent pair of operations on the same machine. This neighborhood is large and not all moves will lead to feasible schedules.

Van Laarhooven et al. (1992) derived an improved neighborhood structure that consists only of those moves that involve swapping any adjacent pair of critical operations which require the same machine. Swapping two adjacent non critical operations will not lead to shorter critical path as the path in the original schedule still exists in the new schedule. Matsuo et al. (1988) further proved that swapping two critical adjacent operations i and j will never give shorter critical path if machine preceding operation of i and succeeding operation of j are also critical.

A subsequent neighborhood enhancement is defined by Nowicki and Smutnicki (1996). This neighborhood is based on the concept of blocks. A block is a maximal sequence (i.e. of at least one) of adjacent critical operations that require the same machine. They proved that swapping the first or last two operations on the first or last block respectively will never lead to an immediate improvement in the makespan if these blocks contain more than two operations. For our bee colony algorithm, we make use of this neighborhood structure.

4 HONEY BEE COLONY ALGORITHMS

This section details algorithms to perform job shop scheduling inspired by the behavior of honey bee colony. There are two major characteristics of the bee colony in searching for food sources: waggle dance and forage (or nectar exploration). We will discuss in separate sub-sections on how we map these characteristics of a bee colony to job shop scheduling.

4.1 Waggle Dance

A forager f_i on return to the hive from nectar exploration will attempt with probability p (see Table 2) to perform waggle dance on the dance floor with duration $D = d_i A$, where d_i changes with profitability rating while A denotes waggle dance scaling factor. A forager will also attempt with probability r_i to observe and follow a randomly selected dance. The probability r_i is dynamic and also changes with profitability rating. If a forager chooses to follow a selected dance, it will use the 'path' taken by the forager performing the dance to guide its direction for flower patches. We refer this path as the 'preferred path' of the forager. The preferred path for a forager is a series of landmarks from a source (hive) to a destination (nectar). While a forager will try to follow the preferred path, it may stray off the preferred path from time to time.

For job shop scheduling, the profitability rating is related to the objective function, which in our case, is makespan. Let Pf_i denote the profitability rating for a forager at time t, it is given by:

$$Pf_i = \frac{1}{C^i_{\max}}$$

where,

$C^i_{\max}$ = makespan of the schedule generated by a forager f_i at time t.

The bee colony's average profitability rating, Pf_{colony} is given by:

$$Pf_{colony} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{C^i_{\max}}$$

where,

n = number of waggle dance at any time t (we only consider those bees that are dancing when computing profitability rating). Note that this represents a sample of the colony's actual profitability

The dance duration, d_i, is given by:

$$d_i = \frac{Pf_i}{Pf_{colony}}$$

The probability r_i of following a path is adjusted according to the profitability ratings of a forager and the colony based on the lookup Table 1 (adopted from Nakrani and Tovey 2004). Essentially, a forager is more likely to randomly observe and follow a waggle dance on the dance floor if its profitability rating is low as compared to the colony's.

Table 1: Lookup Table for Adjusting Probability of following a Waggle Dance

Profitability Rating	r_i
$Pf_i < 0.5Pf_{colony}$	0.60
$0.5Pf_{colony} \le Pf_i < 0.65Pf_{colony}$	0.20
$0.65Pf_{colony} \le Pf_i < 0.85Pf_{colony}$	0.02
$0.85Pf_{colony} \le Pf_i$	0.00

4.2 Forage (Nectar Exploration)

For nectar foraging, a population of foragers will cyclically construct solutions for job shop scheduling problems. The foraging task is based on a neighborhood structure. In the algorithm, each forager keeps a list of 'preferred' neighborhood moves to guide its search for better solutions. A move denotes two successive operations in a critical block, that can potentially improve the objective function. Each forager maintains its own preferred move list, which can be adopted by other foragers during a waggle dance. We define this set of preferred moves as ψ_{ij}. The list of moves is updated whenever a move is taken by the forager.

The foraging algorithm for each forager starts with an initial schedule generated by a priority dispatching rule, which is selected from a pre-defined set. By having a set of dispatching rules, a better diversification in searching can be achieved. A list of allowed moves, defined as φ_{ij}, is then derived from the generated schedule. These moves are based on the neighborhood structure of Nowicki and Smutnicki (1996). Denoting the number of elements in a set or list as $\#$, it can be implied that $\#(\psi_{ij} \cap \varphi_{ij})$= 1 or 0.

A forager subsequently chooses a move from the allowed move list (φ_{ij}) according to the following rule at time t:

$$P_{ij}(t) = \frac{[\rho_{ij}(t)]^{\alpha} \cdot [a_{ij}]^{\beta}}{\sum_{j \in \varphi_{ij}} [\rho_{ij}(t)]^{\alpha} \cdot [a_{ij}]^{\beta}}$$

where,

ρ_{ij} = rating of move$_{ij}$ of operations i and j (operation i precedes operation j)

a_{ij} = attractiveness of move$_{ij}$

P_{ij} = probability to apply move$_{ij}$

The rating ρ_{ij} of a move$_{ij}$ is provided by:

$$\rho_{ij} = \begin{cases} \gamma, & move_{ij} \in \psi_{ij} \\ \dfrac{1-\gamma \cdot \#(\psi_{ij} \cap \varphi_{ij})}{\#(\psi_{ij}) - \#(\psi_{ij} \cap \varphi_{ij})}, & move_{ij} \notin \psi_{ij} \end{cases}$$

where,

γ = value to be assigned to moves found in the list of preferred moves that are used to guide a forager, γ < 1.0.

It should be noted that for the first nectar exploration expedition by the foragers, ρ_{ij} will be assigned the same value for all allowed moves (since $\#(\psi_{ij})$ = 0, and thus $\#(\psi_{ij} \cap \varphi_{ij})$ = 0).

The derivation of the attractiveness is based on the localized effect of a move. The computation of the attractiveness of a move is defined as follows:

$$a_{ij} = \begin{cases} \dfrac{1}{n_{ij}}, n_{ij} > 0 \\ -n_{ij}, n_{ij} \le 0 \end{cases}$$

where,

n_{ij} = localized net change of start and end times of neighborhood

The localized net change is associated with the sum of the start time changes of the operation succeeding operation j on the same machine, and the operations succeeding operations i and j of the respective jobs, before and after operation move$_{ij}$. Figure 1 shows the schedules before and after move $\{O_{j,k}, O_{p,q}\}$. Note the start time changes for operations $O_{x,y}$, $O_{j,k+1}$ and $O_{p,q+1}$.

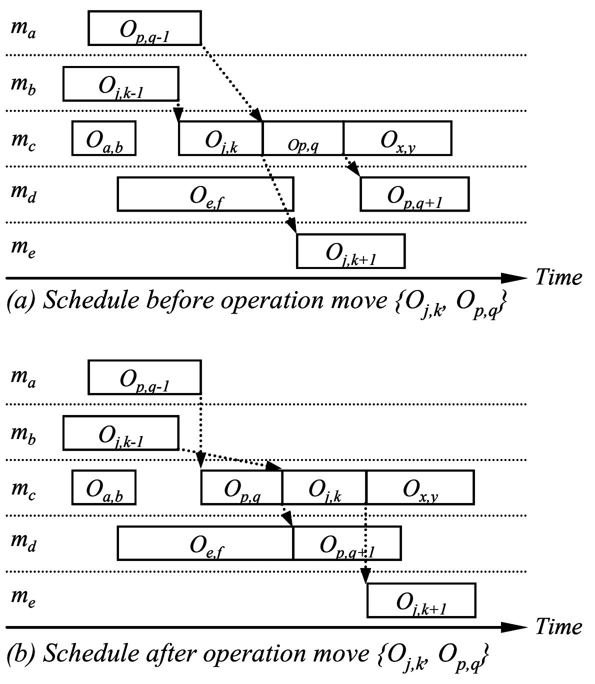

Figure 1: Schedules Before and After an Operation Move. (a) Schedule Before an Operation Move; (b) Schedule After an Operation Move

Once a move in the neighborhood is selected, the selected move is evaluated exactly by updating the start and end times of all the operations after the move is performed, and the makespan for the new schedule is computed. The list of preferred moves for the forager under consideration will be updated accordingly. The parameters α and β adjust the relative importance of the weight in the move that is found in the list of preferred moves versus the attractiveness of a move.

This foraging algorithm differs from our earlier foraging algorithm (Chong et al. 2006). The earlier algorithm is based on a list of preferred operations instead of preferred moves, and the attractiveness of the operation is related to its processing time. The previous algorithm always construct new schedules from scratch, based on a list of preferred operations for the foragers.

4.3 Algorithmic Framework

A combination of waggle dance and forage algorithms constitutes one cycle (or iteration) in this evolutionary computation approach. This computation will run for a specific number of iterations, N_{max}. The global best solution after the N_{max} iterations will be presented as the final schedule at the end of run. In the algorithm, there are l number of honey bees, each of these bees will first observe the waggle dance to select a path to follow. The bees will then leave the hive to forage for nectar. The global best solution will be updated and saved.

5 IMPLEMENTATION DETAILS

The honey bee colony algorithms are developed using Java. A list of elite solutions is used to denote foragers that are performing waggle dance on the dance floor. The duration of a waggle dance is linked to the number of iterations that an elite solution is to be kept in the list. Each elite solution contains forager's preferred list of moves, its makespan, maximum number of iterations allowed and the current iteration number (i = 1 to N_{max}) when a solution is added into the list. After every foraging cycle, the list is updated to remove elite solutions that have exceeded the maximum number of iterations allowed.

In our implementation, the list of preferred moves is stored in a lookup table. This table contains moves that link two operations together. Each operation can only be linked to another operation, and this link may be changed or updated when a a new move is selected and taken. The initial solutions are generated using several priority dispatching rules. These rules are FIFO (first in first out), SPT (shortest processing time), SRPT (shortest remaining processing time), MWKR (most work remaining), MOR (most operations remaining) and RND (random dispatching, i.e. randomly select a job). The rules are randomly selected by the foragers.

6 EXPERIMENTAL EVALUATION

In this section we describe the benchmark problems, benchmark algorithms and present experimental results.

6.1 Problem Instances

The job shop scheduling is a very well studied problem and has a large number of standard benchmarks for which the optimal value or an upper bound is known. We have selected a set of 82 job shop problem instances based on the paper by Ganesan et al. (2004). The sizes of these problems range from 6 to 50 jobs and 5 to 20 machines.

6.2 Benchmark Algorithms

To evaluate the performance of the proposed bee colony algorithm, we have included four other approaches in our experimental study. The first is our previous version of bee colony foraging model based on preferred operation path (Chong et al. 2006). The second is an ant colony algorithm by Dorigo et al. (1996). The third algorithm is a tabu search algorithm developed by Nowicki and Smutnicki (1996). The fourth is a shifting bottleneck procedure (SBP) proposed by Adams et al. (1988). The latter three algorithms are programmed based on the information from the published papers.

6.3 Experiments and Results

Since bee colony and ant colony are probability based algorithms, a total of 5 replication runs have been performed for each job shop problem to obtain average results. To differentiate between the two versions of our bee colony algorithms, we term the previous version in the paper by Chong et al. (2006) as bee 1 while the new version as bee 2. We have performed fine tuning on the parameters for some of the algorithms, and the final settings of major parameters are presented in Table 2. The parameter tunning is performed by changing each individual parameter on a range of discrete values while fixing other parameters. Based on the experiments, we then choose a better value of each of these parameters. In general, the performance of the algorithms are dependent (or sensitive) to the values of these parameters. Most of the values are kept the same for both bee 1 and bee 2. An example is the dance duration, which is associated with the the waggle dance scaling factor, A is kept to be the same value for bee 1 and bee 2. Note that no parameter settings are required for SBP.

Table 2: Parameter Settings for the Algorithms

Parameter	**Bee 1**	**Bee 2**	**Ant**	**Tabu**
Iterations, N_{max}	200	1250	500	50
Population, l	No. of jobs	No. of jobs	No. of jobs	
Alpha, α	1.0	1.0	1.0	
Beta, β	1.0	1.0	1.0	
Rating, ρ_{ij}	0.99	0.99		
Waggle dance scaling factor, A	100	100		
Probability to perform waggle dance, p	0.001	0.001		
Evaporation coefficient, ρ			0.01	
Max. no.. of elite solution	20	20		20
Max. size of tabu list				8

Table 3 summarizes the relative performance in terms of percentage pertaining to makespan for bee 1, bee 2, ant colony, tabu search and shifting bottleneck heuristics. The results show the average, minimum and maximum percentage differences from the best known makespan for the 82 job shop problems. The second last row records the number of best solutions achieved among the 5 heuristics. The last row exhibits the relative execution time for the 5 heuristics in comparison to the fastest heuristic, bee 2.

Table 3: Relative Performance of Bee 1, Bee 2, Ant Colony, Tabu Search and SBP

Relative Improve-ment	**Bee 1**	**Bee 2**	**Ant**	**Tabu**	**SBP**
Mean (%)	12.32	7.74	11.85	8.06	7.42
Min. (%)	0	0	0	0	0
Max. (%)	40.08	29.35	38.24	38.86	37.14
Best solutions	13	19	14	21	23
Execution time	1.95x	1x	1.86x	1.25x	1.59x

From the results, bee 2 is ranked second in terms of mean percentage improvement, falling behind SBP and ranked first in terms of maximum percentage improvement. Although bee 2 performs slightly worse than SBP, bee 2 achieves its results in the shortest execution time. In addition, SBP will terminate when successive iterations do not give improved results. Thus the results represent the best performance this version of SBP can give and no further improvement possible even with additional computation time available. In contrast, other heuristics should improve with increasing computation time. The better results of bee 2 as compared to bee 1 can be attributed to the efficient critical block neighborhoods and multiple initial solutions generated by priority dispatching rules. Although both bee 2 and tabu search use the same neighborhood structure, the former is able to take advantage of starting with multiple initial solutions due to its inherently population-based approach.

Comparing the performance of peers, bee 1 and ant colony heuristics, bee algorithm performs slightly worse than ant algorithm in terms of mean and maximum percentage improvement. The time taken to solve the job shop problems for both heuristics is very similar with ant colony being slightly faster. Evidently, both bee 1 and ant colony heuristics under perform bee 2, tabu search and shifting bottleneck procedures primarily due to: 1) solutions are always constructed from scratch; 2) problems specific knowledge of job shop such as neighborhood structure and machine bottlenecks is not used in the heuristics.

7 CONCLUSIONS AND FUTURE WORK

We have implemented job shop scheduling algorithm based on self-organization of honey bee colony for solving job shop scheduling problems. The algorithm is founded on neighborhood search based foraging. This proposed algorithm outperforms the previous version of our algorithm which always constructs new schedules based on preferred operation sequence.

The experimental studies show that the effectiveness of finding better solutions is dependent on the efficiency of constructing new solutions and escaping local optimums. Modifying previous solutions to obtain new solutions (i.e. bee 2, tabu search and shifting bottleneck) instead of constructing new solutions from scratch (i.e. bee 1 and ant colony) performs better in terms of makespan and computational time. The strategy of using tabu list to avoid local optimums is observed to be more effective as compared to probabilistic-based approaches. Further, job shop problem related knowledge such as neighborhood structure in tabu search and bee 2, and machine bottlenecks in shifting bottleneck can result in significant improvement in the performance of heuristics.

With the performance of bee colony heuristics comparable to well known heuristics such as tabu search and shifting bottleneck, we intend to apply the bee colony heuristic to more realistic semiconductor manufacturing scheduling problems. One of the works we intend to pursue is to deploy the algorithms in a distributed computing environment using software agents (Low et al. 2005).

REFERENCES

Adams, J., E. Balas, and D. Zawack. 1988. "The Shifting Bottleneck Procedure for Job Shop Scheduling." *Management Science*, Vol.34, No.3, 391-401.

Baker, K.R. 1984. "Sequencing Rules and Due-date Assignments in a Job Shop." *Management Science*, Vol.30, No.9, 1093-1104.

Balas, E. and A. Vazacopoulos. 1998. "Guided Local Search with Shifting Bottleneck for Job Shop Scheduling." *Management Science*, Vol.44, No.2, 262-275.

Blackstone, J.H., D.T. Phillips and G.L. Hogg. 1982. " A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job-Shop Operations." *International Journal of Production Research* Vol.20, No.1, 27-45.

Blum, C. and M. Sampels. 2004. "An Ant Colony Optimization Algorithm for Shop Scheduling Problems." *Journal of Mathematical Modelling and Algorithms*, Vol.3, No.3, 285-308.

Brucker, P., B. Jurisch and B. Sievers. 1994. "A Branch and Bound Algorithm for the Job Shop Scheduling Problem." *Discrete Applied Mathematics*, Vol.49, No.1, 109-127.

Campos, M., E. Bonabeau, G. Théraulaz and J.-L. Deneubourg. 2001. "Dynamic scheduling and division of labor in social insects." Adaptive Behavior, Vol.8, No.2, 83-92.

Chong, C.S., Y.H. Low, A.I. Sivakumar, and K.L. Gay. 2006. "A Bee Colony Optimization Algorithm to Job Shop Scheduling." in *Proceedings of the 2006 Winter Simulation Conference*,

Dorigo, M., V. Maniezzo and A. Colorni. 1996. "Ant System: Optimization by a Colony of Cooperating Agents." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol.26, No.1, 29-41.

Ganesan, V.K., A.I. Sivakumar, and G. Srinivasan. 2006. "Hierarchical Minimization of Completion Time Variance and Makespan in Jobshops." *Computers & Operations Research*, Vol.33, No.5, 1345-1367.

Hansen, P. and N. Mladenović. 2001. "Variable Neighborhood Search: Principles and Applications." *European Journal of Operational Research*, Vol.130, No.3, 449-467.

Jain. A.S. and S. Meeran. 1999. "Deterministic Job Shop Scheduling: Past, Present and Future."

European Journal of Operational Research, Vol.113, No.2, 390-434.

Low, Y.H., K.W. Lye, P. Lendermann, S.J. Turner, S. Leo and R. Chin. 2005. "An Agent-based Approach for Managing Symbiotic Simulation of Semiconductor Assembly and Test Operations." in *Proceedings of the 2005 International Conference on Autonomous Agent and Multiagent Systems (AAMAS)*, July 25-29, 2005, Utrecht, The Netherlands.

Matsuo, H., C.J. Suh, and R.S. Sullivan. 1988. "A Controlled Search Simulated Annealing Method for the General Job-Shop Scheduling Problem." *Working Paper #03-04-88 (1988)*, Graduate School of Business, The University of Texas at Aus-tin, Austin, Texas, USA.

Nakrani, S. and C. Tovey. 2004. "On Honey Bees and Dynamic Allocation in Internet Hosting Centers." *Adaptive Behavior*, Vol.12, No.3-4, 223-240.

Nowicki, E. and C. Smutnicki. 1996. "A Fast Taboo Search Algorithm for the Job Shop Problem." *Management Science*, Vol.42, No.6, 797-813.

Park, S.C., N. Raman and M.J. Shaw. 1997. "Adaptive scheduling in Dynamic Flexible Manufacturing Systems: a Dynamic Rule Selection Approach." *IEEE Transactions Robotics and Automation*, Vol.13 No.4 , 486-502.

Pinedo, M. 1995. *Scheduling Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, NJ., 1995

Rajendran, C. and O. Holthaus. 1999. "A Comparative Study of Dispatching Rules in Dynamic Flowshops and Jobshops." *European Journal of Operational Research*, Vol.116, No.1, 156-170.

Seeley, T.D., *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*, Harward University Press, Feb. 1996.

Teodorovic, D. and M. Dell'orco. 2005. "Bee colony optimization - A cooperative learning approach to complex transportation problems." *Advanced OR and AI Methods in Transportation*, 51-60.

Ten Eikelder, H.M.M., B.J.M. Aarts, M.G.A. Verhoeven and E.H.L. Aarts. 1997. "Sequential and Parallel Local Search Algorithms for Job Shop Scheduling." In *MIC'97 2nd International Conference on Meta-heuristics*, Sophia-Antipolis, France, Jul 21-24, 75-80.

Van Laarhooven, P.J.M., E.H.L. Aarts and J.K. Lenstra. 1992. "Job Shop Scheduling by Simulated Annealing." *Operations Research*, Vol.40, No.1, 113-125.

AUTHOR BIOGRAPHIES

CHIN SOON CHONG obtained his degree in Electrical and Electronics Engineering from the City University of London, UK. He obtained his Master of Engineering in Computer Integrated Manufacturing from Nanyang Technological University, Singapore. He has been involved in simulation, scheduling and optimization related projects in logistic and manufacturing IT domains. His current research interest includes simulation, planning, scheduling, optimization in the area of manufacturing, logistic, and supply chain. His e-mail address is : cschong@simtech.a-star.edu.sg.

MALCOLM YOKE HEAN LOW is an Assistant Professor in the School of Computer Engineering at the Nanyang Technological University (NTU), Singapore. He received his Bachelor and Master of Applied Science in Computer Engineering from NTU in 1997 and 1999 respectively. In 2002, he received his D.Phil. degree in Computer Science from Oxford University. His current research interests are in the application of parallel/distributed simulation, grid computing and agent technology for the modeling, simulation, analysis and optimization of complex systems. His e-mail address is: yhlow@ntu.edu.sg and his web-page can be found at www.ntu.edu.sg/home/yhlow/.

APPA IYER SIVAKUMAR (Senior member IIE) is an Associate Professor in the School of Mechanical and Production Engineering at Nanyang Technological University (NTU), Singapore, and a Fellow of Singapore Massachusetts Institute of Technology (MIT) Alliance (SMA). His research interests are in the area of simulation-based optimization of manufacturing performance, supply chain, and dynamic schedule optimization. Prior to joining NTU, he held various management positions including technical manager and project manager for nine years at Lucas Systems and Engineering and Lucas Automotive, UK. He received a Bachelors of Engineering from University of Bradford, UK and a PhD in Manufacturing Systems Engineering from University of Bradford, UK. His e-mail address is : msiva@ntu.edu.sg and his web-page can be found at www.ntu.edu.sg/home/MSiva/.

KHENG LENG GAY obtained his B. Eng, M. Eng and PhD degrees at the University of Sheffield, England. He was awarded the Grouped Scholarship in Engineering and Metallurgy by the University of Sheffield from 1967 to 1970. Since obtaining his PhD, he has been involved in Education and R&D working in institutions such as Singapore University (1972-1979), Rutherford and Appleton Laboratory (England, 1979-1982), NTU (1982-1995 and 1999-present) and Singapore Institute of Manufacturing Technology (1989-1999). He has also been actively involved in promoting innovation in Singapore through work in various committees: Science Quiz (MOE), Science Centre Board, National CAD/CAM (NCB), Tan Kah Kee Young Inventors Award (TKK Foundation & NSTB) and so on. Currently Professor Gay is in the School of EEE at NTU. He has more than a hundred publications in journals, conference proceedings and books. His e-mail address is : eklgay@ntu.edu.sg> and his Web-page can be found at www.ntu.edu.sg/eee/icis/cv/robertgay.html.