Scientia Iranica, Vol. 12, No. 1, pp 90–98

SCIENTIA
IRANICA

Research Note

A Genetic Algorithm for Resource Investment Problems, Enhanced by the Revised Akpan Method

S. Shadrokh* **and F. Kianfar**[1]

In this paper, a genetic algorithm for solving a class of project scheduling problems, called Resource Investment Problems, is presented. Tardiness of the project is permitted with a defined penalty. The decision variables are the level of resources and the start times of the activities. The objective is to minimize the sum of resources and delay penalty costs, subject to the activities' precedence relations and some other constraints. A revised form of the Akpan heuristic method for this problem is used to find better chromosomes. Elements of the algorithm, such as chromosome structure, unfitness function, crossover, mutation, immigration and local search operations, are explained. The performance of this genetic algorithm is compared with that of other published algorithms for Resource Investment Problems. Also, more than 700 problems are solved using an enumerating algorithm and their optimal solutions are used for the performance tests of the genetic algorithm. The tests results are quite satisfactory.

INTRODUCTION

The Resource Constrained Project Scheduling Problem (RCPSP) is one of the main branches of project scheduling. Some other types of scheduling problem, such as Flow-Shop and Job-Shop, can be modeled as RCPSP (see, for example, [1]). A Resource Investment Problem (RIP) is a RCPSP in which the cost of total resources used for the project is part of the objective function, which should be minimized. Mohring [2] has discussed this problem and shows that RIP is NP-hard. He considers a tight deadline for the project completion time. It is more realistic to permit project tardiness and assign a positive penalty to each unit of time delay. In this paper, RIP is considered under the condition that delay is permissible. Since scheduling problems are usually Np-hard, heuristic algorithms, such as Genetic Algorithms (GA), have gained considerable attention in recent years. For example, Hartmann [3] uses GA for Multiple Mode Resource Constrained Project Scheduling (MMRCPSP).

In this paper, a genetic algorithm for a Resource Investment Problem is presented, which is an enhancement of the authors' previous work on this problem. Tardiness in the project finishing time is permitted with a defined penalty. Whereas in the previous genetic algorithm each chromosome consisted of two parts, one for determination of start times and the other for the level of resources, in the new algorithm, chromosomes have only one part, which gives the sequence of the activities to be scheduled. Using a revised version of the Akpan method, the schedule and resource capacities for the chromosomes are, subsequently, determined. Elements of the algorithm, such as chromosome structure, unfitness function, crossover, mutation and immigration operations, are explained.

To adjust the parameters of the genetic algorithm and also to test the performance, 720 problems were solved to optimality, using an enumeration process. Also, the published results of the other algorithms are compared against the performance of the genetic algorithm.

PROBLEM DESCRIPTION

There is a set of $V = \{0, 1, \ldots, n, n+1\}$ activities in a project that should be completed before a predetermined deadline, T. Otherwise, with a constant cost, C_d, for each unit of time delay, some amount

*. *Corresponding Author, Faculty of the Ghazvin Azad University, Ghazvin, I.R. Iran.*

1. *Department of Industrial Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

of penalty must be paid. The precedence relations of activities are shown by activity on the node network with no loops that has $n+2$ nodes. Nodes 0 and $n+1$, the initial and terminal nodes, respectively, are dummies. A set K of ρ renewable resources, each at a constant level, is allocated to this project. It means that, after determination of the resource levels, they remain constant during project execution. Activities are not preemptive and have one mode of execution with a constant rate of consumption of each resource during execution time. All parameters are deterministic. The problem is to find the start time of activity $i, S_i, i = 0, \ldots, n+1$, given the resource level $R_k, k = 1, \ldots, \rho$, such that the precedence relations of activities are satisfied and the total cost of all resources, plus delay penalties are minimized. Let:

C_k = cost of each unit of available capacity for resource $k, (k \in K)$,
P_i = set of predecessors of activity i,
r_{ik} = rate of the kth resource usage by activity i,
D_i = ith activity duration time,
x_{it} = 1, if activity i starts at time t and 0 otherwise.

Then $S_i = \sum_{t=0}^{T} t \times x_{it}$ and the model is as follows [4]:

$$\min \left\{ \sum_{k=1}^{\rho} C_k R_k + C_d \times \max\{0, S_{n+1} - T\} \right\}, \qquad (1)$$

$$\sum_{t=0}^{T} t \times x_{it} \geq D_j + \sum_{t=0}^{T} t \times x_{jt},$$

$$j \in P_i, \quad i = 1, \cdots, n+1, \qquad (2)$$

$$\sum_{i=1}^{n} \sum_{u=t-D_i+1}^{t} r_{ik} \times x_{iu} \leq R_k,$$

$$t = 0, \cdots, T, \quad k \in K = \{1, \cdots, p\}, \qquad (3)$$

$$S_i = \sum_{t=0}^{T} t \times x_{it}, \quad i = 1, \cdots, n+1, \qquad (4)$$

$$\sum_{t=0}^{T} x_{it} = 1, \quad i = 1, \cdots, n+1, \qquad (5)$$

$$x_{01} = 1, \qquad (6)$$

$$x_{it} \in \{0, 1\}, \quad i = 1, \cdots, n+1, \quad t = 0, \cdots, T, \qquad (7)$$

$$R_k \geq 0, \quad k \in K = \{1, \cdots, p\}, \qquad (8)$$

The objective (Formula 1) is to minimize the total cost of all resources and the probable penalties. If project finish time is within the deadline, the objective function is equal to $\sum_{k=1}^{\rho} C_k R_k$, otherwise it is $\sum_{k=1}^{\rho} C_k R_k + C_d(S_{n+1} - T)$. Constraint in Formula 2 takes into consideration the precedence relation between each pair of activities (i, j), where j immediately precedes i. The constraint set in Statement 3 limits the total resource usage within each period to the available amount. Constraint in Equation 4 calculates the start times. Constraint in Formula 5 guarantees that each activity, i, can only have one start time. Sets of constraints in Formulae 7 and 8 denote the domain of variables. Shadrokh and Kianfar [5] called the above problem a resource investment problem, while tardiness is permitted (RIPT).

GENETIC ALGORITHM

Basic Scheme

The New Genetic Algorithm (NGA) uses a combination of the Shadrokh and Kianfar Genetic Algorithm (SKGA) [5] and the Akpan Algorithm (AA) [6]. The basic scheme of NGA is similar to that of SKGA. Each chromosome in SKGA has two sections, an activity list section and a resource capacity list section. In NGA, each chromosome has one section, i.e., activity list, and the resource capacities are computed using a modified version of AA that was derived from AA, so that it could be applicable to RIPT. Each chromosome gives a unique value to the decision variables, S_is and R_ks of RIPT and, hence, specifies a unique schedule. The basic scheme of the algorithm is as follows.

Each individual within the first generation is created randomly and its unfitness is calculated. Size of population, POP, is a parameter of the algorithm and remains constant for all generations. Each new generation is made from existing generations by using three operations: Crossover, mutation and immigration. In the crossover operation, the existing generation is randomly partitioned into POP/2 pairs of parents and, on each pair, the crossover operation is performed with probability P_{cr}. If a pair is not selected for crossover, each individual in the pair is considered for the mutation operation with probability P_{mu}. The crossover operation on a pair of parents, Pa_1 and Pa_2, gives two children, CH_1 and CH_2. Let $f(I)$ be the unfitness value of individual I. If $(f(Pa_i)+f(\mathrm{CH}_i)) \times r_i \leq f(Pa_i)$, then, CH_i will go to the new generation and Pa_i dies out $(i = 1, 2)$, where r_i is a random number generated from the interval [0,1] for each i, otherwise CH_i dies out and Pa_i is considered for mutation with probability P_{mu}. After constructing each generation by these operations, the immigration operation is also performed before the cycle of producing the new

generation is finalized. In the immigration operation, an immigrating chromosome is generated randomly, which is called a new chromosome. An individual I, is randomly selected from the current population. Let $P_{\text{leave}}(I, \text{new}) = f(I)/(f(I) + f(\text{new}))$. A random number, r, is generated in the interval [0,1]. If $r < P_{\text{leave}}(I, \text{new})$, immigrant new replaces I, otherwise, new is discarded. The immigration operation gives a chance for the non-existing desirable characteristics in a population to come to it. The number of generations created for solving a problem is a parameter of the algorithm and is adjustable. The chromosome with the least unfitness value in the final generation is the solution given by the algorithm. Note that POP, P_{mu} and P_{cr} are also adjustable parameters of the algorithm.

Chromosome Structure and Unfitness Function

In each generation, each individual, I, is a unique solution to RIPT and is represented by its chromosome as: $I = (j_0^I, \cdots, j_{n+1}^I)$. Here, $j_i^I (i = 0, \ldots, n+1)$ is the number of the activity, which is the ith activity in the scheduling sequence. As noted in the section of Basic Scheme, this chromosome doesn't contain a resource capacity section, like SKGA. The order of activities from left to right in the chromosome string is such that all of the predecessors of activity j_i^I are located at the left side of it. For determining the resource capacity levels in NGA, AA were modified and employed for NGA chromosomes. This modified version of AA is denoted as MAA and is discussed in the section of Setting Resource Levels. When genotype is transformed to phenotype, the Serial Schedule generation Scheme (SSS) [7-9] method, along with the MAA, is used to determine the start times, $S_1^I, \cdots, S_n^I$. Kolisch [8] shows that finding a schedule from the activity list SSS is more effective than a Parallel Schedule generation Scheme (PSS). This matter is confirmed by Shadrokh and Kianfar [5]. They show that SSS is much better than PSS in their genetic algorithm. Let R_k^I be the capacity of resource k for chromosome I, which is determined by MAA and which will remain constant throughout the project execution. For each chromosome, I, the value of unfitness function $f(I)$, is:

$$f(I) = \sum_{k=1}^{\rho} C_k R_k \quad \text{if } T \geq S_{n+1}^I,$$

otherwise:

$$f(I) = \sum_{k=1}^{\rho} C_k R_k + C_d \times (S_{n+1} - T).$$

If tardiness is prohibited, C_d can be selected large enough in relation to C_ks. But, if C_d is selected large, right from the beginning, the infeasible Is, with probable good characteristics, are discarded very early. To avoid this, an annealing process is used in the algorithm. In this process, C_d is not very large for the first generation. At each iteration of the algorithm for finding the new generation, C_d increases by Δ_{inc} and, then, in the final generation all infeasible I's are discarded.

More than one I may give the same schedule, i.e., the relation is one to many. For example, consider a single resource project with 4 real activities, whose activity on the node network is as shown in Figure 1.

Now, consider two chromosomes $(0, 1, 2, 3, 4, 5)$ and $(0, 1, 3, 2, 4, 5)$. The schedules (phenotypes) of these two genotypes are the same, as shown in Figure 2. Nodes 0 and 5 are dummies and not shown.

Generating Chromosomes and Setting Their Resource Levels

Generating Chromosomes

For generating a chromosome $I = (j_0^I, \cdots, j_{n+1}^I)$, at the stage of determination of j_a^I, when $j_0^I, \cdots, j_{a-1}^I$ are known, let Eligible Activity Set (EAS) be the set EAS=$\{u | P_u \subset \{j_0^I, \cdots, j_{a-1}^I\}\}$ and N(EAS) be the number of activities in EAS. Then, the following two steps [3] are used:

1. Initial step: $j_0^I = 0$, EAS= $\{u, | P_u = \{0\}\}, N = N(\text{EAS}), a = 1$,
2. Main step: If $N = 0$, stop. Otherwise, select one

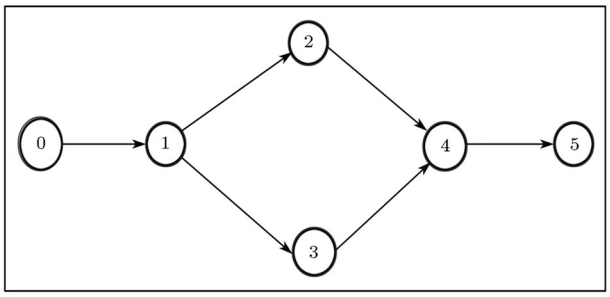

Figure 1. Activity on the nodes network.

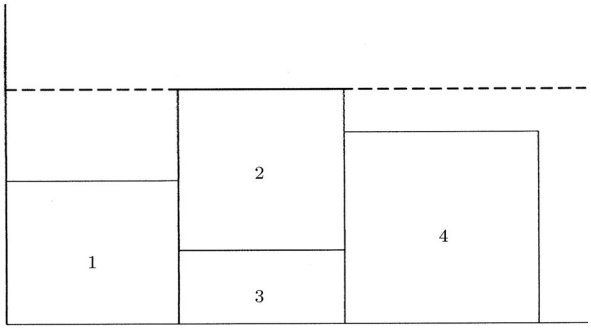

Figure 2. Resulted schedule from two different chromosomes.

of the activities from EAS and let it be activity i. Set $j_a^I = i$ and EAS= $\{u|u \notin \{j_0^I, \cdots, j_a^I\}\}, P_u \subset \{j_0^I, \cdots, j_a^I\}\}, N = N(\text{EAS})$, $a = a + 1$ and then repeat the main step.

For selecting an activity from EAS, two methods are used, whose performances are compared later on. The first method, which is suggested in [3], uses the efficient priority rule of minimum Latest Finish Time (LFT) to derive the probabilities of selecting activities from EAS. This method is referred to as RLFT. In RLFT, a probability for selecting each activity from EAS is assigned, proportional to the inverse value of the Latest Finish Time of that activity. The second method is suggested by Shadrokh and Kianfar [5], which is called WS. This method assigns the probabilities for selection of activities from EAS, such that chromosomes have more tendencies to appear with equal chance. It is shown that random selection of activities from EAS does not give an equal selection chance to all activity lists, i.e. chromosomes. Having the same chance for each chromosome to appear in the chromosome generating process gives an opportunity for the genetic algorithm to guide its population to the promising areas of later generations.

To define WS, the following definitions are used: On an activity on the node network of a project, node j is reachable by node i [10], if there is at least one directed path with origin i and terminus j. Let the set of reachable nodes by node i be RN_i and the number of elements of this set be $N(RN_i)$. The rightmost place on the activity list, where an activity, i, could be placed, is $n + 2 - N(RN_i)$. Then, for an activity, i, the bigger the value of $N(RN_i)$, the closer it would be to the left of the activity list and, hence, the number of location choices for this activity would be smaller. Noting this property, WS employs a simple heuristic for selecting activities from EAS, which gives each activity of EAS the chance of being selected proportional to its number of elements in the set of reachable nodes.

Setting Resource Levels

Each resource capacity should be determined, for each chromosome I. For this purpose, the Akpan algorithm [6] was modified such that it would be applicable for RIPT.

Let $TU_k, k \in K = \{1, \cdots, \rho\}$ be the total amount of usage of resource k, which is equal for all chromosomes and can be calculated as $TU_k = \sum_{i=1,\cdots,n} r_{ik} \times D_i$. Also, let RES be a set containing all resources. At this stage, RES $= K$. The resource capacities for chromosome I are determined using the following steps:

Step 1 For chromosome I, schedule activities at the earliest possible time, regardless of any resource restriction;

Step 2 Let all resource capacities be at their minimum possible levels. The minimum possible level for resource k is the maximum level of resource k used in the earliest start time schedule along the time axis. Calculate the current unfitness value, UF_{cu}, of this schedule, according to the previous section;

Step 3 Reduce one unit from the capacity of the resource that has a minimum value of $TU_k/(S_{n+1} \times R_k)$ [6], $k \in$ RES. In other words, if $\min_{k \in \text{RES}}(TU_k/(S_{n+1} \times R_k)) = TU_v/(S_{n+1} \times R_v)$, reduce one unit from resource capacity v;

Step 4 Consider the above values of the resource capacities as fixed. Using SSS on chromosome I, find the start time of the activities. Let $\overline{S}_1, \cdots, \overline{S}_{n+1}$ be the calculated start times. Calculate the unfitness value for this new schedule, UF_{new};

Step 5 If $\text{UF}_{\text{new}} > \text{UF}_{cu}$, delete resource v from set RES and add one unit to its capacity. If set RES is empty, stop, the current resource capacities and the current unfitness value, UF_{cu}, belong to chromosome I and one also has its phenotype; otherwise, go to Step 3. If $\text{UF}_{\text{new}} \leq \text{UF}_{cu}$, let $\text{UF}_{cu} = \text{UF}_{\text{new}}$ and go to Step 6;

Step 6 If $\overline{S}_{n+1} > S_{n+1}$, set RES $= K$ and go to Step 3.

Crossover

Here, two crossovers, i.e. one-point crossover [3] and two-point crossover [5], are considered and discussed below.

One-Point Crossover

Let $Pa_1 = (j_0^1, \cdots, j_{n+1}^1)$ and $Pa_2 = (j_0^2, \cdots, j_{n+1}^2)$ be a pair of parents selected for crossover. Select integer number r randomly from the interval [1,n]. Two children, CH_1 and CH_2, are defined from this crossover, whose activity list is $(\overline{j_0^1}, \cdots, \overline{j_{r-1}^1}, j_r^1, \cdots, j_{n+1}^1)$ for CH_1 and $(\overline{j_0^2}, \cdots, \overline{j_{r-1}^2}, j_r^2, \cdots, j_{n+1}^2)$ for CH_2, where $\overline{j_0^1}, \cdots, \overline{j_{r-1}^1}$ are $j_0^1, \cdots, j_{r-1}^1$, arranged in the order they are in Pa_2 and $\overline{j_0^2}, \cdots, \overline{j_{r-1}^2}$ are $j_0^2, \cdots, j_{r-1}^2$, arranged in the order they are in Pa_1. For example, if $n = 5, r = 4, (j_0^1, \cdots, j_6^1) = (0, 1, 5, 3, 4, 2, 6)$ and $(j_0^2, \cdots, j_6^2) = (0, 5, 2, 1, 4, 3, 6)$, then $(\overline{j_0^1}, \overline{j_1^1}, \overline{j_2^1}, \overline{j_3^1}, j_4^1, j_5^1, j_6^1) = (0, 5, 1, 3, 4, 2, 6)$ and $(\overline{j_0^2}, \overline{j_1^2}, \overline{j_2^2}, \overline{j_3^2}, j_4^2, j_5^2, j_6^2) = (0, 1, 5, 2, 4, 3, 6)$. Resource capacities and unfitness values for children are determined as explained previously.

Two-Point Crossover

Again, let $Pa_1 = (j_0^1, \cdots, j_{n+1}^1)$ and $Pa_2 = (j_0^2, \cdots, j_{n+1}^2)$ be a pair of parents. Two integer random numbers, r_1 and r_2, $r_1 < r_2$, are generated from the interval [1,n]. Two children, CH_1 and CH_2, which result from this crossover, are $(\overline{j_0^1}, \cdots, \overline{j_{r_1-1}^1}, \overline{j_{r_1}^1}, \cdots, \overline{j_{r_2}^1}, \overline{j_{r_2+1}^1}, \cdots, \overline{j_{n+1}^1})$ for CH_1 and $(\overline{j_0^2}, \cdots, \overline{j_{r_1-1}^2}, \overline{j_{r_1}^2}, \cdots, \overline{j_{r_2}^2}, \overline{j_{r_2+1}^2}, \cdots, \overline{j_{n+1}^2})$ for CH_2, where $\overline{j_0^1}, \cdots, \overline{j_{r_1-1}^1}$ are $j_0^1, \cdots, j_{r_1-1}^1$, arranged according to the order they appear in Pa_2 and $\overline{j_{r_2+1}^1}, \cdots, \overline{j_{n+1}^1}$ are $j_{r_2+1}^1, \cdots, j_{n+1}^1$, again, arranged according to the order they are shown in Pa_2. The definitions of $\overline{j_i^2}$'s are similar.

For instance, consider two parents $(j_0^1, \cdots, j_{10}^1) = (0,2,1,5,7,3,6,8,4,9,10)$ and $(j_0^2, \cdots, j_{10}^2) = (0,4,6,8,2,5,7,1,3,9,10)$ and two integers $r_1 = 4$ and $r_2 = 6$, then, $(\overline{j_0^1}, \overline{j_1^1}, \overline{j_2^1}, \overline{j_3^1}, j_4^1, j_5^1, \overline{j_6^1}, \overline{j_7^1}, \overline{j_8^1}, \overline{j_9^1}, \overline{j_{10}^1}) = (0,2,5,1,7,3,6,4,8,9,10)$ and $(\overline{j_0^2}, \overline{j_1^2}, \overline{j_2^2}, \overline{j_3^2}, j_4^2, j_5^2, j_6^2, \overline{j_7^2}, \overline{j_8^2}, \overline{j_9^2}, \overline{j_{10}^2}) = (0,6,8,4,2,5,7,1,3,9,10)$.

Because of the existence of precedence relations and the fact that two different chromosomes could have the same phenotypes, when the number of activities is not big enough (i.e., less than 30 activities), the two-point crossover usually will not give a new schedule. In such cases, even if a new child is produced, it is usually very similar to one of the parents and, therefore, it would be better to use a one-point crossover. On the other hand, if the number of activities are not small (more than 100), the one point method leads to children that may be very different from their parents and are similar to cases when a chromosome is produced randomly. Here, for saving the characteristics of the parents and for avoiding randomness behavior, it is better to use the two-point crossover.

Mutation

Let $I = (j_0^I, \cdots, j_{n+1}^I)$ be the chromosome that is selected for mutation. An integer random number, a, is generated from the interval [1,n]. Let activity j_b^I be the last predecessor of activity j_a^I and activity j_c^I the first successor of j_a^I in chromosome I. Then, another integer random number, d, is generated from the interval $[b+1, c-1]$. If $d < a$, then chromosome I is replaced by $(j_0^I, \cdots, j_{d-1}^I, j_a^I, j_d^I, \cdots, j_{a-1}^I, j_{a+1}^I, \cdots, j_{n+1}^I)$, but, if $d > a$, it would be replaced by $(j_0^I, \cdots, j_{a-1}^I, j_{a+1}^I, \cdots, j_{d-1}^I, j_a^I, j_{d+1}^I, \cdots, j_{n+1}^I)$. The mutated chromosome replaces I with a probability proportional to its unfitness. The process is the same as that of immigration, explained previously.

For example, consider chromosome $(0,1,5,3,7,2,4,6,8,11,9,10,12)$. If $a = 6$ and activities 1 and 3 are the predecessors of 4; and 10 and 11 are its successors, a random number from 4 to 8 is produced. If $d = 4$ the mutated chromosome will be $(0,1,5,3,4,7,2,6,8,11,9,10,12)$.

TESTING THE ALGORITHM

There is a dearth of published research on the Resource Investment Problem, RIP. This is especially true about RIP benchmark problems with a known optimal solution. In particular, no algorithm has been presented to find the optimal solution of RIPT. For adjusting the parameters of NGA, one needs to solve RIPT problems. Except for the 90 problems on RIP solved by Möhring [2], no other published set could be found. Shadrokh and Kianfar [5], show that this set of problems is simple and is not suitable for this purpose. The enumeration procedure was used on the resource capacities, together with the branch-and-bound procedure of Demeulemeester and Herroelen [11] and more than 700 problems were solved. ProGen software [8] was used to generate these problems. The enumeration procedure is as follows.

In solving each problem, the decision variables are the resource capacity and activity start times. Given a set of resource capacities, the value of the RIPT objective function is minimized by minimizing S_{n+1}. Therefore, the Single Mode Resource Constrained Project Scheduling Problem (SMRCPSP) algorithms [11] can be used to find the minimum objective function value of the RIPT, when resource capacities are set. Using this fact, RIPT can be solved by enumerating all possible combinations of resource capacities. It would be enough to consider all combinations of resource capacities between a lower bound, $\underline{R_k}$, and an upper bound, $\overline{R_k}$. For RIPT, R_ks $(k = 1, \cdots, \rho)$ cannot be less than $\max_{i=1,\cdots,n}\{r_{ik}\}$ and, hence, $\underline{R_k} = \max_{i=1,\cdots,n}\{r_{ik}\}$. For RIP, R_ks $(k = 1, \cdots, \rho)$ cannot be less than $\max_{i=1,\cdots,n}\{r_{ik}\}$ or less than $\sum_{i=1}^{n}(r_{ik} \times D_i)/T$, hence:

$$\underline{R_k} = \max\{\sum_{i=1}^{n}(r_{ik} \times D_i)/T, \max_{i=1,\cdots,n}\{r_{ik}\}\}.$$

The available capacity of resource k in the earliest start time schedule is considered for $\overline{R_k}(k = 1, \cdots, \rho)$. Problems with 10, 14 and 20 activities were generated, solved and used for this analysis.

The 480 instances presented in [12] were also used, which can be found in the project scheduling problem library PSBLIB [13], as will be explained in the next sections. Also, various sensitivity analyses of computational performance on some problem parameters have been presented.

All the computations were performed on an IBM-compatible PC with a Pentium III, 1 GHz CPU speed and 256 MB RAM memory, under Windows 2000 as the operating system. Also, all procedures were coded in

ANSI C and compiled with the Microsoft Visual C++ 6.0 compiler.

ProGen Instances with 20 Non-Dummy Activities

20 instances, with 20 non-dummy activities and 4 resources, were generated with ProGen software [8]. The user can set a number of parameters in this program. Three important parameters that can be set are as follows:

1. NC (Network Complexity): The average number of non-redundant arcs per node, including the dummy activities;
2. RF (Resource Factor): The average percentage of different resource types for which each non-dummy activity has a non-zero resource demand;
3. RS (Resource Strength): It shows the availability amount of resources. An RS value of zero defines the capacity of the resource to be no more than the maximum demand over the set of all activities, while an RS value of 1 defines the capacity of each resource to be equal to the demand imposed by the earliest start time schedule.

For these problems, activity duration and resource requirements for each of the 4 resource types are integer values within the range [1,10], according to a discrete uniform distribution. The project networks have three (non-dummy) start activities and three (non-dummy) finish activities. The Network Complexity, NC, is set at 1.5 and the Resource Factor, RF and Resource Strength, RS, are set to 1 and 0.2, respectively (for more details see [8]). The maximum number of predecessors/successors is three. The original resource availabilities of SMRCPSPs, which are generated by ProGen, are used as the unit costs of the corresponding resource types. Also, the tardiness cost, C_d, is considered as 1/8 of the sum of the unit cost of the resources. If the selected parameter is too high, the solution of the RIPT comes close to that of RIP and, if it is too low, the resource availabilities tend toward their lower bounds and tardiness becomes too long.

To get a RIPT instance, one additionally needs the due date, T. In this study, $T = \theta^*$EFT was set where EFT is the earliest finish time of the project having infinite resource capacities and $\theta \in \{1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$. Using the enumeration procedure, the optimal solution and the value of the objective function were obtained for these 120 problems. Having the optimal solutions, one can adjust the parameters of the genetic algorithm. The following selections were used for the parameter values: POP $\in \{10, 20\}, P_{mu} \in \{0.1, 0.2\}$ and $P_{\rm cr} \in \{0.5, 1\}$. A complete factorial design of the above parameters is considered, so that a total number of $2^*2^*2^*120 = 960$ problems have been tested using NGA for RIPT. For adjusting the parameters, POP, P_{mu} and $P_{\rm cr}$, SSS and WS were employed for selecting activities from EAS. Then, average and maximum percent deviation from the optimal solution and the percentage of the problems that reached optimality were calculated. The average number of generations for 2 seconds computing time was 121.13. For the statistical analysis of the results on the percent deviation from the optimal solution, knowledge of its distribution function is needed. Although this random variable is not necessarily normally distributed, since the average percent deviation of 120 problems is being studied based on the central limit theorem, one can safely assume that this average percentage is normally distributed. This assumption was supported using the goodness of fit test. In the statistical analysis, in order to find good combinations of the NGA parameters, multi factor analysis of variance was employed. Analysis of variance showed that all of the above factors are statistically significant. Using a Duncan multiple range test, the following rankings were observed: for POP; $10 \succ\succ 20$, for $P_{mu}; 0.1 \succ 0.2$ and for $P_{\rm cr}; 1 \succ\succ 0.5$, where "$\succ$" denotes better and "$\succ\succ$" denotes significantly better. Table 1 shows the combinations of the parameter values and their results with 2 seconds computing time. Without going into detail, it is interesting to note that if these problems were considered as RIP, the percentage of problems to reach optimality would be between 88 and 95.5. The last column of Table 1 shows the percentage of problems that reached optimality when considered as RIP.

For testing the effects of crossover operator, selection method from EAS and computing time limit, two crossover operators were considered, i.e., one-point crossover (1P), two-point crossover (2P) and two selection methods, i.e. WS and RLFT.

For this test, POP = 10, $P_{mu} = 0.1$ and $P_{\rm cr} = 1$ were considered. Again, a complete factorial design was employed and $2^*2^*120 = 480$ problems were evaluated. Average percent deviation from the optimal solution was calculated for a 0.4, 0.8, 1.2, 2, 4 and 10 second computing time limit. Figure 3 shows the results: The X-axis shows the computation times and the Y-axis shows the average percent deviations from the optimal solution. A multi factor analysis of variance was conducted. For all selected time limits and with a confidence level of less than 1%, there was a statistically significant difference between the crossover operators and, also, between the selection methods.

Figure 3 shows that RLFT is better for time limits less than 1.2 seconds and WS outperforms the others for time limits exceeding 1.2 seconds. This was expected, since WS tries to assign uniform probabilities to all possible activity lists. This property gives the

Table 1. Combinations of parameter values.

P_{mu}	P_{cr}	POP	Av. Dev.%	Max. Dev.%	RIPT Optimal%	RIP Optimal%
0.1	0.5	10	0.28	2.9	49	92.2
0.1	0.5	20	0.30	3.1	48.3	88.2
0.1	1	10	0.24	2.5	54.3	95.5
0.1	1	20	0.28	3.0	51.1	90.1
0.2	0.5	10	0.29	2.9	49.2	89.3
0.2	0.5	20	0.30	3.0	48.5	89
0.2	1	10	0.25	2.6	49.2	91.1
0.2	1	20	0.28	2.8	47.8	90.2

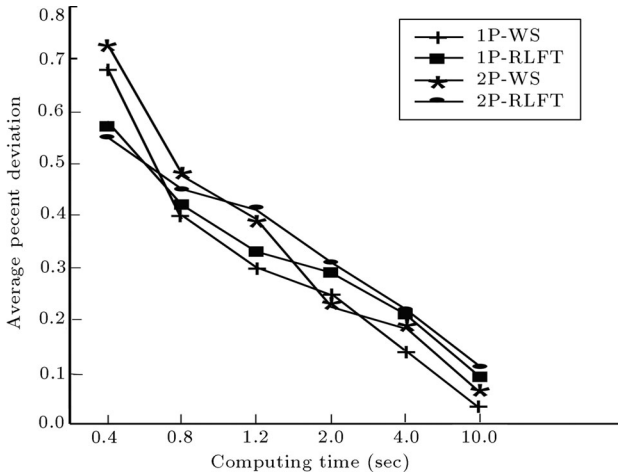

Figure 3. Average percent deviation from optimal solution for four combinations of crossover operators and selection methods from EAS for RIPT.

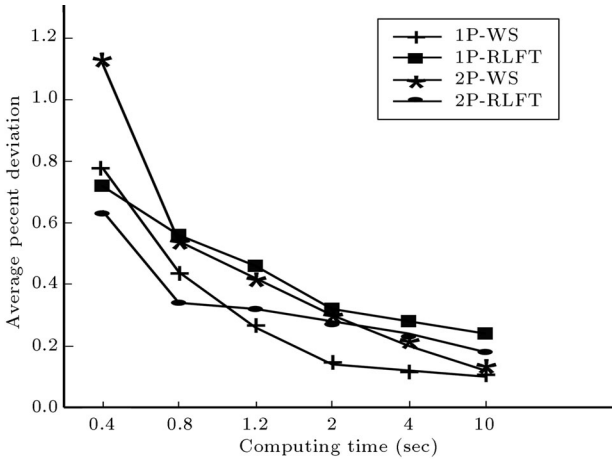

Figure 4. Average percent deviation from optimal solution for four combinations of crossover operators and selection methods from EAS for RIP.

NGA a chance to find the promising areas of the feasible solutions, provided it has enough time to reach them.

On the other hand, RLFT gives a higher chance to those activity lists that seem promising and, hence, it is expected that RLFT will perform better for shorter time limits. However, the optimal solution may not be among the activity lists that RLFT selects with a higher probability.

The above problems were also tested when the tardiness penalty is very large. In this case, in fact, one has RIP. The results are shown in Figure 4. Multiple factor analysis of variance showed a statistically significant difference between the selection methods from EAS with a confidence level of less than 1% for all time limits, but no significant difference was observed between different crossover operators for time limits greater than 0.8 seconds.

ProGen Instances with 10 and 14 Non-Dummy Activities

For more analysis of the performance of the NGA, 20 instances with 10 activities and 20 instances with 14 activities were generated and tested, each with 4 resources using ProGen, with the same parameters used for the 20 activity problems mentioned previously.

Here, again, one set $T = \theta^*$ EFT, $\theta \in \{1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$. Therefore 2*20*6 = 240 problems were generated and solved, both as RIP and RIPT (480 problems altogether), using the enumeration procedure described before. These problems were given to the NGA with the following parameters: POP =10, $P_{mu} = 0.1$ and $P_{\text{cr}} = 1$.

Tables 2 and 3 show the results of this test for RIPT and RIP, respectively, considering a computing time limit of 0.4 seconds. These tables also contain the results of the 20 activity problems mentioned previously.

Table 2. Results of NGA on RIPT with 10, 14 and 20 problems within 0.4 seconds.

No. of Activities	Av. Dev. %	Max. Dev. %	Optimal %
10.00	0.04	0.25	50.50
14.00	0.05	0.29	43.03
20.00	0.55	3.95	38.21

Table 3. Results of NGA on RIP with 10, 14 and 20 problems within 0.4 seconds.

No. of Activities	Av. Dev. %	Max. Dev. %	Optimal %
10.00	0.00	0.00	100.00
14.00	0.04	3.98	97.80
20.00	0.63	7.20	69.70

ProGen Instances with 30 Non-Dummy Activities

Drexl and Kimms [12] used mathematical programming to find the lower bound, LB, and the upper bound, UB, for RIP. They employed two mathematical programming techniques, Lagrangian relaxation and Column generation, to construct two algorithms for finding these bounds. To measure the effectiveness of their algorithms, they used the two following measures of performance:

percentage of improvement of the upper bound

$$= \frac{UB_0 - UB}{UB_0} \times 100,$$

Percentage of improvement of the lower bound

$$= \frac{LB - LB_0}{LB_0} \times 100,$$

where UB_0 is the objective function value of the problem, if one sets R_ks at their obvious upper bounds, $\overline{R_k}$s, and LB_0 is the value of objective function, if one sets R_ks at their obvious lower bounds, $\overline{R_k}$s ($\underline{R_k}$s and $\overline{R_k}$s were given in previous sections).

The two algorithms were tested on 480 instances with 30 non-dummy activities and 4 resources from the project scheduling problem library PSBLIB [13]. The resource limits are ignored in their test, since the resource limitation is irrelevant in RIP. For these problems, the ProGen parameters were: NC $\in \{1.5, 1.8, 2.1\}$ and RF$\in \{0.25, 0.5, 0.75, 1.0\}$. For each combination of NC and RF, 40 instances exist in PSBLIB, which gives a total of 3*4*40 = 480 instances. For each instance, the value of the project due date T, is considered as $T = \theta^*$ EFT, where $\theta \in \{1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$. This gives a test bed of 6*480 = 2880, RIP problems. Also, a random number was chosen from the interval [1,10] as the unit resource cost for each resource [14]. For each investigated instance, the average percentage of improvement of the lower bound and the upper bound, using Lagrangian relaxation and column generation methods, is computed.

All these 2880 problems were given to the NGA with the following parameters: POP $= 10, P_{mu} = 0.1, P_{\rm cr} = 1, 2P$ and WS, with a computing time limit of 2 seconds. The result of the NGA for each instance is the best feasible individual in the final population and the unfitness value of this individual is an upper bound for the instance. The NGA algorithm does not produce a lower bound, therefore, to evaluate the performance of the NGA using the above problems, only the percentage improvement of the upper bound ($100^*(UB_0 - UB)/UB_0$) is compared with that of Drexl and Kimms [12]. Table 4 shows the percentage improvement of the upper bound using NGA.

The average number of generations for these problems was 102.82. The comparison of the entries in Table 4 with those of Drexl and Kimms [12] and, also, with SKGA shows that the NGA performs much better in finding upper bounds.

CONCLUSION

A new genetic algorithm is presented for solving the resource investment problem when tardiness is permitted with a penalty. This algorithm has many parameters. To adjust the parameters 120 RIPT

Table 4. Average percent improvement of the upper bound using NGA.

$n = 30$	θ	1	1.1	1.2	1.3	1.4	1.5
NC = 1.5	RF = 0.25	34.60	40.56	42.46	43.36	43.86	44.12
	RF = 0.5	32.25	41.54	46.60	50.36	53.56	55.84
	RF = 0.75	36.27	42.95	47.43	51.23	54.27	56.95
	RF = 1	38.20	43.68	48.30	52.00	55.13	57.50
NC = 1.8	RF = 0.25	29.68	36.99	40.04	41.11	41.45	41.60
	RF = 0.5	30.26	39.00	44.17	48.56	51.08	53.56
	RF = 0.75	33.53	39.79	44.51	48.09	51.07	53.78
	RF = 1	29.69	37.36	42.05	46.40	49.83	52.52
NC = 2.1	RF = 0.25	24.79	34.70	38.30	38.92	39.18	39.40
	RF = 0.5	27.88	36.66	42.49	46.75	49.79	52.52
	RF = 0.75	26.50	35.15	40.41	44.56	48.14	50.90
	RF = 1	29.80	37.50	42.71	46.63	49.49	52.16

and 120 RIP were solved to optimality, each with 20 activities. These solved problems were also used in testing the computational performance of NGA. Additionally, for the computational performance test; 240 RIPT and 240 RIP were also solved, with 10 and 14 activities to optimality. The NGA results on these problems, compared with SKGA results and their optimal solutions, were quite satisfactory.

In addition, Drexl and Kimms problems [12] were used for test purposes only, on the basis of the upper bound on the optimal value of the objective function. The performance of the NGA on these problems was also very good.

REFERENCES

1. Baker, K., *Introduction to Sequencing and Scheduling*, John Wiley and Sons, Inc. (1974).
2. Mohring, R.H. "Minimizing costs of resource requirements in project networks subject to a fix completion time", *Operations Research*, **32** pp 89-120 (1984).
3. Hartmann, S. "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, **45**, pp 733-750 (1998).
4. Pritsker, A.A.B., Watters, L.J. and Wolfe, P.M. "Multi-project scheduling with limited resources", *Management Science*, **16**, pp 93-108 (1969).
5. Shadrokh, S. and Kianfar, F. "A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty", Industrial Engineering Department, Sharif University of Technology, *European Journal of Operational Research* (in press).
6. Akpan, E.O.P. "Optimal resource determination for project scheduling", *Production Planning and Control*, **8**(5), pp 462-468 (1997).
7. Kelley, J.E., Jr. "The critical path method: Resources planning and scheduling", *Industrial Scheduling*, Prentice-Hall, J.F. Muth and G.L. Thompson, Eds., New Jersey, pp 347-365 (1963).
8. Kolish, R. "Serial and parallel resource constrained project scheduling methods revisited: Theory and computation", *European Journal of Operational Research*, **90**, pp 320-333 (1996).
9. Kolisch, R. and Hartmann, S. "Heuristic algorithms for the resource constrained project scheduling problem: Classification and computational analysis", *Project Scheduling*, J. Weglarz, Ed. (1999).
10. Schwindt, C. "Generation of resource constrained scheduling problems with minimal and maximal time lags", *Report WIOR-489*, Universität Karlsruhe (1996).
11. Demeulemeester, E. and Herroelen, W. "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science*, **38**, pp 1803-1818 (1992).
12. Drexl, A. and Kimms, A. "Optimization guided lower & upper bounds for the resource investment problem", *Journal of the Operational Research Society*, **52**, pp 340-351 (2001).
13. Kolisch, R. and Sprecher, A. "PSBLIB- A project scheduling problem library", *European Journal of Operational Research*, **96**, pp 205-216 (1997).
14. Kimms, A., *Personal communication* (2002).