Transaction in Scheduling and Hybrid Optimization
ISSN: 1985-9406 Online Publication, June 2010
www.pcoglobal.com/gjto.htm
SO-O14/GJTO

Global Journal on Technology & Optimization
Volume 1, 2010

GA FOR THE RESOURCE SHARING AND SCHEDULING PROBLEM

Gaby Pinto[1], Uriel Israeli[2], Inessa Ainbinder[3], Gad Rabinowitz[3]

[1]Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Israel
[2]Department of Technology Management, Holon Institute of Technology, Israel
[3]Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Israel
Email: pintog@bgu.ac.il, uriel@hit.ac.il, inessaai@bgu.ac.il, rgadi@bgu.ac.il

Received May 2008, Revised September 2008, Accepted March 2009

Abstract
In this paper we consider the resource-sharing and scheduling problem, with makespan minimization as an objective. Although this problem was optimally solved through a customized branch-and-bound algorithm, its complexity motivated the use of heuristics such as genetic algorithms. A previous genetic algorithm used for solving this problem was significantly faster than the branch-and-bound algorithm; however, it suffered from a high rate of infeasible offspring. We propose a new genetic approach, which produces only feasible offspring via a much more compact, genotype representation of the solution. While in the previous genetic algorithm the chromosome consisted of all the solution 0-1 variables (genotype=phenotype), in the new algorithm we define a much smaller chromosome (genotype) that stores sufficient information for efficiently generating a solution for the 0-1 variables (phenotype).

Keywords: genetic algorithm, resource-sharing, scheduling problem, mixed integer linear programming,

1. Introduction

Within the resource-sharing and scheduling problem (RSSP) as modeled by Rabinowitz et al. [1], a set of precedence constrained operations should be scheduled such that the makespan is minimized. Renewable resources (RRs) required for accomplishing the operations should also be scheduled. An operation may share different RRs simultaneously, each of which may be used for a portion of the operation time. Once started, an operation may not be interrupted (i.e., non-preemptive). Each operation can be performed in one of several modes (Elmaghraby [2]), with each mode representing an alternative assignment of different RRs with different durations. Also, operations may have several sequencing alternatives on each RR according to their precedence constraints. The objective of the scheduling problem is to allocate resources to operations and schedule them to minimize the makespan.
The RSSP arises in various practical scenarios such as production scheduling. However, as shown by Samaddar et al. [3,4], the branch and bound (B&B) algorithm, which is the only optimization procedure available for solving the RSSP, was able to find optimal schedules for only small problems within satisfactory runtimes. Efficient and effective heuristic procedures, therefore, must be developed for larger problems.
A heuristic procedure for solving the RSSP has been proposed by Pinto et al. [5]. They proposed a genetic algorithm (GA) which was based on the mixed integer linear programming (MILP) model of Rabinowitz et al. [1], the constraints of which are categorized into two groups, selection-type and time-type constraints. The selection-type constraints guarantee a feasible selection of one mode for each operation and the assignment of tasks for the RRs required by each operation-mode. These constraints contain only 0-1 variables and form the main layer of the model. The time-type constraints involve continuous time variables in addition to the 0-1 variables. Pinto et al. [5] referred to the 0-1 variables and the selection-type constraints as the GA phenotype and proposed a GA based on a phenotype representation (encoding). Once the result of the selection-type constraints and the 0-1 variables is obtained by the GA, production of the time-type constraints and the continuous time variables is easy. This GA suffers from a high rate of infeasible offspring, which is a major issue in the genetic process. Arbitrary crossovers of two feasible solutions often produce infeasible offspring. This can result from one of three possible situations: (1) two operation-modes are assigned as the same resource task number; (2) violation of the precedence constraints of the operation, and (3) gaps of unused resource task numbers. In order to ensure feasibility of the offspring, the GA applies a repair operator to the newly produced offspring.
This paper introduces a new GA approach for solving the RSSP. The stepping stone is producing only feasible offspring by using a genotype representation, successfully employed by Hartmann [6] for solving the multi-mode resource-constrained project scheduling problem (MRCPSP). Adopting this approach, we develop a genotype representation that consists of a mode assignment for each operation and a precedence feasible operation list for each resource. The phenotype, i.e., schedule, related to a genotype is generated using the MILP model for the

RSSP as presented by Rabinowitz et al. [1].
The remainder of this paper is organized as follows. In Section 2, we briefly present the RSSP. Section 3 describes a new GA for the RSSP. Section 4 presents the basic assumptions, definitions, and theorems regarding the feasibility of the offspring. Finally, Section 5 summarizes the study.

2. The RSSP

We consider a problem where a predetermined job requires a set of non-preemptive operations $I = \{i\}$ with precedence relationship constraints defined by the immediate predecessor-successor set S. A set of RRs of the same or different types $R = \{r\}$ is used to perform these operations. Each operation i can be performed using an alternative mode $M_i = \{m\}$. A mode m specifies the subset of resources $R_{i,m}$, as well as the exact timing $t^s_{i,m,r}$ and $t^f_{i,m,r}$ of the use of r during an operation i. An operation may share different resources simultaneously, each of which may be used for a portion of the operation time. Each resource r has L_r sequence locations for its tasks, which are labeled $l = 1,\dots,L_r$. The set D_r defines sequence-dependent delay requirements between operations on a specific resource r. The processing time is assumed to be deterministic and continuous. The problem is to select a single mode for each operation and, accordingly, to allocate and schedule the resources to minimize the makespan time.

3. A New Genetic Agorithm for the RSSP

This section presents a new GA for solving the RSSP. Introduced by Holland [7], GAs play an important role in solving hard optimization problems. Following the law of nature, the GA recombines existing solutions to obtain new ones. The idea is to produce better solutions by selecting and recombining the best solutions of the current generation. For an introduction to GAs, see Goldberg [8] and Chambers [9]. In this paper, j defines the current individual, and the stopping criteria of our GA was a pre-specified number of generations G.

3.1. Genotype Representation

Our main goal was to develop an individual representation that will produce only feasible offspring. With this idea in mind, we used the genotype representation successfully employed by Hartmann [6] to solve the MRCPSP. We made some changes to Hartmann's genotype representation, however, because of the differences between the MRCPSP and the RSSP.
The individual genotype representation $j = (m, V_r, \forall r \in R)$ comprises a precedence feasible operation sequence list $V_r = (i_1,\dots,i_{|I|})$ for performing on each resource $r \in R$ and a mode assignment $m = (m(i_1),\dots,m(i_{|I|}))$. The mode determines which resources will actually perform the operation. If S is the set of the immediate precedence relationship between operations, then for each $i \in I$, there exists a set $S_i = \{i_1,\dots\}$ which is the set of all predecessors of operation i. An operation sequence list is precedence feasible if all the predecessors of an operation are sequenced before this operation in the list. That is, if S_{i_p} is the set of all predecessors of the pth operation in the operation sequence list, then $S_{i_p} \subseteq \{i_1,\dots,i_{p-1}\}$ for $p \in 1,\dots,|I|$. A mode assignment m is a mapping that assigns one of its modes $m(i) \in M_i$ to each operation $i \in I$. We will use the following alternative notation for the individuals:

$$J = \begin{array}{c} \\ m \\ 1 \\ \vdots \\ |R| \end{array}\begin{array}{c} \begin{array}{ccc} 1 & \cdots & |I| \end{array} \\ \begin{pmatrix} m(1) & \cdots & m(|I|) \\ l_{11} & \cdots & l_{1|I|} \\ & \cdots & \\ l_{|R|I} & \cdots & l_{|R||I|} \end{pmatrix} \end{array}$$

Since operation i in mode m is not assigned to all the resources, but only to $R_{i,m} \subseteq R$ in contrast to Hartmann [6], for the sequencing step on each r we consider only the operations that were assigned to it by activating the following operator for each i. For each $r \in R_{i,m}$, the corresponding operation i is marked with an asterisk $(\overset{*}{i})$ in the genotype operation sequence list V_r.
For each genotype $j = (m, V_r, \forall r \in R)$ we produce its phenotype by transforming the genotype into a MILP formulation of the RSSP. The fitness (T) of individual j is then calculated via a customized critical path method.

3.2. Initial Generation

The initial individual for the first generation is produced by repeatedly and randomly selecting the next schedulable operation, choosing its mode, and scheduling the operation as the next task for its resources. This process is repeated until $|J|$ individuals are produced. Each individual is assigned a reproduction probability based on its fitness function value, which in our case is the schedule makespan time.

3.3. Reproduction Probability and Selection

Since the fitness function is of a minimization type, the lower the value, the higher the reproduction probability is expected to be. Therefore, there is a need to apply a transformation function. We construct such a transformation function as follows (Oĝuz and Ercan [10]):

1. Let $f(j) = 1/T$ denote the transformation function value of individual j; $j = 1\dots J$.
2. Find the total value (F) of the generation $F = \sum_{j=1}^{J} f(j)$,
3. Calculate the reproduction probability ($p_j = f(j)/F$) for individual j, $j = 1\dots J$.

Several selection methods were considered, such as the "roulette wheel sampling" method (Goldberg [8]), and all of them follow Darwin's strategy of "natural selection". The selection determines the two parent solutions for "breeding" according to their reproduction probabilities. A parent can be selected more than once or not at all. This selection is repeated $|J|/2$ times per generation.

3.4. Crossover and Mutation

Once the two parent individuals are selected we use the crossover and mutation operators to produce the two offspring. As the genotype representation of our problem is very unique, we had to develop special crossover and mutation operators. The new operators we present here are based on those developed by Hartmann [6].

Two parent individuals are selected for crossover, $j^{P1} = (m^{P1}, V_r^{P1}, \forall r \in R)$ and $j^{P2} = (m^{P2}, V_r^{P2}, \forall r \in R)$. Then two random integers q_1 and q_2 with $1 \le q_1, q_2 \le I$ are generated. From the two parent individuals, two offspring are produced, $j^{O1} = (m^{O1}, V_r^{O1}, \forall r \in R)$ and $j^{O2} = (m^{O2}, V_r^{O2}, \forall r \in R)$. The crossover algorithm contains two stages, the assignment stage and the sequencing stage. The stages for producing offspring j^{O1} are described as follows (offspring j^{O2} is produced in the same way).

<u>Assignment Stage</u> – The mode assignments of the offspring are produced as follows: $m^{O1}(1\ldots q_2) = m^{P1}(1\ldots q_2), m^{O1}((q_2+1)\ldots|I|) = m^{P2}((q_2+1)\ldots|I|)$.

<u>Sequencing Stage</u> – The operation sequence list ($V_r = (i_1,\ldots,i_{|I|})$) for each resource $r \in R$ of the offspring is defined and the operations assigned to resource r are marked by an asterisk as follows:

Set $V^{O1}(1\ldots q_1) = V^{P1}(1\ldots q_1), \forall r \in R$, and the operation sequence list of positions $i = (q_1+1)\ldots|I|$ in $V^{O1}, \forall r \in R$ is taken from V_r^{P2} by the following procedure:

```
For $r \in R$ do: $t = q_1 + 1$;
    For $i = 1$ to $|I|$ do: If $V_r^{P2}(i) \notin V_r^{O1}$ then $V_r^{O1}(t) = V_r^{P2}(i)$ and set $t = t + 1$;
    For $t = 1$ to $|I|$ do: If $(V_r^{O1}(t), m^{O1}(V_r^{O1}(t))) \in R_{i,m}$ then mark $V_r^{O1}(t)$ by an asterisk
End For
```

A mutation approach is applied for some of the newly produced offspring. We used two mutation operators, one for the mode assignment m and another for the operation sequence list $V_r, \forall r \in R$. The offspring to be mutated are chosen by a small mutation probability p_m^{Mu} and p_l^{Mu} for the mode assignment and for the operation sequence list accordingly. The mutation operator for the mode assignment m changes the modes assigned for each operation randomly. For each $i \in I$ with $|M_i| \neq 1$ and a mode $m(i)$, a different or the same mode assignment m is selected. The mutation operator for the operation sequence list $V_r, \forall r \in R$ changes the sequence list of operations on resource r (randomly selected). For all the resource tasks $l = 1\ldots|I-1|$, operation $V_r(l)$ and $V_r(l+1)$ are swapped only if a precedence feasible operation list is received.

4. Fesibility of the Offspring

In this chapter, we provide the basic assumptions, definitions and theorems regarding the feasibility of the offspring. Due to the limits on the size of the paper, the proofs are not provided.

Assumption 1 – For an assignment problem with $I = \{i\}$ and $M_i = \{m\}, \forall i \in I$, all the combinations of modes are legal.

This means that no special interrelation constraints among the operation-modes (i,m)s are considered in this paper.

Assumption 2 – When sequencing a set $I = \{i\}$ of operations with $|S| = \emptyset$ on a single resource ($|R| = 1$), all the permutations are legal.

Definition – A *feasible genotype* is a $j \in J$ with $\phi(j) = true$, where ϕ is a Boolean function on j and the pair $\langle j, \phi \rangle$ represents the genotype and its feasibility status.

The function ϕ receives $true$ when offspring j is feasible, namely, it has a legal precedence operation sequence list $V_r = (i_1,\ldots,i_{|I|})$ for each resource $r \in R$ and a legal mode assignment $m = (m(i_1),\ldots,m(i_I))$. We note that although in most scenarios a feasible genotype will lead to a feasible phenotype, this is not always guaranteed, as will be elaborated later.

Theorem 1 – For an RSSP with $I = \{i\}$, $M_i = \{m\}$, $S \neq \emptyset$ and $|R| \ge 1$, from two feasible parent genotypes ($\phi(j^{P1}) = true$ and $\phi(j^{P2}) = true$), the GA produces two feasible offspring genotypes ($\phi(j^{O1}) = true$ and $\phi(j^{O2}) = true$.

Notice that when the order between two operations is not restricted by S, there may be task sequences of different rs that contradict each other by dictating conflicting orders for performing these two operations. Let $R_{a,b} = R_{a,m_a} \cap R_{b,m_b}$ be the set of shared resources of operations-modes (a, m_a) and (b, m_b) in $j = (m, V_r, \forall r \in R)$.

Definition – A *cross schedule* is an opposite sequence order between two operations-modes on two different resources as follows. For $|R_{a,b}| \ge 2, \{(a,b),(b,a)\} \notin S, r_n, r_k \in R_{a,b}, n \neq k$ the sequence on resource r_n is $V_n = \{\ldots,(a,m_a),\ldots,(b,m_b),\ldots\}$ and the sequence on resource r_k is $V_k = \{\ldots,(b,m_b),\ldots,(a,m_a),\ldots\}$.

Theorem 2 – For an RSSP, the genotype-phenotype transformation algorithm produces a single feasible phenotype (of the earliest time for each operation) from a feasible genotype that does not have any cross schedule.

When a *cross schedule* does exist in a genotype, the existence of a feasible phenotype depends on the specific problem parameters.

5. Summary

In this paper we proposed a solution to the RSSP via a GA that produces only feasible genotype offspring. The GA preserves offspring feasibility by employing a proper problem representation and using suitable genetic operators. The representation of the problem plays a major role in the success of the GA. In addition, the specific crossover and mutation operators that we used guarantee the feasibility of the offspring without loosing the evolutionary nature of the GA. Development of a generic tool for applying our GA to solve RSSPs and conducting empirical experiments is left for future research.

6. References

[1] G. Rabinowitz, A. Mehrez and S. Samaddar, Scheduling model for multi-robot assembly cells, *The International Journal of Flexible Manufacturing Systems* **3**, pp. 149-180, 1991.

[2] S.E. Elmaghraby, *Activity network: project planning and control by network models. John Wiley*, New York, 1977.

[3] S. Samaddar, G. Rabinowitz and A. Mehrez, Resource sharing and scheduling for cyclic production in a computer-integrated manufacturing cell, *Computers & Industrial Engineering* **36** (3), pp. 525-547, 1999.

[4] S. Samaddar, G. Rabinowitz and G.P. Zhang, An experimental analysis of solution performance in a resource sharing and scheduling problem, *European Journal of Operational Research* **165**, pp. 139-156, 2005.

[5] G. Pinto, I. Ainbinder and G. Rabinowitz, A Genetic Algorithm-Based Approach for Solving the Resource Sharing and Scheduling Problem, *Computers & Industrial Engineering Journal*, 57 (3), pp. 1131-1143, 2009.

[6] S. Hartmann, Project scheduling with multiple modes: a genetic algorithm, *Annals of Operations Research* **102**, pp. 111-135, 2001.

[7] H.J. Holland, *Adaptation in natural and artificial systems*, Ann Arbor, Michigan: The University of Michigan Press, 1975.

[8] D.E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Massachussetts, 1989.

[9] L. Chambers, *Practical Handbook of Genetic Algorithms 1, Applications*, CRC Press, Boca Raton, Florida, 1995.

[10] C. Oĝuz and F. Ercan, A Genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks, *Journal of Scheduling* **8**, pp. 323-351, 2005.