European Journal of Operational Research 201 (2010) 409–418

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem

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

[a] Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium
[b] Vlerick Leuven Gent Management School, Reep 1, 9000 Gent, Belgium

ARTICLE INFO

Article history:
Received 3 November 2007
Accepted 17 March 2009
Available online 31 March 2009

Keywords:
Project scheduling
Genetic algorithm
Multi-mode RCPSP
Preemption

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. We also introduce the preemptive extension of the problem which allows activity splitting (P-MRCPSP). To solve the problem, we apply a bi-population genetic algorithm, which makes use of two separate populations and extend the serial schedule generation scheme by introducing a mode improvement procedure. We evaluate the impact of preemption on the quality of the schedule and present detailed comparative computational results for the MRCPSP, which reveal that our procedure is amongst the most competitive algorithms.

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).

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 distinguished (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).

Since doubly constrained resources can be considered as a combination of renewable and nonrenewable resources, we do not consider them explicitly. 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** – The first solution method for the multi-mode problem can be found in Slowinski (1980), who presented a one-stage and two-stage lineair programming approach. Talbot (1982) and Patterson et al. (1989) 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 branch-and-bound algorithms, while Zhu et al. (2006) proposed a branch-and-cut algorithm. However, none of these procedures can be used for solving large-sized realistic projects, since they are unable to find an optimal solution in a

* Corresponding author. Address: Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium. Tel.: +32 92643569.
E-mail addresses: mario.vanhoucke@UGent.be, mario.vanhoucke@vlerick.be (M. Vanhoucke).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.03.034

Table 1
Overview of the literature.

Author	Year	Method	R/NR	dataset	No. of act	m	R	NR
Slowinski	1980	LP	RNR	Own	–	–	–	–
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
Zhu et al.	2006	B&C	RNR	PSPLIB	20,30	3	2	2
Lova et al.	2006	Heur	R	Boctor	50,100	1–4	1,2,4	0
Jarboui et al.	2008	PS	RNR	PSPLIB	10,12,14,16,18,20,30	3	2	2
Ranjbar et al.	2008	SS	RNR	PSPLIB	10,12,14,16,18,20	3	2	2
Lova et al.	2009	GA	RNR	PSPLIB/Boctor (1993)	10,12,14,16,18,20,30/50,100	3/1–4	2/1,2,4	2/0

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; and NR = number of nonrenewable resources.

reasonable computation time. Therefore, different single-pass heuristic and meta-heuristic 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, Knotts et al. (2000) evaluated different agent-based algorithms and Lova et al. (2006) designed several multi-pass heuristics based on priority rules for solving the MRCPSP.
- **Meta-heuristics** – Genetic algorithms have been presented by Mori and Tseng (1997), Özdamar (1999), Hartmann (2001), Alcaraz et al. (2003) and Lova et al. (2009). 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) and Jarboui et al. (2008) presented the methodology of particle swarm optimization for solving the MRCPSP. Ranjbar et al. (2008) used a hybrid scatter-search to tackle the MRCPSP, using the path relinking methodology as a solution combination method.

The basic RCPSP and MRCPSP assume that each activity, once started, will be executed until its completion. The extension to the preemptive multi-mode version (P-MRCPSP) allows activities to be preempted at any time instance and restarted later on at no additional cost. For the single-mode case, Kaplan (1988) and Demeulemeester and Herroelen (1996) presented exact algorithms, Ballestin et al. (2008) and Vanhoucke and Debels (2008) proposed heuristics, while Damay et al. (2007) presented a lineair programming based algorithm. For the multi-mode case, Buddhakulsomsiri and Kim (2006) proved that preemption is very effective to improve the optimal project makespan in the presence of resource vacations and temporary resource unavailability and that the makespan improvement is dependent on the parameters that impact resource utilization. To the best of our knowledge, no further research has been performed on the P-MRCPSP.

In this paper, we introduce a genetic algorithm approach for solving the MRCPSP and the P-MRCPSP, based on different components found in literature. In contrast to a regular GA, our procedure is based on the bi-population approach, as proposed by Debels and Vanhoucke (2005) for the RCPSP. We have extended the bi-population genetic algorithm (BPGA) to the MRCPSP and the P-MRCPSP. We have also introduced a crossover and mutation operator, compared the penalty functions of Hartmann (2001) and Alcaraz et al. (2003) and extended the serial generation scheme by using a mode improvement procedure, adapted from the single-pass improvement procedure of Hartmann (2001). The remainder of the paper is organised as follows: Section 2 describes the general formulation of the problem, while Section 3 describes the mode improvement procedure and the different steps in our genetic algorithm in detail. In Section 4 the results of the computational experiments are reported as well as the comparison with other heuristics and an overview of the influence of the introduction of preemption. Finally, we summarize the conclusions of the present work.

2. Problem formulation

The MRCPSP can be stated as follows. The project is represented as an activity-on-the-node network $G(N, A)$, where N is the set of activities and 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 activities, numbered from 1 to $|N|$ with a dummy start node 0 and a dummy end node $|N|+1$, is to be scheduled on a set R^{ρ} of renewable and R^{ν} of nonrenewable resource types. Each activity $i \in N$ is performed in a mode m_i, which is chosen out of a set of $|M_i|$ different execution modes $M_i = \{1,\ldots,|M_i|\}$. 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 a vector denoting its corresponding execution modes m_i. A schedule 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.

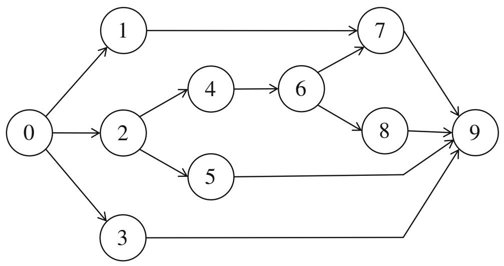

Fig. 1. Network.

Table 2
Project information.

Act i	Mode m_i	d_{im_i}	r^{ρ}_{ij}	r^{ν}_{ij}
0	1	0	0	0
1	1	4	3	3
	2	5	2	4
2	1	1	3	4
	2	2	2	3
3	1	1	2	3
	2	2	1	1
4	1	2	5	4
	2	3	4	3
5	1	2	4	6
	2	5	3	2
6	1	1	1	4
	2	3	1	3
7	1	1	3	3
	2	3	2	2
8	1	2	3	4
	2	2	3	3
9	1	0	0	0

The MRCPSP can be conceptually formulated as follows:

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

$$\text{s.t. } s_i + d_{im_i} \leqslant s_j \quad \forall (i,j) \in A, \tag{2}$$

$$\sum_{i \in S(t)} r^{\rho}_{im_ik} \leqslant a^{\rho}_k \quad \forall k \in R^{\rho},\ \forall m_i \in M_i, \tag{3}$$

$$\sum_{i=1}^{|N|} r^{\nu}_{im_il} \leqslant a^{\nu}_l \quad \forall l \in R^{\nu},\ \forall m_i \in M_i, \tag{4}$$

$$m_i \in M_i \quad \forall i \in N, \tag{5}$$

$$s_0 = 0, \tag{6}$$

$$s_i \in \text{int}^+ \quad \forall i \in N, \tag{7}$$

where $S(t)$ denotes the set of activities in progress in period $[t-1, t[, t \in \{1,\ldots,s_{n+1}\}$. The objective function (1) minimizes the total makespan of the project. Constraint set (2) takes the finish-start precedence relations with a minimal time lag of zero into account. Constraints (3) and (4) take care of the renewable and nonrenewable resource limitations, respectively. Each activity i has to be performed in exactly one mode m_i (constraint (5)). Constraint (6) forces the project to start at time instance zero and constraint (7) ensures that the actvity start times assume nonnegative integer values. A schedule which fullfills all the constraints (1 to 7), is called *optimal*.

In the P-MRCPSP, activities are allowed to be preempted at any time and restarted later on at no additional cost. Therefore, each duration unit v of an activity i scheduled in mode m_i (with $v \in \{0,\ldots,d_{im_i}-1\}$) is assigned a starting time s_{iv}. The preemptive problem can be formulated as follows:

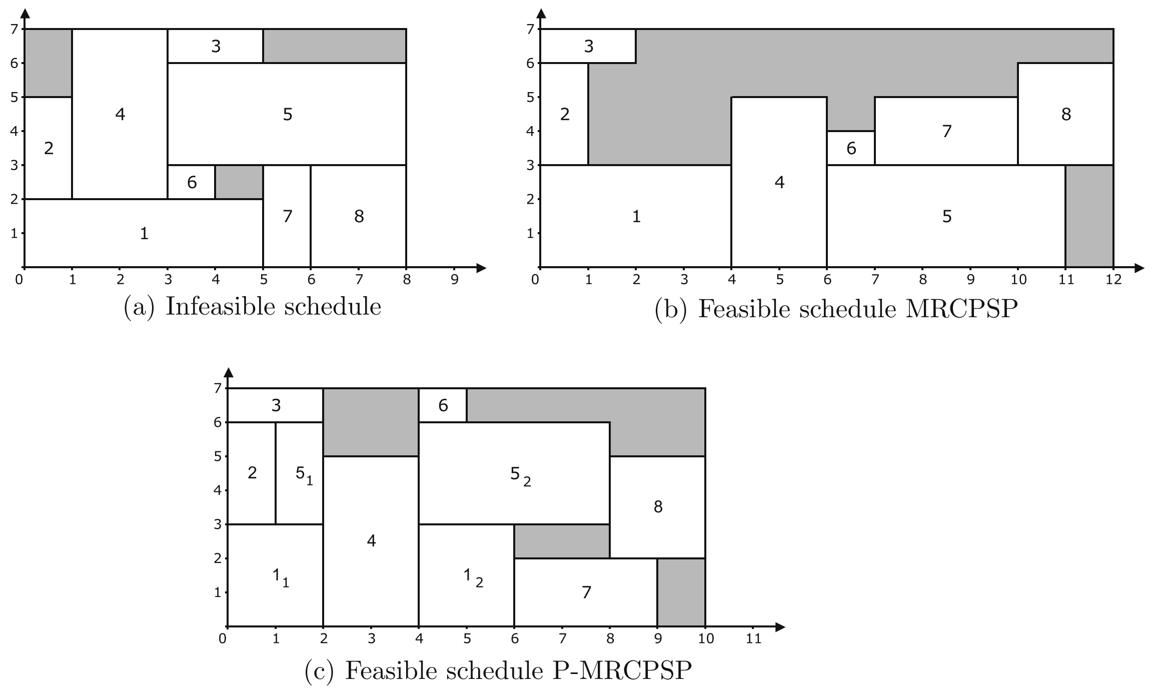

Fig. 2. Schedules of the example.

$$\text{Min. } s_{n+1,0}, \tag{1'}$$

s.t.

$$s_{i,d_{im_i}-1} + 1 \leqslant s_{j,0} \quad \forall (i,j) \in A, \tag{2'}$$

$$s_{i,v-1} + 1 \leqslant s_{i,v} \quad \forall i \in N, \ \forall v \in \{1, d_{im_i} - 1\}, \tag{2''}$$

$$\sum_{i \in S(t)} r^{\rho}_{im_ik} \leqslant a^{\rho}_k \quad \forall k \in R^{\rho}, \ \forall m_i \in M_i, \tag{3'}$$

$$\sum_{i=1}^{|N|} r^{v}_{im_il} \leqslant a^{v}_l \quad \forall l \in R^{v}, \ \forall m_i \in M_i, \tag{4'}$$

$$m_i \in M_i \quad \forall i \in N, \tag{5'}$$

$$s_{0,0} = 0, \tag{6'}$$

$$s_{i,v} \in \text{int}^{+} \quad \forall i \in N, \ \forall v \in \{0, d_{im_i} - 1\}. \tag{7'}$$

The constraints mentioned in the problem formulation are similar to the constraints for the MRCPSP-formulation. Except for constraint (2), which must be replaced by constraints (2′) and (2″). In constraint set (2′), the earliest start time of an activity j cannot be smaller than the finish time for the last unit of duration of its predecessor i. Constraint set (2″) guarantees that the start time for every time instance of an activity has to be at least one time-unit larger than the start time for the previous unit of duration.

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 (nonrenewable) resource is 7 (23). The activity-on-the-node network is shown in Fig. 1. In Table 2 the duration d_{im_i} and resource requirements (r^{ρ}_{ij} and r^{v}_{ij}) for mode m_i of activity i are shown. Fig. 2a depicts a schedule with a makespan of 8 days. However, this schedule is infeasible with respect to the nonrenewable resource. The schedule uses 25 nonrenewable resource units, which exceeds the available 23 nonrenewable resources. Fig. 2b shows a schedule with a makespan of 12 days. This schedule is feasible because it uses exactly 23 nonrenewable resource units. If we relax the problem to the P-MRCPSP, a schedule as shown in Fig. 2c can be generated. In Section 3.2 we present the optimal solution for both problems.

3. A genetic algorithm for the MRCPSP

Introduced by Holland (1975), genetic algorithms (GAs) use techniques and procedures inspired by evolutionary biology to solve complex optimization problems. Several selection 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 the bi-population genetic algorithm (BPGA), as introduced for the single-mode RCPSP by Debels and Vanhoucke (2005), has been extended to the multi-mode case. The BPGA 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 procedure starts with an initial population POP_L of left-justified schedules. On this population the iterative forward–backward procedure of Li and Willis (1992) is applied: the backward procedure is used to feed the population of right-justified schedules POP_R with combinations of population elements of POP_L that are scheduled backwards with the serial SGS. The forward procedure is applied on the population elements of POP_R for updating POP_L. This process is repeated until the stop criterium is met. In the remainder of this section we first examine the representation of the individuals, based on the random key representation with topological ordering notation proposed by Valls et al. (1999). We go more deeply into the schedule generation scheme, which is extented with a local improvement search, adapted from Hartmann (2001). Finally, we also reveal the algorithmic details of our BPGA.

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 embedded. 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 sequence in which the activities are scheduled is based on the priority value attributed to each 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 unique RK representation also leads to promising results thanks to the use of the topological ordering (TO) notation (Valls et al., 1999). A topological order of the activities is an order which is compatible with the precedence relations of the project. It implies that for all activities i and j for which $s_i < s_j$, activity i should have a higher priority than activity j.

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 and a mode list μ.

Each priority value in the vector λ is linked to an activity and based on these values, the sequence in which the activities will be scheduled is assigned. Next to the vector λ, each population element has also a mode execution list μ. This mode list μ assigns a mode m_i to each activity i $(i \in \{1, \ldots, |N|\})$ and determines if the schedule will be feasible with respect to the nonrenewable resources. For example, schedule Fig. 2a is based on the mode list (2,1,2,1,2,1,1,2), while schedule Fig. 2b and c are based on the mode list (1,1,2,1,2,1,2,2).

The number of requested nonrenewable resource units that exceeds the capacity a^{v}_l, $l \in R^{v}$, is defined as the excess of resource request $ERR(\mu)$. An $ERR(\mu) = 0$ means that the solution is feasible. If $ERR(\mu)$ is larger than 0, the solution is infeasible. The formula of the $ERR(\mu)$ can be stated as follows:

$$ERR(\mu) = \sum_{j=1}^{l} (\max(0, \sum_{i=1}^{|N|} (r^{v}_{im_ij}) - a^{v}_j)) \quad l \in R^{v}. \tag{8}$$

Schedule Fig. 2a has an $ERR(\mu)$ of 2 (25 requested versus 23 available nonrenewable resource units) and is therefore infeasible. Schedules Fig. 2b and c on the other hand are feasible, because the required nonrenewable resource units do not exceed the availability, with an $ERR(\mu)$ of 0.

In order to allow activity preemption, the original activity network need to be converted to a new network. From the moment a mode m_i is assigned to an activity i using the mode list μ, its duration d_{im_i} is known. Afterwards, each activity is split into d_i subactivities with a unit duration of one, resulting in the new constructed network (see Demeulemeester and Herroelen (1996) and Vanhoucke and Debels (2008)). The vector λ, which now determines a schedule sequence for all subactivities of the new project network, determines the subactivity starting times and hence the algorithm can now be used with no further changes.

3.2. Extended generation scheme

A schedule generation scheme (SGS) translates the vector λ and the mode list μ into a schedule S. Two different types of SGSs exist in the 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 constraints. However, we have extended the serial SGS with the following two elements.

First, our procedure makes use of the forward/backward scheduling technique, a well-known local search technique for RCPSP meta-heuristics. The idea of scheduling activities in backward and forward direction has been introduced by Li and Willis (1992) and then used as improving method by Tormos and Lova (2001). Afterwards several authors as Alcaraz and Maroto (2001), Valls et al. (2005) and Debels et al. (2006) amongst others have also used similar procedures. The forward (backward) procedure is applied on the population $POP_R(POP_L)$ of right-justified (left-justified) schedules and is used to build the left-justified (right-justified) schedules which are stored in the population $POP_L(POP_R)$. To embed the TO condition in the random key representation, the start (finish) times of the activities of the schedules in $POP_L(POP_R)$ are used as the priority values of the random key. The sequence of the activities is made by sorting the activities in (reverse) chronological order of their starting (finish) times (Debels and Vanhoucke, 2007).

Second, our serial generation scheme is extended with a procedure to improve the mode selection. For every activity $i \in \{1, \ldots, |N|\}$ that will be scheduled, both in the forward and in the backward procedure, there is a probability of P_{modimp} that the mode improvement 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 improvement 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 $k = 1, \ldots, |M_i|$ 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.

For the preemptive case, the mode improvement procedure is only applied to activities that have been scheduled completely (i.e. for which all subactivities are scheduled).

The pseudocode for the mode improvement procedure for an activity i of a project with an excess resource request of $ERR(\mu)$ is shown below:

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

This mode improvement procedure is a combination of a mode selection rule of Lova et al. (2006) on the one hand and the single-pass improvement local search of Hartmann (2001) on the other hand. Lova et al. (2006) extended the serial generation scheme using the 'Earliest Feasible Finish Time' as a mode selection rule. During the generation of the schedule, this rule selects for each activity the execution mode such that it is scheduled with the smallest feasible finish time as possible. The local search of Hartmann (2001) is based on the multi-mode left shift of Sprecher (1994). For each completely generated feasible schedule, the procedure checks for every activity whether a multi-mode left shift can be performed. A multi-mode left shift of an activity j is an operation on a given schedule which reduces the finish time of activity j without changing the modes or finish times of the other activities and without violating the constraints. Unlike the procedure of Hartmann, our procedure tries to improve the makespan during the generation of the schedule.

One could think that the mode improvement 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. 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 Fig. 2. Suppose that activities 4 and 7 are subject to the mode improvement procedure. After scheduling activities 1, 2 and 3 (without applying the procedure), the mode improvement procedure is used for 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 (see Fig. 3a). When changing the mode of activity 4 to mode 2, the $ERR(\mu)$ remains equal, because the required nonrenewable resource units (22) are smaller

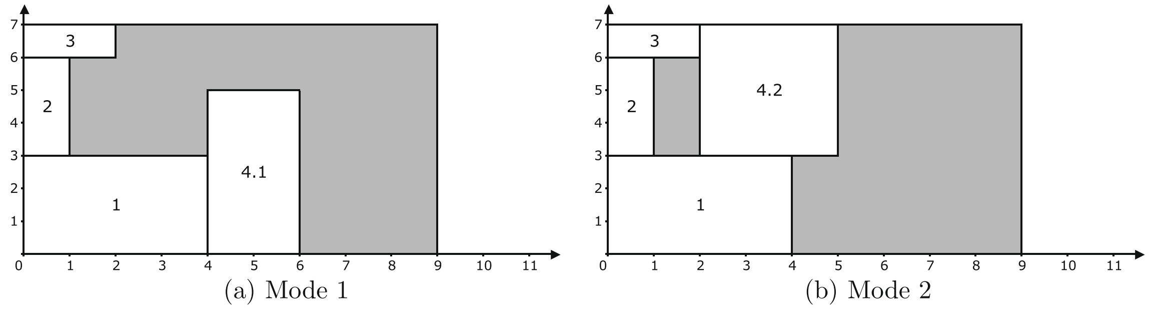

Fig. 3. Mode improvement of activity 4.

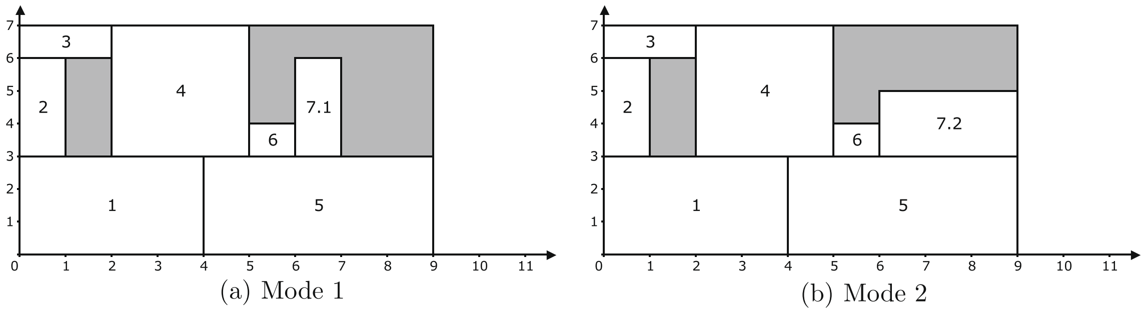

Fig. 4. Mode improvement of activity 7.

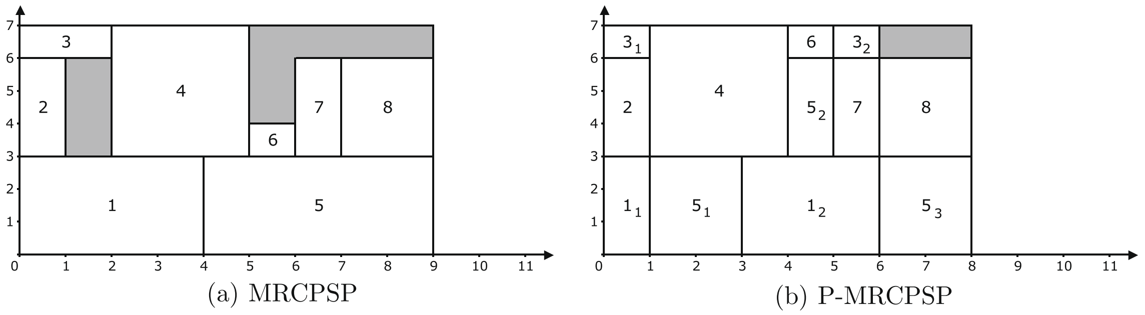

Fig. 5. Optimal solutions.

than the available ones (23). Although the duration of mode 2 is larger than mode 1, the required renewable resource units also decrease and the activity can be scheduled in parallel with activity 3. The finish time for mode 2 is now 5 (see Fig. 3b). Because mode 2 has a lower finish time and the $ERR(\mu)$ does not deteriorate, the mode list is adapted. After scheduling activities 5 and 6, the mode improvement is applied on activity 7. For activity 7, mode 2 is currently selected and when inserting the activity in the current partial schedule, this activity has a finish time of 9 (see Fig 4b). However, when choosing mode 1, the finish time can be reduced to 7, because no renewable resource conflict occurs (see Fig. 4a). The nonrenewable resource requirements increase to 23 and hence, the $ERR(\mu)$ remains 0. As the finish time is smaller for mode 1, the mode list for activity 7 is adapted to mode 1. After scheduling activity 8, the optimal solution for this specific problem is obtained, as shown in Fig. 5a. The optimal solution for the preempted problem is shown in Fig. 5b.

3.3. Details of the genetic algorithm

3.3.1. Preprocessing

Before the genetic algorithm is started, 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 there is another mode of the same activity with the same or smaller duration and no more requirements for all resources. A mode is called *non-executable* if its execution would violate the renewable or nonrenewable resource constraints in any schedule. 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 Fig. 1. Mode 1 of activity 8 can be called *inefficient* because both its duration and its resource requirements are equal or larger than those of mode 2. Also mode 1 of activity 5 can be excluded, because this mode is *non-executable* with respect to the nonrenewable resource. If it were executed, the project would require at least 24 nonrenewable resource units, while only 23 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, $i \in \{1, \ldots, |N|\}$, an execution mode m_i is randomly selected. To minimize the number of infeasible solutions in the initial population, 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 assignment have been made. In this paper, J equals to four 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 has been 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 C_{max}. 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 and revealed 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:

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

If the schedule is feasible ($ERR(\mu) = 0$), the fitness function is equal to the makespan of the project $C_{max}(\lambda, \mu)$ obtained from the vector λ and mode list μ. If the schedule is infeasible, the fitness function is equal to the sum of the maximal durations of the activities (T) plus the $ERR(\mu)$ of the mode list μ. The lower the fitness value of a certain schedule is, the better the quality of the related schedule is. However, Alcaraz et al. (2003) point out that two different individuals with the same excess of resource request but with a different makespan have 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:

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

where $max\_feas\_C_{max}$ 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. The other parent represents the mother.

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 vector λ and mode list μ are adapted. 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. All data left from that point, both from the activity 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 crossover, as tested in Özdamar (1999) or the peak crossover (Valls et al. (2008), 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 vector λ and mode list μ is done randomly. However, computational tests have revealed that the probabilities P_{mut_act} and P_{mut_mod} to obtain the best results are equal to 4% and 2%, respectively.

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 two 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. However, to avoid loosening high-quality schedules, we do not perform a replacement if the individual corresponds to a schedule with the best found makespan 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. In Section 4.1, we configure the parameters of our genetic algorithm on a new generated dataset. In Section 4.2, we compare our results with two benchmark datasets available in literature, the PSPLIB and the dataset of Boctor (1993). In Section 4.3, we study the influence of the introduction of preemption.

4.1. Configuring the algorithm

In this section, we configure the parameters of our genetic algorithm. Therefore we have generated a training set with projects of 10, 15, 20, 25 and 30 activities, each with 3 modes and 2 renewable and 2 nonrenewable resources. The order strength (OS) equals 0.25, 0.50 or 0.75. The renewable and nonrenewable resource strength equals 0.25, 0.50, 0.75 or 1 and the resource factor equals 0.5 or 1. In total, 960 different project instances were generated. For the generation of the instances, we have used the RanGen project scheduling instances generator developed by Vanhoucke et al. (2008) and extended the projects to a multi-mode version. On this dataset, we test the mode improvement procedure, the introduction of two populations and the different fitness computations. In Table 3 the percentage deviation from the best found solution is shown. The following remarks can be made:

Table 3
Results for the training set – configuration of the algorithm – average % deviation best found solution – 5000 schedules.

# Populations	Fitness comp.	P_{modimp}										
		0 (%)	10 (%)	20 (%)	30 (%)	40 (%)	50 (%)	60 (%)	70 (%)	80 (%)	90 (%)	100 (%)
1 Population	Alcaraz	7.34	1.21	**0.70**	0.81	0.87	1.01	0.82	0.80	1.27	1.12	1.41
	Hartmann	8.43	1.16	1.15	**0.71**	0.92	1.01	0.82	1.05	1.30	0.94	1.27
2 Populations	Alcaraz	0.85	0.11	0.02	**0.00**	0.19	0.02	0.07	0.01	0.04	0.01	0.06
	Hartmann	1.56	0.44	0.15	0.14	0.09	0.18	**0.04**	0.05	0.21	**0.04**	0.08

Table 4
PSPLIB instances.

	Number of activities						
	10	12	14	16	18	20	30
Number of instances	536	547	551	550	552	554	552
Average number of modes	2.801	2.832	2.849	2.865	2.871	2.869	2.883

Table 5
Comparison with other heuristics – average % optimality gap – 5000 schedules.

	Instances set					
	J10	J12	J14	J16	J18	J20
This work	**0.01**	**0.09**	**0.22**	**0.32**	**0.42**	**0.57**
Lova et al. (2009)	0.06	0.17	0.32	0.44	0.63	0.87
Jarboui et al. (2008)	0.03	0.09	0.36	0.44	0.89	1.10
Ranjbar et al. (2008)	0.18	0.65	0.89	0.95	1.21	1.64
Alcaraz et al. (2003)	0.24	0.73	1.00	1.12	1.43	1.91
Jozefowska et al. (2001)	1.16	1.73	2.60	4.07	5.52	6.74
Optimal (%)	99.63%	98.17%	94.56%	92.00%	88.95%	85.74%
CPU-time (seconds)	0.12	0.13	0.14	0.15	0.16	0.17

Mode improvement procedure: The value of the probability of applying the mode improvement procedure P_{modimp} is varying between 0% and 100% in steps of 10%. If we compare the results of the algorithm with and without the mode improvement procedure, the table shows that the introduction of the mode improvement procedure ameliorates the solution quality significantly.

Number of populations: Although a bi-populational genetic algorithm is used, we also check if the one-populational genetic algorithm reveals other results. As shown in Table 3, 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.

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, the fitness value of Alcaraz et al. (2003) reveals better results than the one of Hartmann (2001).

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 improvement procedure, P_{modimp}, equal to 30%.

When configuring the algorithm, we have noticed that the population size has an influence on the performance of the algorithm. We examined that the population size is negatively related to the number of activities. Simular results were found in Debels and Vanhoucke (2005), Hartmann (2001) and Alcaraz and Maroto (2001), amongst others. A nonlinear least squares regression based on the best population size values for the different number of activities revealed a negative relationship between the population size (POP) and the number of activities ($|N|$) which can be defined as follows:

$$POP = e^{3.551+\frac{22.72}{|N|}}.$$

A large population size avoids homogeneity of the population and this becomes more important for small problem instances. However, when the population size becomes too large, only few generations can be computed within the time limit and the advantages of the genetic algorithm cannot be fully exploited.

4.2. Comparison with other heuristics

In this section, we compare our algorithm with the best performing heuristics and meta-heuristics in literature. As a benchmark, we use the well-known PSPLIB dataset (Kolisch and Sprecher, 1996) and the dataset of Boctor (1993) to compare with other existing procedures from the literature. The PSPLIB 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).

A comparison of the different algorithms on the J10 to J20 datasets of PSPLIB is made for the authors mentioned in Table 5. To make a fair comparison between the different heuristics, the evaluation is stopped after 5000 generated schedules. As Kolisch and Hartmann (2006) stated, one schedule corresponds to (at most) one start time assignment per activity, as done by a SGS. The number of generated schedules can be calculated as the sum of times each activity of the project has obtained a feasible start time divided by the number of activities of the project. As can be seen in Table 5, the results for our algorithm outperform the other heuristics. In the second part of the table, we also present the percentage of optimal solutions found and the computation time for each of the datasets.

We have also tested our algorithm on the J30 dataset. This is the dataset with the largest instances in PSPLIB and contains 640 data instances of 30 activities (from which for 552 instances a feasible solution is found). The best known solutions for these instances can be found on the PSPLIB-server. In Table 6, the column with label '% Dev. BKS' shows the average deviation from the best known solutions (BKS) till now[1] for 1000, 5000 and 50,000 schedules computed. However, as comparison of these deviations is not possible (due to the change in the BKS), the percentage increase of the project duration above the minimal critical path length and the CPU-time needed is also indicated. This information can be used by other researchers to compare with their results. In the last two columns of the table, the percentage of instances which results in an equal or worse result than the best known solution is shown. No improvements in the best known solutions were found.

[1] Results on the PSPLIB-server on 20 February 2009.

Table 6
Results for the J30-dataset.

	J30				
	Dev. BKS (%)	Incr. CP (%)	CPU (seconds)	Worse	Equal
1000 schedules	2.27	15.30	0.05	40.4	59.6
5000 schedules	1.08	13.75	0.24	29.0	71.0
50,000 schedules	0.73	13.31	2.46	20.5	79.5

Table 7
Results for the Boctor dataset – average% deviation from critical path length.

	Boctor 50		Boctor 100	
	1000 sched. (%)	5000 sched. (%)	1000 sched. (%)	5000 sched. (%)
This work	27.36	23.41	29.70	24.67
Lova et al. (2009)	24.89	23.70	26.96	24.85
Alcaraz et al. (2003)	33.83	26.52	41.85	29.16

The second benchmark dataset on which we have tested our algorithm is the dataset of Boctor (1993). This dataset contains 240 instances and contains the largest instances of problem MRCPSP. There is one set composed of 120 instances of 50 activities and one set composed of 120 problems of 100 activities. For each project, one, two or four resource types were used. However, the dataset does not include nonrenewable resources. More information concerning the dataset can be found in Boctor (1993). Alcaraz et al. (2003) and Lova et al. (2009) reported the percentage increase of project duration above the critical path length. The comparison can be found in Table 7.

4.3. Preemption

This section presents computational results for the MRCPSP with activity preemption. Table 8 summarizes the results obtained from tests on the PSPLIB data instances J10, J20 and J30 with and without the presence of the nonrenewable resources. Results are also shown for the Boctor50 and Boctor100 data instances (Boctor, 1993), which contain projects without nonrenewable resources. The second and third column display the average deviation from the critical path based lower bound for the MRCPSP and P-MRCPSP, respectively. The column with label 'Av.Impr.' displays the average makespan improvement for the P-MRCPSP relative to the MRCPSP. The columns with labels 'Better', 'Equal' and 'Worse' show the number of preemptive solutions with a lower, equal or higher project makespan than the solutions found for the MRCPSP. The results in Table 8 can be used for comparison purposes for future research in this area. The computational results for the different datasets can be downloaded from www.projectmanagement.ugent.be/mrcpsp.html.

Table 8
Results for different datasets with and without preemption – 5000 schedules.

		MRCPSP (%)	P-MRCPSP (%)	Av. Impr. (%)	Better	Equal	Worse
NR	J10	32.27	31.54	0.49	47	488	1
	J20	17.76	17.03	0.55	99	432	23
	J30	13.75	13.23	0.41	108	404	40
No NR	Boctor50	23.41	21.52	1.55	111	9	0
	Boctor100	24.67	21.91	2.22	120	0	0
	J10[a]	15.49	14.94	0.42	36	500	0
	J20[a]	8.27	7.61	0.53	68	486	0
	J30[a]	5.60	5.06	0.44	74	478	0

[a] Ignoring the nonrenewable resources (NR).

Table 8 shows that activity preemption obviously leads to an overall average makespan improvement. However, the PSPLIB instances show that a fraction of the projects shows a higher makespan with preemption compared to the best found MRCPSP solution. Indeed, introducing activity preemption results in a larger project network, since each non-preempted activity needs to be splitted into subactivities, leading to an increase of the random key size. This larger RK search space is responsible for the fraction of solutions displayed in the 'Worse' column.

The table also shows that activity preemption always leads to better solutions (the column 'Worse' is equal to 0) when no nonrenewable resources are taken into account. In that case, infeasible mode assignments will never occur, leaving more room to the algorithm to select low duration modes. Obviously, when nonrenewable resources are taken into account, the low durations correspond to relatively high nonrenewable resource demand, leading to infeasible mode assignments. We indeed observed that the best found solutions without nonrenewable resources contain many mode assignments with low activity durations, which is in line with the study of Boctor (1993) who has shown that the 'shortest feasible mode' selection rule performs best.

5. Conclusions

In this paper we have designed a genetic algorithm for the multi-mode resource-constrained project scheduling problem and its extension to the preempted case. 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 improves the mode selection by choosing that feasible mode of a certain activity that minimizes the finish time of that activity. The computational results for the MRCPSP with and without activity preemption show that the introduction of these two elements increases the performance of the genetic algorithm. The results for the MRCPSP outperforms all the existing algorithms in literature, within reasonable computation times. Areas of further research are e.g. the adaption of the algorithm to other project scheduling problems, such as the DTRTP.

References

Alcaraz, J., Maroto, C., 2001. A robust genetic algorithm for resource allocation in project scheduling. Annals of Operations Research 102, 83–109.

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.

Ballestin, F., Valls, V., Quintanilla, S., 2008. Pre-emption in resource-constrained project scheduling. European Journal of Operational Research 189, 1136–1152.

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.

Buddhakulsomsiri, J., Kim, D., 2006. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. European Journal of Operational Research 175, 279–295.

Damay, J., Quilliot, A., Sanlaville, E., 2007. Linear programming based algorithms for preemptive and non-preemptive rcpsp. European Journal of Operational Research 182, 1012-–1022.

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.

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.

Demeulemeester, E., Herroelen, W., 1996. An efficient optimal procedure for the preemptive resource-constrained project scheduling problem. European Journal of Operational Research 90, 334–348.

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. A classification scheme for project scheduling. In: Handbook of Recent Advances in Project Scheduling. Kluwer Academic Publishers, pp. 1–26.

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

Jarboui, B., Damak, N., Siarry, P., Rebai, A., 2008. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Applied Mathematics and Computation 195, 299–308.

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.

Kaplan, L., 1988. Resource-Constrained Project Scheduling with Preemption of Jobs. Ph.D. Thesis, University of Michigan.

Kelley Jr., J.E., 1963. The critical-path method: Resources planning and scheduling. In: Industrial Scheduling. Prentice-Hall, New Jersey, 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. Heuristic algorithms for solving the resource-constrained project scheduling problem. In: Project Scheduling – Recent Models, Algorithms and Applications. Kluwer Academic Publishers, Boston, MA, pp. 147–178.

Kolisch, R., Hartmann, S., 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, 23–37.

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.

Lova, A., Tormos, P., Barber, F., 2006. Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. Inteligencia Artificial 30, 69–86.

Lova, A., Tormos, P., Cervantes, M., Barber, F., 2009. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. International Journal of Production Economics 117 (2), 302–316.

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). Technical Report, 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. An algorithm for a general class of precedence and resource constrained scheduling problem. In: Advances in Project Scheduling. Elsevier, Amsterdam, pp. 3–28.

Ranjbar, M., De Reyck, B., Kianfar, F., 2008. A hybrid scatter-search for the discrete time/resource trade-off problem in project scheduling. European Journal of Operational Research 193 (1), 35–48.

Slowinski, R., 1980. Two approaches to problems of resource allocation among project activities – A comparative study. Journal of Operational Research Society 8, 711–723.

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., 1994. Resource-constrained project scheduling: Exact methods for the multi-mode case. Lecture Notes in Economics and Mathematical Systems, vol. 409. Springer, Berlin.

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.

Tormos, P., Lova, A., 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research 102, 65–81.

Valls, V., Laguna, M., Lino, P., Prez, A., Quintanilla, S., 1999. Project scheduling with stochastic activity interruptions. In: Project Scheduling Recent Models, Algorithms and Applications. Kluwer Academic Publishers, pp. 333–354.

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., Ballestín, F., Quintanilla, S., 2008. A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 185 (2), 495–508.

Vanhoucke, M., Debels, D., 2008. The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects. Computers and Industrial Engineering 54, 140–154.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., Tavares, L., 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. European journal of operational research 187 (2), 511–524.

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

Zhu, G., Bard, J., Tu, G., 2006. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. Journal on Computing 18 (3), 377–390.