J Heuristics (2011) 17:705–728
DOI 10.1007/s10732-010-9152-0

Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem

Vincent Van Peteghem · Mario Vanhoucke

Received: 16 May 2010 / Revised: 4 October 2010 / Accepted: 1 December 2010 /
Published online: 22 December 2010

Abstract In the past decades, resource parameters have been introduced in project scheduling literature to measure the scarceness of resources of a project instance. In this paper, we incorporate these resource scarceness parameters in the search process to solve the multi-mode resource constrained project scheduling problem, in which multiple execution modes are available for each activity in the project. Therefore, we propose a scatter search algorithm, which is executed with different improvement methods, each tailored to the specific characteristics of different renewable and non-renewable resource scarceness values. Computational results prove the effectiveness of the improvement methods and reveal that the procedure is among the best performing competitive algorithms in the open literature.

Keywords Multi-mode · Scheduling · Scatter-search · Resource characteristics

1 Introduction

The single-mode resource-constrained project scheduling problem (RCPSP) is a well-known optimization problem in the project scheduling literature. This problem type aims at minimizing the total duration or makespan of a project subject to precedence relations between the activities and limited renewable resource availabilities, and is known to be NP-hard (Blazewicz et al. 1983). Several solution procedures have

V. Van Peteghem
Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2,
9000 Gent, Belgium
e-mail: vincent.vanpeteghem@ugent.be

M. Vanhoucke (✉)
Operations and Technology Management Centre, Vlerick Leuven Gent Management School, Reep 1,
9000 Gent, Belgium
e-mail: mario.vanhoucke@ugent.be

been developed for this problem and for its extensions to other optimization problems (reviews can be found in Brucker et al. 1999 and Herroelen et al. 1999). One of the extensions is the multi-mode resource-constrained project scheduling problem (MRCPSP). The MRCPSP is a generalized version of the single-mode RCPSP, where each activity can be performed in one out of a set of modes, each mode with a different activity duration and different renewable and nonrenewable resource requirements. The objective of this optimization problem is to minimize the makespan of the problem. The multi-mode resource-constrained project scheduling problem can be represented as $m, 1T|cpm, disc, mu|C_{max}$ using the classification scheme of Herroelen et al. (1999) or as $MPS|prec|C_{max}$ following the classification scheme of Brucker et al. (1999).

Several exact, heuristic and meta-heuristic procedures to solve the MRCPSP have been proposed in the recent years. Branch-and-bound procedures were introduced by Sprecher et al. (1997), Hartmann (1998) and Sprecher and Drexl (1998), while Zhu et al. (2006) proposed a branch-and-cut algorithm. Boctor (1996a), Drexl and Grünewald (1993), Knotts et al. (2000), Kolisch and Drexl (1997), Lova et al. (2006) and Özdamar and Ulusoy (1994) presented single or multi-pass heuristics. Mori and Tseng (1997), Özdamar (1999), Hartmann (2001), Alcaraz et al. (2003), Lova et al. (2009), Tseng and Chen (2009) and Van Peteghem and Vanhoucke (2010) presented a genetic algorithm, Slowinski et al. (1994), Boctor (1996b), Józefowska et al. (2001) and Bouleimen and Lecocq (2003) used the simulated annealing approach, Nonobe and Ibaraki (2002) proposed a tabu search procedure and Zhang et al. (2006) and Jarboui et al. (2008) applied the methodology of particle swarm optimization to the MRCPSP. Recently, Ranjbar et al. (2009) proposed a hybridized scatter search procedure to solve the MRCPSP.

Different research papers have described valuable insights in the relation between project characteristics and the performance of solution procedures for both the single-mode and multi-mode RCPSP (see Herroelen and De Reyck 1999 and Kolisch et al. 1995). To the best of our knowledge, for the multi-mode RCPSP only one paper has used project characteristics to steer the algorithmic procedures towards promising solution areas. Buddhakulsomsiri and Kim (2007) proposed the moving resource strength, which helps the priority rule-based heuristic for the MRCPSP with activity splitting and resource vacation to identify in which project situations activity splitting is likely to be beneficial during scheduling.

The main contribution of the paper is threefold: first, a scatter search procedure to solve the scheduling problem is proposed. While a scatter search is proven to be successful to deal with combinatorial problems, it is—to the best of our knowledge—not used earlier to solve the MRCPSP. Second, three different solution improvement methods are developed, each based on the information obtained from the renewable and nonrenewable resource characteristics. Third, the proposed algorithm provides state-of-the-art results for the available benchmark datasets.

The outline of this paper is as follows: In Sect. 2 the MRCPSP is described while in Sect. 3 a resource scarceness matrix is presented which gives insight into the influence of the scarceness of the renewable and nonrenewable resources on the search focus of the algorithm. Section 4 proposes a scatter search for the MRCPSP, for which different solution improvement methods tailored to the resource scarceness characteristics of a project, are presented. In Sect. 5, computational results for the configuration of the scatter search are given, the influence of the resource scarceness on the solution quality is tested and the results of the algorithm on the well-known PSPLIB dataset are shown. Finally, in the last section overall conclusions and suggestions for future research are presented.

2 Problem formulation

We consider 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 each non-preemptable activity i, when executed in mode m_i, is d_{im_i}. An activity i, executed in mode m_i, requires $r^\rho_{im_ik}$ renewable resource units of type k ($k \in R^\rho = \{1, \ldots, |R^\rho|\}$) and $r^\nu_{im_il}$ nonrenewable resource units of type l ($l \in R^\nu = \{1, \ldots, |R^\nu|\}$). Each renewable resource k has a constant availability a^ρ_k throughout the project horizon, while the overall capacity of each nonrenewable resource of type l is given by a^ν_l. The number of requested nonrenewable resource units that exceeds the capacity a^ν_l, $l \in R^\nu$, is defined as the Excess of Resource Request (*ERR*). If the nonrenewable resource request exceeds the availability, *ERR* will be larger than 0 and the solution will be infeasible. The formula of the *ERR* can be stated as follows:

$$ERR = \sum_{l=1}^{|R^\nu|}\left(\max\left(0, \sum_{i=1}^{|N|}(r^\nu_{im_il}) - a^\nu_l\right)\right)$$

The objective of the MRCPSP is to find a feasible schedule, represented by a vector of corresponding start times s_i and a vector denoting its corresponding execution modes m_i, such that the makespan of the project is minimized, subject to the precedence constraints and (renewable and nonrenewable) resource constraints.

The MRCPSP can be conceptually formulated as follows:

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

s.t.

$$s_i + d_{im_i} \le s_j \quad \forall (i, j) \in A \tag{2}$$

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

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

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

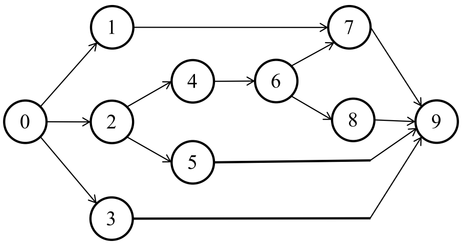


Fig. 1 Network of the example project

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

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

where $S(t)$ denotes the set of activities in progress in period $[t-1, t[$, $t \in \{1, s_{n+1}\}$. The project is minimized in objective function (1). 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 activity start times assume nonnegative integer values. A schedule which fulfills all the constraints (1) to (7), is called *optimal*.

In Fig. 1, an example project is presented which will be used throughout the remainder of this paper. The project network contains 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). In Table 1, the duration d_{im_i} and resource requirements ($r^{\rho}_{im_i}$ and $r^{\nu}_{im_i}$) for mode m_i of activity i are shown.

3 Resource scarceness matrix

Several resource parameters have been introduced in the past decades to measure the scarceness of resources of a project instance. These parameters are determined by the resource consumption and the resource availability. For a constant resource availability, the scarceness will increase for an increasing resource consumption. Moreover, the more restrictive a resource type becomes, the higher the makespan of the project will be.

Since the resource scarceness can be applied on both the renewable and the nonrenewable resources, a brief description for both resource types is given.

- **Renewable resources**
  When the scarceness of the renewable resources is low, the project is hardly restricted by its resources. The makespan of the project will mainly be determined by the precedence relations of the project and each activity will be scheduled at or close to its critical path start time. When, however, the scarceness of renewable resources is high, the influence of the resource constraints will overrule the precedence constraints. Due to the relatively low resource availability, most of the

Table 1 Information of the example project

act i	mode m_i	d_{im_i}	$r^{\rho}_{im_i}$	$r^{\nu}_{im_i}$
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
Available			7	23

activities will be scheduled one after the other. An increase from a low to a high resource scarceness also leads to an increasing deviation of the project makespan above the minimal critical path duration.

The renewable resource scarceness might influence the performance of a solution procedure. Projects with a low scarceness might need a search procedure which focuses on the neighbourhood of the minimal critical path mode assignment (i.e. the critical path using the minimal duration of activities), while a search procedure that will mainly focus on the limitations imposed by the renewable resource availabilities might be more effective for projects with a high scarceness of the renewable resources.

– **Nonrenewable resources**
The feasibility of a mode assignment is determined by the nonrenewable resource consumption. A mode combination is feasible if the sum of the requested nonrenewable resources is smaller than or equal to the nonrenewable resource availabilities (i.e. if *ERR* is equal to zero). In case the scarceness of the nonrenewable resources is low, many mode assignment combinations will be feasible. The more the scarceness of the nonrenewable resources increases, the more mode combinations will become infeasible. Consequently, the higher the scarceness of the nonrenewable resources, the more the search procedure should focus on the search for feasible mode assignments.

Combining the information of the resource scarceness of both the renewable and the nonrenewable resources, a *resource scarceness matrix* can be presented, as shown

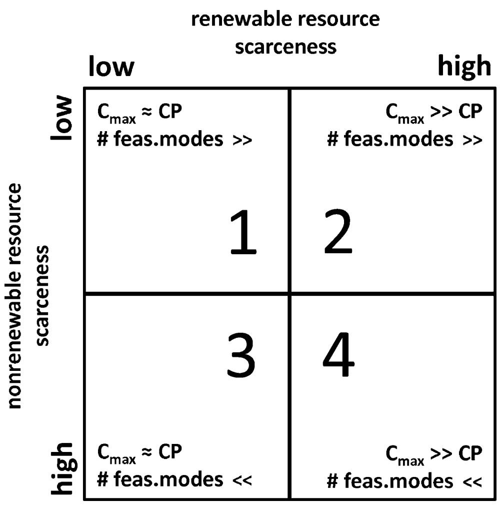


Fig. 2 The resource scarceness matrix

in Fig. 2. On the horizontal axis the scarceness of the renewable resources, moving from low to high, is presented, while on the vertical axis the scarceness of the nonrenewable resources is shown. The matrix can be divided into four quadrants: in quadrants 1 and 2, the number of feasible modes (# feas.modes) is high (indicated as '$\gg$'), while the amount of feasible modes is more limited ('$\ll$') in quadrants 3 and 4, due to the high nonrenewable resource scarceness. In quadrants 1 and 3, the makespan (C_{max}) of the projects with a low renewable resource scarceness will be close to the critical path duration (CP), while the makespan of the projects with high resource scarceness values in quadrants 2 and 4 might deviate significantly from the minimal critical path duration.

In the next section, a scatter search heuristic and different improvement methods are presented. The improvement methods are based on the resource scarceness characteristics of each quadrant to increase the effectiveness of the search procedure in each of the quadrants.

4 Scatter search

Scatter search is a population-based meta-heuristic, proposed by Glover et al. (2000), in which solutions are intelligently combined to yield better solutions. The scatter search method involves deterministic procedures that can include problem specific knowledge (Pinol and Beasley 2006) and can therefore be implemented in a variety of ways and degrees of sophistication.

Scatter search algorithms are often classified as so-called evolutionary methods. However, the scatter search algorithm contrasts with other evolutionary procedures, such as genetic algorithms, by providing unifying principles of joining solutions based on generalized path constructions in Euclidian space and by utilizing strategic designs where other approaches resort to randomization (Glover et al. 2000).

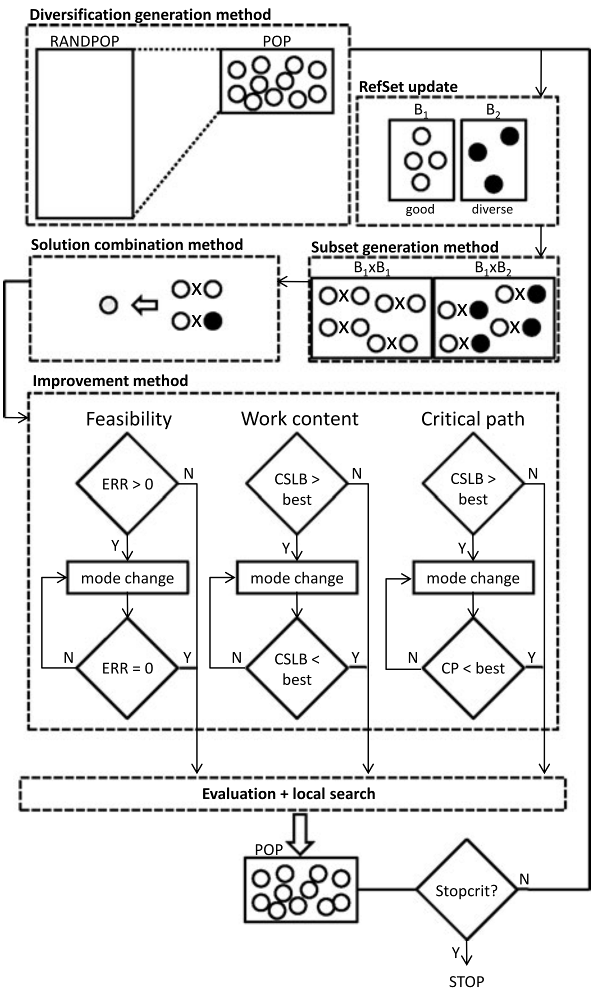


Fig. 3 A conceptual overview of the scatter search procedure

For an overview of the basic and advanced features of the scatter-search, we refer to Glover et al. (2000) and Marti et al. (2006). The scatter search we present in this paper has a generic procedure as outlined in the pseudo-code below.

```
1. Diversification Generation Method
While Stop Criterium not met
2. Subset Generation Method
3. Solution Combination Method
4. Improvement Method
5. Reference Set Update Method
Endwhile
```

In Fig. 3, a conceptual overview of the different steps in our scatter search procedure is shown. In the remainder of this section, each of these different steps is explained in detail.

Table 2 Solution vector representation

Activity	1	2	3	4	5	6	7	8
RK	12	18	21	22	23	29	30	35
ML	2	1	2	1	2	1	1	2

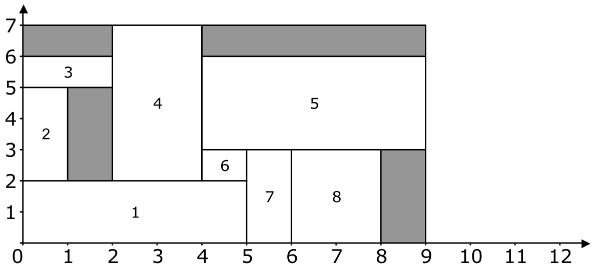


Fig. 4 Schedule of example project

4.1 The diversification generation method

4.1.1 Schedule representation

A schedule is represented by a solution vector, which contains two lists: a *random key* (RK), which determines the sequence in which the activities are scheduled, and a *mode list* (ML), which determines the duration and the resource requirements for each activity. In the RK representation, the sequence in which the activities are scheduled is based on the priority value attributed to each activity. It is assumed that a low RK value corresponds to a high priority. The mode list represents the execution modes of the activities in ascending order, i.e. the first number in the list indicates the mode in which the first activity will be executed, the second number the execution mode of the second activity, etc. An example of solution vector representation is given in Table 2.

A schedule generation scheme (SGS) translates the solution vector into a schedule. In this paper, we make use of a serial scheduling generation scheme (Kelley 1963) which sequentially adds activities to the schedule one-at-a-time. In each iteration, the next activity is chosen based on the priorities in the random key and that activity is assigned to the schedule as soon as possible within the precedence and resource constraints.

Figure 4 depicts a schedule which is based on the solution vector as proposed in Table 2. The schedule has a makespan of 9 days and is generated by using a serial schedule generation scheme. However, this schedule is infeasible with respect to the nonrenewable resources since 25 nonrenewable resource units are used, while only 23 nonrenewable resources are available (which means that *ERR* is equal to 2).

4.1.2 Initial population

In this first step, a pool P of *Psize* solution vectors is generated. To generate the mode list of the solution vectors, a controlled mode assignment procedure is designed. This procedure relies on the results of a computational experiment performed on 90 project instances containing 20 activities, 3 modes and 2 renewable and nonrenewable resources. For each project instance, 100 unique and feasible mode lists were generated, leading to 9,000 different combinations. The idea behind this research is that if a relationship between a mode list characteristic and the makespan of a project that results from that specific mode list could be found, the search space of the problem could be reduced to the most promising search regions.

In a first phase, the near-optimal project makespan is calculated with the bi-population genetic algorithm for the single-mode RCPSP of Debels and Vanhoucke (2005). In a second phase, a set of measures has been calculated to characterize the mode lists. The following characteristics have been analyzed:

- **Sum of Durations** (SOD), defined as $\sum_{i=1}^{n} d_{im_i}$. This characteristic is in line with the priority rule for the mode assignment proposed by Boctor (1993), who concluded that choosing the shortest feasible execution mode is the most appropriate rule to minimize the project duration.
- **Total Work Content** (TWC), defined as $\sum_{i=1}^{n}\sum_{k=1}^{|R^\rho|} r^{\rho}_{im_ik} d_{im_i}$. This characteristic assumes that mode lists with a lower total work content will result in lower makespans.
- **Mean relative consumption** (MRC), defined as $\frac{1}{|R^\rho|}\sum_{i=1}^{n}\sum_{k=1}^{|R^\rho|}\frac{r^{\rho}_{im_ik}}{a^{\rho}_{k}}$. This characteristic is based on the priority rule proposed by Heilmann (2001). It is assumed that mode lists with a lower relative resource consumption will result in lower makespans.

A statistical analysis investigated the relation between the mode lists characteristics and the project makespans obtained during the experiment. A Pearson correlation test revealed that the correlations for the SUD, TWC and MRC with the project makespan are 0.74, 0.61 and 0.40, respectively (all correlations are significant with $p < 0.01$). These results indicate that the significant and proportional relationship between the sum of all activity durations and the project makespan is the strongest. We therefore use this information during the generation of the initial mode lists population, leading to the controlled mode assignment procedure presented hereunder.

A *controlled mode assignment procedure* can be formulated in the three following steps:

1. A start population of mode lists, called *RANDPOP*, is created, with a large number of randomly generated feasible mode lists (in this paper: |*RANDPOP*| is equal to 4 times the number of populations elements *Psize*).
2. Each mode assignment list has a value for the sum of all activity durations, which is likely to lead to smaller project makespans.
3. The *Psize* mode lists with the lowest values for the sum of all activity durations are selected for entrance in the initial population.

Once the mode assignment lists are generated, a duration and resource consumption can be assigned to each activity for each population element. Since 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, infeasible solutions are accepted in the initial population, but are penalized with the penalty function of Alcaraz et al. (2003), which can be formulated as follows:

$$penalty = \begin{cases} C_{max} & \text{if feasible} \\ C_{max} + max\_feas\_C_{max} - CP^{min} + ERR & \text{otherwise} \end{cases}$$

where $max\_feas\_C_{max}$ gives the maximal makespan of the feasible schedules related to solution vectors of the current generation and CP^{min} is the critical path using the minimal duration of each activity. The excess of resource request *ERR* is defined as the number of requested nonrenewable resource units that exceeds the capacity a_l^{ν}, $l \in R^{\nu}$ (see Sect. 2).

Once the mode lists are generated, the activity lists are generated randomly, assigning a priority value to each activity.

4.1.3 Reference sets

After the generation of *Psize* population elements and the evaluation of the solution vectors with the serial schedule generation scheme (SGS), which translates the solution vector into a schedule, two diverse populations are constructed from P: a set B_1, with the b_1 best solutions of the solution set P and a set B_2, with b_2 diverse solutions. For the subset B_1, a threshold t_1 on the minimal distance between the elements is imposed in pursuit of diversity. The subset B_2 contains the b_2 best solutions from $P\backslash B_1$ that are sufficiently distant from the elements of B_1. The diversity in B_2 is achieved by a threshold t_2 on the smallest distance to any element in B_1 with $t_2 > t_1$. The distance between two solutions is a measure for diversity and is calculated according to the following two distance functions. The first distance function, $d^s_{p_1,p_2}$, calculates the distance as the sum of the differences between the start times of the activities and can be formulated as follows:

$$d^s_{p_1,p_2} = \sum_{i=1}^{|N|} |s_i^{p_2} - s_i^{p_1}|$$

with p_1 and p_2 two population elements and s^{p_1} and s^{p_2} their according start times. The second distance function, $d^m_{p_1,p_2}$, calculates the distance based on the difference in mode assignments and is formulated as follows:

$$d^m_{p_1,p_2} = \sum_{i=1}^{|N|} \begin{cases} 0 & \text{if } m_i^{p_1} = m_i^{p_2} \\ 1 & \text{otherwise} \end{cases}$$

In the computational results section both distance function are compared and analyzed.

If there are less solutions in B_2 than the predefined number b_2, the set B_2 is filled up with randomly generated schedules.

4.2 The subset generation method

After the initialization phase, a new pool of solutions is created by combining pairs of reference solutions in a controlled way. New solutions are created from all two-element subsets. First, all pairs in B_1 containing at least one new solution compared to the previous generation are considered. From each such pair, two children are produced. Second, from each combination of one element from B_1 and one from B_2 two offsprings are constructed. Choosing the two reference solutions out of the same cluster stimulates intensification, while choosing them from different clusters stimulates diversification.

4.3 The solution combination method

In the solution combination phase, the two selected population elements produce a new offspring which inherits parts of their parents characteristics. Several crossover operators were tested and tests revealed that the two-point crossover method clearly outperforms other crossover operators. In the *two-point crossover scheme*, two crossover points are randomly chosen and the characteristics between them are exchanged. As the procedure works on both the activity-list and the mode-list, the crossover considers random key values and modes simultaneously (i.e. using the same crossover points).

In Fig. 5, an example of a two-point crossover is presented. Two solution vectors (SV_1 and SV_2) are used to create a new solution vector SV_{cross}. Two crossover point p_1 and p_2 are chosen randomly and are equal to 2 and 7, which means that the first and the last 2 values are chosen out of solution vector SV_1 and the other values are selected from solution vector SV_2. The new solution vector SV_{cross} is presented below. Due to the use of random keys, each solution vector remains precedence feasible and no repair functions need to be used. In Fig. 6, the resulting feasible schedule is represented.

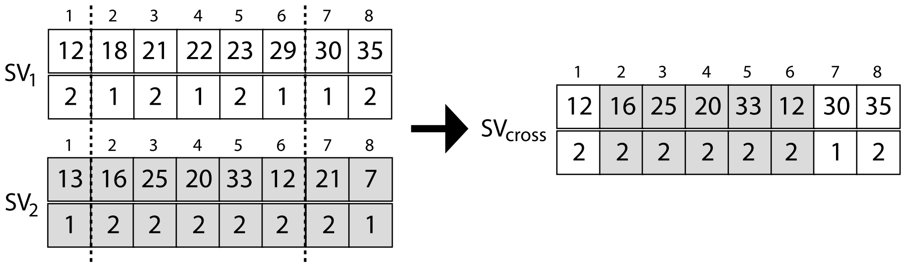
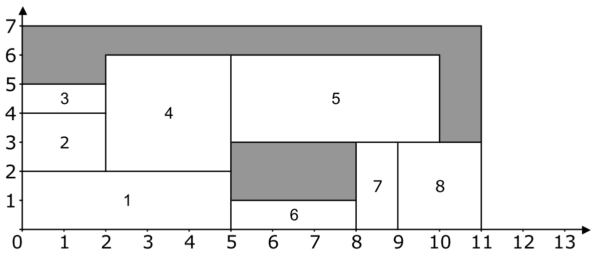


Fig. 5 Solution improvement methods

Fig. 6 Schedule of solution vector SV_{cross}



4.4 The improvement method

In this section we propose different solution improvement methods, which are applied on the solution vectors that are generated in the solution combination method. Every new solution vector consists of a new generated activity list and a new generated mode list. Since the mode list determines the duration and the renewable and nonrenewable resource requirements for each activity, quick tests can be used to indicate whether the solution vector has potential to improve the current best solution found so far, without actually using the schedule generation scheme. The two quick tests are:

Feasibility test This test is related to the **nonrenewable resources** and checks whether the mode assignment is feasible or not. If the test reveals an infeasible solution vector ($ERR > 0$), the *feasibility improvement method* is performed.

Lower bound This test is related to the **renewable resources** and checks whether a new generated mode assignment (i.e. a duration and renewable resources) can lead to an improvement in the total project makespan. If the critical sequence lower bound, proposed by Stinson et al. (1978), is larger than the best makespan found so far, two specific improvement methods are performed to improve the solution vector: the *critical path improvement method* tries to minimize the critical path length of the solution vector, while the *work content improvement method* tries to minimize the total work content of the proposed solution vector.

In case one of the these two tests is positive, one of the three improvement methods discussed below will be called in order to obtain a modified solution vector which will likely result in a decrease of the project makespan compared to the best known solution found so far. Consequently, each of the improvement methods will modify the mode assignment list of the solution vector in such a way to maximize the probability that these modifications lead to a better project makespan. Therefore, a probability $p_{(i,j)}$ is calculated in order to determine which activity/mode combinations will be the subject to a change. The activity/mode combinations with a higher $p_{(i,j)}$ value will have a higher priority to be modified. The probability $p_{(i,j)}$ is defined as follows:

$$p_{(i,j)} = \frac{\Delta_{i,j}}{\sum_{i=1}^{N} \sum_{j=1}^{|M_i|} \Delta_{i,j}} \tag{8}$$

with $\Delta_{i,j}$ the *improvement value* for each activity i/mode j combination. Changes are made until a stop criterion defined by the improvement method is met.

Feasibility improvement method (FIM) The purpose of this improvement method is to decrease the value of *ERR*. The improvement value $\Delta_{i,j}$ is formulated as follows:

$$\Delta_{i,j} = \max\{0, ERR_{old} - ERR_{new}\} \tag{9}$$

with ERR_{new} equal to the *ERR*-value based on the activity i/mode j combination, holding all the other modes equal. Obviously, the value of ERR_{new} is equal to the value of ERR_{old} if the current mode m_i of activity i is chosen. Once the mode assignment becomes feasible or no further improvements can be made, the feasibility improvement method is stopped.

Critical path improvement method (CPIM) The purpose of this improvement method is to minimize the critical path length of the solution vector. The improvement value $\Delta_{i,j}$ for this improvement method is calculated as follows:

$$\Delta_{i,j} = \max\{0, CP_{old} - CP_{new}\} \tag{10}$$

with CP_{new} the critical path based on the duration of the activity i/mode j combination and holding all the other modes equal. The improvement method stops when CP_{new} is smaller than the best found makespan or when no further improvements can be found.

Work content improvement method (WCIM) The purpose of this improvement method is to minimize the total work content of the proposed solution vector. The work content is calculated as the total required resources needed to execute the project, given the chosen mode per activity, i.e. $\sum_{i=1}^{n}\sum_{k=1}^{|R^\rho|} r^{\rho}_{im_ik} d_{im_i}$. The improvement value $\Delta_{i,j}$ for this improvement method is calculated as follows:

$$\Delta_{i,j} = \max\{0, WC_{old} - WC_{new}\} \tag{11}$$

with WC_{new} the needed work content based on the duration and resource demand of the selected mode, holding all other modes equal. The improvement method stops when the critical sequence lower bound is smaller than the best found makespan or when no further improvements can be found.

These improvement methods perfectly fit into the renewable and nonrenewable resource scarceness matrix presented in Fig. 2. Since the feasibility improvement method will try to solve nonrenewable resource infeasibilities expressed by positive ERR values, it will lie its focus on the third and fourth quadrants of the matrix. The focus of the critical path improvement method is to make changes in the critical path length, and hence lies its focus on the left part of the resource scarceness matrix. The work content improvement method puts a focus on the total work content of the activity/mode combinations, and consequently, will be fully exploited for project instances classified in the right part of the resource scarceness matrix. Figure 7 shows by means

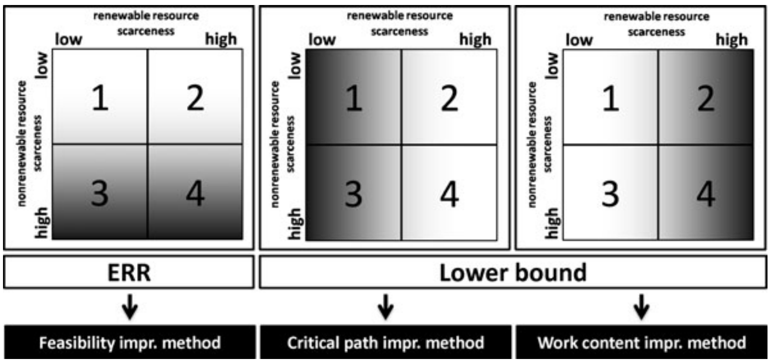

renewable resource scarceness
low
high
nonrenewable resource scarceness
low
high
1
2
3
4
ERR
Lower bound
Feasibility impr. method
Critical path impr. method
Work content impr. method

Fig. 7 Solution improvement methods

of the dark shaded areas what to focus of each improvement method is. The contribution of this approach on the solution quality will be tested in the computational results section.

4.5 The reference set update method

The population evolves over time with the entrance of new solution vectors and the removal of old solutions, searching to improve the quality of the best known solution. A new solution is introduced as a member in the reference set either if the solution vector has a better objective function value than the solution vector with the worst objective function value in B_1 or if the solution point is more diverse with respect to B_1 than the least diverse solution point in B_2.

4.6 Local searches

In Sect. 4.4, we have proposed different solution improvement methods. These methods were applied on the infeasible solution vectors, before they were actually scheduled, using the serial schedule generation scheme. However, different local searches were already proposed in literature which are applied on partially or fully scheduled projects. The local search of Józefowska et al. (2001) and Bouleimen and Lecocq (2003) search in the neighbourhood of a schedule by changing the activity list and mode list randomly. Kolisch and Drexl (1997) determine a probability function based on an approximation of the change in the objective function to determine which activity-mode pair will be changed. These local searches were coded and tested but revealed inferior results in the scatter search procedure with respect to the local search procedures of Hartmann (2001) and Van Peteghem and Vanhoucke (2010). In what follows, we explain both local search procedures briefly. Both local search procedures will be tested in Sect. 5.

Hartmann (2001) The local search of Hartmann (2001) is based on the multi-mode left shift of Sprecher (1994). 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 precedence and resource constraints. For each feasible schedule, the procedure checks for every activity whether a multi-mode left shift can be performed. For each activity, the first feasible multi-mode left shift is applied to the schedule. It is called a single-pass procedure, because every activity is considered only once for a multi-mode left shift.

Van Peteghem and Vanhoucke (2010) The local search of Van Peteghem and Vanhoucke (2010) selects an activity with a certain probability and evaluates during the generation of a schedule all feasible mode assignments of the selected activity. For each new mode assignment the *ERR* is calculated. If the *ERR* is equal to or smaller than the current one, the local search checks if an improvement can be made in the finish time of that activity. The mode with the lowest finish time, which does not increase the *ERR*, is chosen.

5 Computational results

In this section we configure the algorithm and evaluate its performance. The scatter search has been coded and compiled in Visual C++ 6.0 and used in the computational tests on a personal computer with a 1.75 GHz processor. In Sect. 5.1 we present two datasets which are generated for this research, while in Sect. 5.2, the stopping criterion is discussed. In Sect. 5.3, the Taguchi method is used to configure the algorithmic parameter settings. The influence of the improvement methods on the different quadrants in the resource scarceness matrix is tested in Sect. 5.4. The analysis of the local searches is presented in Sect. 5.5, while the introduction of an integrated solution procedure is presented in Sect. 5.6. Finally, in Sect. 5.7 the results of the scatter search algorithm are compared with the results of other existing procedures from the literature on the well-known PSPLIB benchmark dataset.

5.1 Dataset generation

5.1.1 Introduction

In this section two datasets are proposed, each containing a large set of data instances based on different complexity project parameters. A first dataset is used to configure the proposed scatter search algorithm, the other dataset is used to analyse the influence of the improvement methods and the local searches on the resource scarceness matrix.

In literature, two of the most used parameters to calculate the scarceness of the resources for single-mode projects are the *Resource Strength* (*RS*), introduced by Cooper (1976) and later on redefined by Alvarez-Valdes and Tamarit (1989) and Kolisch et al. (1995), and the *Resource Constrainedness* (*RC*), proposed by Patterson (1976). Since no formula is known for the *resource constrainedness* as a resource parameter for multi-mode resource-constrained projects, we will use in the remainder of this paper the resource strength as a parameter to calculate the scarceness of the renewable and nonrenewable resources. Kolisch et al. (1995) and Demeulemeester et al. (2003) defined this parameter for multi-mode projects as follows:

$$RS_k = \frac{a_k - r_k^{min}}{r_k^{max} - r_k^{min}} \tag{12}$$

where a_k denotes the total availability of renewable resource type k, r_k^{min} is formulated as $\max_{i=1,\ldots,n;m=1,\ldots,|M_i|} r_{ikm_i}$ and r_k^{max} denotes the peak demand of renewable resource type k in the precedence preserving earliest start schedule, where each activity has a duration which corresponds to a maximum allocation of resources (Demeulemeester et al. 2003). Kolisch et al. (1995) defined r_k^{min} as $\max_{i=1,\ldots,n}\{\min_{m=1,\ldots,M} r_{ikm}\}$. However, for low values of RS_k, the use of this definition will lead to different non-executable modes, which means that its execution would violate the renewable (or nonrenewable) resource constraints in any schedule (Sprecher 2000).

For the *nonrenewable resources* the minimum and maximum consumption can be obtained by cumulating the consumptions obtained when performing each activity in

Table 3 Parameter setting for the different datasets

Dataset 1		Dataset 2	
OS	0.25–0.50–0.75	*OS*	0.25–0.50–0.75
RS_R	0.25–0.50–0.75	RS_R	0 to 1 (0.10)
RS_{NR}	0.25–0.50–0.75	RS_{NR}	0 to 1 (0.10)
RF	0.50–1.00	*RF*	0.50–1.00
#	540	#	7,260

the mode having minimum and maximum consumptions. The resource strength *RS* varies between zero and one. A *RS* close to zero indicates that the scarceness of the resource is high, while a *RS* close to 1 implies that the resource is hardly restrictive.

5.1.2 *Parameter settings*

For the generation of the instances of both datasets, we have used the RanGen project scheduling instances generator developed by Vanhoucke et al. (2008) and extended the projects to a multi-mode version. Each instance contains 50 activities, with three modes, two renewable resource and two nonrenewable resources. Dataset 1 is used in Sect. 5.3 for the configuration of the algorithmic parameters, while dataset 2 is used in Sects. 5.4, 5.5 and 5.6 to analyze the performance of the improvement methods and local searches.

The following network and resource parameters were used for the two datasets. The values for each of these project characteristics are presented in Table 3.

1. The network complexity is described by the *order strength*, which is defined as the number of precedence relations (including the transitive ones but not including the arcs connecting the dummy start or end activity) divided by the theoretical maximum number of precedence relations (Mastor 1970). In both datasets, the *OS* is set at 0.25, 0.50 or 0.75.
2. In dataset 1, the resource strength is set at 0.25, 0.50 or 0.75, while in the other dataset, the parameter varies between 0 and 1 in steps of 0.10. The same parameter values are used for the nonrenewable resource strength in both datasets.
3. The *resource factor* (*RF*) indicates the percentage of resources that are required per activity and is set at 0.50 or 1.
4. For each problem class, 5 instances were generated. In the last row of Table 3, the *total number of instances* generated is shown.

5.2 Stopping criterion

Since it is assumed that the computational effort for constructing one schedule is similar in most heuristics and in order to make a fair comparison, the evaluation is stopped after a predefined number of generated schedules, in casu 5,000 schedules. According to Kolisch and Hartmann (2006), the advantage of the number of schedules as stop criterion is twofold: first, it is *platform independent* and second, future studies can easily make use of the *benchmark results* by applying the same stop criterion. However, the stop criterion also has a few shortcomings. First, it cannot be applied to all different heuristic strategies. Second, the required time to compute one schedule might differ between metaheuristics. Nevertheless, Kolisch and Hartmann (2006) conclude that limiting the number of schedules is the best criterion available for a broad comparison, which motivated us to use this stop criterion in all computational experiments.

To measure the number of schedules, the definition of one schedule should be defined. In their RCPSP review paper, Kolisch and Hartmann (2006) state that one schedule corresponds to (at most) one start time assignment per activity, as done by a SGS. However, measuring the number of schedules according to this rule means that for every mode change in a local search procedure a new schedule should be counted. Therefore, Lova et al. (2009) define the number of generated schedules 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. Assume a project with eight activities, each with three modes. Suppose that the SGS generates a schedule based on an activity and mode list (8 start times are assigned) and that a local search procedure has also analyzed the two other (feasible) modes for three of the activities. This means that the procedure has generated and analyzed $(8 + 2 \times 3)/8 = 1.75$ schedules. In the remainder of this paper, the last definition is used to define the number of schedules.

5.3 Impact of the algorithmic parameters

In order to determine a suitable set of parameters for our scatter search algorithm, the Taguchi method is used (Montgomery 2005). This method involves reducing the variation in the process through robust design of experiments (DOE). The DOE is carried out on dataset 1.

The algorithm contains the following seven key parameters: the values B, b_1 and b_2, which determine the total population size ($POP = B(b_1 + b_2)$), the value DF, which determines which distance function is used, the value IMG, which determines if the initial mode lists are generated randomly or by using the controlled mode assignment procedure, and the values v_1 and v_2, which determine the threshold values $t_1 = v_1|N|$ and $t_2 = v_2|N|$.

The values v_1 and v_2 depend on the chosen distance function. If the distance function $d^m_{p_1,p_2}$ is chosen, the values v_1 and v_2 indicate a percentage of the total number of activities for which the execution mode is equal. In the other situation, the values v_1 and v_2 indicate the average time per activity the start times of an activity in the two schedules may vary. Since there is a relationship between the value DF and the values v_1 and v_2, two Taguchi experiments are executed, one with the distance function $d^m_{p_1,p_2}$ and one with the distance function $d^s_{p_1,p_2}$.

Each experiment therefore contains 6 parameters (B, b_1, b_2, IMG, v_1 and v_2). According to the number of levels (2 for the parameter IMG and 5 for the others), the orthogonal array L25 is chosen. An overview of the different levels for each parameter is given in Table 4. The values v_1 and v_2 are divided by distance function.

Computational results clearly indicates that the experiment with the distance function d_s outperforms the experiment with the distance function d_m (a minimum deviation of 190.75% versus 191.62% and an average deviation 200.70% versus 201.15%).

Table 4 Different parameter values

Factor level	Parameters				$d^s_{p_1,p_2}$		$d^m_{p_1,p_2}$	
	B	b_1	b_2	IMG	v_1	v_2	v_1	v_2
1	5	8	3	0	0.25	1	0.1	0.3
2	10	10	5	1	0.50	2	0.2	0.4
3	15	12	7		0.75	3	0.3	0.5
4	20	15	9		1	4	0.4	0.6
5	25	20	11		1.5	5	0.5	0.7

Table 5 Response table with 5,000 schedules

Factor level	Parameters					
	B	b_1	b_2	IMG	v_1	v_2
1	197.33%	201.15%	201.05%	200.75%	203.11%	205.50%
2	**193.65%**	200.13%	199.87%	**200.62%**	201.07%	200.73%
3	195.28%	**198.16%**	202.91%		**198.58%**	**197.73%**
4	201.01%	200.27%	**199.45%**		201.57%	200.01%
5	216.23%	203.78%	200.22%		199.16%	199.53%
Δ	22.58%	5.62%	3.46%	1.13%	4.53%	7.78%

In Table 5, an overview is given of the average of the average deviation from the minimal critical path after 5,000 schedules for each of the different parameters and levels. According to the factor level trends, the best combinations of parameter values for the our algorithm are determined, as indicated in bold in Table 5.

5.4 Influence of the improvement method

In this section, the influence of the different improvement methods on the solution quality in the different quadrants of the resource scarceness matrix is tested. The results for the improvement methods applied on dataset 2 after 5,000 schedules and after 1 second are mentioned in Table 6. The sum of all the project makespans is presented in the column 'Sum', while the column 'Dev.CP' (%) contains the average deviation from the minimal critical path length. The column 'Dev.Best' contains the average deviation from the solutions of the best procedure, which will be presented later. Since the CPU-time needed to execute each of the improvement methods differs significantly (as can be seen in column 'CPU-time'), computational tests were performed with a fixed time limit of 1 second in order to make a fair comparison. However, similar conclusions can be drawn.

In this section, we will consider the first 5 results. The other results will be discussed in the following sections.

Figure 8 indicates which improvement method obtains the lowest average deviation from the minimal critical path for the project instances in a specific RS_R/RS_{NR}-combination. E.g. for the instances with a $RS_R = 80$ and a $RS_{NR} = 40$, the critical path

Table 6 Influence of the improvement methods and local searches

Result	Impr. method	Local Search	5,000 schedules Sum	Dev.CP (%)	Dev.Best (%)	CPU-time (s)	1 second Sum (%)	Dev.CP (%)	Dev.Best (%)
1	–	–	283,006	48.70%	3.92%	0.31	278,246	45.92%	3.16%
2	FIM	–	280,459	47.27%	2.97%	0.33	275,284	44,26%	2.07%
3	WCIM	–	280,311	47.17%	2.58%	0.67	278,437	45.98%	3.18%
4	CPIM	–	278,389	46.09%	1.62%	0.51	274,417	43.76%	1.22%
5	COMB	–	275,217	44.33%	0.95%	0.92	273,690	43.30%	1.01%
6	–	HART	276,689	45.10%	1.48%	0.25	271,616	42.06%	0.61%
7	–	VPV	275,455	44.43%	1.05%	0.29	271,342	42.03%	0.56%
8	–	COMB	274,998	44.19%	0.58%	0.26	270,718	41.70%	0.29%
9	COMB	COMB	273,047	43.09%	0.00%	0.73	270,232	41.37%	0.00%

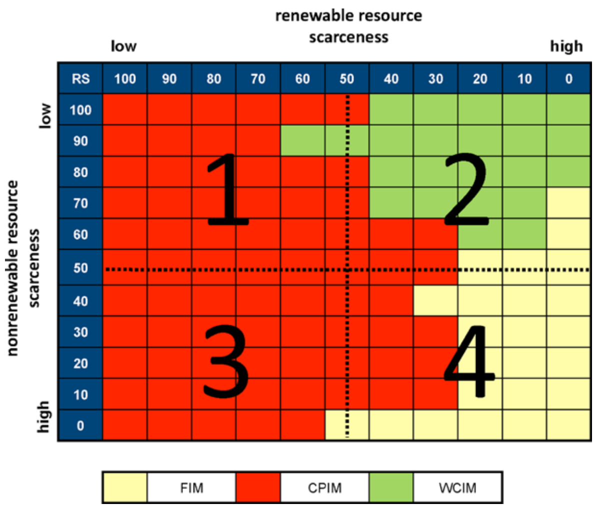


Fig. 8 Distribution of the most effective improvement methods over the resource scarceness matrix

improvement method outperforms the other two improvement methods. The results corresponds with the resource scarceness characteristics as mentioned in Sect. 4.4:

- The lower the renewable resource scarceness of a project instance is, the more effective the critical path improvement method is.
- The higher the renewable resource scarceness of a project instance is, the more effective the work content improvement method is.
- The higher the nonrenewable resource scarceness of a project instance is, the more effective the feasibility improvement method is.

Based on these findings, a combined improvement method can be proposed. For this combined improvement method, a probability is assigned to each of the improvement methods, based on the value V which is assigned according to the following formula:

$$V_{CPIM} = RS_R$$
$$V_{FIM} = 100 - RS_{NR}$$
$$V_{WCIM} = 100 - RS_R$$

with RS_R the value of the renewable resource strength and RS_{NR} the value of the nonrenewable resource strength. The probability P that an improvement method q ($q = CPIM$, FIM or $WCIM$) is chosen is equal to:

$$P_q = \frac{V_q}{V_{CPIM} + V_{FIM} + V_{WCIM}}$$

If the improvement is selected and finds an improvement in the solution vector, the improvement value V is increased such that the probability of selection in a next iteration is increased. In doing so, this probability selection method aims at selecting the improvement methods that fits best for the project under study given the renewable and nonrenewable resource characteristics as shown in the resource scarceness matrix. Table 6 shows that this method results in the best performing results found (result 5).

5.5 Introduction of local searches

Results 6 and 7 in Table 6 show the impact on the solution quality for the two local searches described in Sect. 4.6, i.e. the local search of Hartmann (*HART*) and the local search of Van Peteghem and Vanhoucke (*VPV*). The table shows that the local of search of Van Peteghem and Vanhoucke outperforms on average the solutions of Hartmann. However, looking to the specific $RS_{NR} - RS_R$-combinations in the resource scarceness matrix, a distinction can be made: in (parts of) quadrants 2 and 3 the local search of Hartmann (*HART*) outperforms the local search of Van Peteghem and Vanhoucke (*VPV*), while in the other two quadrants, the latter seems more effective than the former (see Fig. 9). Based on these findings, a controlled local search procedure is proposed (result 8), where in the specific regions of quadrant 2 and 3 the local search of Hartmann is used, while in the other situations, the local search of Van Peteghem and Vanhoucke is performed.

5.6 An integrated solution procedure for the MRCPSP

An integrated procedure is developed by integrating the combined local search and the combined improvement method in our scatter search algorithm. This integrated procedure follows the different steps as indicated in Fig. 3. Once a solution vector is generated in the solution combination method, the integrated procedure is applied, along the following steps:

1. Based on the probability P (as defined in Sect. 5.4), one of the three improvement methods is applied on the solution vector.

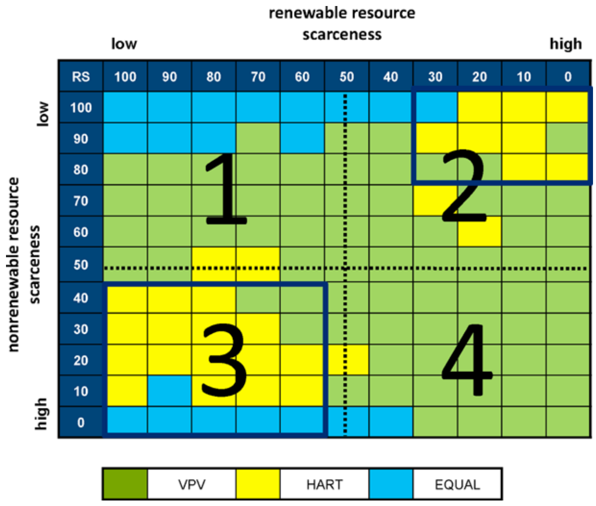

renewable resource scarceness
low
high
nonrenewable resource scarceness
low
high
RS
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
VPV
HART
EQUAL

Fig. 9 Influence of the local searches on the resource scarceness matrix

2. The solution vector is evaluated. Depending on the values of RS_R and RS_{NR}, the local search of Hartmann (if $RS_R < 40$ and $RS_{NR} > 70$ or if $RS_R > 50$ and $RS_{NR} < 50$) or the local search of Van Peteghem and Vanhoucke (otherwise) is applied.
3. The solution vector is added to the population.

Once all the solution vectors of the subsets are combined, the population is updated, new subsets are generated and new solution vectors are generated with the solution combination method. On each solution vector, the integrated procedure is applied. This procedure continues until the stopping criterion is met.

The results for the overall procedure are mentioned in Table 6 (result 9). In the following section, tests will be performed on the PSPLIB dataset using this integrated solution procedure.

5.7 Comparison with other heuristics

In this section we compare the scatter search algorithm with other procedures available in literature. We test the algorithm on the well-known PSPLIB benchmark dataset of Kolisch and Sprecher (1996). This 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 known. For each problem size, 640 instances were generated. Due to infeasibility of some instances, not all these instances could be solved.

In Table 7 an overview of the average deviation from the optimal solution on the J10 to J20-datasets for different algorithms and based on 5,000 schedules is presented. As can be seen in both tables, the results for our algorithm outperforms the

Table 7 Comparison with other heuristics—average % deviation from optimum—5,000 schedules

	Instances set					
	J10	J12	J14	J16	J18	J20
Józefowska 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
Ranjbar et al. (2009)	0.18	0.65	0.89	0.95	1.21	1.64
Lova et al. (2006)	0.06	0.17	0.32	0.44	0.63	0.87
Tseng and Chen (2009)	0.33	0.52	0.91	1.08	1.30	1.71
Van Peteghem and Vanhoucke (2010)	0.01	0.09	0.22	0.32	0.42	0.57
This work	**0.00**	**0.02**	**0.08**	**0.15**	**0.23**	**0.32**
Optimal	100%	99.45%	97.28%	96.73%	94.75%	87.73%
CPU-time (ms)	95	105	118	130	143	155

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

	1,000 schedules	5,000 schedules	50,000 schedules
	Av. % Dev. CP	Av. % Dev. CP	Av. % Dev. CP
Lova et al. (2009)	16.65%	14.77%	–
Tseng and Chen (2009)	–	18.33%	15.68%
Van Peteghem and Vanhoucke (2010)	15.30%	13.75%	13.31%
This work	15.02%	13.66%	12.72%

other heuristics. The percentage of optimal solutions found and the computation time (in ms) for each of the datasets for 5,000 schedules is presented in the bottom two rows.

For the J30-dataset only the best known solutions (*BKS*) are available. Since comparison of the deviations from the *BKS* is not desirable (due to the change over time in the values of the *BKS*), only the percentage increase of the project duration above the minimal critical path length can provide a fair comparison between different heuristic procedures. Therefore, these deviations for the results after 1,000, 5,000 and 50,000 schedules computed compared to two other solution procedures are mentioned in Table 8 and indicate the strong performance of our algorithm. For one instance, an improvement in the best known solution was found.

6 Conclusions

In this paper we have designed a promising scatter search procedure for the well-known multi-mode resource-constrained project scheduling problem. The added value of this algorithm lies in the steering power of three proposed improvement methods, each tailored to the specific characteristics of different renewable and non-renewable resource scarceness values. Computational results confirm the influence of these improvement methods on projects situated in one of the quadrants of our resource scarceness matrix. The combination of these improvement methods and the introduction of two local searches into one overall solution procedure leads to promising computational results.

Further research efforts will focus on both the extension of the multi-mode project scheduling problem by taking other criteria into account, like the maximisation of the net present value and introduction of activity splitting, and on the applicability of these solution improvement methods on the influence of other network or resource parameters on the search strategy of solution procedures.

References

Alcaraz, J., Maroto, C., Ruiz, R.: Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. J. Oper. Res. Soc. **54**, 614–626 (2003)

Alvarez-Valdes, R., Tamarit, J.: Heuristic algorithms for resource-constrained project scheduling: A review and empirical analysis. In: Slowinski, R., Weglarz, J. (eds.) Advances in Project Scheduling. Elsevier, Amsterdam (1989)

Blazewicz, J., Lenstra, J., Rinnooy Kan, A.: Scheduling subject to resource constraints: Classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)

Boctor, F.: Heuristics for scheduling projects with resource restrictions and several resource-duration modes. Int. J. Prod. Res. **31**, 2547–2558 (1993)

Boctor, F.: A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. Eur. J. Oper. Res. **90**, 349–361 (1996a)

Boctor, F.: Resource-constrained project scheduling by simulated annealing. Int. J. Prod. Res. **34**, 2335–2351 (1996b)

Bouleimen, K., Lecocq, H.: A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. Eur. J. Oper. Res. **149**, 268–281 (2003)

Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: notation, classification, models, and methods. Eur. J. Oper. Res. **112**, 3–41 (1999)

Buddhakulsomsiri, J., Kim, D.: Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. Eur. J. Oper. Res. **178**, 374–390 (2007)

Cooper, D.: Heuristics for scheduling resource-constrained projects: an experimental investigation. Manag. Sci. **22**, 1186–1194 (1976)

Debels, D., Vanhoucke, M.: A bi-population based genetic algorithm for the RCPSP. Lect. Notes Comput. Sci. **3483**, 378–387 (2005)

Demeulemeester, E., Vanhoucke, M., Herroelen, W.: A random network generator for activity-on-the-node networks. J. Sched. **6**, 13–34 (2003)

Drexl, A., Grünewald, J.: Nonpreemptive multi-mode resource-constrained project scheduling. IIE Trans. **25**, 74–81 (1993)

Glover, F., Laguna, M., Marti, R.: Fundamentals of scatter search and path relinking. Control Cybern. **29**, 653–684 (2000)

Hartmann, S.: A competitive genetic algorithm for resource-constrained project scheduling. Nav. Res. Logist. **45**, 733–750 (1998)

Hartmann, S.: Project scheduling with multiple modes: A genetic algorithm. Ann. Oper. Res. **102**, 111–135 (2001)

Heilmann, R.: Resource-constrained project scheduling: A heuristic for the multi-mode case. OR Spektrum **23**, 335–357 (2001)

Herroelen, W., De Reyck, B.: Phase transitions in project scheduling. J. Oper. Res. Soc. **50**, 148–156 (1999)

Herroelen, W., Demeulemeester, E., De Reyck, B.: A classification scheme for project scheduling problem. In: Weglarz, J. (ed.) Project Scheduling—Recent Models, Algorithms and Applications, pp. 1–26. Kluwer Academic, Dortrecht (1999)

Jarboui, B., Damak, N., Siarry, P., Rebai, A.: A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Appl. Math. Comput. **195**, 299–308 (2008)

Józefowska, J., Mika, M., Rózycki, R., Waligóra, G., Weglarz, J.: Simulated annealing for multi-mode resource-constrained project scheduling. Ann. Oper. Res. **102**, 137–155 (2001)

Kelley, J.: The Critical-path Method: Resources Planning and Scheduling. Prentice-Hall, New Jersey (1963)

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

Kolisch, R., Drexl, A.: Local search for nonpreemptive multi-mode resource-constrained project scheduling. IIE Trans. **29**, 987–999 (1997)

Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. Eur. J. Oper. Res. **174**, 23–37 (2006)

Kolisch, R., Sprecher, A.: PSPLIB—a project scheduling problem library. Eur. J. Oper. Res. **96**, 205–216 (1996)

Kolisch, R., Sprecher, A., Drexl, A.: Characterization and generation of a general class of resource-constrained project scheduling problems. Manag. Sci. **41**, 1693–1703 (1995)

Lova, A., Tormos, P., Barber, F.: Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules. Intel. Artif. **30**, 69–86 (2006)

Lova, A., Tormos, P., Cervantes, M., Barber, F.: An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. Int. J. Prod. Econ. **117**, 302–316 (2009)

Marti, R., Laguna, M., Glover, F.: Principles of scatter search. Eur. J. Oper. Res. **169**, 359–372 (2006)

Mastor, A.: An experimental and comparative evaluation of production line balancing techniques. Manag. Sci. **16**, 728–746 (1970)

Montgomery, D.C.: Design and Analysis of Experiments. Wiley, New York (2005)

Mori, M., Tseng, C.: A genetic algorithm for the multi-mode resource constrained project scheduling problem. Eur. J. Oper. Res. **100**, 134–141 (1997)

Nonobe, K., Ibaraki, T.: Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Ribeiro, C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics. Kluwer Academic, Dordrecht (2002)

Özdamar, L.: A genetic algorithm approach to a general category project scheduling problem. IEEE Trans. Syst. Man Cybern. **29**, 44–59 (1999)

Özdamar, L., Ulusoy, G.: A local constraint based analysis approach to project scheduling under general resource constraints. Eur. J. Oper. Res. **79**, 287–298 (1994)

Patterson, J.: Project scheduling: The effects of problem structure on heuristic scheduling. Nav. Res. Logist. **23**, 95–123 (1976)

Pinol, H., Beasley, J.: Scatter search and bionomic algorithms for the aircraft landing problem. Eur. J. Oper. Res. **171**, 439–462 (2006)

Ranjbar, M., De Reyck, B., Kianfar, F.: A hybrid scatter-search for the discrete time/resource trade-off problem in project scheduling. Eur. J. Oper. Res. **193**, 35–48 (2009)

Slowinski, R., Soniewicki, B., Weglarz, J.: DSS for multi-objective project scheduling subject to multiple-category resource constraints. Eur. J. Oper. Res. **79**, 220–229 (1994)

Sprecher, A.: Resource-constrained Project Scheduling: Exact Methods for the Multi-mode Case. Lecture Notes in Economics and Mathematical Systems. Springer, Berlin (1994)

Sprecher, A.: Scheduling resource-constrained projects competitively at modest memory requirements. Manag. Sci. **46**, 710–723 (2000)

Sprecher, A., Drexl, A.: Multi-mode resource-constrained project scheduling with a simple, general and powerful sequencing algorithm. Eur. J. Oper. Res. **107**, 431–450 (1998)

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

Stinson, J., Davis, E., Khumawala, B.: Multiple resource-constrained scheduling using branch-and-bound. IIE Trans. **10**, 252–259 (1978)

Tseng, L.Y., Chen, S.C.: Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem. IEEE Trans. Evol. Comput. **13**, 848–857 (2009)

Van Peteghem, V., Vanhoucke, M.: A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problems. Eur. J. Oper. Res. **201**, 409–418 (2010)

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., Tavares, L.: An evaluation of the adequacy of project network generators with systematically sampled networks. Eur. J. Oper. Res. **187**, 511–524 (2008)

Zhang, H., Tam, C., Li, H.: Multi-mode project scheduling based on particle swarm optimization. Comput.-Aided Civ. Infrastruct. Eng. **21**, 93–103 (2006)

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