A Knowledge-Migration-Based Multi-Population Cultural Algorithm to Solve Job Shop Scheduling

Mohammad R. Raeesi N. and **Ziad Kobti**
School of Computer Science, University of Windsor, Windsor, ON, Canada N9B 3P4
raeesim@uwindsor.ca and kobti@uwindsor.ca

Abstract

In this article, a multipopulation Cultural Algorithm (MP-CA) is proposed to solve Job Shop Scheduling Problems (JSSP). The idea of using multiple populations in a Cultural Algorithm is implemented for the first time in JSSP. The proposed method divides the whole population into a number of sub-populations. On each sub-population, a local CA is applied which includes its own population space as well as belief space. The local CAs use Evolutionary Programming (EP) to evolve their populations, and moreover they incorporate a local search approach to speed up their convergence rates. The local CAs communicate with each other using knowledge migration which is a novel concept in CA. The proposed method extracts two types of knowledge including normative and topographic knowledge and uses the extracted knowledge to guide the evolutionary process to generate better solutions. The MP-CA is evaluated using a well-known benchmark. The results show that the MP-CA outperforms some of the existing methods by offering better solutions as well as better convergence rates, and produces competitive solutions when compared to the state-of-the-art methods used to deal with JSSPs.

Introduction

Job Shop Scheduling Problem (JSSP) is a combinatorial optimization problem which is well-known in different areas, specially manufacturing systems. JSSP is the task of scheduling different operations to be processed on different machines. The main goal of this type of problems is minimizing the maximum completion time of all the operations. The maximum completion time of a schedule is also called $makespan$. JSSP is still an open problem. It is proved that the job shop scheduling systems with more than two machines are NP-complete (Garey, Johnson, and Sethi 1976), which means that there is no method capable to find the best solution for all the scheduling problems in an acceptable time.

There are various types of algorithms proposed to deal with JSSPs including heuristic approaches, meta-heuristic methods and Evolutionary Algorithms (EAs). In the area of Evolutionary Computation, there are various algorithms with different versions proposed to solve JSSPs including Genetic Algorithm (GA), hybrid GA, Ant Colony Optimization (ACO), Memetic Algorithm (MA), and Cultural Algorithm (CA). Each method has its own strengths and weaknesses. However, combinations of different types of algorithms work better.

In this article, a new CA is proposed to solve JSSPs. The proposed method incorporate a multipopulation design which is called multipopulation CA (MP-CA). In this design, there are a number of sub-populations incorporating local CAs to cooperate with each other to generate better solutions. The sub-populations communicate with each other by exchanging their extracted knowledge every predefined number of generations.

The structure of this article is as follows. Section present the existing Evolutionary Algorithm introduced in the area of Job Shop Scheduling, which is followed by the definition of the classical JSSPs in Section . Section describes the proposed MP-CA in details, and Section shows the results of evaluating the MP-CA. Finally, the conclusions are represented in section .

Related Work

The application of EAs in JSSPs is first introduced by Lawrence (1985) as a GA. Hasan et al. (2008) combined a GA with different priority rules including Partial Reordering, Gap Reduction, and Restricted Swapping. An ACO method is proposed by Wang, Cao, and Dai (2005), and recently we proposed a MA to solve JSSPs (Raeesi N. and Kobti 2011).

Becerra and Coello (2005) introduced the application of CA in JSSP for the first time. CA, developed by Reynolds (1994), is an EA which extracts knowledge to improve its search mechanism as well as its convergence rate. CA consists of population space and belief space. Population space contains individuals which are evolving to generate the optimal solution. The knowledge is extracted from the best individuals every generation. The extracted knowledge which is also called belief is recorded in the belief space to be used in the next generations to direct the evolutionary process. The link from population space to belief space which sends the best individuals in order to update the extracted knowledge is called acceptance function, and the link from belief space

to population space which sends the updated knowledge to guide the evolution is called influence function.

The CA proposed by Becerra and Coello (2005) incorporates Evolutionary Programming (EP) for population space evolution which only uses the mutation operator to generate the offspring population. The belief space of their proposed method only records the best individual as the situational knowledge. An improved version of the first CA (Becerra and Coello 2005) in JSSP is proposed by Corts, Becerra, and Coello (2007). The newer method is similar to the first version with the main differences being in the implementation of the mutation operator and influence function.

Ho and Tay (2004) proposed a CA to solve Flexible Job Shop Scheduling which is called GENACE. Like (Becerra and Coello 2005), they used EP and situational knowledge. The next version of GENACE is also proposed by Ho and Tay (2005) which is called LEGA.

All of the proposed CAs in JSSP use a single population in their population space. Our proposed method would be the first attempt to use a CA with multiple populations which is called MP-CA. However, there are a number of CAs in different fields which are using multiple populations. Digalakis and Margaritis (2002) introduced the multipopulation concept in CA for the first time by proposing a master-slave design. The master processor generates the initial population and manages it, while the slave processors execute different CAs on different sub-populations. The communication among sub-populations is implemented using Message-Passing Interface (MPI). The latter also used the situational knowledge.

One of the main issues in MP-CAs is the communication among the sub-populations. Most of the existing methods consider exchanging the best individuals as the communication process. But there is a more powerful strategy which exchanges the extracted knowledge instead of the best individuals. The new strategy is called knowledge migration which was first proposed by Guo et al. (2009). The extracted knowledge can incorporate more useful information about the previous generations to be used to direct the evolutionary process. Consequently, it would be more effective to use knowledge migration as the sub-populations communication mechanism.

Problem Definition

Classical JSSP is defined as a process of assigning different jobs to be processed on different machines (Baker 1974). There are M machines denoted by m_k and N jobs denoted by J_i, where k is the machine index and i is the job index. Each job is defined by a fixed sequence of operations. Each operation is denoted by O_{ij} where i is the job index and j is the operation index in that job. Each operation can be processed on only one machine in a known processing time. In other words, the route of operations of each job through the given machines is predefined.

In classical JSSP, there are some assumptions which are as follows: All the jobs are available at the starting point which are independent from each other; the machine set up time and part movement time between machines are negligible; each job is processed only one time on each machine; the machines can process only one operation at a time which cannot be interrupted; and there is no due date for the jobs.

Table 1: A sample classical job shop scheduling problem

Operation Index	1	2	3
J_1	$m_2, 1$	$m_1, 2$	$m_3, 3$
J_2	$m_1, 2$	$m_2, 1$	$m_3, 2$
J_3	$m_1, 2$	$m_3, 4$	$m_2, 1$

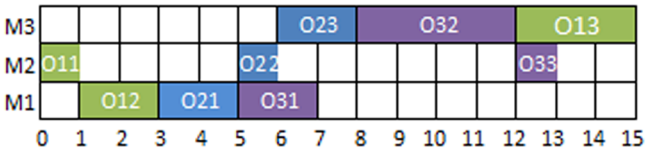

Figure 1: Sample Schedule

Table 1 presents a sample classical JSSP which contains 3 jobs to be processed on 3 machines. This table shows the applicable machine with the corresponding processing time for each operation. The third operation of the second job O_{23} , for example, is applicable on the third machine, and it needs 2 time units to be processed. A sample schedule for this example is shown in Figure 1.

It should be mentioned that we use the active schedule concept defined by Croce, Tadei, and Volta (1995), and used by Hasan et al. (2008) as Gap Reduction rule and by Becerra and Coello (2005) as permissible left shift.

Proposed Cultural Algorithm

In this article, a MP-CA is proposed to deal with JSSPs. The proposed method is the first attempt to incorporate multiple populations in JSSP. In this method, the whole population is divided into a number of sub-populations. Each sub-population incorporates a local CA which contains its own belief space. The local CAs are cooperating to find the optimal solutions. They are communicating with each other by exchanging their own extracted knowledge which is called knowledge migration.

The architecture of the MP-CA is represented in Figure 2. As it is illustrated in the figure, like other CAs, each local CA has its own population space and belief space (Reynolds 1994).

Moreover, the overall framework of the proposed MP-CA is represented in Figure 3. The number of sub-populations, the number of iterations which is the termination criterion, and the migration frequency are denoted by the $SubPopulationsNo$, $IterationNo$, and $ExchangeRate$ parameters, respectively. The parameters values which are used in our experiments are presented in Table 2.

Population Space

Population space is a set of individuals which are evolving using EP. Our EP uses the selection and mutation as regular EP, and incorporates a local search heuristic to speed up the convergence rate.

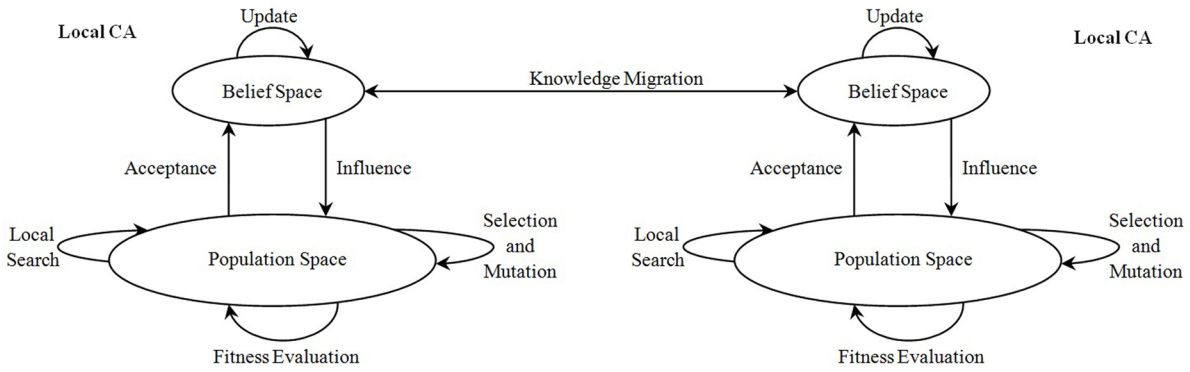

Figure 2: MP-CA Architecture

PROCEDURE: MP-CA Framework
INPUT: Test Problems and Algorithm Parameters
OUTPUT: Optimal or Near-Optimal Schedules

Generate $SubPopulationsNo$ sub-populations.
FOR ($IterationNo$)
  FOR ($each\ subpopulation$)
    Evaluate all individuals and sort them.
    Apply mutation and local search method to generate offspring population.
    Update belief space.
  END
  IF ($IterationNo \bmod ExchangeRate = 0$)
    Exchange knowledge.
  END
END
Output the best found individual so far.

Figure 3: MP-CA Framework

Chromosome Representation Chromosome representation is one of the main characteristics of an EA. In the literature, there are various chromosome representations, with nine of them described by Cheng, Gen, and Tsujimura (1996). Recently, we introduced Machine Operation Lists (MOL) representation (Raeesi N. and Kobti 2011). MOL is an extended version of preference list-based representation such that it adds the concept of fixed list, the operation sequence of which cannot be changed unless due to the permissible left shift. In (Raeesi N. and Kobti 2011), we showed that MOL representation outperforms preference list-based representation by yielding better solutions.

In our proposed algorithm, we incorporate the MOL representation. MOL considers a list of operations for each machine determining the sequence of operations to be processed on that machine. Because each machine only processes one operation of each job, the operations in the list can be denoted by their job indices. For example, the sample schedule illustrated in Figure 1 is represented as follows.

$$\{(1,2,3),(1,2,3),(2,3,1)\}$$

Evolutionary Programming The proposed method uses EP to evolve the population space. The EP incorporates only the mutation operator as a genetic operator. The mutation operator uses the knowledge recorded in belief space to influence the direction of evolution. The EP applies the mutation operator on all the individuals to generate new ones. In our proposed MP-CA, there are two mutation operators which will be described in details in subsection .

Local Search After generating the offspring population, a number of best individuals will be selected to be investigated by a local search heuristic. We use the same local search method as we used before (Raeesi N. and Kobti 2011) which is compatible with MOL representation. The search method reassigns all the operations of a randomly selected job to decrease the makespan. It has been shown by the authors that the time complexity of the local search method is negligible compared to the number of fitness evaluations in each generation.

Belief Space

Each local CA has its own belief space which space gets updated every generation using the acceptance function. The acceptance function passes a number of best individuals from the sub-population to the belief space. We consider the top 20 percent of the individuals in our implementation. The belief space extracts both normative and topographic knowledge from the individuals, and updates its own belief by the extracted knowledge. The knowledge stored in the belief space is incorporated to direct the mutation operator using the influence function.

The local CAs migrate their knowledge to each other to improve the search exploration in different sub-populations. The knowledge migration occurs every predefined number of generations. In knowledge migration, the sub-population which finds the best individual so far sends its own knowledge to others. The sub-populations which receive the migrated knowledge replace their own knowledge with migrated one.

Normative Knowledge We use normative knowledge to improve the search exploration of our proposed method. Since normative knowledge records the feasible search space, it is used to explore all the feasible search space uniformly. In our method, we consider the position of each operation in its corresponding machine list as the search variable. So we have $M \times N$ variables for a system with M machines and N jobs. For each operation O_{ij}, there are a lower position $L_{O_{ij}}$ and an upper one $U_{O_{ij}}$. The normative knowledge is initialized using the best individual as follows:

$$\begin{aligned} L_{O_{ij}} &= P_{O_{ij}} - 0.5 \\ U_{O_{ij}} &= P_{O_{ij}} + 0.5 \end{aligned}$$

where $P_{O_{ij}}$ denotes the position of operation O_{ij} in its corresponding machine list in the best individual at the first iteration. The 0.5 is used to provide a range of 1 for each position. The normative knowledge gets updated every generation by each individual passed by acceptance function using:

$$\begin{aligned} L_{O_{ij}} &= min\left(L_{O_{ij}}, P_{O_{ij}} - 0.5\right) \\ U_{O_{ij}} &= max\left(U_{O_{ij}}, P_{O_{ij}} + 0.5\right) \end{aligned}$$

where $P_{O_{ij}}$ denotes the position of operation O_{ij} in its corresponding machine list in that individual.

Topographic Knowledge We use the topographic knowledge to exploit certain regions which have more individuals. The topographic knowledge for each operation keeps the record of all positions used by all the individuals passed through the acceptance function. For each operation, there is a list of N available positions where N is the number of jobs. The topographic knowledge counts the number of occurrences for each position. Since the greater number of occurrences determines the existence of more good individuals in that region, the topographic knowledge is used to exploit those regions. Like the normative knowledge, topographic knowledge is also updated every generation, and it has its own mutation operator which influences the evolution to exploit certain regions.

Influence Function Each type of knowledge has its own mutation operator, one mutation for normative knowledge and another one based on topographic knowledge. To generate a new individual, each machine list of an existing individual is mutated using one of both mutation operators which is selected randomly with the same chance.

The normative-knowledge-based mutation operator calculates a position for each operation using the following formula, and then it sequences the operations in their operation lists based on their calculated position values.

$$P_{O_{ij}} = \begin{cases} P_{O_{ij}} + R \times \left(U_{O_{ij}} - L_{O_{ij}}\right) & P_{O_{ij}} < L_{O_{ij}}, \\ P_{O_{ij}} - R \times \left(U_{O_{ij}} - L_{O_{ij}}\right) & P_{O_{ij}} > U_{O_{ij}}, \\ P_{O_{ij}} + G & otherwise, \end{cases}$$

where R is a random number uniformly distributed between 0 and 1, and G is a random number in a normal distribution with mean 0 and variance 1.

In topographic knowledge, there is a list of position occurrences for each operation such that we need to use the position with greater occurrence to generate new individuals. Here we use roulette-wheel selection strategy to choose the new position for each operation.

Table 2: Parameters of the proposed algorithm

Parameters	Value
$SubPopulationsNo$	7
$SubPopSize$	142
$IterationNo$	200
$ExchangeRate$	20

Table 3: Sample Results on LA Benchmark

Problem	Algorithm	Best	Median	Worst
la02	CA	**655**	660.5	667
655	MP-CA	**655**	655.0	655
la20	CA	907	912.6	924
902	MP-CA	**902**	907.0	907
la40	CA	1256	1277.4	1328
1222	MP-CA	1228	1234.0	1245

Results

The proposed algorithm is implemented and evaluated using the java programming language version 1.6.0.18 on a system with Intel(R) Core(TM)2Quad 2.50GHz CPU and 8.00GB RAM. Table 2 presents the parameters used in our experiments which are adjusted using extensive experiments. In our experiments, we generate 7 sub-populations with 142 individuals, approximately 1000 individuals in total. We run the method for 200 iterations such that the knowledge migration occurs every 20 iterations.

In our experiments, we consider a well-known benchmark (Lawrence 1984) for classical JSSP. The benchmark consists of 40 different problems with different size. All the experiments are done 10 times independently.

The experiment results show that the proposed MP-CA finds the optimal solution for 28 test problems out of 40. To show the performance of the proposed method, we compare our method with another CA recently published by Corts, Becerra, and Coello (2007). The authors claim that their method finds the optimal solution for 26 test problems. However, our MP-CA outperforms their method by finding the best solution for 2 more test problems as well as by offering better statistical results (lower median and worst solutions) for those 26 test problems which means that the convergence rate of our proposed method is better than theirs. Moreover, for the remaining test problems, our method offers better solutions comparing to theirs, for all the best, median and worst solutions. Table 3 shows three sample results of our proposed MP-CA compared to the result of Corts, Becerra, and Coello (2007)[1]. The results show that incorporating multi-population and using knowledge exchange offers better solution and improves the convergence rate.

To have a fair comparison, we compare our results with their results for 200,000 fitness evaluations, not for more than 2,000,000 evaluations (Corts, Becerra, and Coello 2007). The maximum number of evaluations in our method is 200,000 which is 1000 evaluations in 200 iterations.

[1]Please refer to http://cs.uwindsor.ca/∼raeesim/Flairs-25/Statictical-Analysis.pdf to see the complete results.

Table 4: Comparison among Different Evolutionary Algorithms proposed recently to solve JSSP on LA Benchmark

Problem	Size	Best Known	Hybrid GA	MA	CA	Proposed MP-CA
LA20	10×10	**902**	907 (0.55%)	907 (0.55%)	907 (0.55%)	**902** (0.00%)
LA21	15×10	**1046**	1046* (0.00%)	1057 (1.05%)	1059 (1.24%)	1048 (0.19%)
LA22	15×10	**927**	935 (0.86%)	935 (0.86%)	947 (2.16%)	932* (0.54%)
LA24	15×10	**935**	953 (1.93%)	944 (0.96%)	950 (1.60%)	943* (0.86%)
LA25	15×10	**977**	986 (0.92%)	983* (0.61%)	998 (2.15%)	983* (0.61%)
LA26	20×10	**1218**	**1218** (0.00%)	**1218** (0.00%)	1219 (0.08%)	**1218** (0.00%)
LA27	20×10	**1235**	1256* (1.70%)	1269 (2.75%)	1279 (3.56%)	1264 (2.35%)
LA28	20×10	**1216**	1232 (1.32%)	1223 (0.58%)	1236 (1.64%)	1219* (0.25%)
LA29	20×10	**1157**	1196 (3.37%)	1191 (2.94%)	1219 (5.36%)	1182* (2.16%)
LA36	15×15	**1268**	1279 (0.87%)	1281 (1.03%)	1296 (2.21%)	1274* (0.47%)
LA37	15×15	**1397**	1408* (0.79%)	1429 (2.29%)	1416 (1.36%)	1415 (1.29%)
LA38	15×15	**1196**	1219 (1.92%)	1208 (1.00%)	1231 (2.93%)	1202* (0.50%)
LA39	15×15	**1233**	1246 (1.05%)	1248 (1.22%)	1269 (2.92%)	1240* (0.57%)
LA40	15×15	**1222**	1241 (1.55%)	1234 (0.98%)	1256 (2.78%)	1228* (0.49%)
Average Error Rate			1.20%	1.20%	2.18%	0.73%
Average Ranking (Friedman)			2.39	2.46	3.79	1.36

Furthermore, different Evolutionary Algorithms are considered to be compared with our MP-CA including a hybrid GA (Goncalves, de Magalhaes Mendes, and Resende 2002), our published MA (Raeesi N. and Kobti 2011), and the CA (Corts, Becerra, and Coello 2007). All the four algorithms find the optimal solution for certain 26 test problems including la01-la19, la23, and la30-la35. Table 4 represents the results for the remaining 14 test problems. The Error Rate (ER) is considered to be able to compare the results over all the 14 problems which is defined as follows.

$$ER = \frac{C - LB}{LB} \times 100\%$$

where LB is the best-known solution, and C is the best solution found by the algorithms. The ER values are represented in brackets in Table 4. To compare the 4 algorithms on the 14 problems, we used the non-parametric procedure incorporated by Garca et al. (2009) with the same levels of significance as theirs which are $\alpha = 0.05$ and $\alpha = 0.10$. To do so, first the algorithms are ranked using Friedman's test. The p-value for this test is less than 0.0001 which is also less than both significance levels; this means there are significant differences between the compared algorithms. Then, the algorithm with the minimum average ranking is selected as the control algorithm which is our MP-CA. Afterwards, the critical differences (CD) for control algorithm MP-CA are calculated using Bonferroni-Dunns test which are:

$$CD = 1.17 \quad \text{for} \quad \alpha = 0.05$$
$$CD = 1.09 \quad \text{for} \quad \alpha = 0.10$$

Now, the algorithms with the average rankings greater than the summation of the control algorithm ranking and CDs are considered as the algorithms with worse performance than the control algorithm. In other words, the summation of the average ranking of the MP-CA and the CD for each significance level is a threshold to find the worse algorithms. So the thresholds are as follows.

$$Threshold = 2.53 \quad \text{for} \quad \alpha = 0.05$$
$$Threshold = 2.45 \quad \text{for} \quad \alpha = 0.10$$

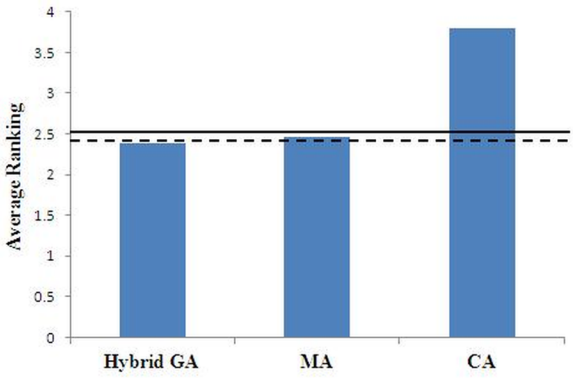

Figure 4: The Graphical Representation of Statistical Result of Friedman's test and Bonferroni-Dunn's method

Figure 4 represents this concept graphically. The thresholds for $\alpha = 0.05$ and $\alpha = 0.10$ are illustrated using a solid line and a dashed line, respectively. This figure shows that the control algorithm MP-CA outperforms the algorithms whose bar exceeds the threshold line. So based on the Friedman's test and Bonferroni-Dunn's approach, we can claim that our proposed MP-CA outperforms the CA proposed by Corts, Becerra, and Coello (2007) with significance level $\alpha = 0.05$ and outperforms our published MA (Raeesi N. and Kobti 2011) with level $\alpha = 0.10$. However, while the average ER and average ranking of the proposed MP-CA are better than the hybrid GA (Goncalves, de Magalhaes Mendes, and Resende 2002), these statistical tests show that the differences are not significant enough to say their performance is worse than our method's performance.

Finally, we compare our algorithm with others mutually over all the 40 problems. The comparison results illustrated in Table 5 shows that, for example, the MP-CA works similar to the hybrid GA (Goncalves, de Magalhaes Mendes, and Resende 2002) for 27 problems, outperforms it for 10

Table 5: Overall Mutual Comparison

Proposed MP-CA	Win	Tie	Lose
HGA	10	27	3
MA	12	28	0
CA	14	26	0

problems, and works worse for the remaining 3 problems.

Conclusions

In this article, we propose a MP-CA to solve JSSP which is a combinatorial optimization problem proved to be NP-Complete. This article introduces application of the MP-CA in JSSPs for the first time. Using multipopulation we improve the search exploration. The proposed method incorporates local CAs on the sub-populations cooperating to find the optimal solution. In addition, it uses knowledge migration among sub-populations as the communication link. In the proposed method, we consider the extraction of normative and topographic knowledge.

The experiments show that the proposed MP-CA offers better solutions. Moreover, it improves the convergence rate. The comparison of the MP-CA with the traditional CA shows that using multiple populations as well as incorporating knowledge migration enhance the search space exploration and help the algorithm to avoid trapping into local optimal solutions.

The statistical comparisons reveal that the proposed method outperforms some existing methods very well and offers competitive solutions compared to other state-of-the-art methods. Furthermore, while the statistical comparisons do not show the significant differences between the proposed MP-CA and those methods, the results show that it outperforms others by offering lower error rates.

The proposed method is also applicable for other types of JSSPs, but since it uses the MOL representation and this representation does not cover other JSSP types, we need to use another representation or provide an extension for MOL, in future work.

Acknowledgments

This work is made possible by a grant from the National Science Foundation and NSERC Discovery No. 327482.

References

Baker, K. R. 1974. *Introduction to Sequencing and Scheduling*. Wiley.

Becerra, R., and Coello, C. 2005. A cultural algorithm for solving the job-shop scheduling problem. In *Knowledge Incorporation in Evolutionary Computation, Studies in Fuzziness and Soft Computing*, volume 167, 37–55. Springer.

Cheng, R.; Gen, M.; and Tsujimura, Y. 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms i: representation. *Computers and Industrial Engineering* 30 (4):983–997.

Corts, D.; Becerra, R.; and Coello, C. 2007. Cultural algorithms, an alternative heuristic to solve the job shop scheduling problem. *Engineering Optimization* 39 (1):69–85.

Croce, F.; Tadei, R.; and Volta, G. 1995. A genetic algorithm for the job shop problem. *Computers in Operations Research* 22:15–24.

Digalakis, J., and Margaritis, K. 2002. A multipopulation cultural algorithm for the electrical generator scheduling problem. *Mathematics and Computers in Simulation* 60(3-5):293–301.

Garca, S.; Molina, D.; Lozano, M.; and Herrera, F. 2009. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics* 15:617–644.

Garey, M. R.; Johnson, D. S.; and Sethi, R. 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research* 1:117–129.

Goncalves, J.; de Magalhaes Mendes, J.; and Resende, M. G. C. 2002. A hybrid genetic algorithm for the job shop scheduling problem. Technical Report TD-5EAL6J, AT&T Labs.

Guo, Y.-N.; Cao, Y.-Y.; Lin, Y.; and Wang, H. 2009. Knowledge migration based multi-population cultural algorithm. In *Fifth International Conference on Natural Computation (ICNC 09)*, 331–335.

Hasan, S.; Sarker, R.; Essam, D.; and Cornforth, D. 2008. Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing* 1:69–83.

Ho, N., and Tay, J. 2004. Genace: An efficient cultural algorithm for solving the flexible job-shop problem. In *IEEE Congress on Evolutionary Computation (CEC)*, 1759–1766.

Ho, N., and Tay, J. 2005. Lega: an architecture for learning and evolving flexible job-shop schedules. In *IEEE Congress on Evolutionary Computation (CEC)*, 1380–1387.

Lawrence, S. 1984. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Master's thesis, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Lawrence, D. 1985. Job shop scheduling with genetic algorithms. In *First international conference on genetic algorithms*, 136–140.

Raeesi N., M. R., and Kobti, Z. 2011. A machine operation lists based memetic algorithm for job shop scheduling. In *IEEE Congress on Evolutionary Computation (CEC)*.

Reynolds, R. G. 1994. An introduction to cultural algorithms. In Sebald, A. V., and Fogel, L. J., eds., *Thirs Annual Conference on Evolutionary Programming*, 131–139. River Edge, New Jersey: World Scientific.

Wang, C.; Cao, Y.; and Dai, G. 2005. Bi-directional convergence aco for job-shop scheduling. *Computer Integrated Manufacturing Systems* 10(7):820–824.