UNIVERSITEIT GENT

FACULTEIT ECONOMIE
EN BEDRIJFSKUNDE

TWEEKERKENSTRAAT 2
B-9000 GENT
Tel. : 32 - (0)9 – 264.34.61
Fax. : 32 - (0)9 – 264.35.92

WORKING PAPER

A Genetic Algorithm for the Multi-Mode Resource-Constrained Project Scheduling Problem

Vincent Van Peteghem [1]
Mario Vanhoucke [2]

January 2008

2008/494

[1] Faculty of Economics and Business Administration, Ghent University, Gent, Belgium
Vincent.VanPeteghem@UGent.be
[2] Faculty of Economics and Business Administration, Ghent University, Gent, Belgium
Operations & Technology Management Centre, Vlerick Leuven Gent Management School, Gent, Belgium
Mario.Vanhoucke@UGent.be

D/2008/7012/03

A Genetic Algorithm for the Multi-Mode Resource-Constrained Project Scheduling Problem

Vincent Van Peteghem [a] and Mario Vanhoucke [a,b,*]

[a] *Ghent University, Tweekerkenstraat 24, 9000 Gent*

[b] *Vlerick Leuven Gent Management School, Reep 1, 9000 Gent*

Abstract

In this paper we present a genetic algorithm for the multi-mode resource-constrained project scheduling problem (MRCPSP), in which multiple execution modes are available for each of the activities of the project. In contrast to a conventional genetic algorithm, we apply a bi-population genetic algorithm, which makes use of two seperate populations. We extend the serial schedule generation scheme by introducing a mode optimization procedure. We present detailed comparative computational results, which reveals that our procedure is among the most competitive algorithms for the MRCPSP.

Key words: project scheduling, genetic algorithm, multi-mode RCPSP

1 Introduction

Resource-constrained project scheduling has been a research topic for many decades, resulting in a wide variety of optimization procedures. The main focus on project lead time minimization has led to the development of various exact and (meta-)heuristic procedures for scheduling projects with tight resource constraints under a wide variety of assumptions. The basic problem type in project scheduling is the well-known resource-constrained project scheduling problem (RCPSP). This problem type aims at minimizing the total duration or makespan of a project subject to precedence relations between the activities and the limited renewable resource availabilities, and is known to be NP-hard (Blazewicz et al., 1983).

* Corresponding author. E-mail: mario.vanhoucke@ugent.be

The multi-mode problem (MRCPSP) is a generalized version of the RCPSP, where each activity can be performed in one out of a set of modes, with a specific activity duration and resource requirements. Three different categories of resources can be distinghuised (Slowinski et al., 1994): renewable resources, which are limited per time-unit (e.g. manpower, machines), nonrenewable resources, which are limited for the entire project (e.g. budget) and doubly constrained resources, which are limited both per time-unit and for the total project duration (e.g. cash-flow per time-unit). The objective of the MRCPSP is to find a mode and a start time for each activity such that the makespan is minimized and the schedule is feasible with respect to the precedence and renewable and nonrenewable resource constraints. As this problem is a generalisation of the RCPSP, the MRCPSP is also NP-hard. Moreover, if there is more than one nonrenewable resource, the problem of finding a feasible solution for the MRCPSP is NP-complete (Kolisch and Drexl, 1997). The problem is denoted as $m, 1T|cpm, disc, mu|C_{max}$ using the classification scheme of Herroelen et al. (1999) and is denoted as $MPS|prec|C_{max}$ by Brucker et al. (1999).

Several exact and heuristic approaches to solve the MRCPSP have been proposed in recent years. In Table 1, an overview of the available exact and heuristic procedures in literature is given.

- **Exact procedures** – Talbot (1982) was the first to present an enumeration scheme for solving the problem. Patterson et al. (1989) also presented an enumeration scheme-based procedure. Speranza and Vercellis (1993) proposed a depth-first branch-and-bound algorithm, but Hartmann and Sprecher (1996) have shown that this algorithm may be unable to find the optimal solution for instances with two or more renewable resources. More recently, Sprecher et al. (1997), Hartmann and Drexl (1998) and Sprecher and Drexl (1998) presented a branch-and-bound algorithm. However, none of these can be used for solving real life projects, since they are unable to find an optimal solution in a reasonable computation time. Therefore, different single-pass heuristic and metaheuristic procedures are presented.
- **Heuristics** – Talbot (1982) and Sprecher and Drexl (1998) proposed to impose a time limit on their exact branch-and-bound procedure. Boctor (1993) tested 21 heuristic scheduling rules and suggested a combination of 5 heuristics which have a high probability of giving the best solution. Drexl and Grünewald (1993) proposed a biased random sampling approach, while Özdamar and Ulusoy (1994) proposed a local constraint based analysis approach. Boctor (1996b) presented a heuristic algorithm based on the Critical Path Method computation, Kolisch and Drexl (1997) suggested a local search method with a single-neighbourhood search and Knotts et al. (2000) evaluated different agent-based algorithms for solving the MRCPSP.
- **Meta-heuristics** – Genetic algorithms have been presented by Mori and Tseng (1997), Özdamar (1999), Hartmann (2001) and Alcaraz et al. (2003). Slowinski et al. (1994), Boctor (1996a), Jozefowska et al. (2001) and Bouleimen

and Lecocq (2003) used the simulated annealing approach, while Nonobe and Ibaraki (2001) proposed a tabu search procedure. Zhang et al. (2006) introduced the methodology of particle swarm optimization for solving the MRCPSP.

In this paper, we introduce a genetic algorithm approach for solving the MR-CPSP. In contrast to a regular GA, we use a bi-population genetic algorithm (BPGA). We have introduced a cross-over and mutation operator, compared different penalty functions and extended the serial generation scheme by using a mode optimization procedure. The remainder of the paper is organised as follows: The next section describes the general formulation of the problem. The third section describes, in detail, the mode optimization procedure and the different steps in our genetic algorithm. In the fourth section the results of the computational experiments are reported as well as the comparison with other heuristics. Finally, we summarize the conclusions of the present work.

2 Problem formulation

The MRCPSP can be stated as follows. We present a project network $G(N, A)$ in an activity-on-the-node format where A is the set of pairs of activities between which a finish-start precedence relationship with a minimal time lag of 0 exists. A set of N activities, numbered from a dummy start node 0 to a dummy end node $n+1$, is to be scheduled without pre-emption on a set R^ρ of renewable and R^ν of nonrenewable resource types. Each activity $i \in N$ can be performed in m_i different execution modes, $m_i \in M_i = \{1, ..., |M_i|\}$. The duration of activity i, when executed in mode m_i, is d_{im_i}. Each mode m_i requires $r^\rho_{im_ik}$ renewable resource units ($k \in R^\rho$). For each renewable resource $k \in R^\rho$, the availability a^ρ_k is constant throughout the project horizon. Activity i, executed in mode m_i, will also use $r^\nu_{im_il}$ nonrenewable resource units ($l \in R^\nu$) of the total available nonrenewable resource a^ν_l. A schedule S is defined by a vector of activity start times s_i and its corresponding finish times f_i and is said to be feasible if all precedence and renewable and nonrenewable resource constraints are satisfied. The objective of the MRCPSP is to minimize the makespan of the project.

Table 1. Overview literature

Author	Year	Method	R/NR	dataset	nr. of act	m	R	NR
Talbot	1982	enum	RNR	Own	10,20,30	1-3	3	0
Patterson et al.	1989	enum	RNR	-	-	-	-	-
Speranza and Vercellis	1993	B&B	RNR	Own	10-20	≥ 2	1-6	1
Boctor	1993	heur	R	Own	50,100	1-4	1,2,4	0
Drexl and Grünewald	1993	heur	RNR	Own	10/10	2-4/2-4	3/3	1/3
Özdamar and Ulusoy	1994	heur	RNR	Own	20-57	1-3	1-6	1-6
Slowinski et al.	1994	SA	RNR	Own	30	2	3	3
Boctor	1996a	SA	R	Boctor (1993)	50,100	1-4	1,2,4	0
Boctor	1996b	heur	R	Boctor (1993)	50,100	1-4	1,2,4	0
Sprecher et. al.	1997	B&B	RNR	PSPLIB	10	3	2	2
Mori and Tseng	1997	GA	R	Own	20,30,40,50,60,70	2-4	4	0
Kolisch and Drexl	1997	heur	RNR	PSPLIB	10,30	3	2	2
Hartmann and Drexl	1998	B&B	RNR	PSPLIB	10,12,14,16	3	2	2
Sprecher and Drexl	1998	B&B	RNR	PSPLIB/Own	10,12,14,16,18,20	3/1-5/3	2/1-5/2	2/1-3/0
Özdamar	1999	GA	RNR	PSPLIB/Own	10/90	3/2	2/2	2/2
Knotts et al.	2000	heur	R	Maroto and Tormos (1994)	50	2	3	0
Nonobe and Ibaraki	2001	TS	R	PSPLIB	30	3	2	2
Jozefowska et al.	2001	SA	RNR	PSPLIB	10,12,14,16,18,20,30	3	2	2
Hartmann	2001	GA	RNR	PSPLIB	10,12,14,16,18,20,30	3	2	2
Bouleimen and Lecocq	2003	SA	RNR	PSPLIB	10,12,14,16,18,20,30	3	2	2
Alcaraz et al.	2003	GA	RNR	PSPLIB/Boctor (1993)	10,12,14,16,18,20,30/50,100	3/1-4	2/1,2,4	2/0
Zhang et al.	2006	PS	RNR	PSPLIB	10,12,14,16,18,20	3	2	2

R/NR = applicable on datasets with only renewable resources (R) or with both renewable and nonrenewable (RNR) resources

m = number of modes

R = number of renewable resources

NR = number of nonrenewable resources

The MRCPSP can be conceptually formulated as follows:

$$\text{Min. } s_{n+1} \tag{1}$$

s.t.

$$
\begin{aligned}
& s_i + d_{im_i} \leq s_j && \forall (i,j) \in A && (2)\\
& \sum_{i \in S(t)} r^{\rho}_{im_ik} \leq R^{\rho}_k && \forall k \in R^{\rho}, \forall m_i \in M_i && (3)\\
& \sum_{i \in S(t)} r^{\nu}_{im_il} \leq R^{\nu}_l && \forall l \in R^{\nu}, \forall m_i \in M_i && (4)\\
& m_i \in M_i && \forall i \in N && (5)\\
& s_0 = 0 && && (6)\\
& s_i \in \text{int}^+ && \forall i \in N && (7)
\end{aligned}
$$

where $S(t)$ denotes the set of activities in progress in period $]t-1, t]$. The objective is to minimize the total makespan of the project (equation 1). Equation (2) takes the finish-start precedence relations with a minimal time lag of zero into account. The renewable resource constraints are satisfied thanks to equation (3), where the nonrenewable ones are fullfilled in (4). Each activity i has to be performed in exactly one mode m_i (eq. 5). Equation (6) forces the project to start at time instance zero and equation (7) ensures that the actvity start times assume nonnegative integer values. A schedule which fullfills all the equations, is called *optimal*. When equation 1 is not met, the schedule is called *feasible*.

Consider an example project that will be used throughout the remainder of this paper with 8 non-dummy activities, each with 2 modes. For each mode, 1 renewable resource and 1 nonrenewable resource is indicated. The availability for the renewable resource is 5 and for the nonrenewable 20. The activity-on-the-node network is shown in figure 2. In Table 2 the different modes of each activity with their corresponding duration and resource requirements are shown. Schedule (a) has a schedule with a makespan of 7 days. However, this schedule is infeasible with respect to the nonrenewable resource. The schedule uses 22 nonrenewable resource units, which exceeds the available 20. Schedule (b) shows a schedule with a makespan of 11 days. This schedule is feasible because it uses exactly 20 nonrenewable resource units. In section 3.2 we present the optimal solution for this problem.

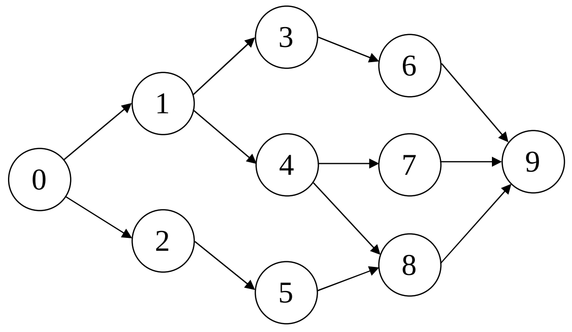
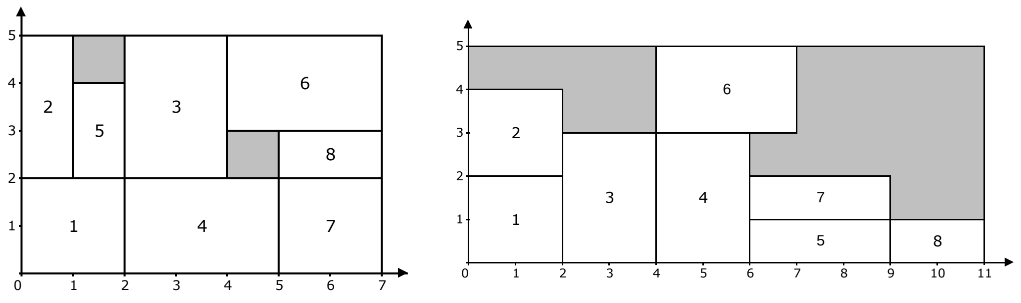


Fig. 1. Network

act i	mode j	dur_{ij}	r^{ρ}_{ij}	r^{ν}_{ij}
0	0	0	0	0
1	1	2	2	3
	2	3	2	3
2	1	1	3	4
	2	2	2	2
3	1	2	3	4
	2	3	2	3
4	1	2	3	4
	2	3	2	3
5	1	1	2	3
	2	3	1	2
6	1	1	3	5
	2	3	2	2
7	1	2	2	3
	2	3	1	2
8	1	1	2	3
	2	2	1	1
9	0	0	0	0

Table 2. test



(a) Infeasible schedule

(b) Feasible schedule

Fig. 2. Schedules of the example

3 A genetic algorithm for the MRCPSP

Introduced by Holland (1975), genetic algorithms (GAs) are used to solve complex optimization problems by using procedures that simulate biological evolution. Several selections mechanisms, such as natural selection, crossover and mutation, are applied to recombine existing solutions to obtain new ones and to find an optimal solution. In this paper we use, in contrast to a regular GA, the BPGA that makes use of two different populations: a population POP_R that only contains right-justified schedules and a population POP_L that only contains left-justified schedules. Both populations have the same population size POP. The idea of alternatively justifying the schedule to the right and to the left is proposed by Valls et al. (2005). In the remainder of this section we first examine the representation of the individuals and the schedule generation scheme, after which we reveal the further algorithmic details of our bi-populational genetic algorithm.

3.1 Representation

The representation of an individual is crucial for the performance of the genetic algorithm. Kolisch and Hartmann (1999) distinguished five different schedule representations in the RCPSP literature, from which the activity-list (AL) representation and the random-key (RK) representation are the most widespread. In both representations, a priority structure between the activities is embeded. In the AL representation, the position of an activity in the AL determines the relative priority of that activity versus the other activities, while in the RK representation the position of an activity is based on the priority value attributed to an activity. Hartmann and Kolisch (2000) concluded from experimental tests that procedures based on AL representations outperform the other procedures. However, Debels et al. (2006) illustrated that the RK also leads to promising results thanks to the use of the topological ordering (TO) notation (Valls et al., 2003).

As the multi-mode is an extension of the single-mode RCPSP, most authors, such as Slowinski et al. (1994), Bouleimen and Lecocq (2003), Hartmann (2001), Jozefowska et al. (2001) and Alcaraz et al. (2003), have used an extension of the standard AL representation. In this paper however, we use the RK representation. An individual is therefore represented by a vector λ of priority values. The advantage of this representation is that no repair requirements are needed for preserving precedence and resource feasibility in the offspring (Özdamar, 1999). Next to the vector λ, each population element has also a randomly generated mode list μ. The mode list μ assigns a mode m_i to each activity i ($i \in \{1, ..., N\}$) and determines if the schedule will be feasible with respect to the nonrenewable resources. The number of requested nonrenewable resource units that exceed the capacity a_l^ν, $l \in R^\nu$, is defined as the excess of resource request $ERR(\mu)$. An $ERR(\mu)$=0 means that the solution is feasible. Otherwise, the solution is infeasible. The formula of the $ERR(\mu)$ can be stated as follows:

$$ERR(\mu) = \sum_{j=1}^{l}(\min(0, a_j^\nu - \sum_{i=1}^{N}(r_{im_ij}^\nu))) \qquad l \in R^\nu \qquad (8)$$

Schedule 2(a) is scheduled according to a topological ordered random key (1,2,5,3,4,6,7,8) and the mode list is (1,1,1,2,1,2,1,2). The schedule has an $ERR(\mu)$ of 3 (23 requested versus 20 available nonrenewable resource units) and is therefore infeasible. Schedule 2(b) on the other hand is feasible, because the required nonrenewable resource units do not exceed the availability, so the schedule has an $ERR(\mu)$ of 0. Schedule (b) is scheduled according to the topological ordered random key (1,4,2,3,6,5,7,8) and the mode list (1,2,1,1,2,2,2,2).

A schedule generation scheme (SGS) translates the lists λ and μ into a schedule S. Two different types of SGSs exist in literature: the serial SGS (Kelley, 1963) and the parallel SGS (Bedworth and Bailey, 1982). As it is sometimes impossible to reach an optimal solution with the parallel SGS (Kolisch, 1996), we use in this paper the serial SGS, where all activities are scheduled one at a time and as soon as possible within the precedence and resource constaints. However, we have extended the serial SGS with the following two elements.

First, our procedure makes use of two different procedures to build right-justified and left-justified schedules, the forward and backward procedure respectively. Both procedures are based on the well-known local search technique of Li and Willis (1992). The forward-procedure is applied on the left-justified population POP$_L$ and is used to build a right-justified schedule. The finish times are used as the priority values for our random key. The sequence of the activities is made by sorting the eligible activities in decreasing order of their finish times. Each activity is scheduled as late as possible. After the genetic operators are applied on the right-justified schedules, the backward-procedure is applied on the population of right-justified schedules POP$_R$ (Debels and Vanhoucke, 2007).

Second, our serial generation scheme is extended with a procedure to optimize the mode selection. For every activity $i \in \{1, ..., N\}$ that will be scheduled, both in the forward and in the backward procedure, there is a probability of P_{modopt} that the mode optimization procedure will be executed. We use m_i to refer to the current mode of activity i and μ to refer to the current mode vector for all project activities. The mode optimization procedure evaluates for each activity i whether a new mode selection m_i' leads to an improvement in the $ERR(\mu')$. Consequently, this procedure evaluates all possible mode assignments $k = 1, ..., |M_i|$ of an activity i and calculates for each new mode assignment the corresponding mode vector μ' and the $ERR(\mu')$. If the $ERR(\mu')$ is equal or smaller than the current $ERR(\mu)$, the procedure checks if an improvement can be made in the finish time of that activity. The mode m_i' with the lowest finish time f_i', which does not increase the $ERR(\mu)$, is chosen. If no improvement can be made, the mode selection is retained. The pseudocode for the mode optimization procedure for an activity i of a project with an excess resource request of $ERR(\mu)$ is shown in Table 3.

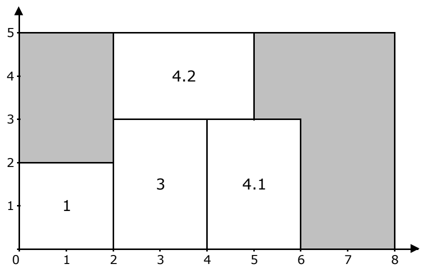
One could think that the mode optimization procedure will always chose the mode with the lowest duration. However, a mode with a shorter duration often requires more nonrenewable resource units and gives cause to an increase of the $ERR(\mu)$ and the infeasibility of the schedule. In addition, a mode with a longer duration requires less resource units, both renewable and nonrenewable.

Table 3
Pseudocode of the mode optimization procedure for activity i

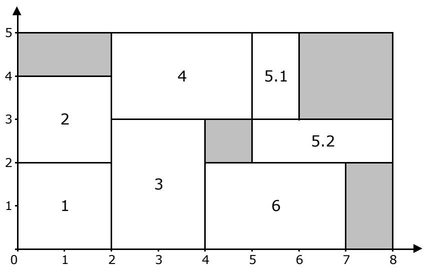
```
FOR k=1,...,$|M_i|$
{
        Set new mode $m'_i = k$ and create a new mode vector $\mu'$
        Compute $ERR(\mu')$
        IF $(ERR(\mu')) \leq ERR(\mu))$
              Compute $f'_i$
              IF$(f'_i < f_i)$
                   $f_i = f'_i$
                   $m_i = m'_i$
                   $ERR(\mu) = ERR(\mu')$
}
```

The procedure will therefore be able to schedule the activity more easily in parallel with other activities, because - due to the lower resource requirements - less resource conflicts will occur. Retake schedule (b) of figure 2. Suppose that activities 4 and 5 are subject to mode optimization. According to our random key λ, activities 1 and 3 are already scheduled, when the mode optimization procedure is applied on activity 4. With the current mode for activity 4 (mode 1), the schedule has an $ERR(\mu)$ of 0. The finish time of mode 1 is 6. When changing the mode of activity 4 to 2, the $ERR(\mu)$ remains equal, because the required nonrenewable resource units (19) are smaller than the available ones (20). Although the duration of mode 2 is larger than mode 1, the required renewable resource units also decreases and the activity can be scheduled in parallel with activity 3. The finish time for mode 2 is now 5 (see Fig 3(a)). Because mode 2 has a lower finish time and the $ERR(\mu)$ does not deteriorate, the modelist is adapted. After scheduling activities 2 and 6, the mode optimization is applied on activity 5. For activity 5, mode 2 is currently selected and when inserting the activity in the current partial schedule, this activity has a finish time of 8 (see Fig 3(b)). However, when choosing mode 1, the finish time can be reduced to 6, because no renewable resource conflict occurs. The nonrenewable resource requirements increase to 20 and hence, the $ERR(\mu)$ remains 0. As the finish time is smaller for mode 1, the mode list for activity 5 is adapted to mode 1. After scheduling activities 7 and 8, the optimal solution for this specific problem is obtained, as shown in figure 3(c).

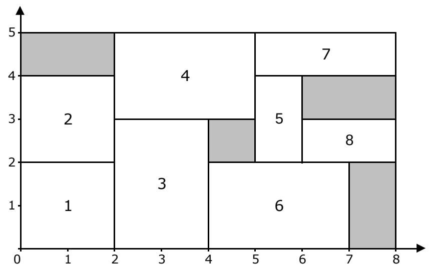
This mode optimization procedure shows similarities with the single-pass improvement of Hartmann (2001). This procedure however is a local search method which tries to improve the schedule after construction by the SGS. However, tests has shown that our procedure outperforms the local search of Hartmann.



(a) Mode-optimization of activity 4



(b) Mode-optimization of activity 5



(c) Optimal solution

Fig. 3. Mode optimization procedure

3.3 Details of the genetic algrithm

3.3.1 Preprocessing

Before we start the genetic algorithm, the reduction procedure of Sprecher et al. (1997) is applied. This procedure excludes those modes which are inefficient or non-executable and those resources which are redundant. As defined by Sprecher et al. (1997), a mode is called *inefficient* if its duration is not shorter and its resource needs are not less than those of another mode of the same activity. A mode is called *non-executable* if the renewable or nonrenewable resource constraints are violated. A nonrenewable resource is called *redundant* if the sum of the maximal requests for that nonrenewable resource does not exceed its availability. Excluding these modes or nonrenewable resources does not affect the set of feasible or optimal schedules.

Consider the project instance given in figure 1. Mode 2 of activity 1 can be called *inefficient* because both its duration and its resource requirements are equal or larger than those of mode 1. Also mode 1 of activity 6 can be excluded, because this mode is *non-executable* with respect to the nonrenewable resource. If it was executed, the project would require at least 21 nonrenewable resource units, while only 20 are available. The mode can therefore be deleted.

3.3.2 Initial population

The genetic algorithm is started by building an initial population POP_L of left-justified schedules. First, the random key λ is generated randomly. Second, for each activity i, $i \in \{1, ..., N\}$, an execution mode m_i is randomly selected. Kolisch and Drexl (1997) mentioned that finding a feasible solution for the MRCPSP is a NP-complete problem if at least two nonrenewable resources are given. Therefore, infeasible solutions are accepted in our initial population. However, to minimize the number of infeasible solutions, the local search procedure of Hartmann (2001) is applied to transform infeasible solutions into feasible ones. The procedure chooses an activity randomly and for that activity, a different mode is chosen. If the $ERR(\mu)$ remains the same or decreases, the mode for that activity is changed. This step is repeated until the mode assignment is feasible ($ERR(\mu)$=0) or until J consecutive unsuccesfull trials to improve the mode assigment have been made. In this paper, J equals to 4 times the number of activities in the project. Based on the random key λ and the mode list μ, a schedule is constructed with the extended serial generation scheme.

3.3.3 Evaluation

Once the initial population is generated, each of the schedules must be evaluated. Therefore, a fitness value is calculated for each individual. In the single-mode RCPSP, the fitness value normally equals the makespan of the project. It is a good measure for sequencing the different individuals according to their contribution to the objective of the problem. However, using the makespan as fitness value for the multi-mode RCPSP is inappropriate. As infeasible schedules (with an $ERR(\mu) > 0$) can be included in the population, infeasible schedules must be penalised, otherwise they will displace the feasible schedules in the population. Jozefowska et al. (2001) examined the differences between a fitness function with penalty function and one without. Tests have indicated that the fitness function with penalty function clearly performs better.

A good fitness function gives appropriate feedback to the genetic algorithm. Hartmann (2001) defined the fitness function for an individual as follows:

$$f(n) = \begin{cases} C_{max}(\lambda, \mu) & \text{if } ERR(\mu) = 0 \\ T + ERR(\mu) & \text{otherwise} \end{cases}$$

with T equal to the upper bound of the project's makespan that is given by the sum of the maximal durations of the activities. The lower the fitness value of a certain schedule is, the better the quality of the related schedule is. However, Alcaraz et al. (2003) points out that two different individuals with the same excess of resource request but with a different makespan has the same fitness value. They also mention that the upper bound T is a poor bound and indicate that the probability of the infeasible solutions to survive will be near zero. Therefore, they defined a new fitness function that can be presented as follows:

$$f(n) = \begin{cases} C_{max}(\lambda, \mu) & \text{if } ERR(\mu) = 0 \\ C_{max}(\lambda, \mu) + max\_feas\_C_{max}(\lambda, \mu) & \\ -CP^{min} + ERR(\mu) & \text{otherwise} \end{cases}$$

where $max\_feas\_C_{max}(\lambda, \mu)$ gives the maximal makespan of the feasible schedules related to individuals of the current generation and CP^{min} is the critical path using the minimal duration of each activity. According to Alcaraz et al. (2003), this fitness computation revealed better results than the one of Hartmann (2001). In this paper we will test both methods.

3.3.4 Parent selection

For each population element i of POP$_L$ (POP$_R$) we create a set of 2 right-justified (left-justified) children that are candidates to enter POP$_R$ (POP$_L$). To create a child out of i, another parent j from POP$_L$ (POP$_R$) is selected by using the 2-tournament selection procedure. In this selection procedure two population-elements are chosen randomly and the element with the best fitness function value is selected. Afterwards, we determine randomly whether i or j represents the father S_f. The other parent represents the mother S_m.

3.3.5 Crossover

The crossover operation is applied on each pair of parents from POP$_L$ (POP$_R$), producing two children which inherit parts of their characteristics. When a crossover is used, both the activity list and mode list are adapted. Receiving characteristics from one of the parents, implies the reception of the information from both the activity list and the mode list. We have tested different possible crossover operators. However, the so-called one-point crossover revealed the best results. In the one-point crossover, an integer number r is randomly selected from the interval [1,n]. All data left from that point, both from the actvity list and from the mode list, is copied from the father's chromosome. Beyond the point, the data of the mother is copied. Other crossover operators, such as the two-point crossover, as used in Alcaraz et al. (2003), the uniform cross-over, as tested in Özdamar (1999) or the peak cross-over (Valls (2002), Debels et al. (2006)), did not reveal better results than the one-point crossover.

3.3.6 Mutation

Mutation is applied to reintroduce lost genetic material into the population and creates variation in the different individuals. It introduces partial activity sequences and unselected mode choices into the population which could not have been produced by the crossover operator. In our genetic algorithm, two mutation operators are executed. The first one has a probability of P_{mut_act} and modifies the selected random key λ. The mode assignment is not affected. The second mutation operator modifies the mode list with a probability of P_{mut_mod}. The mutation in both the activity list and the mode list is done randomly. However, computational tests have revealed that the probabilities P_{mut_act} and P_{mut_mod} to obtain the best results are always smaller than 4%.

3.3.7 Update

In our algorithm, the parent population is replaced by the offspring population, which means that the population size remains the same. For the replacement, we follow the survival-of-the-fittest strategy. For every individual i out of the population, a partner is chosen (see 3.3.4) and 2 children are generated. The child with the lowest fitness value is selected and replaces individual i in the population, even if there is a deterioration. It is obvious that i is not replaced when it is the best-found schedule so far.

4 Computational results

In this section we evaluate the performance of our algorithm. The genetic algorithm has been coded and compiled in Visual C++ 6.0 and used in the computational tests on a Dell Dimension DM051 with a Pentium D with a 2.80 GHz processor. As a benchmark, we use the well-known PSPLIB dataset (Kolisch and Sprecher, 1996) to test our parameters and compare with other existing procedures from the literature. The dataset is generated with the project generator ProGen (Kolisch et al., 1995) and is available in the project scheduling problem library PSPLIB from the ftp server of the University of Kiel (http://129.187.106.231/psplib/). The dataset for the multi-mode RCPSP contains project instances with 10, 12, 14, 16, 18, 20 and 30 activities, each with 2 renewable and 2 nonrenewable resources. The duration of the activities varies from 1 to 10. For the instances with 30 activities only a set of best known heuristic solutions is available, for the other instances the optimal solutions are available. For each problem size, 640 instances were generated. Due to infeasibility of some instances, not all these instances could be solved. Therefore, these infeasible instances are excluded from further research. Table 4 shows the number of instances for which at least one feasible solution has been found and also indicates the average number of modes per activity after applying the reduction procedure of Sprecher et al. (1997).

Table 4
PSPLIB instances

	Number of activities						
	10	12	14	16	18	20	30
Number of instances	536	547	551	550	552	554	552
Av. number of modes	2,927	2,952	2,964	2,984	2,986	2,984	2,999

4.1 Configuring the algorithm

In this section, we configure the parameters of our genetic algorithm. We test the mode optimization procedure, the introduction of two populations and the different fitness computations. The tests are performed on the J20-dataset, which is the largest dataset for which optimal solutions are available. The results of these tests are shown in Table 5.

Mode optimization procedure The value of the probability of applying the mode optimization procedure P_{modopt} is varying between 0% and 100% in steps of 10%. If we compare the results of the algorithm with and without the mode optimization procedure, we see the enormous influence of the procedure in the results. The deviation of the makespan has decreased from 1,36% (the best found solution when no mode optimization procedure is carried out) to 0,45%.

Number of populations Although a bi-populational genetic algorithm is used, we also check if the one-populational genetic algorithm reveals other results. As we see in Table 1, the best solution of the bi-populational GA clearly outperforms the best solution of the one-populational GA. In addition, statistical tests reveal a very significant ($p < 0,001$) contribution of the introduction of two populations in the genetic algorithm. These results are consistent with the results for the RCPSP (Debels and Vanhoucke, 2005).

Fitness computation The different fitness functions are also compared. As we have mentioned in section 3.3.3, two fitness functions were proposed in literature, i.e. the one proposed by Hartmann (2001) and the one proposed by Alcaraz et al. (2003). Looking at the best results for both fitness computations, we see that the fitness value of Alcaraz et al. (2003) reveals better results than the one of Hartmann (2001). Moreover, a significant difference ($p < 0,05$) is mentioned between those two solutions.

The best results are obtained by using two populations, the fitness function according to Alcaraz et al. (2003) and setting the probability of applying the mode optimization procedure, P_{modopt}, equal to 30%.

Table 5. Configuration of the algorithm

		P_{mod_opt}										
# populations	Fitness comp.	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1 population	Alcaraz	6,96%	1,71%	1,41%	1,39%	1,40%	1,44%	**1,35%**	1,47%	1,70%	1,55%	1,80%
	Hartmann	6,57%	1,75%	1,40%	**1,29%**	1,46%	1,50%	1,40%	1,48%	1,61%	1,63%	1,90%
2 populations	Alcaraz	1,36%	0,69%	0,53%	**0,45%**	0,55%	0,59%	0,58%	0,56%	0,60%	0,56%	0,66%
	Hartmann	1,38%	0,60%	0,57%	0,60%	0,56%	0,56%	**0,53%**	0,63%	0,57%	0,60%	0,69%

4.2 Comparison with other heuristics

In this section, we compare our algorithm with some of the best performing heuristics and metaheuristics in literature. The procedures are chosen on the basis of their good results and the used stopping criterion. Only those procedures which reported their results based on a number of schedules computed are taken into account. The results from the algorithms were obtained from Alcaraz et al. (2003). Hartmann (2001) explains how the results of Kolisch and Drexl (1997) and Özdamar (1999) were computed. In Table 6 we present the average deviation from the optimal solution (Av. Dev. (%)) and the percentage of optimal solutions found (Optimal (%)) for the J10-dataset of PSPLIB. For this test, 6000 schedules are computed. We see that our algorithm outperforms the other heuristics.

Table 6
Comparison with other heuristics - J10 - 6000 schedules

Author	Av. Dev. (%)	Optimal (%)
Van Peteghem and Vanhoucke	**0,03**	**99,8%**
Hartmann (2001)	0,10	98,1%
Alcaraz et al. (2003)	0,19	96,5%
Kolisch and Drexl (1997)	0,50	91,8%
Özdamar (1999)	0,86	88,1%

A comparison of the different algorithms on the J10 to J20-datasets of PSPLIB and based on 5000 schedules is made for the authors mentioned in Table 7. The results for our algorithm outperforms the other heuristics. We also present the percentage of optimal solutions found, the optimal population size and the computation time for each of the datasets for 5000 schedules.

Table 7
Comparison with other heuristics - average % deviation from optimal - 5000 schedules

	Instances set					
	J10	J12	J14	J16	J18	J20
Jozefowska et al. (2001)	1,16	1,73	2,60	4,07	5,52	6,74
Alcaraz et al. (2003)	0,24	0,73	1,00	1,12	1,43	1,91
This work	**0,03**	**0,08**	**0,18**	**0,29**	**0,47**	**0,45**
Optimal (%)	99,81%	98,17%	95,28%	92,73%	88,77%	87,91%
CPU-time (sec)	0,18	0,19	0,21	0,25	0,27	0,28
Population size	250	225	200	175	150	125

Table 8
% increase of project duration above critical path length and CPU-time for J30

	J30		
	% Incr. CP	Av. % Dev.	CPU (sec)
1000 schedules	14,86%	1,93%	0,06
5000 schedules	13,77%	1,10%	0,31
50000 schedules	13,25%	0,71%	2,96

We have also tested our algorithm on the J30-dataset for which no optimal solutions are available. This is the largest dataset in the PSPLIB and contains 640 data instances with 30 activities each, 3 modes per activity and 2 renewable and 2 nonrenewable resources. No optimal solutions for this dataset are available, only the best known results till now, which can be found on the PSPLIB-server. Comparison of the results on this dataset however is much more difficult because most researchers, such as Jozefowska et al. (2001), Nonobe and Ibaraki (2001) and Bouleimen and Lecocq (2003), only report the number of improvements made with respect to the set of - till then - best-known solutions. Because this set of solutions changes, comparison is impossible. Therefore, we computed the percentage increase of the project duration above the minimal critical path length for 1000, 5000 and 50000 schedules computed. As can be seen in Table 8, we also indicate the percentage improvement we have made above the best known solutions till now [1] . This information can be used by other researcher to compare with their results.

5 Conclusions

In this paper we have designed a new genetic algorithm for the multi-mode resource-constrained project scheduling problem. For this algorithm we have used two populations, one with left-justified schedules and one with right-justified schedules. We have also introduced an extended serial schedule generation scheme, which optimizes the mode selection by choosing that feasible mode of a certain activity that minimizes the finish time of that activity. The computational results have shown that the introduction of these two elements increases the performance of the genetic algorithm enormously. Our algorithm outperforms all the existing algorithms in literature, within reasonable computation times. Areas of further research are e.g. the introduction of pre-emption and the adaption of the algorithm to other project scheduling problems, such as the discrete time/resource trade-off problem.

[1] Results on the PSPLIB-server on 5 October 2007

References

Alcaraz, J., Maroto, C., Ruiz, R., 2003. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society 54, 614–626.

Bedworth, D., Bailey, J., 1982. Integrated Production Control Systems - Management, Analysis, Design. Wiley, New York.

Blazewicz, J., Lenstra, J., Rinnooy Kan, A., 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 11–24.

Boctor, F., 1993. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. International Journal of Production Research 31 (11), 2547–2558.

Boctor, F., 1996a. An adaption of the simulated annealing for solving resource-constrained project scheduling problems. International Journal of Production Research 34, 2335–2351.

Boctor, F., 1996b. A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. European Journal of Operational Research 90, 349–361.

Bouleimen, K., Lecocq, H., 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 149, 268–281.

Brucker, P., Drexl, A., Möhring, R., Neuman, K., Pesch, E., 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, 3–41.

Debels, D., De Reyck, B., Leus, R., Vanhoucke, M., 2006. A hybrid scatter-search/electromagnetism meta-heuristic for the resource-constrained project scheduling problem. European Journal of Operational Research 169 (3), 638–653.

Debels, D., Vanhoucke, M., 2005. A bi-population based genetic algorithm for the resource-constrained project scheduling problem. Lecture Notes on Computer Science 3483, 378–387.

Debels, D., Vanhoucke, M., 2007. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. Operations Research 55 (3), 457–469.

Drexl, A., Grünewald, J., 1993. Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions 25, 74–81.

Hartmann, S., 2001. Project scheduling with multiple modes: a genetic algorithm. Annals of Operations Research 102, 111–135.

Hartmann, S., Drexl, A., 1998. Project scheduling with multiple modes: a comparison of exact algorithms. Networks 32, 283–297.

Hartmann, S., Kolisch, R., 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 127, 394–407.

Hartmann, S., Sprecher, A., 1996. A note on "hierarchical models for multi-project planning and scheduling". European Journal of Operational Research 94, 377–383.

Herroelen, W., De Reyck, B., Demeulemeester, E., 1999. Handbook of Recent Advances in Project Scheduling. Kluwer Academic Publishers, Ch. A Classification Scheme for Project Scheduling, pp. 1–26.

Holland, H., 1975. Adaption in Natural and Artifical Systems. University of Michigan Press.

Jozefowska, J., Mika, M., Rozycki, R., Waligora, G., Weglarz, J., 2001. Simulated annealing for multi-mode resource-constrained project scheduling. Annals of Operations Research 102, 137–155.

Kelley, J.E., J., 1963. Industrial Scheduling. Prentice-Hall, New Jersey, Ch. The critical-path method: Resources planning and scheduling, pp. 347–365.

Knotts, G., Dror, M., Hartman, B., 2000. Agent-based project scheduling. IIE Transactions 32 (5), 387–401.

Kolisch, R., 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research 43, 23–40.

Kolisch, R., Drexl, A., 1997. Local search for nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions 29, 987–999.

Kolisch, R., Hartmann, S., 1999. Project Scheduling - Recent Models, Algorithms and Applications. Kluwer Academic Publishers, Boston, MA, Ch. Heuristic algorithms for solving the resource-constrained project scheduling problem, pp. 147–178.

Kolisch, R., Sprecher, A., 1996. PSPLIB - a project scheduling problem library. European Journal of Operational Research 96, 205–216.

Kolisch, R., Sprecher, A., Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41, 1693–1703.

Li, K., Willis, R., 1992. An iterative scheduling technique for resource-constrained project scheduling. European Journal of Operational Research 56, 370–379.

Maroto, C., Tormos, P., 1994. Project management: an evaluation of software quality. International Transactions in Operational Research 1, 209–221.

Mori, M., Tseng, C., 1997. A genetic algorithm for multi-mode resource constrained project scheduling problem. European Journal of Operational Research 100, 134–141.

Nonobe, K., Ibaraki, T., 2001. Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP). Tech. rep., Kyoto University.

Özdamar, L., 1999. A genetic algorithm approach to a general category project scheduling problem. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 29, 44–59.

Özdamar, L., Ulusoy, G., 1994. A local constraint based analysis approach to project scheduling under general resource constraints. European Journal of Operational Research 79, 287–298.

Patterson, J., Slowinski, R., Talbot, F., Weglarz, J., 1989. Advances in Project Scheduling. Elsevier, Amsterdam, Ch. An algorithm for a general class of precedence and resource constrained scheduling problem, pp. 3–28.

Slowinski, R., Soniewicki, B., Weglarz, J., 1994. DSS for multiobjective project scheduling. European Journal of Operational Research 79, 220–229.

Speranza, M., Vercellis, C., 1993. Hierarchical models for multi-project planning and scheduling. European Journal of Operational Research 64, 312–325.

Sprecher, A., Drexl, A., 1998. Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107, 431–450.

Sprecher, A., Hartmann, S., Drexl, A., 1997. An exact algorithm for the project scheduling with multiple modes. OR Spektrum 19, 195–203.

Talbot, F., 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. Management Science 28 (10), 1197–1210.

Valls, V., Ballestin, F., Quintanilla, S., 2005. Justification and RCPSP: A technique that pays. European Journal of Operational Research 165 (2), 375–386.

Valls, V., Quintanilla, S., Ballestin, F., 2003. Resource-constrained project scheduling: A critical activity reordering heuristic. European Journal of Operational Research 149, 282–301.

Zhang, H., Tam, C., Li, H., 2006. Multimode project scheduling based on particle swarm optimization. Computer-Aided Civil and Infrastructure Engineering 21, 93–103.