ELSEVIER

European Journal of Operational Research 100 (1997) 134–141

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A genetic algorithm for multi-mode resource constrained project scheduling problem

Masao Mori [1], Ching Chih Tseng [2]

Department of Industrial Engineering and Management, Tokyo Institute of Technology, Meguro-ku, Tokyo 152, Japan

Received 11 April 1995; accepted 13 May 1996

Abstract

This article considers a general class of nonpreemptive multi-mode resource-constrained project scheduling problems in which activity durations depend on committed renewable resources (multi-mode time resource tradeoff). We propose a genetic algorithm for these problems and compare it with a stochastic scheduling method proposed by Drexl and Gruenewald. Computational results show that the proposed genetic algorithm is superior to the stochastic scheduling method. © 1997 Elsevier Science B.V.

Keywords: Resource-constrained project scheduling; Direct chromosome representation; Genetic operators

1. Introduction

A traditional resource-constrained project scheduling problem is composed of activities subject to technological precedence constraints (i.e. an activity can start only if all its predecessor activities have been completed) and which cannot be interrupted once they begin (i.e. no preemption is allowed). In this problem, an activity can be performed in one or more combinations of duration and resource requirements. Any activity, once initialized in a specific mode, must be fixed without changing its mode until it is completed. Resources are available in a constant amount per period. The problem described above can be called a **single-mode** or **multi-mode** resource-constrained project scheduling problem depending on an activity performed in exactly one or more ways. Interested readers should examine both single-mode works [2,4,6] and multi-mode works [7,10,12,14].

The following example, similar to Talbot [12], illustrates the abovementioned multi-mode problem. In Fig. 1 and Table 1 , the project is shown to consist of seven activities (including one dummy activity), each of which is successfully accomplished in one of two modes. One renewable resource is required by each activity. The quantity of resource available in each period of the project is 10. With simple computing, we can find that the optimal project makespan is 11, when activity 4 and activity 5 are processed with mode 2, and all of others are scheduled with mode 1. The critical path represented by activity number is $1 \rightarrow 5 \rightarrow 6 \rightarrow 7$. However, if the size of the project becomes larger or the resource types get more, an efficient approach is needed.

The ideas involved in Genetic Algorithms (GAs) were originally developed by Holland [9] and described in greater detail by Goldberg [8]. Genetic al-

[1] Corresponding author.

[2] Present address: Chung Shan Institute of Science and Technology, P.O. Box 90008-6-8, Lungtan, Taoyuan 325, Taiwan, R.O.C.

PII S0377-2217(96)00180-4

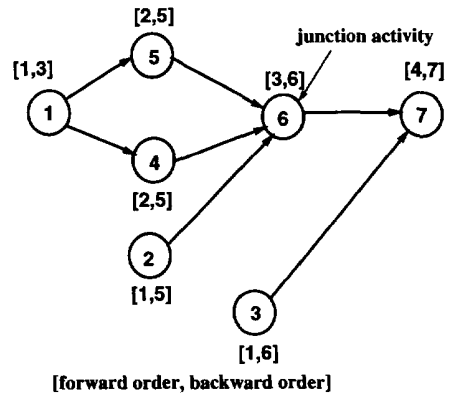

Fig. 1. A 7-activity project.

gorithms are search techniques for global optimization in a complex search space. As the name suggests, they employ the concepts of natural selection and genetics. Using past information they direct the search such that the expected performance will be improved. Although genetic algorithms have already been applied to a wide range of different problem domains, only a few approaches have tried to apply them to scheduling problems until now and, moreover, most of them have been restricted to job shop, flowshop scheduling problems [3,11] or production scheduling problems [1].

As known, a multi-mode resource-constrained project scheduling problem (MRCPSP) is one of the combinatorial problems which can be theoretically found out the optimal solution through finite steps. But for a large and complex project, it becomes computationally-infeasible. As far, a few deterministic and stochastic scheduling rules have been developed to find an approximate solution for the MRCPSP, genetic algorithms are not concerned with. This paper introduces a genetic algorithm for the MRCPSP. The approach is based on the incorporation of problem-specific knowledge of the application domain in the genetic algorithm. In particular, this leads to a new complex non-standard representational scheme for chromosomes that comprises all information relevant to the search task. The introduction of this expanded representation requires the definition of new domain-dependent crossover and mutation operators which take advantage of the additional information represented in the chromosomes.

Table 1
The data of activity

acti. no. i	mode j	dura. d_{ij}	required resource q_{ij}
1	1	3	6
	2	5	4
2	1	2	4
	2	3	3
3	1	5	4
	2	7	3
4	1	4	3
	2	3	4
5	1	5	4
	2	4	5
6	1	4	6
	2	6	4
7	1	0	0
Resource available (Q)			10

The remainder of the paper is organized as follows: Section 2 describes the problem considered. In Section 3 a genetic algorithm for multi-mode resource-constrained project scheduling problems is introduced. Section 4 provides a comparison of the genetic algorithm and the stochastic scheduling method proposed by Drexl and Gruenewald [7].

2. Problem statement and heuristic algorithms

2.1. Problem statement

We consider the problem described as follows:

- A project consists of N activities, activities are labeled form 1 to N, with activity N being the unique terminal job without successors.
- Activity i may be performed in any one of the modes $j = 1, \ldots, M_i$. Each job, once initiated in a specific mode, must be finished without changing mode.
- Scheduling activity i in mode j takes d_{ij} time units (duration).
- Activity i cannot start unless all of its predecessors have been completed.
- Activity preemption is not allowable.
- There are K kinds of the renewable resources, where resource k is available in quantity Q_k per period. Scheduling activity i in mode j uses q_{ijk}

resources units per period for resource k.

- The objective is to **minimize the project duration.**

2.2. Heuristic algorithms for MRCPSP

In accordance with resource restrictions, the necessary sequence should be decided if any resource conflict occurs among the currently schedulable activities. (Activities are said to be conflicting when their resource requirements exceed the currently available resource levels of at least one resource type.) There are a few rules have been developed. Talbot [12] proposed *deterministic scheduling rules*, and compared eight heuristic scheduling rules. The rule **MIN** LF_i was shown to behave best regarding the average quality of solutions.

MIN LF_i: The eligible job i with minimum latest finish time LF_i is scheduled first. The latest finish times LF_i are calculated by taking

$$\overline{T} := \sum_{i=1}^{N} \max\{d_{ij} \mid j = 1, \ldots, M_i\}$$

as an upper bound for project makespan and performing a traditional backward recursion using shortest durations $dmin_i := \min\{d_{ij} \mid j = 1, \ldots, M_i\}$ for all activities i.

Drexl and Gruenewald [7] presented a *stochastic scheduling method* (a weighted random selection technique). The stochastic nature of this method emerges from using some criteria measuring the impacts of job selection and mode assignment in a probabilistic way.

More precisely, they calculate

$$\omega_{ij} := (\max\{d_{hk} \mid h \in EJ,\ k = 1, \ldots, M_h\} - d_{ij} + \epsilon)^{\alpha} (i \in EJ,\ j = 1, \ldots, M_i), \quad (1)$$

where EJ corresponds to the set of the eligible jobs and job modes $j = 1, \ldots, M_i$ taken appropriately. This equation estimates the worst-case consequence of assigning mode j *not* to job i with respect to job durations; $\epsilon > 0$ makes ω_{ij} to be positive; $\alpha \geq 0$ transforms the term (.) in an exponential way, thus diminishing or enforcing the difference between the mode-dependent job durations for $\alpha < 1$ or $\alpha > 1$, respectively. In this sense it suggests itself to use ω_{ij} for stochastic job selection and (or) mode assignment with probabilities proportional to ω_{ij} for all $i \in EJ$ and $j = 1, \ldots, M_i$.

Alternatively, use

$$\gamma_i := (\max\{LF_k \mid k \in EJ\} - LF_i + \epsilon)^{\alpha},\ i \in EJ, \quad (2)$$

for selecting job $i^* \in EJ$ randomly with probabilities proportional to γ_i for all $i \in EJ$ in a first stage and then assign mode j to job i^* at random based on ω_{i^*j} in a second stage.

For convenience they denoted both stochastic job selection and mode assignment procedures with STOCOM (STOchastic COnstruction Method), whereas STOCOM1 as well as STOCOM2 specifically denote, whenever necessary, the one based on (1) as well as (2) and (1), respectively.

Drexl and Gruenewald [7] compared the performance between STOCOM and **MIN** LF_i and concluded that STOCOM is highly superior to other well-known existing deterministic scheduling rules.

3. Genetic algorithm for MRCPSP

Before a GA can be run, a suitable representation for the MRCPSP must be devised. We also require a fitness function, which assigns the project makespan to each representation. During the run, parents must be selected for reproduction, and use genetic operators to generate offspring. Before we describe these aspects, one parameter of the representation which represents activity scheduling order is introduced.

3.1. Scheduling order interval of the activity

A scheduling order of an activity means a priority of the activity in a schedule. Suppose that an activity with smaller order number is scheduled before than anyone with larger order number. Let P_i denote the set of activities immediately preceding activity i, and S_i be the set of activities immediately succeeding activity i. In accordance with precedence constraints, the activity scheduling order interval between forward scheduling order f_i and backward scheduling order b_i of activity i can be determined (ref. Tavares [13]) as follows.

For a N-activity project, the forward scheduling order of activity i ($i = 1, \ldots, N$) is defined by

(a) $f_i = 1$ iff $P_i = \emptyset$,

(b) $f_i = m$ iff $\max_{j \in P_i}\{f_j\} = m - 1$.

Then given any integer I subject to $f_N \leq I \leq N$, the backward scheduling order of activity i is defined by

(a) $b_i = I$ iff $S_i = \emptyset$,

(b) $b_i = n$ iff $\min_{j \in S_i}\{b_j\} = n + 1$.

Based on the previous description, scheduling order intervals $[f_i, b_i]$ of all activities i of the example shown in Fig. 1 can be obtained given I=7.

3.2. Direct representation of MRCPSP

In a direct problem representation, the MRCPSP itself is used as a chromosome. Thus, no decoding procedure is necessary. All information relevant to the MRCPSP at hand is included in the problem representation. The genetic algorithm is the only method that carries out search since the represented information comprises the entire search space. Neither a transformation procedure nor a schedule builder are necessary anymore.

A complete schedule for the MRCPSP comprises all activities with associated mode and scheduling order assignments and time intervals of activities (start and finish times). The mode is chosen at random from its available modes and the scheduling order is assigned randomly within its scheduling order interval for each activity. The scheme of the direct representation is sketched as below. Each cell (i.e., gene) which describes an activity i contains three elements: the upper one, AiMα, means that activity i is scheduled in mode Mα; the middle one, order?, denotes the scheduling order of activity i; the lower one, [start,finish], represents the time interval between start and finish time of activity i. For example, A1Mα/order1/[start,finish] represents that activity 1 is scheduled with mode α and is processed before the activities with scheduling order 2, and that [start,finish] means the start and finish time of activity 1 will be derived later. That is, once modes and scheduling orders of all activities are given, the start and finish time of each activity and project makespan can be obtained if the generated schedule is feasible.

We represent the MRCPSP as a list of activity-mode/order/start-finish-time, as shown in Fig. 2.

A1Mα	A2Mβ		ANMγ
order 1	order 2	.....	order Q
[start, finish]	[start, finish]		[start, finish]

Fig. 2. Direct representation of a schedule.

The quality of a chromosome (i.e. schedule) is measured by means of an fitness (or evaluation) function, which assigns a numerical value to a schedule. Several different evaluation criteria can be relevant for a particular application, e.g. project duration, or maximum lateness. In this paper, we evaluate the duration of the project. The goal of scheduling is to construct a feasible project schedule, which minimizes the chosen evaluation function.

Remark: A project schedule is called *feasible*, if the precedence relations of the activities are maintained and the resource constraints are met.

3.3. Initialization

The initial generation of complete and consistent (i.e., no precedence violation) schedules can be generated as follows.

For each schedule of initial generation, we first randomly decide the backward scheduling order of activity N within the interval between f_N and N. Then the scheduling order intervals of all activities are obtained. For each locus:

- the activity number is placed in ascending order.
- the mode of each activity is assigned randomly from its available modes.
- the scheduling order of each activity is determined randomly from its scheduling order interval.

After the mode and scheduling order of each activity is given, we can calculate the start and finish time of each activity and project makespan if the generated schedule is feasible. Generate a new feasible schedule successively until the first generation is produced. In this way a random and diverse generation of schedules can be produced, since even slightly different mode or scheduling order usually cause the generation of radically different schedules.

Parent Schedule 1

A 1 M 2	A 2 M 1	A 3 M 2	A 4 M 1	A 5 M 2	A 6 M 1	A 7 M 1
order 1	order 1	order 2	order 3	order 2	order 4	order 5
[0, 5]	[0, 2]	[2, 9]	[9, 12]	[5, 9]	[12, 19]	[19, 19]

Parent Schedule 2

A 1 M 1	A 2 M 2	A 3 M 2	A 4 M 2	A 5 M 1	A 6 M 2	A 7 M 1
order 1	order 2	order 1	order 2	order 3	order 4	order 5
[0, 3]	[3, 6]	[0, 7]	[3, 6]	[6, 10]	[10, 16]	[16, 16]

↓ Crossover

Offspring Schedule

A 1 M 1	A 2 M 2	A 3 M 2	A 4 M 2	A 5 M 1	A 6 M 2	A 7 M 1
order 1	order 2	order 2	order 2	order 3	order 4	order 5
[-, -]	[-, -]	[-, -]	[-, -]	[-, -]	[-, -]	[-, -]

Fig. 3. An example of crossover.

3.4. Sorting

Once one generation which consists of a certain number of feasible schedules is constructed, we sort the chromosomes in ascending order of project duration.

3.5. Genetic operators

The introduction of a non-standard chromosome representation necessitates the definition of new crossover and mutation operators which are usually more complicated than traditional ones. The selection operator need not to be changed since it works only on fitness values and is therefore independent of any specific chromosome representation.

Recombination operators have been developed to operate on the direct representation of the MRCPSP. They have been designed to take advantage of all information represented in the chromosome, i.e. to address the entire search space. To overcome the problem of illegal solutions, each operator creates offspring schedules in a manner that guarantees that precedence constraint specified in the scheduling problem remains satisfied.

(1) **Crossover**

The crossover operator should generate an offspring schedule by combining features of two selected parent schedules. The basic features of the MRCPSP are temporal mode and scheduling order assignments of each activity.

The goal is to minimize the project duration. Thus the junction activity with the smaller start (or finish) time is a very important criterion. Therefore, in order to support the inheritance of the good features of a schedule, the crossover operator has been designed to maintain the modes and scheduling orders of the predecessors of the junction activity. The scheme of the crossover operator is depicted as follows:

Select two parent schedules: one is the first schedule with minimum project duration; the other is chosen randomly from those whose project durations are not minimum.

(a) Choose one junction activity with lower start time at random from one parent. The offspring schedule inherits the corresponding predecessor activities of the junction activity chosen.

(b) Take missing activities from the other parent; and insert into the offspring schedule

The part taken from the chosen parent schedule in step (a), i.e. the chosen junction activity and the predecessor activities of it with their modes and scheduling orders, builds a consistent partial schedule for the offspring. Step (b) enables the missing activities to be inherited from the second parent schedule. If this offspring schedule is inconsistent, we adjust the scheduling orders of the activities inherited from the second parent schedule to satisfy precedence constraints. For example, suppose we choose two parent schedules from a 7-activity project example described in Section 1 (Fig. 3). Activity 6 is a junction. The start time of the activity 6 of parent schedule 2 is smaller than parent schedule 1. According to the crossover devised, activities 1, 2, 4, 5 and 6 of offspring schedule will inherit from parent schedule 2. Activities 3 and 7 will inherit from parent schedule 1.

This approach ensures that the application of the crossover operator always yields consistent offspring schedules.

(2) **Mutation**

The mutation operator must be able to alter all information represented in the chromosome. It must also provide the possibility of reintroducing lost genetic material. Thus we devise two types, one is to change modes for the selected activities, the other is to generate a new schedule, as follows:

(a) Select one schedule randomly;

Parent Schedule

A 1 M 2	A 2 M 1	A 3 M 2	A 4 M 1	A 5 M 2	A 6 M 1	A 7 M 1
order 1	order 1	order 2	order 3	order 2	order 4	order 5
[0, 5]	[0, 2]	[2, 9]	[9, 12]	[5, 9]	[12, 19]	[19, 19]

↓ Mutation

Offspring Schedule

A 1 M 2	A 2 M 2	A 3 M 2	A 4 M 2	A 5 M 1	A 6 M 1	A 7 M 1
order 1	order 1	order 2	order 3	order 2	order 4	order 5
[-, -]	[-, -]	[-, -]	[-, -]	[-, -]	[-, -]	[-, -]

Fig. 4. An example of mutation.

(i) select n (less than N) activities at random, and

(ii) change mode for the selected activities.

For instance, suppose we select a parent schedule from a 7-activity project presented in Fig. 4 and choose 3 activities such as activities 2, 4, and 5 randomly. Then their modes are changed and an offspring schedule is produced.

The mutation operator changes the mode assignments of the activities selected. The result of mutation is always a consistent schedule.

(b) To avoid falling into a local optimum and accelerate finding a better solution, a new schedule is produced randomly by the same method used in initialization.

3.6. New generation

A new generation is produced by the operators described as follows:

(1) Duplicate P_1 offspring schedules from the parent schedule with minimum makespan (i.e. elitest preserving strategy).

(2) Produce P_2 offspring schedules with the **crossover** operator.

(3) Produce P_3 offspring schedules with the **mutation** operator.

(4) Generate P_4 offspring schedules at **random**.

$P_1 + P_2 + P_3 + P_4 = P$, where P indicates the number of schedules in the new generation.

4. Computational results

The GA as well as the stochastic scheduling method have been coded in C and implemented on a SunOS (UNIX) Release 4.1.3. Similar to [7], the test data generated for comparative purposes may be characterized as follows (project summary measures):

- The problem size, in terms of the number of jobs N, is the first project summary measure.
- The network complexity C= number of arcs (precedence relations)/number of nodes (jobs) affects the performance of scheduling procedures.
- An activity network is generated using *A Random Activity Network Generator* proposed by Demeulemeester et al. [5] when N and C are given.
- The number of modes of job i is generated randomly such that $2 \leq M_i \leq 4$.
- The total number of renewable resource types $K = 4$ is fixed.
- The (integer) duration d_{ij} of job i being scheduled in mode j is generated at random such that $5 \leq d_{ij} \leq 10$.
- Job-mode-dependent (integer) resource demands (requirements) are randomly generated such that $0 \leq q_{ijk} \leq 5$ for all resources.
- The availability Q_k of renewable resource type k is determined by multiplying the peak resource requirement

$$Q_k' := \max\{q_{ijk} \mid i = 1, \ldots, N,\ j = 1, \ldots, M_i\},$$

with R1, $1.5 \leq R1 \leq 3.3$, thus getting

$$Q_k := Q_k' \cdot R1, \quad k = 1, \ldots, K.$$

Drexl and Gruenewald generated and solved to optimality with different project summary measures. They found the smallest average percentage deviation of objective function values with $\epsilon = 1$ and $\alpha = 2.0$ among $\alpha = 0.0, 0.5, 1.0, 1.5, 2.0$.

Once test data is generated, we first obtain the minimal solution from 100 successful[3] iterations in STOCOM stage with $\epsilon = 1$ and $\alpha = 2$. In the GA stage, we initialize 40 schedules as the first generation. To compare the minimal solution of each generation in GA, we stop searching if the solution is smaller than that produced by STOCOM, else we continue to do at most 50 generations.

[3] By successful, we mean finding a feasible solution.

Table 2
Comparison of STOCOM and GA

Complexity (C)	N=20 3.5	N=30 4.0	N=40 4.0	N=50 4.0	N=60 4.0	N=70 4.0
A	10	63	57	50	78	77
B	6	18	19	21	20	12
C	30	8	2	4	0	0
D	17	4	8	7	0	2
E	37	7	14	18	2	9
Total	100	100	100	100	100	100

A : Number of times in which GA found smaller optimum than STOCOM and GA took equal or less CPU time than STOCOM.
B : Number of times in which GA found smaller optimum than STOCOM and GA took more CPU time than STOCOM.
C : Number of times in which GA found same optimum with STOCOM and GA took equal or less CPU time than STOCOM.
D : Number of times in which GA found same optimum with STOCOM and GA took more CPU time than STOCOM.
E : Number of times in which GA optimum found is greater than STOCOM.

Table 3
CPU-times of STOCOM and GA in seconds

Number of activities	STOCOM			GA		
	MIN	MEAN	MAX	MIN	MEAN	MAX
N=20	0.52	1.03	4.45	0.61	1.40	6.23
N=30	1.02	1.19	2.39	0.68	0.99	5.95
N=40	2.13	2.65	3.85	0.77	2.91	11.35
N=50	4.11	4.51	5.92	0.98	3.82	15.26
N=60	7.06	7.34	14.49	1.21	3.63	15.74
N=70	11.54	12.27	15.72	1.97	5.90	32.17

The results of Table 2 are obtained from 100 different project measures (generated randomly) under one generated activity network (given N and C) by running GA with $P_1 = 20, P_2 = 10, P_3 = 7, P_4 = 3$.

The CPU-times required by the rule of STOCOM and by GA are given in Table 3. Table 3 presents minimum, mean, maximum CPU-times (test data generating time excluded) in seconds (MIN, MEAN, MAX) to obtain minimal solution for STOCOM and GA using the abovementioned procedure. Table 3 shows that the mean times of GA are faster than those of STOCOM for large number of activities (such as N equals to 50, 60 or 70).

Acknowledgements

The authors would like to thank two anonymous referees for their valuable comments and suggestions which greatly enhance the clarity of the paper.

References

[1] Bruns, R. (1993), "Direct chromosome representation and advanced genetic operators for production scheduling", in: Forrest, S., ed., *Proceedings of the Fourth International Conference of Genetic Algorithms*, Morgan Kaufmann Publishers, CA, 352–359.

[2] Christofides, N., Alvarez-Valdes, R., and Tamarit, J.M. (1987), "Project scheduling with resource constraints: a branch and bound approach", *European J. of Operational Research* 29, 262–273.

[3] Croce, F.D., Tadei, R., and Volta, G. (1995), "A genetic algorithm for the job shop problem", *Computers & Operations Research* 22/1, 15–24.

[4] Davis, E.W., and Heidorn, G.E. (1971), "Optimal project scheduling under multiple resource constraints", *Management Science* 17, B803–B816.

[5] Demeulemeester, E., Dodin, B., and Herroelen, W. (1993), "A random activity network generator", *Operations Research* 41/5, 972–980.

[6] Demeulemeester, E., and Herroelen, W. (1992), "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science* 38, 1803–1818.

[7] Drexl, A., and Gruenewald, J. (1993), "Nonpreemptive multi-mode resource-constrained project scheduling", *IIE Transactions* 25, 74–81.

[8] Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

[9] Holland, J.H. (1976), *Adaptation in Nature and Artificial Systems*, University of Michigan Press, Ann Arbor.

[10] Patterson, J.H., Talbot, F.B., Slowinski, R., and Weglarz, J. (1990), "Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems", *European J. of Operational Research* 49, 68–79.

[11] Reeves, C.R. (1995), "A genetic algorithm for flowshop sequencing", *Computers & Operations Research* 22/1, 5–13.

[12] Talbot, F.B. (1982), "Resource-constrained project scheduling with time-resource tradeoffs: the nonpreemptive case", *Management Science* 28, 1197–1210.

[13] Tavares, L.V. (1990), "A multi-stage non-deterministic model for project scheduling under resources constraints", *European J. of Operational Research* 49, 92–101.

[14] Tseng, C.C., Mori, M., and Yajima, Y. (1995), "A project scheduling model considering the success probability", in: Fushimi, M., and Tone, K., eds., *Proceedings of APORS'94*, World Scientific Publishing Company, Singapore, 399–406.