A Genetic Algorithm for Scheduling Alternative Tasks subject to Technical Failure

Dalila B.M.M. Fontes and José Fernando Gonçalves

Abstract Nowadays, organizations are often faced with the development of complex and innovative projects. This type of projects often involves performing tasks which are subject to failure. Thus, in many such projects several possible alternative actions are considered and performed simultaneously. Each alternative is characterized by cost, duration, and probability of technical success. The cost of each alternative is paid at the beginning of the alternative and the project payoff is obtained whenever an alternative has been completed successfully. For this problem one wishes to find the optimal schedule, i.e., the starting time of each alternative, such that the expected net present value is maximized. This problem has been recently proposed in [22], where a branch-and-bound approach is reported. Since the problem is NP-*Hard*, here we propose to solve the problem using genetic algorithms.

1 Introduction

Companies must plan and optimize their activities in a uncertain environment. The uncertainties may come from several different parts of their business. The uncertainties most commonly addressed in the literature are related to the costs and returns associated with the business. Regarding scheduling problems the most frequently studied uncertainties are resource breakdowns and duration variability. However, other sources of uncertainty exist. For example, Research and Development (R&D) companies, highly dependent on innovation, also face uncertainty regarding the success of their initiatives. These initiatives, usually called projects, may fail. Thus, in

Dalila B.M.M. Fontes
Faculdade de Economia da Universidade do Porto and LIAAD INESC TEC, Rua Dr. Roberto, Frias 4200-464 Porto e-mail: fontes@fep.up.pt

José Fernando Gonçalves
Faculdade de Economia da Universidade do Porto and LIAAD INESC TEC, Rua Dr. Roberto, Frias 4200-464 Porto e-mail: jfgoncal@fep.up.pt

order to deal with this kind of uncertainty companies may have to consider several alternative ways of developing their projects (see e.g. [27, 28]). In this type of projects, the alternatives are of the same kind, although different, and pursue a similar goal. For example, their execution may represent the repetition of trials until success in one is achieved. Usually, the alternatives are related and some alternatives may imply the execution of some other alternatives, i.e. there are precedence relations between some of the alternatives.

This work addresses the scheduling of alternatives subject to technical failure, in order to maximize the expected Net Present Value (NPV) of the projet. The NPV of a project is the discounted value of the project cash flows. The NPV is affected by the project schedule and in capital-intensive industries, the timing of expenditures has a major impact on project feasibility and profitability.

Most of the relevant sources of literature considering activity failure come from chemical engineering applications, where Grossmann and his colleagues have been addressing such problem. In [25] a mixed integer linear programming model was proposed to schedule the activities of a single product considering precedence constraints. Activities have associated a cost, a duration, and a probability of success. The objective was to minimize the expected cost. This model was subsequently used on a specific application [26]. In [20] the authors propose a two stage stochastic optimization approach to account for the uncertainty in the outcome of the trials. A recent survey on optimization challenges and opportunities in the pharmaceutical industry can be found in [19].

Other scheduling problems involving activity failures have been addressed, see the survey in [7]. De Reyck and colleagues study the scheduling of activities with uncertain outcomes, where project success is achieved only if all individual activities succeed. In [8], the authors have considered the project scheduling problem with uncertain activity outcomes and known durations. This work was extended in [6, 4] where activity durations are stochastic. More recently, in [2] the scheduling of projects subject to failure has been considered. In this problem, several projects, each consisting of several activities, have to be scheduled. If an activity of a project fails, the project fails. The authors also consider resource constraints and the possibility of outsourcing. Modular projects, i.e., projects that include the execution of several modules, each of which consisting of several activities, have been considered in [3, 5]. For such a project to be successful every module must succeed. A module succeeds if at least one of its activities succeeds. In the former work, activity durations are deterministic and activities must be performed sequentially, while in the latter the durations are stochastic and the resources unlimited.

Following on the work of Ranjbar and Morteza [23], we focus on a single firm facing a R&D project or the development of a new product. There are several alternatives of executing the project and its success requires the successful execution of at least one of the available alternatives. Each alternative consists of a single activity and is characterized by a cost, a duration, and a probability of technical success. The successful completion of the project provides a given payoff. These alternatives can be pursued either in parallel or sequentially. The objective is to schedule the activities in such a way as to maximize the expected Net Present Value (eNPV) of the project. The eNPV takes into account the activity costs, the cash flows generated by the successful completion of project, the activity durations and starting times, and the probability of failure of each of the activities. Some alternatives may imply the implementation of other alternatives. This is a recently proposed problem and it has been addressed by exact methods only [22, 23]. Since this is a NP-*hard* problem (see [23]), an exact algorithm without an exponential time complexity is unlikely to exist. Thus, in here we propose a genetic algorithm since only heuristic methods are able to solve real sized problems.

Section 2 defines and provides a mathematical programming model for this problem. Section 3 discusses the methodology proposed to solve the problem and in Section 4 the computational experiments are reported. Finally in Section 5 some conclusions are drawn.

2 Problem Definition and Formulation

Given a project for which there are several alternative ways of execution, one wishes to determine the order in which these alternatives should be executed such that the project expected net present value is maximized. Alternatives pursue a similar target and consist of one activity[1]. Activities should be executed without interruption and are characterized by a cost, a duration, a set of precedence constraints, and a probability of technical success. Activity costs are to be paid at the start of the activity. The outcomes of the different tasks are considered to be independent. The successful completion of a project provides a payoff and is achieved if at least one alternative is successfully executed.

Before introducing the mathematical programming model, let us illustrate the problem by resorting to the example used in [23]. Consider a project consisting of 5 alternatives, for which the information is given in Table 1. Note that the execution of activity 4 requires activity 1 to be previously executed. Nevertheless, activity 4 can be executed and be successful regardless of the outcome of the execution of activity 1. It is assumed a 5% monthly discount rate, a project deadline of 29 months, and a project payoff, achieved in case of technological success, of 2770 dollars.

These alternatives can be scheduled in many different ways. The two extreme ones being, the parallel and the sequential schedules. These schedules are given in Fig. 1.

Note that, while the parallel schedule anticipates the project completion and thus the net payoff is larger, it also leads to the highest costs since it starts activities without waiting to find out if the previously started one has had success. Thus, in this type of schedules some, in progress, alternatives of the project will be ignored. For instances, if activity 5 has success activities 1 and 2, which have already been initiated and paid for will be ignored. Here the costs are typically higher. In the serial schedule, since only one activity is being performed at any time this risk does not

[1] Since each alternative consists of a single activity, here and hereafter we will use indifferently alternative and activity.

Table 1 Alternatives data (the costs are given in dollars and the duration in months).

Alternative number	Costs ($)	Duration (months)	PTS	Precedent activities
1	51	8	0.73	–
2	31	6	0.62	–
3	87	3	0.91	2
4	28	7	0.57	1
5	80	4	0.86	–

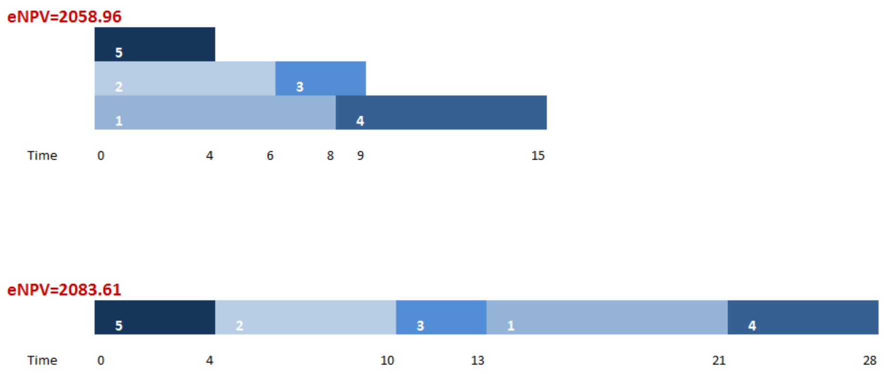

Fig. 1 Extreme Schedules: Parallel and Serial Schedules.

exist. Therefore, it is more conservative in terms of costs. However, in this case if an alternative fails it takes longer to be able to have another tried and thus, the project payoff is typically smaller since it is obtained later. Therefore, a trade-off between costs and project payoff (project duration) must be searched for.

At time t the project payoff C is obtained if and only if at least one of the activities finishing at time t (A_t) succeeds and all activities that have finished before time t (B_t) have failed; otherwise the payoff had already been received. Thus, the expected payoff at time t is given by:

$$\prod_{j \in B_t} (1-p_j) \times \left(1 - \prod_{k \in A_t} (1-p_k)\right) \times C. \tag{1}$$

As said before, each activity i has a cost (c_i) associated to its execution that must be paid upfront, i.e., at the time that the activity is started (s_i). In addition, an activity is only started if all activities that have finished before (B_{s_i}) or at (A_{s_i}) its starting time have failed; otherwise the project had already been concluded. Thus, the expected cost incurred with activity i at its starting time s_i can be written as:

$$\prod_{j \in \{B_{s_i} \bigcup A_{s_i}\}} (1-p_j) \times c_i. \tag{2}$$

A summary of the notation used is provided in Table 2.

Table 2 Notation used for the mathematical programming model).

Symbols	Description
N	set of available alternatives.
i, j, k	alternative indices.
c_i	cost of alternative i.
d_i	duration of alternative i.
p_i	probability of technical success of alternative i.
t_{max}	project deadline.
t	time index.
C	Project payoff.
r	discount rate.
A	set of precedence constraints.
B_t	auxiliary decision variable: set of alternatives finishing before t.
A_t	auxiliary decision variable: set of alternatives finishing at t.
s_i	decision variable: starting time of alternative i.

The project net value is then obtained by subtracting all expected costs from all expected payoffs. However, since we are maximizing the project expected net present value, the costs and payoffs given by equations (1) and (2) need to be discounted. The scheduling decisions are only constrained by the precedence relations amongst the alternatives. Therefore, the complete mathematical model is as given in equations (3) to (5).

$$\text{Minimize} \quad \sum_{t=1}^{t_{max}} \left(\prod_{j \in B_t} (1-p_j) \times \left(1 - \prod_{k \in A_t} (1-p_k)\right) \times C \times e^{-rt} - \prod_{j \in \{B_{s_i} \bigcup A_{s_i}\}} (1-p_j) \times c_i \times e^{-rs_i} \right) \tag{3}$$

Subject to

$$s_i + d_i \leq s_j, \qquad \forall j \in A \text{ and } \forall i \in N. \tag{4}$$

$$s_i \in \mathbb{N}, \qquad \forall i \in \mathbb{N}^+. \tag{5}$$

3 Methodology

In this section, we provide an overview of the proposed solution process. This is followed by a discussion on the proposed Biased Random-Key Genetic Algorithm (BRKGA), including detailed descriptions of the solution encoding and decoding procedures, evolutionary process, and fitness function.

3.1 Overview

The new approach is based on a constructive heuristic algorithm which inserts activities, one at a time, in a partial schedule for the problem. Once all the activities are inserted, a solution is obtained. The new approach proposed in this paper combines a BRKGA with a novel insertion decoding procedure. The role of the genetic algorithm is to evolve the encoded solutions, or *chromosomes*, which represent the *parameters* that will be used by the solution builder to construct a schedule. For each chromosome, the following phases are applied to decode the chromosome:

1. *Decoding of the parameters*: this first phase decodes the chromosome into a sequence of activities, as well as each activity scheduling mode (SM). The former determines the activities to be started, while the latter determines whether each activity is scheduled forward or backward.
2. *Construction of a solution*: The second phase makes use of the activities and SMs defined in phase 1 and uses the solution builder procedure to construct a schedule.
3. *Fitness evaluation*: The final phase computes the fitness of the solution, by computing the expected net present value as given in equation (3).

Fig. 2 illustrates the sequence of decoding steps applied to each chromosome generated by the BRKGA. The remainder of this section describes in detail the genetic algorithm.

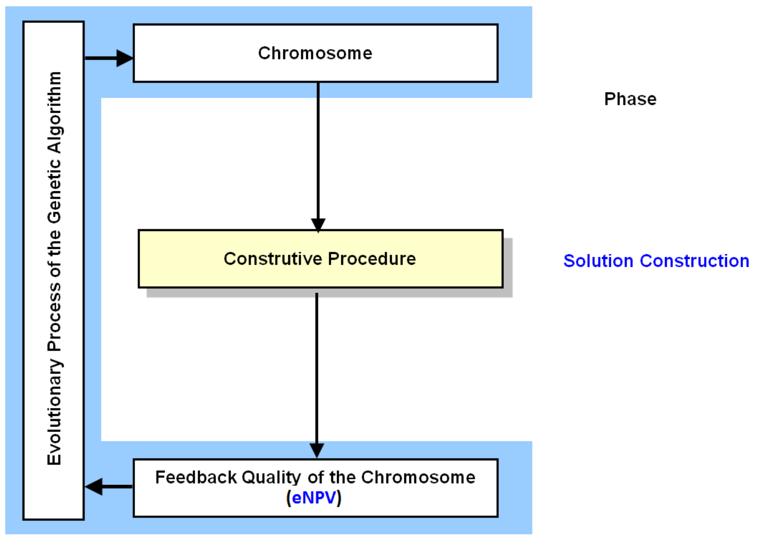
Evolutionary Process of the Genetic Algorithm
Chromosome
Phase
Construtive Procedure
Solution Construction
Feedback Quality of the Chromosome
(eNPV)

Fig. 2 Architecture of the algorithm.

3.2 Biased random-key genetic algorithm

Random-key genetic algorithms (RKGA) or genetic algorithms with random keys were introduced in [1] for solving sequencing or optimization problems whose solutions can be represented as permutations. In a RKGA, chromosomes are represented as vectors of randomly generated real numbers in the interval $[0,1]$. A deterministic algorithm, the *decoder*, takes as input a chromosome and associates with it a solution of the combinatorial optimization problem for which an objective value or fitness value can be computed.

RKGAs are particularly attractive for sequencing problems and/or when the chromosomes have several parts (see for example [15], [18], [16], [17], [11], [13], [10], [24], and [12]). Unlike traditional GAs, which need to use special repair procedures to handle permutations or sequences, RKGAs move all the feasibility issues into the objective evaluation procedure and guarantee that all offspring formed by crossover result into feasible solutions. When the chromosomes have several parts, traditional GAs need to use different genetic operators for each part. However, since RKGAs use the *parameterized uniform crossover* of [29] (instead of the traditional one-point or two-point crossovers), they do not need to have different genetic operators for each part.

A RKGA evolves a *population* of random-key vectors over a number of *generations* (iterations). The initial population is made up of p vectors of r random keys. Each component of the solution vector, or random key, is generated independently at random in the real interval $[0,1]$. After the fitness of each individual is computed by the decoder in generation g, the population is partitioned into two groups of individuals: a small group of p_e *elite* individuals, i.e., those with the best fitness values, and the remaining set of $p - p_e$ *non-elite* individuals. To evolve a population g, a new generation of individuals is produced. All elite individuals of the population of generation g are copied without modification to the population of generation $g+1$. RKGAs implement mutation by introducing *mutants* into the population. A mutant is a vector of random keys generated in the same way that an element of the initial population is generated. At each generation, a small number p_m of mutants is introduced into the population. With $p_e + p_m$ individuals accounted for in population $g+1$, $p - p_e - p_m$ additional individuals need to be generated to complete the p individuals that make up population $g+1$. This is done by producing $p - p_e - p_m$ offspring solutions through the process of *mating* or *crossover*.

A BRKGA [14], differs from a RKGA in the way parents are selected for mating. While in the RKGA of [1] both parents are selected at random from the entire current population, in a BRKGAs each element is generated by combining a parent selected at random from the elite partition of the current population with another from the rest of the population, also randomly selected. Repetition in the selection of a parent is allowed and therefore an individual can produce more than one offspring in the same generation. As in RKGAs, parameterized uniform crossover is used to implement mating in BRKGAs. Let ρ_e be the probability that the vector component of an elite parent is inherited by the offspring. For $i = 1,\dots,r$, the i-th component $c(i)$ of the offspring vector c takes on the value of the i-th component $e(i)$ of the elite parent e

with probability ρ_e and the value of the i-th component $\bar{e}(i)$ of the non-elite parent $\bar{e}$ with probability $1-\rho_e$.

Once the next population is complete, the corresponding fitness values are computed for all of the newly created random-key vectors and the population is partitioned into elite and non-elite individuals to start a new generation.

A BRKGA searches the solution space of the combinatorial optimization problem indirectly by searching the r-dimensional continuous hypercube, using the decoder to map solutions in the hypercube to solutions in the solution space of the combinatorial optimization problem where the fitness is evaluated.

To specify a BRKGA, one simply needs to specify how solutions are encoded and decoded and how their corresponding fitness values are computed. This is done in the next sections.

3.2.1 Chromosome representation and decoding

A chromosome encodes a solution to the problem as a vector of random keys. In a direct representation, a chromosome represents a solution to the original problem, and is called *genotype*, while in an indirect representation it does not and special procedures are needed to obtain from it a solution called a *phenotype*. In the present context, the solutions will be represented indirectly by parameters that are later used by a decoding procedure to obtain a solution. To obtain the solution (phenotype) we use the decoding procedures described in Section 3.2.2.

In this paper, a solution to the problem is represented indirectly by the chromosome structure given in Fig. 3, where n is the number of activities. Overall the chromosome has $n+(n-1)2n$ genes.

$$
\begin{aligned}
\textbf{Chromosome} = (\ & \text{gene}_1 , \dots , \text{gene}_n , \\
& \text{gene}_{n+1} , \dots , \text{gene}_{n+n} , \ \text{gene}_{n+n+1} , \dots , \text{gene}_{n+n+n} \\
& \dots \\
& \text{gene}_{n+(n-2)2n+1} , \dots , \text{gene}_{n+(n-2)2n+n} , \ \text{gene}_{n+(n-2)2n+n+1} , \text{gene}_{n+(n-2)2n+n+n} \ )
\end{aligned}
$$

Fig. 3 Chromosome representation.

The genes in blue are used by the solution builder (decoding procedure) to determine which activity or activities are to be scheduled at each iteration into the partial schedule and the genes in red are used to decide whether the activity chosen is going to be scheduled forward or backward. Note that in the first iteration the activities must always be scheduled forward. An activity is considered schedulable if all of its predecessors have already been scheduled and if its blue gene value is greater than or equal to 0.5. If the value of the red gene is greater than or equal to 0.5 then the chosen activity is scheduled backward; otherwise it is scheduled forward.

The decoding (mapping) of each chromosome into a schedule is performed by the solution builder, which described in the next section.

3.2.2 Solution builder

The solution builder follows a sequential process that inserts activities into a partial schedule. The order in which activities are inserted into the partial scheduled and the corresponding mode (forward or backward) are evolved by the BRKGA. The solution builder is comprised of the following two main steps:

1. Selection of activities to be inserted;
2. Selection of the mode used for the insertion in the partial solution of the activities selected in step 1.

The possible insertion times for scheduling an activity are provided by the starting (S) or ending times (E) of the activities already scheduled. Amongst these, we are only interested on the ones that are feasible regarding the precedence relations between activities.

To illustrate how the solution builder works we used again the example provided in Table 1. A solution will be constructed using the chromosome in Fig. 4.

1

Act
0.7
0.2
0.1
0.25
0.14

2

Act	S/E
0.2	0.4
0.1	0.3
0.33	0.3
0.46	0.2
0.85	**0.62**

3

Act	S/E
0.12	0.41
0.9	**0.3**
0.23	0.3
0.26	0.2
0.15	0.62

4

Act	S/E
0.12	0.41
0.09	0.03
0.63	**0.37**
0.76	**0.14**
0.05	0.62

5

Act	S/E
0.4	0.23
0.55	0.58
0.6	0.55
0.5	0.95
0.4	0.12

Fig. 4 Chromosome example used in the illustration of the solution builder.

Initially only time zero is available for scheduling one or more activities. According to the precedence constraints the activities which are schedulable are 1, 2, and 5. However, only activity 1 has a blue gene value greater than or equal to 0.5 (0.7), so activity 1 is the only one selected for insertion into the partial schedule. Since this activity is scheduled at time 0, it must be scheduled forward. At this point the partial schedule looks like the one given in Fig. 5.

The next time to be considered for insertion is time 8. Only activities 2, 4, and 5 can be considered since due to the precedence constraints activity 3 cannot yet be scheduled. According to the second column only activity 5 has a blue gene value greater than or equal to 0.5 (0.85), thus only activity 5 can be scheduled. Given that the value in sub-column S/E of column 2 for activity 5 is 0.62 (> 0.5), then activity 5 is scheduled backward. At this point the partial schedule looks like the one given in Fig. 6.

The next time to be considered for insertion is time 4, the only one available not yet considered. According to the precedence constraints only activity 2 can be

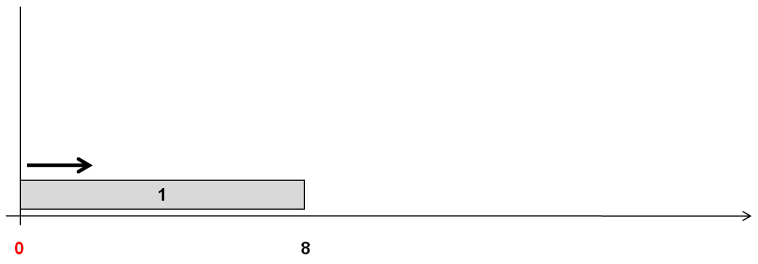

Fig. 5 Step 1 of solution builder - Partial schedule after inserting activity 1.

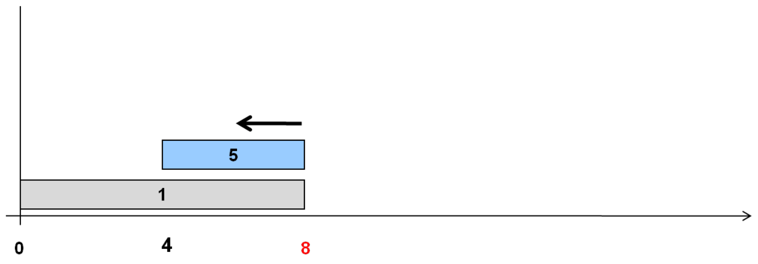

Fig. 6 Step 2 of solution builder - Partial schedule after inserting activity 5.

started and its blue gene has a value greater than or equal to 0.5 (0.9), thus activity 2 is inserted into the partial schedule. Since the value in sub-column S/E of column 3 for activity 2 is smaller than 0.5 (0.3), then activity 2 is scheduled forward. The partial schedule obtained is illustrated in Fig. 7.

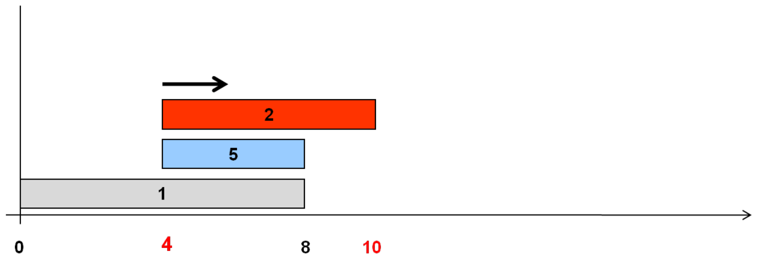

Fig. 7 Step 3 of solution builder - Partial schedule after inserting activity 2.

The next time to be considered for insertion is time 10. According to the precedence constraints and the fourth column of the chromosome, the activities which can be started are activities 3 and 4. Since the value for both in sub-column S/E is smaller than 0.5 (0.37 and 0.14, respectively), both are scheduled forward. Given that there are no more activities to be scheduled the final schedule is the one given in Fig. 8.

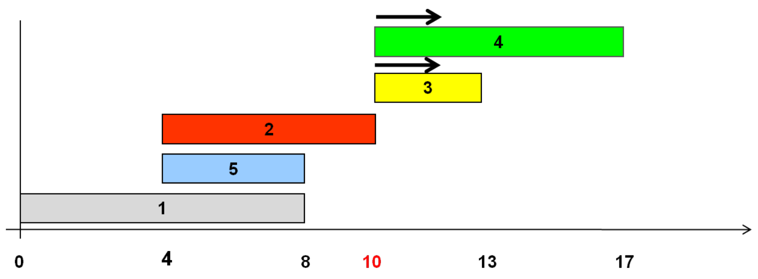

Fig. 8 Step 4 of solution builder - Final schedule after inserting activities 3 and 4.

Finally, the fitness of the solution, i.e., the project expected net present value, is computed according to equation (3) and in this case it is 1702.87 dollars.

4 Computational Experiments

The methodology proposed here was tested on the randomly generated test problem instances used by Ranjbar and Morteza [23].

The 60 problem instances used have been generated using RanGen ([9]). Four different problem sizes and three different order strength values[2] have been considered. For each of these 12 combinations five problem instances were generated by choosing uniform random values for durations, costs, and PTS in the intervals [1,10], [10,100], and [0.5,1], respectively. For each problem instance the payoff has been chosen to be 5 times the sum of the alternatives cost and the discount rate was set to 5%.

The genetic algorithm has been coded using C++ and the experiments have been carried out on a computer with an Intel Core i7-2630QM @2.0 GHZ CPU running the Linux operating system with Fedora release 16.

We compare the best solutions obtained with the genetic algorithm with those of the branch and bound developed by Ranjbar and Morteza [23]. The BRKGA was able to find an optimal solution to 47 of the 60 problem instances considered. The computational time required by the BRKGA was always below 10 seconds and on average was about 7 seconds.

The BRKGA proposed here, when compared to the best alternative method [23], provides an enormous improvement since it improves substantially the computational time performance. In addition, it finds very good solutions, actually optimal for most problems. It should be noticed that for the worst case class of problems (problems with 12 alternatives and order strength of 0.4), the alternative method

[2] The number of precedence-related activity pairs divided by the theoretically maximum number of such pairs in the network [21].

takes around 1 hour and 45 minutes. The optimality gaps are always below 2.5% and the average optimality gap for the 60 problems solved is below 0.2%.

5 Conclusions

We have presented a genetic algorithm for scheduling projects with alternative tasks subject to technical failure. This is a newly proposed problem and thus far only branch-and-bound algorithms have been proposed. Results obtained compare favorably with the ones reported in current literature.

The genetic algorithm proposed finds nearly optimal solutions, actually optimal for most solved problem instances. In addition, the computational time requirements are greatly improved. Specially, for larger size problem instances. The magnitude of the improvement grows with problem size. For the 12 alternative problems with order strength of 0.4 the BRKGA requires less than 10 seconds, while the literature reports about 1 hour and 44 minutes. Nevertheless, the average gap is only about 0.2%.

Acknowledgements This work has been partially supported by projects PTDC/EGE-GES/117692/2010 and NORTE-07-0124-FEDER-000057 funded by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF) and the Programme COMPETE, and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

References

1. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6**, 154–160 (1994)
2. Colvin, M., Maravelias, C.T.: R&d pipeline management: Task interdependencies and risk management. European Journal of Operational Research **215**(3), 616–628 (2011)
3. Coolen, K., Wei, W., Nobibon, F.T., Leus, R.: Scheduling modular projects on a bottleneck resource. Journal of Scheduling **17**(1), 67–85 (2014)
4. Creemers, S., De Reyck, B., Leus, R.: R&d project planning with multiple trials in uncertain environments. In: Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on, pp. 325–329. IEEE (2009)
5. Creemers, S., De Reyck, B., Leus, R.: Project planning with alternative technologies in uncertain environments. FEB Research Report KBI_1314 (2013)
6. Creemers, S., Leus, R., De Reyck, B., Lambrecht, M.: Project scheduling for maximum npv with variable activity durations and uncertain activity outcomes. In: Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on, pp. 183–187. IEEE (2008)
7. De Reyck, B., Grushka-Cockayne, Y., Leus, R.: A new challenge in project scheduling: The incorporation of activity failures. Tijdschrift voor economie en management **52**(3), 411 (2007)
8. De Reyck, B., Leus, R.: R&d project scheduling when activities may fail. IIE transactions **40**(4), 367–384 (2008)

9. Demeulemeester, E., Vanhoucke, M., Herroelen, W.: Rangen: A random network generator for activity-on-the-node networks. Journal of Scheduling **6**(1), 17–38 (2003)
10. Fontes, D.B.M.M., Gonçalves, J.F.: A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in nonlinear cost flow networks. Optimization Letters **7**(6), 1303–1324 (2013)
11. Gonçalves, J., Resende, M.: A parallel multi-population biased random-key genetic algorithm for a container loading problem. Computers & Operations Research **39**(2), 179–190 (2012)
12. Gonçalves, J.F., Costa, M.D., Resende, M.G.C.: A biased random-key genetic algorithm for the minimization of open stacks problem. International Transactions in Operational Research (2014). To appear
13. Gonçalves, J.F., Resende, M.G.: A biased random key genetic algorithm for 2d and 3d bin packing problems. International Journal of Production Economics (2013)
14. Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms forcombinatorial optimization. Journal of Heuristics **17**, 487–525 (2011)
15. Gonçalves, J.F., Almeida, J.R.: A hybrid genetic algorithm for assembly line balancing. Journal of Heuristics **8**, 629–642 (2002)
16. Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C.: A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research **167**, 77–95 (2005)
17. Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C.: A genetic algorithm for the resource constrained multi-project scheduling problem. European Journal of Operational Research **189**, 1171–1190 (2009)
18. Gonçalves, J.F., Resende, M.G.C.: An evolutionary algorithm for manufacturing cell formation. Computers and Industrial Engineering **47**, 247–273 (2004)
19. Laínez, J.M., Schaefer, E., Reklaitis, G.V.: Challenges and opportunities in enterprise-wide optimization in the pharmaceutical industry. Computers & Chemical Engineering **47**, 19–28 (2012)
20. Maravelias, C.T., Grossmann, I.E.: Simultaneous planning for new product development and batch manufacturing facilities. Industrial & engineering chemistry research **40**(26), 6147–6164 (2001)
21. Mastor, A.A.: An experimental investigation and comparative evaluation of production line balancing techniques. Management Science **16**(11), 728–746 (1970)
22. Ranjbar, M.: A branch-and-bound algorithm for scheduling of new product development projects. International Transactions in Operational Research **20**(2), 251–266 (2013)
23. Ranjbar, M., Davari, M.: An exact method for scheduling of the alternative technologies in r&d projects. Computers & Operations Research **40**(1), 395–405 (2013)
24. Roque, L.A.C., Fontes, D.B.M.M., Fontes, F.F.A.C.C.: A hybrid biased random key genetic algorithm approach for the unit commitment problem. Journal of Combinatorial Optimization **28**(1), 140–166 (2014)
25. Schmidt, C.W., Grossmann, I.E.: Optimization models for the scheduling of testing tasks in new product development. Industrial & Engineering Chemistry Research **35**(10), 3498–3510 (1996)
26. Schmidt, C.W., Grossmann, I.E., Blau, G.E.: Optimization of industrial scale scheduling problems in new product development. Computers & chemical engineering **22**, S1027–S1030 (1998)
27. Sobek, D.K., Ward, A.C., Liker, J.K.: Toyota's principles of set-based concurrent engineering. Sloan management review **40**(2), 67–84 (1999)
28. Sommer, S.C., Loch, C.H.: Selectionism and learning in projects with complexity and unforeseeable uncertainty. Management Science **50**(10), 1334–1347 (2004)
29. Spears, W.M., Dejong, K.A.: On the virtues of parameterized uniform crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 230–236 (1991)