KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 153–160

GENETIC CHROMODYNAMICS FOR THE JOB SHOP SCHEDULING PROBLEM

D. DUMITRESCU$^{(1)}$, CATALIN STOEAN $^{(2)}$, AND RUXANDRA STOEAN $^{(2)}$

Abstract. A novel evolutionary computing approach to the job shop scheduling problem is proposed. The new technique is based on a recent metaheuristic which, through its nature and mechanisms, manages to maintain the diversity in the population, fact that conducts to the discovery of several local optima and, thus, avoids the blockage of the entire population into a local optima region, which represents a major concern when designing an algorithm for scheduling. Furthermore, the aim of present paper is to find multiple optimal schedules within one problem and thus bring several options to its managers. The newly evolutionary approach to the job shop scheduling problem is validated on an instance, which is sufficient to be considered NP-hard. Results demonstrate the success of this first attempt and prove the promise of the new approach.

1. Introduction

Scheduling is a well-known problem that deals with the efficient allocation of resources with respect to time in order to perform a collection of tasks. Scheduling problems appear in many real life situations. In manufacturing, tasks correspond to parts that need to be processed on a set of machines. In hospitals, tasks are patients and resources are doctors, nurses, hospital beds or medical equipment. In education, tasks are classes and resources can be teachers, classrooms, and students. Finally, in transportation, problems include material transportation, airport terminal scheduling, train scheduling [7].

The generalized version of scheduling is the job shop scheduling problem (JSSP) and is a widely studied NP-hard problem. Various solutions have been brought to JSSP, among which evolutionary algorithms (EA) make a significant part.

In present paper, a new method based on a recently developed evolutionary metaheuristic, called genetic chromodynamics (GC) [2], is proposed as a solver of JSSP. The novel technique is very suitable for the given task due to its multimodal

2000 *Mathematics Subject Classification.* 68T20, 68T05.
Key words and phrases. job shop scheduling, evolutionary computation, multimodality, genetic chromodynamics.

nature that is both able to direct search to multiple optimal regions of the solutions space and to escape local optima.

Experiments are conducted on an 3×3 instance of JSSP and results validate the assumptions. However, this is only a first attempt with the new technique. Many of its components and corresponding values still remain to be improved. Also, validation on other examples has to be continued.

The paper is structured as follows. Section 2 gives the definition of JSSP, while section 3 briefly describes some evolutionary approaches to the problem. Section 4 presents the support evolutionary metaheuristic, whereas section 5 outlines the new approach to JSSP. Section 6 presents the experimental results and the paper ends with conclusions and ideas for future work.

2. Job Shop Scheduling Problem

The JSSP can be enunciated in the following manner. Suppose there are m manufacturing machines, M_i, $i = 1, 2, ..., m$, and n jobs to be performed, J_k, $k = 1, 2, ..., n$. Each job has an ordering of m tasks of specified duration, T_{ki}, $k = 1, 2, ..., n$, $i = 1, 2, ..., m$ and each task must be performed using the corresponding machine. Each job visits each machine exactly once. The tasks of each job have to be performed in the specified order only. A machine can perform only one task at a time and cannot be interrupted. The task is to make such a *schedule* that minimizes the time that is required to process all jobs. An example of a JSSP is outlined in the beginning of section 6.

3. Evolutionary Computing Approaches to Job Shop Scheduling Problem

The crucial issue in any technique that addresses JSSP is represented by the blockage into local optima. Consequently, some mechanism to handle this problem has to be implemented in any such approach. In addition, the ability of a heuristic to find a means to discover multiple optimal schedules would constitute an advantage.

Apart from deterministic techniques ranging from the classical backtracking to simulated annealing or tabu search, several heuristics and metaheuristics from the field of EAs (which are half deterministic, half probabilistic algorithms) have been proposed to solve JSSP.

An EA encodes the space of candidate solutions to the problem to be solved into a population of *individuals*. The values for the *genes* of the individuals in the initial population are randomly chosen. Subsequently, through the means of a fitness function that measures the quality of individuals (and thus the quality of solutions) and through the use of selection for reproduction and variation operators (crossover and mutation), candidate solutions evolve and, in the end, reach the optimum. The optimum can be considered as the best individual from the last generation or the best individual from all generations [4], [5].

We will enumerate only a number of the evolutionary attempts to JSSP. One approach was from the travelling salesman problem point of view [10]. In a different view, a simple representation for individuals is chosen and it is interpreted by a schedule builder and specialized variation operators are employed [10]. Other attempts prefer to incorporate problem specific knowledge both into operators and representation [1], [6]. In other methods, a disjunctive graph model is used where each arc is binary labelled to define ordering and thus a schedule can be represented by a binary string and evolved by an EA with corresponding representation. A recent algorithm uses a representation which is much simpler from an EA point of view [11]. As it is very often the case that a machine has to choose among many jobs that are in queue, an individual encodes the machines and, for each of them, a vote which points for the job in the line that wins the right to use the machine. The JSSP is then solved by evolving genetic algorithms by a particular genetic programming method.

4. Genetic Chromodynamics Metaheuristic

Generally speaking, if, for some problem, multiple optimal solutions are possible then a multimodal EA may be employed in order to get the whole picture of possibilities before making a decision. Moreover, multimodal evolutionary heuristics can be used even for unimodal tasks as they have the ability to escape local optima.

The novel GC [2] has demonstrated its suitability with respect to problems exhibiting issues discussed above through the numerous results obtained by its application to different types of problems like function optimization, clustering or classification [2], [3], [8], [9].

GC represents a multimodal evolutionary metaheuristics based on radii. Only individuals that are similar under a given radius can recombine. Moreover, individuals that resemble each other under another specified radius are merged.

Algorithm 1 Merging procedure within GC

```
repeat
    A chromosome $c$ is considered to be the current one;
    Select all individuals in the merging region of $c$, including itself;
    Remove all but the best chromosome from the selection;
until the merging region of each chromosome remains empty
```

Depending on the encoding for the individuals, some distance between them has to be defined (Hamming for binary encoding, Euclidean or Manhattan or any other distance for real encoding). Within one generation, distance is computed twice - once when the selection for mating is done and once when the merging process takes place.

Evolution in GC takes place as in Algorithm 2. The initial population is randomly generated. Each individual is then taken into account for forming the new generation. For each such stepping stone, all individuals in the mating region are found and one of them is selected for recombination by means of proportional selection. Recombination takes place and competition for survival of the fittest is held between the offspring and the first parent only. If there are no individuals in the mating region of the current individual then the current individual suffers mutation. Obtained mutated offspring is inserted in the population and takes the place of its parent only if it is better than the current individual. Before proceeding to the next generation, the merging procedure is applied, so that unfit individuals are removed from the population (Algorithm 1).

Algorithm 2 GC Algorithm

```
Initialize population;
while termination condition is not satisfied do
  Evaluate each chromosome;
  for all chromosomes $c$ in the population do
    if mating region of $c$ is empty then
      Apply mutation to $c$;
      if obtained chromosome is fitter than $c$ then
        Replace $c$;
      end if
    else
      Select one chromosome from the mating region of $c$ for crossover;
      Obtain and evaluate one offspring;
      if offspring is fitter than $c$ then
        Replace $c$;
      end if
    end if
  end for
  Merging
end while
```

As a consequence of the GC mechanisms, subpopulations appear and, with each iteration, they become more and more separated. Depending on the choice of values for the two GC radii, sooner or later each subpopulation contains only one individual that suffers only mutation, leading thus to refinements in the final solutions which are, in the end of the algorithm, each connected to an optimum point in the search space.

5. Genetic Chromodynamics for Job Shop Scheduling Problem (GCJSSP)

The new evolutionary technique for the JSSP acts in the following manner. The multiple optimal possible schedules are encoded into EA individuals in an adapted manner of [11] and evolved through GC.

In what follows, the choice for the evolutionary components with respect to JSSP is outlined.

5.1. **Representation of Individuals.** Each individual in the population contains a number of $m \times n$ genes. The i-th group made of n genes corresponds to machine M_i , $i = 1, 2, ..., m$, and the genes of a group encode votes for jobs that are in queue for M_i. This means that, in case there are multiple jobs - say for instance J_p and J_q - that both start with the task that has to be performed by M_i only, the choice between them is taken by selecting the job corresponding to the maximum value between c_{ip} and c_{iq}.

An individual thus has the form:

$$c = (c_{11}c_{12}...c_{1n}|c_{21}c_{22}...c_{2n}|...|c_{m1}c_{m2}...c_{mn}) \tag{1}$$

Consequently, an individual encodes a schedule, with the succession of jobs given by votes, as will be illustrated by an example in the following subsection. Initially, values for genes are randomly generated using a uniform distribution in the interval [0, 1].

5.2. **Fitness Function.** The quality of an individual c is associated with the total time it takes for all jobs J_k, $k = 1, 2, ..., n$, to be performed using the schedule provided by c and the given duration of each task. As the fitness of individuals corresponds to the time necessary for all jobs to be completed, the task for GCJSSP is to minimize evaluations; we deal therefore with a minimization problem.

In order to outline the way in which quality is computed, we consider a JSSP instance (Figure 1 (a)) and we carry out the evaluation of a certain randomly generated individual $c = (0.5, 0.7, 0.9|0.2, 0.8, 0.3|0.4, 0.9, 0.6)$. There are three jobs which each consist of three tasks to be performed by three corresponding machines. The duration of each task may be found enclosed in parentheses.

Initially, only machines M_1 and M_2 can start working as there is no task for M_3 in the beginning of any job. On the other hand, M_1 may handle either J_1 or J_2; the values corresponding to M_1 in c are 0.5, 0.7 and 0.9 respectively. As M_1 has to choose between J_1 and J_2, J_2 is the job selected first as the second value, 0.7, is greater than 0.5. At the same time, M_2 starts the only job it has in line, which is J_3(Figure 1 (a), (b)).

Machine M_1 finishes the task first (after 2 units of time, e.g. minutes) and is next free to start its task in another job, while machine M_2 still has one minute to go until its first task for job J_3 is done. Until now, the total time spent measures 2 minutes.

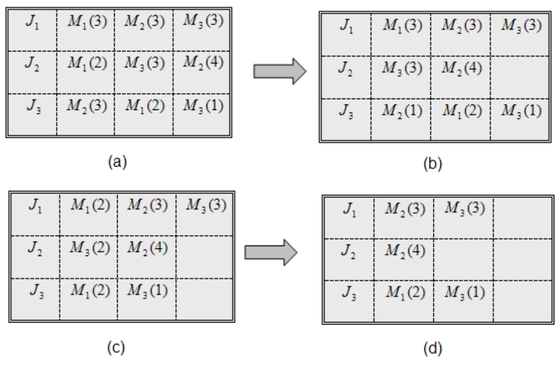

FIGURE 1. An illustration of fitness computation within GCJSSP

At the current moment, M_1 and M_3 are ready to begin another job. Naturally, M_1 selects job J_1 and M_3 starts the task of J_2. After 1 minute, M_2 finishes its task and has to wait as the current tasks of the other jobs J_1 and J_2 are still undertaken by the other two machines for the next 2 minutes.

Figure 1 (c) shows M_1 and M_3 executing their current tasks, while M_2 waits. At the step illustrated in Figure 1 (d), all three machines are available. M_1 naturally starts job J_3, while M_2 has to pick between J_1 and J_2. The latter is selected as its corresponding value in the individual's representation, 0.8, is higher than 0.2. Total time reached 5 minutes.

After two more minutes, M_1 finishes and M_3 starts the last task of J_3. Machine M_3 finishes the final task of J_3 before it ends J_2's last task. After one more minute, M_2 finally selects job J_1 and only after this task is completed, M_3 begins its last task. All jobs are eventually completed. The total time reaches 15 minutes and represents the quality of individual c.

5.3. **Variation operators.** Intermediate crossover and mutation with normal perturbation [4], [5] were experimentally considered for reproduction.

5.4. **Stop condition.** The evolutionary process stops after a predefined number of generations. The last population gives the multiple optimal individuals that are decoded into the corresponding schedules.

6. EXPERIMENTAL RESULTS

An experimental environment was set up and results of GCJSSP were collected. The choices of a specific problem, values for parameters of GC and obtained results are further on presented.

TABLE 1. Manual choice of the values for GCJSSP parameters

Parameters	**Values**
Initial population size	50
Number of generations	100
Mating radius	0.1
Merging radius	0.15
Mutation probability	0.4
Mutation strength	0.15

6.1. **JSSP instance.** The new approach was validated on the JSSP 3×3 problem presented above [12]. The representation encodes the routing of each job, J_k, $k = 1, 2, ..., n$ ($n = 3$ here), through each machine, M_i, $i = 1, 2, ..., m$ ($m = 3$ in this case), and the processing time for each task, T_{ki}, $k = 1, 2, ..., n$, $i = 1, 2, ..., m$, which is written in parentheses. 3×3 JSSP instances have been demonstrated to be already NP-hard.

6.2. **Experimental Setup.** The values for the GCJSSP parameters are depicted in Table 1. Note that, in order to find these values, manual tuning was performed.

6.3. **Results.** GCJSSP was applied to the problem instance for 30 runs. In each run, results gave multiple possible configurations for schedules of total time equal to 12.0 (which is in fact the optimal time for the considered problem stated in [12]). One obtained individual (schedule) is given in Figure 2.

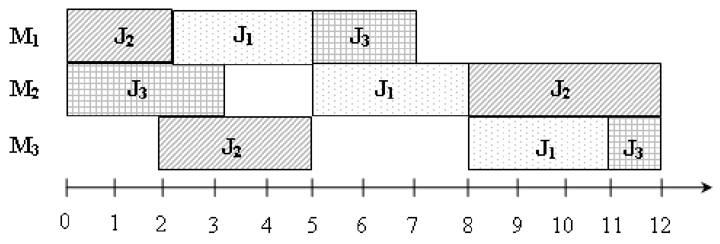

FIGURE 2. Representation of a solution to the chosen 3×3 problem reached by GCJSSP

7. CONCLUSIONS AND FUTURE WORK

Present paper addresses a novel evolutionary approach to the job shop scheduling problem. The technique is based on a recent powerful metaheuristic that is able to cope with the local optima issues and, at the same time, to find multiple optimal configurations of schedules.

Experiments are conducted on an instance of JSSP, whose dimensionality suffices to say the problem is NP-hard. Results demonstrate the suitability of the new approach. Nevertheless, proposed technique is here at its first attempt, thus other choices for parameters and their values together with higher dimensional instances of JSSP have to be investigated in the near future.

References

[1] Bagchi, S., Uckun, S., Miyabe, Y. and Kawamura, K., "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling", Proceedings of the 4th International Conference on Genetic Algorithms, 1991, pp. 10-17.

[2] Dumitrescu, D., "Genetic Chromodynamics", Studia Universitatis Babes-Bolyai, Informatica, 2000, pp. 39 - 50.

[3] Dumitrescu, D., Gorunescu, R., (2004), "Evolutionary clustering using adaptive prototypes", Studia Univ. Babes - Bolyai, Informatica, Volume XL IX, Number 1, 2004, pp. 15 - 20.

[4] Dumitrescu, D., Lazzerini, B., Jain, L., C., Dumitrescu, A., *Evolutionary Computation*, CRC Press, Boca Raton, Florida, 2000

[5] Eiben, A. E., Smith J. E., *Introduction to Evolutionary Computing*, Springer-Verlag Berlin Heidelberg, 2003

[6] Husbands, P., Mill, F., Warrington, S., "Genetic Algorithms, Production Plan Optimization and Scheduling", Proceedings of Parallel Problem Solving from Nature I, 1991, pp. 80-84.

[7] Sadeh, N., *Look-ahead techniques for micro-opportunistic job shop scheduling*, PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1991.

[8] Stoean, C., Preuss, M., Gorunescu, R., Dumitrescu, D., "Elitist Generational Genetic Chromodynamics - a New Radii-Based Evolutionary Algorithm for Multimodal Optimization", Proceedings of the 2005 IEEE Congress on Evolutionary Computation - CEC 2005, Edinburgh, UK, September 2-5, 2005, pp. 1839 - 1846.

[9] Stoean, C., Gorunescu, R., Preuss, M., Dumitrescu, D., "An Evolutionary Learning Classifier System Applied to Text Categorization", Annals of University of Timisoara, Mathematics and Computer Science Series, vol. XLII, special issue 1, 2004, pp. 265-278.

[10] Syswerda, G., "Schedule Optimization Using Genetic Algorithms", Handbook of Genetic Algorithms, Davis, L. (Ed), 1991, pp. 332-349.

[11] Werner, J., *Evolving Genetic Algorithm for Job Shop Scheduling Problems*, Technical Report, School of Computing, Information Systems and Mathematics, South Bank University, 2000

[12] Yamada T., Nakano R., "Genetic Algorithms for Job-Shop Scheduling Problems", Proceedings of Modern Heuristic for Decision Support, Unicom seminar, London, 1997, pp. 67-81.

(1) Babes - Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, Str. M. Kogalniceanu, No. 1, Cluj-Napoca, 400084, Romania

E-mail address: `ddumitr@cs.ubbcluj.ro`

(2) University of Craiova, Faculty of Mathematics and Computer Science, Department of Computer Science, Str. Al. I. Cuza, No. 13, Craiova, 200585, Romania

E-mail address: `{catalin.stoean, ruxandra.stoean}@inf.ucv.ro`