Expert Systems with Applications 39 (2012) 3983–3994

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm

D.C. Paraskevopoulos *, C.D. Tarantilis, G. Ioannou

Management Science Laboratory, Department of Management Science & Technology, Athens University of Economics & Business, Greece

ARTICLE INFO

Keywords:
Project scheduling
Resource constraints
Evolutionary algorithms
Iterated Local Search

ABSTRACT

There are various scheduling problems with resource limitations and constraints in the literature that can be modeled as variations of the Resource Constrained Project Scheduling Problem (RCPSP). This paper proposes a new solution representation and an evolutionary algorithm for solving the RCPSP. The representation scheme is based on an ordered list of events, that are sets of activities that start (or finish) at the same time. The proposed solution methodology, namely SAILS, operates on the event list and relies on a scatter search framework. The latter incorporates an Adaptive Iterated Local Search (AILS), as an improvement method, and integrates an event-list based solution combination method. AILS utilizes new enriched neighborhoods, guides the search via a long term memory and applies an efficient perturbation strategy. Computational results on benchmark instances of the literature indicate that both AILS and SAILS produce consistently high quality solutions, while the best results are derived for most problem data sets.

1. Introduction

Project scheduling addresses the sequencing of a set of activities, performed by a set of resources, and aims at the optimization of a single or multiple relevant objectives. There are various scheduling problems studied in literature (Hsu, de Blois, & Pyle, 2004; Zhou & Zhong, 2007) that are hard combinatorial optimization problems and can be modeled as variations of the Resource-Constrained Project Scheduling Problem (RCPSP) addressed in this paper. The RCPSP can be described as follows. Let a single project with a set $J = \{0, 1, \ldots, n, n+1\}$ of N activities, which have to be executed using k renewable resources in set $K = \{1, \ldots, k\}$. Precedence constraints force activity j to start not before all its immediate predecessor activities have finished. While being processed, activity j requires $r_{j,k}$ units of resource type k during its non-preemptable duration d_j. The term "non-preemptable duration" means that the execution of an activity cannot be interrupted. Resource k has a limited capacity q_k at any point in time, and parameters d_j, r_{jk}, and q_k are deterministic for all $k \in K$. The objective of the RCPSP is to find precedence and resource feasible completion times for all activities, such that the completion time of the project is minimized.

Due to its wide applicability and high complexity (Blazewicz, Lenstra, & Rinnooy Kan, 1983), the RCPSP has attracted substantial research efforts on the development of both exact and heuristic solution methods. Nevertheless, heuristics remain the only reliable methods for producing high quality solutions for large scale instances. The research focus is thus on the design of efficient metaheuristics, including local search methodologies, evolutionary (or population-based) approaches as well as hybrids, in an effort to combine the strengths and eliminate the weaknesses of the previous two groups of methods.

A critical factor, for the successful design of a metaheuristic for solving the RCPSP, is the solution representation (Kolisch & Hartmann, 2006). Typically, the more accurately the representation scheme depicts the RCPSP's solution, the more efficient it is. This paper proposes a new solution representation and an evolutionary algorithm to tackle the RCPSP. The proposed representation, is a list of events, that are sets of activities that start (or finish) at the same time, ordered according to their starting (or finishing) times. The event-list's structure prevents two or more representations from being associated with a single solution, and thus, it is particularly efficient.

The proposed evolutionary algorithm, namely SAILS, relies on a scatter search framework and consists of an initial phase and a scatter search phase. At the initial phase, a construction heuristic is used to produce distant solutions, that constitute the reference set of solutions, in an effort to spread the search over different regions of the solution space. At the scatter search phase an efficient solution combination method is introduced and an Adaptive Iterated Local Search (AILS) is presented as an improvement method. AILS incorporates new enriched neighborhoods, utilizes a short

* Corresponding author. Address: Evelpidon 47A & Lefkados 33, GR11362 Athens, Greece. Tel.: +30 2108203862; fax: +30 2108828078.
E-mail addresses: dparaskevop@aueb.gr (D.C. Paraskevopoulos), tarantil@aueb.gr (C.D. Tarantilis), ioannou@aueb.gr (G. Ioannou).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.09.062

and long term guidance of the search and applies an efficient perturbation strategy to allow the search to overcome local optima. The focus is on the simplicity and the effectiveness of the algorithm in order to support its integration into the scatter search phase; thus, few parameters are introduced, while a neighborhood evaluation mechanism is presented. Furthermore, the reference set of solutions is evolved by employing an efficient solution combination that is in line with scatter search principles and relies on the proposed event-list representation. Numerical experiments on well known benchmarks of literature showed that both AILS and SAILS produce consistently high quality solutions.

The remainder of this paper is organized as follows. Section 2 provides an overview of the latest developments in the field. Section 3 introduces the proposed solution representation, while Section 4 presents the evolutionary algorithm and all of its components. Computational results are given in Section 5. Finally, the paper concludes in Section 6, where pointers for further research are also presented.

2. Literature review

Literature has shown that the most of the local search methods, that have been successfully applied for solving hard combinatorial optimization problems (Bräysy, Porkka, Dullaert, Repoussis, & Tarantilis, 2009; Tarantilis, Zachariadis, & Kiranoudis, 2008), are not competitive when applied for solving the RCPSP, at least by using the existing solution representation schemes (Kolisch & Hartmann, 2006). In this section we review the related literature regarding local search methods, evolutionary approaches and hybrid metaheuristics for the RCPSP.

Thomas and Salhi (1998) were among the first who proposed a Tabu Search (TS) algorithm for solving the RCPSP. Their solution framework utilizes two different neighborhood structures, namely swap and insert, while resource infeasible solutions are considered. In a similar manner, Nonobe and Ibaraki (2002) developed a TS approach for a generalized variant of RCPSP. The proposed TS operates on the Activity List (AL) representation and uses specific rules for defining the neighborhood. The most recent TS for the RCPSP is proposed by Artigues, Michelon, and Reusser (2003). The algorithm incorporates various insertion rules to define multiple neighborhood structures.

Beyond Tabu Search methodologies, other local search methods have been also applied for solving the RCPSP. Bouleimen and Lecocq (2003) proposed a Simulated Annealing method to the RCPSP and its multiple mode version (MRCPSP). Their solution framework operates on the AL representation and performs shifts of activities. Fleszar and Hindi (2004) were the first to apply a variable neighborhood search method for solving the RCPSP. An enhanced neighborhood operator, that moves activities along with their predecessors or successors throughout the AL representation, is employed. Moreover, particular lower bound calculations were used to speed up the search. Pesek, Schaerf, and Zerovnik (2007) also proposed a local search algorithm that uses several enhanced neighborhood structures defined by removing and reinserting a fixed number of activities on the AL representation.

A sophisticated local search for the RCPSP was proposed by Ranjbar (2008). The author developed a filter and fan algorithm that operates on an AL representation and has two main components; the local search and the filter and fan strategy. The first utilizes local move operators to define the solution neighborhood. The second can be viewed as a beam search with multiple paths following a breadth search strategy. It actually uses a list of local optima (obtained during the local search phase) and evaluates a large solution neighborhood composited by the neighboring solutions of all local optima in the list. This tree search is terminated and a local search phase starts again, when an improvement in the current solution is conducted.

To sum up, there are few local search methods for solving the RCPSP (Tabu Search, Simulated Annealing and Variable Neighborhood Search) while the dominant representation scheme is the Activity List representation. Given that the most of these methods are not competitive enough, when compared to other metaheuristics for solving RCPSP, at least by using the existing representation schemes (Kolisch & Hartmann, 2006), the development of a highly competitive local search algorithm for the RCPSP is a worth-pursing research direction.

Considering population-based methods, Hartmann (2002) was among the first to apply such a methodology for the RCPSP. The author proposed a self adaptive Genetic Algorithm (GA) that operates on the AL representation. The algorithm utilizes both the serial and the parallel Schedule Generation Scheme (SGS) to generate active and non delay schedules, respectively. Even though parallel SGS produces non delay schedules, which do not always include the optimum, it utilizes the resources as early as possible, thus, generating compact schedules. Merkle, Middendorf, and Schmeck (2002) were the first to apply an Ant Colony Optimization (ACO) algorithm to the RCPSP, using the AL representation. Their ACO uses two pheromone evaluation methods and a self adaptive mechanism for the ants' decisions.

The first hybrid approach to the RCPSP was presented by Valls, Quintanilla, and Ballestin (2003). The proposed method operates on the Topological Order Random Key (TO-RK) representation (a special case of the RK representation) and integrates a TS into a population based algorithm. The solution neighborhood is defined by two types of operators that use the critical activities and a third operator that applies random sampling within a time window of the current solution. Similarly, Kochetov and Stolyar (2003) developed a GA-Path Relinking methodology that incorporates a TS algorithm with variable neighborhood. A hybrid methodology, based on an adaptive large neighborhood search, was developed by Palpant, Artigues, and Michelon (2004). At each iteration of the algorithm, a part of the current solution is fixed and extracted. The remaining part is solved separately by a heuristic rule or constraint programming or mathematical programming techniques, and an assembly of the subsolutions follows.

Valls, Ballestin, and Quintanilla (2004) developed a population-based solution approach to the RCPSP. The authors used a blend of scatter search and path relinking strategies to evolve the initial population of solutions, while employing a forward and backward scheduling procedure to improve the resource utilization. The algorithm operates on a topological order activity list.

Tseng and Chen (2006) presented a hybrid metaheuristic (ANGEL) for solving RCPSP, which hybridizes ACO GA and a local search method. ACO is used to generate initial solutions which are subject for further improvement through GA. Every time a better solution is obtained the pheromone set in ACO is updated. After GA termination, ACO searches again by using the new pheromone set. The local search is employed for intensification purposes.

A sophisticated way for the solution recombination into a hybrid Scatter Search (SS) methodology, was proposed by Debels, Reyck, Leus, and Vanhoucke (2006). The authors developed a hybrid metaheuristic by combining an SS algorithm and the principles of electromagnetism, for solution selection purposes. A specialized representation was proposed and particular intensification procedures were developed. Moreover, Debels and Vanhoucke (2007) proposed a successful decomposition-based GA for solving RCPSP. In particular, the proposed algorithm splits RCPSP into subproblems and solve these via a GA. An assembly of the subsolutions takes part in the solution framework, as a next step, and an effort is given for further cost improvement.

Valls, Ballestin, and Quintanilla (2008) proposed a hybrid genetic and local search algorithm that operates on the AL representation. A specialized crossover operator and a local improvement operator are presented. In a similar manner, Mendes, Goncalves, and Resende (2009) developed a GA that operates on a RK representation. The algorithm generates parameterized active schedules, while an augmented objective function is proposed.

Recently, Ranjbar and Kianfar (2009) developed a hybrid SS method with Forward Backward Improvement (FBI) local search strategy. FBI performs forward and backward shifts of activities in order to improve the makespan. The topological ordered AL representation is used, both for the forward and backward types of schedules. Similarly, Mahdi Mobini, Rabbani, Amalnik, Razmi, and Rahimi-Vahed (2009) proposed an enhanced scatter search algorithm for solving the RCPSP. Their algorithm operates on the AL representation and employs FBI strategy as an improvement method. Moreover, the solution combination method incorporates a two point cross-over operator, a path relinking strategy and a permutation-based operator.

A different hybrid metaheuristic approach was proposed by Agarwal, Colak, and Erenguc (2011). The authors developed a neurogenetic algorithm for solving RCPSP that operates on an AL representation. Their algorithm hybridizes a population based strategy that relies on GA and a local search algorithm based on neural networks.

Among the latest developments in the field are the papers of Chen, Wub, Wang, and Lo (2010) Chen, Shi, Teng, Lan, and Hu (2010). The first article proposed a Particle Swam Optimization (PSO) methodology for solving RCPSP. The PSO follows the principles of evolutionary strategies and emulates the swarm behaviors of birds flocking. Two scheduling rules (delay local search rule and bidirectional scheduling rule) are proposed. Critical path calculations are used to speed up the procedure of evaluating the makespan of a solution. Simulation results indicate the efficiency of the proposed algorithm in producing high quality solutions. To the other end, Chen et al. (2010) proposed a powerful hybrid metaheuristic, namely ACOSS, that uses the AL representation and combines ACO, SS and a local search method. In a similar manner to the paper of Tseng and Chen (2006), ACO is used for the generation of the initial solutions which are subject for further improvement through SS. The pheromone is updated and the local search is called every time a better solution is found. After SS termination, ACO is re-executed using the updated pheromones.

Finally, Wu, Wan, Shukla, and Li (2011) developed an improved immune algorithm (CBIIA) for solving RCPSP. The immune algorithm belongs to population based metaheuristics and emulates the immune system of living organisms. CBIIA operates on a RK representation, the random numbers of which are obtained by a chaotic generator, and incorporates an innovative hypermutation mechanism. The efficiency of the proposed algorithm is tested through numerical experiments on the benchmark instances of Patterson (1984).

In conclusion, population-based methods as well as hybrid methodologies seem to produce the best results when applied to the RCPSP. Various strategies have been applied, including Genetic Algorithms, Scatter Search frameworks, Ant Colony Optimization and Particle Swam Optimization. Particular mechanisms are incorporated for local search, such as Forward and Backward Improvement, local shifts and activity swaps. Considering the representation schemes, the Activity List representation, the Random Key representation as well as adaptations of these two have been widely utilized for depicting RCPSP's solutions. In this paper, both a new representation scheme, that is based on events instead of activities and random keys, and an evolutionary algorithm are presented.

3. Event-list representation

The efficiency of the representation scheme is expected to be a determinant factor for the performance of a heuristic approach, since the latter operates on the representation of the problem and not directly on the schedule. The dominant representation schemes of literature is the Activity List (AL) and the Random Key (RK) representation (Kolisch & Hartmann, 2006). However, there are various adaptations of these two representations (Debels et al., 2006; Ranjbar & Kianfar, 2009; Valls et al., 2003).

The AL representation is a vector of size N, whose elements depict activities. The order of the elements-activities corresponds to the order that these activities are scheduled using a Schedule Generation Scheme (SGS). To the other end, in the RK representation, a solution corresponds to a point in Euclidian $(N+1)$-space, such that the ith vector element functions as a scheduling priority value for the ith activity. For example, activities of higher priority values will be scheduled first, when using an SGS (Kolisch & Hartmann, 1999). SGS constructs an active schedule by scheduling each activity one-at-a-time and as early as possible (or as late as possible), according to the sequence defined by the incumbent representation scheme. Hartmann and Kolisch (2000) conclude that AL seems to be more efficient than the RK representation, based on computational experiments.

Both AL and RK schemes associate more than one representation with a single schedule. Consequently, the solution space is filled with useless representations that are replicates of particular solutions, and thus, the search efficiency is typically reduced. For example, a local search move may result in the same solution, despite the fact that there have been changes into the representation scheme. The structural inefficiencies of the dominant representations schemes of the literature are summarized as follows:

1. Two different RK schemes may result in the same solution, if the first scheme is a linear combination of the second and vice versa.
2. Precedence constraints are not considered in the RK.
3. If two activities have the same starting time, the positions (for AL) and the priority values (for RK) can be interchanged but the solution will remain the same.
4. Due to precedence constraints, the starting time of an activity remains the same, no matter what is the priority value (for RK) and the list position (for AL).

There have been important achievements in literature that address the aforementioned issues. In particular, Debels et al. (2006) proposed a transformation mechanism that addresses the above deficiencies in a specialized RK representation scheme. Initially, the authors applied a scaling of the Euclidean space to address the first issue and they used a repairing mechanism that considers the precedence constraints for the second. Moreover, the authors assigned the same priority values for the activities that have the same starting times to address the third issue and they finally used a topological order of activities to tackle the last issue.

The new representation scheme, proposed herein, does not use any adjustments and repairing mechanisms and addresses effectively the aforementioned inefficiency issues. Two different solutions can be distinguished by the different events that they comprise, both in terms of events' composition and dates (i.e., start or finish times). The event list's structure thus prevents two different representations from resulting in the same schedule. Moreover, it has been developed to allow local moves of sets of activities, enabling local search methodologies to generate new enriched solution neighborhoods.

Furthermore, the event list empowers evolutionary strategies, since it enables effective inheritance that allows the solution

recombination procedure to overcome identical offspring or mutants. AL-based recombination strategies use activity chains as the solution elements to be combined. This frequently leads to identical offspring, since time information is typically neglected. Similar are the results, when the RK representation is used. On the contrary, when using the event based representation proposed herein, the offspring inherit events from the parent solutions, all information about the starting and finish times is considered, the desired solution properties are properly delivered to the next generations, and thus, an efficient evolutionary process is performed.

The above described event aspect has been also utilized in a similar fashion for deriving alternative RCPSP's mathematical formulations (see Kone, Artigues, Lopez, & Mongeau, 2011; Mingozzi, Maniezzo, Ricciardelli, & Bianco, 1998). In this paper, two different lists of events are proposed: the first is generated by sorting the events, that are created by the serial forward scheduling scheme, according to the activities' starting times, while the second by sorting the events that are created by the serial backward scheduling scheme, according to their finish times. The serial bidirectional scheduling scheme is considered as well, but the resulting schedule is linked to the backward or the forward list. In particular, if the last activities of the bidirectional scheme are scheduled as early as possible the resulting schedule is linked to the forward list. On the contrary, if the first activities of the bidirectional scheme are scheduled as late as possible the resulting schedule is linked to the backward list. Note that, only active schedules are considered in this paper constructed by the serial schedule generation scheme.

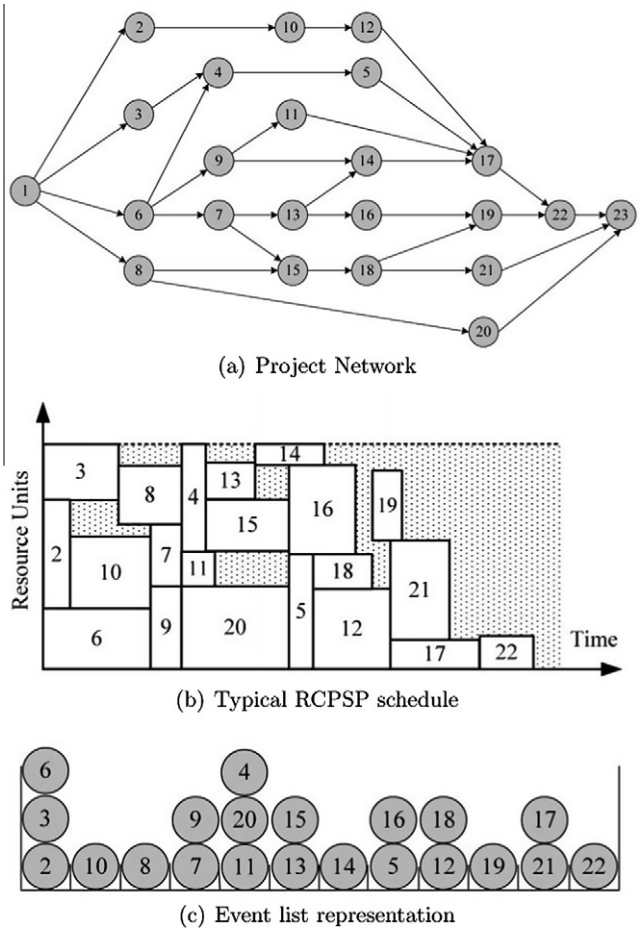

Fig. 1. A typical Resource Constrained Project Scheduling Problem.

Fig. 1 shows a typical project, a feasible solution and its representation by the proposed event list. The sample project consists of 21 activities and uses a single renewable resource type with limited capacity. For the sake of simplicity, the activities' resource consumption and duration are not provided, but shown in Fig. 1b. The precedence constraints are represented by the network arcs, i.e., activity 13 cannot start before 7 has finished. In Fig. 1b a feasible solution of the problem is given, while Fig. 1c depicts the event list of this particular solution. In such a representation, activities with the same starting times are grouped together and the resulting groups are ordered by their starting times. It is worth noting that the feasible solution has been generated by utilizing the serial SGS with a forward planning. Similar are the figures when backward planning is applied. In that case, the events are grouped by the finish time of each activity and they are ordered from the end to the start of the scheduling horizon.

4. Solution methodology

The proposed solution methodology is based on a scatter search framework (Glover, 1977) that operates on the proposed event-list representation and integrates an Adaptive Iterated Local Search (AILS) as an improvement method. In particular, the proposed method consists of an initial phase and a scatter search phase. At the initial phase, a reference set of μ solutions is created by utilizing a construction heuristic that picks at random a precedence feasible activity and schedules it as early as possible. The focus is on the balance between solution quality and diversity. At the scatter search phase, the reference set of solutions is evolved via an efficient solution combination method, while AILS is employed to improve the solution quality. Particular criteria are proposed for the update of the reference set. This framework oscillates by recombining the updated reference set and the procedure terminates when a maximum number of schedules examined is met. In the following, the components of the proposed solution methodology are presented.

4.1. Fitness function

A large part of RCPSP's solution space comprises solutions with the same makespan (Czogalla & Finik, 2009). The question that comes up is: "Between two different solutions with the same makespan, which is more attractive and favorable?". In fact, both solutions are of the same value and accepted if they are optima. If not, the algorithm must pick the one that will aid the search to converge faster in a higher quality region of the solution space. To tackle this fitness function landscape's peculiarity, a lexicographic fitness function is proposed in this paper and can be expressed as follows:

$$f(s) = \left\langle l_{n+1}, \sum_{j=1}^{J} \left\{ \frac{l_j - EFT_j}{EFT_j + 1} \left(\sum_{k=1}^{K} r_{jk} \right) d_j \right\} \right\rangle \quad (1)$$

The first component denotes the finish time of activity $n + 1$, that is, the makespan of the project. The second component depicts the deviation of the finish time of each activity j from the Earliest Finish Time (EFT) calculated by the Critical Path Method (CPM). The latter quantity is multiplied by the total resource consumption r_{jk} for the activity j as well as by the activity's duration d_j. Assuming that high quality solutions are expected to have the least possible deviations from the earliest finish times for each activity, the larger the second component of (1) the lower the quality of the solution, with respect to the same makespan. Moreover, the larger the product of the activity's resource consumption multiplied by its duration, the larger the impact of the activity scheduling times to the solution quality. Despite the fact that (1) is designed for the forward scheduling,

it is also valid to the backward scheduling if instead of the fraction $(l_j - EFT_j)/(EFT_j + 1)$ the $(LFT_j - l_j)/(LFT_j + 1)$ is used, where LFT is the Latest Finish Time calculated by CPM.

4.2. Initial phase

The construction heuristic we propose builds an RCPSP solution step by step, by scheduling one activity at a time and using a serial SGS along with forward, backward and bidirectional planning. The activity to be scheduled is randomly picked from a list of all precedence feasible activities at each step. The main focus is on the generation of distant/dissimilar good quality solutions. The dissimilarity of two solutions can be calculated using the distance function proposed by Chen et al. (2010) and is given as follows:

$$DSM(s, s') = \sum_{j=1}^{J} |l_{sj} - l_{s'j}| \quad (2)$$

where l_{sj} and $l_{s'j}$ are the finish times of activity j in solutions s and s', respectively.

For the generation of μ initial solutions, characterized by high quality properties as well as diversity, an iterative selection procedure is employed. First, a population of μ random solutions is initialized. Subsequently, an iterative procedure constructs new solutions at each iteration and the reference set update method (see below for details) is called to determine whether the current solution will be inserted to the reference set or not. The procedure terminates after λ iterations. The pseudo-code is as follows:

Initialization(μ, λ)
$M \leftarrow$ Initialize()
counter $\leftarrow 0$
While *counter* $< \lambda$ **do** {
  $s \leftarrow$ ConstructionHeuristic()
  UpdateRefSet(M,s)
Return M

4.2.1. Reference set update method

There are various rationales in the literature for updating the reference set of solutions. In this paper, the candidate for insertion into the reference set is solution s selected according to thefollowing criteria: (a) if the new solution s has better makespan than the ever best found s_{elite}, or (b) if it has better makespan than the worst solution s_{worst} and improves the average dissimilarity of the population. The new average dissimilarity is calculated by inserting s into the reference set and by extracting s_{worst}. The reference set update method described above, is following an elitist rationale towards high quality as well as distant solutions. The main focus is on the spread of the search over different and high quality regions of the solution space.

4.3. Scatter search phase

At the scatter search phase the reference set of solutions is evolved via an efficient solution combination method and an improvement method, that is, the Adaptive Iterated Local Search (AILS). The subset generation method picks at random κ solutions from the reference set M, which comprise the set U, and a linear combination is then applied to produce one single solution. AILS attempts to improve the quality of each recombined solution, and the reference set update method is then called. The components of the scatter search algorithm are presented in the following pseudocode:

The SAILS algorithm
Parameters: λ, μ, δ, κ, ϑ_{max}
Initial phase
$M \leftarrow$ Initialization(λ, μ)
gen $\leftarrow 0$
Scatter search phase
While termination conditions **do** {
$M' \leftarrow$ Recombination(κ, μ,*gen*), *gen*=*gen*+1
  For all individuals s of M' **do** {
    $s' \leftarrow$ AILS($s, \delta, \vartheta_{max}$)
    UpdateRefSet(M,s')
  }
}
Return M

4.4. Solution combination method

The proposed solution combination method relies on scatter search principles and operates on the representation proposed herein. Instead of using activity chains (permutation of activities) as the solution parts to be recombined (i.e., the approach followed in AL-based methodologies), an innovative solution recombination is proposed that treats events as the solutions' elements to be recombined. Both the forward and the backward event lists are considered, though for the sake of simplicity the following discussion is on the forward only.

An event list comprises events sorted according to their starting times. Each event is characterized and identified by its starting time. In the following, the term *similar events* refers to events that have the same starting time but they consist of a different set of activities, while the term *identical events* refers to events that have both the activities and the starting times identical.

At first, the events' starting times for each different solution $s \in U$ (where U is a subset of the reference set) are stored. Each starting time is associated with a value of "preference", that shows the extent at which this particular starting time is desirable for the respective activities to be scheduled. To calculate this value, κ weights (as many as the solutions into the subset U of the reference set) are introduced for each different solution. If a particular activity belongs to more than one *similar events* (or *identical events*) along the parent solutions, its value of "preference" is calculated by the sum of the respective weights. The starting time that the activity will be finally rescheduled is the one that maximizes the sum of the respective weights. The weights are calculated by the following:

$$W(s) = \frac{X(s)}{\sum_{u=1}^{|U|} X(s_u)} \quad (3)$$

where $X(s) = 1/(f(s) + age(s))$. The first component of the denominator of $X(s)$ is the makespan of the solutions, while the second component gives the age of the solution through the evolution of generations. The age of a solution is equal to the number of generations this solution remains into the reference set. The larger the cost of the solution the smaller the respective weight. To the other end, it is assumed that aged solutions have adequately shared their characteristics into the recombination process through the evolution of generations. Thus, as solutions are aging, their contribution to the recombination process is reduced in an effort to increase the probabilities of visiting different regions of the solution space.

There might occur infeasibilities if one attempts to construct a solution by using the "preferred" starting times for each activity, obtained by the procedure described above. To this end, a

construction heuristic is incorporated to ensure feasibility. In particular, it starts with an empty schedule and sequentially schedules activities as early as possible, in an effort to minimize any deviations from the "preferred" starting times.

The recombined solutions are as many as the solutions comprised in the reference set. Given the resulting μ recombined solutions, which are hosted in set M', AILS is applied on each solution s of M' in a sequential fashion, in order to improve the solution quality. The resulting solutions s' obtained by AILS, are subject to evaluation, and the solution framework oscillates until the termination criteria are met.

4.5. AILS improvement method

The Iterated Local Search (ILS), based on the multi-start local search, perturbs the local optima using adaptive memory and restarts from the modified solution (Martin, Otto, & Felten, 1991). In this paper, an Adaptive Iterated Local Search is proposed that utilizes an efficient local search algorithm and an adaptive perturbation strategy. The main focus is on retaining, at each restart, the good properties of the local optima, while adaptively perturbing the current solution towards diversified solution structures. The components of the AILS algorithm are presented in the following pseudocode:

The AILS improvement method
$\vartheta \leftarrow 1$
do $\{s' \leftarrow$ Perturbation$(s, \vec{g}, \vec{h}, \vartheta)$
$(\vec{g}, \vec{h}) \leftarrow 0$
$(s^*, \vec{g}, \vec{h}) \leftarrow$ LocalSearch$(\delta, s', \vec{g}, \vec{h})$
If $f(s^*) > f(s)$ Then $\vartheta = \vartheta + 1$
Else $\vartheta \leftarrow 1, (\vec{g}, \vec{h}) \leftarrow 0, s \leftarrow s^*\}$
While $(\vartheta < \vartheta_{max})$
Return s

AILS has two components; a local search engine and an adaptive perturbation strategy. The local search, performed by AILS, incorporates new neighborhood structures, while utilizes a short term memory to guide the search. The adaptive perturbation strategy partially modifies the current solution according to information gathered during the search (long term memory). To this purpose, two vectors are introduced (see for details Section 4.5.1), $\vec{g}$ and $\vec{h}$, that keep track of the search history. The procedure terminates after δ iterations that an improvement has not been observed.

4.5.1. Local search

The local search, conducted by AILS, is equipped with compound moves and new neighborhood structures (described in Section 4.5.2). It encompasses a frequency based memory to escape from local optima by penalizing previously visited neighbors. To this end, two vectors are used; vector $\vec{g}$ is referred to the list of activities and vector $\vec{h}$ is referred to the schedule horizon. The size of $\vec{g}$ is equal to the size of the problem, that is, the number of activities, whilst the size of $\vec{h}$ is equal to the planning horizon. At first, all elements of $\vec{g}$ and $\vec{h}$ are initialized to zero. Every time an activity, that belongs to an event, is chosen to participate into a local move, the respective element of $\vec{g}$ increments by one, while the element of $\vec{h}$ that represents the time bucket in which the relocated activity is scheduled, increments by one as well. Every time a better solution is found, all elements of both vectors are re-initialized. Let an event e, comprised by m activities and a local move that relocates the event e to a feasible position into the event list. The resulting cost of the move, will be the new cost $f(s')$ of solution s', minus the previous cost $f(s)$ plus the penalty of the move. The penalty of the move is given by the sum of the respective element values for both $\vec{g}$ and $\vec{h}$. The elements of $\vec{g}$ that are associated with the move are these that represent the relocated activities. Similarly, the elements of $\vec{h}$ are the starting times t_i of the relocated activities $i \in e$. An aspiration criterion is used according to which a solution is accepted no matter how large is the penalty, if it improves the ever best found solution. The following equation depicts the local move cost:

$$\Delta f_{move} = f(s') - f(s) + \sum_{i=1}^{m} (g(i) + h(t_i)) \tag{4}$$

4.5.2. Neighborhood structures

The solution neighborhood, proposed in this paper, is built by considering the relocation of all events of a schedule to all possible positions in the event list. An event comprises a set of activities that start at the same time in a schedule (or finish at the same time when a backward SGS is considered). These activities possibly share common network characteristics, i.e. share the same predecessors and/or successors, and thus are considered as a whole. The latter assumption is not always true. There are cases in which some activities start at the same time because of the resource load restrictions and not due to their network characteristics and precedence constraints. However, one can take advantage of the cases where the activities of an event share common network characteristics, consider them as a whole, i.e., as an event, and move them throughout the schedule to generate new neighborhood structures. In these cases the neighborhood is enriched with more neighbors, since a larger number of moves is permitted. If an event comprises activities that do not share common network characteristics, its permitted local moves, in terms of precedence constraints, are limited.

The event list is generated given the starting times of the activities in a schedule. Moreover, an allowable range of positions is denoted in which the event can be relocated. For each event in the list, a set of new events is generated according to all possible combinations of the considered activities. For example, if an event comprises of three activities a, b, and c, all possible combinations-events that occur are seven, that are, a, b, c, ac, ab, bc, abc while the order does not make any difference. Note that events that comprise one activity are considered as well. All generated events at each different position in the event list, take part into an iterative relocation procedure to all precedence feasible positions.

After relocating an event to a specific position into the event list, a serial schedule generation scheme (SGS) is employed to generate the schedule. It is worth mentioning, that relocating an event to a specific position in the event list does not mean that it will inherent the starting time of the particular position, but on the contrary, all of its activities will be rescheduled independently as early as possible (or as late as possible if the backward SGS is considered).

Fig. 2 describes a local move of an event into three distinct steps. The sample project of Fig. 1 is used and its solution represents the initial solution. At the first step, an event is selected and a feasible relocation position is chosen (Fig. 2a). The candidate event for relocation is the shaded one and comprises two activities, that are, the 11^{th} and the 20^{th} activity. Given the relocation position, the event is removed and all of its previous events remain at the same time–buckets. The events that exist between the event position and the relocation position, that are, {13,15}, {14}, {5,16}, {12,18} and {19}, are rescheduled sequentially as early as possible, as Fig. 2b shows, while the removed event {11,20} is standing by. Subsequently, at Fig. 2c the removed event is inserted into the sub-

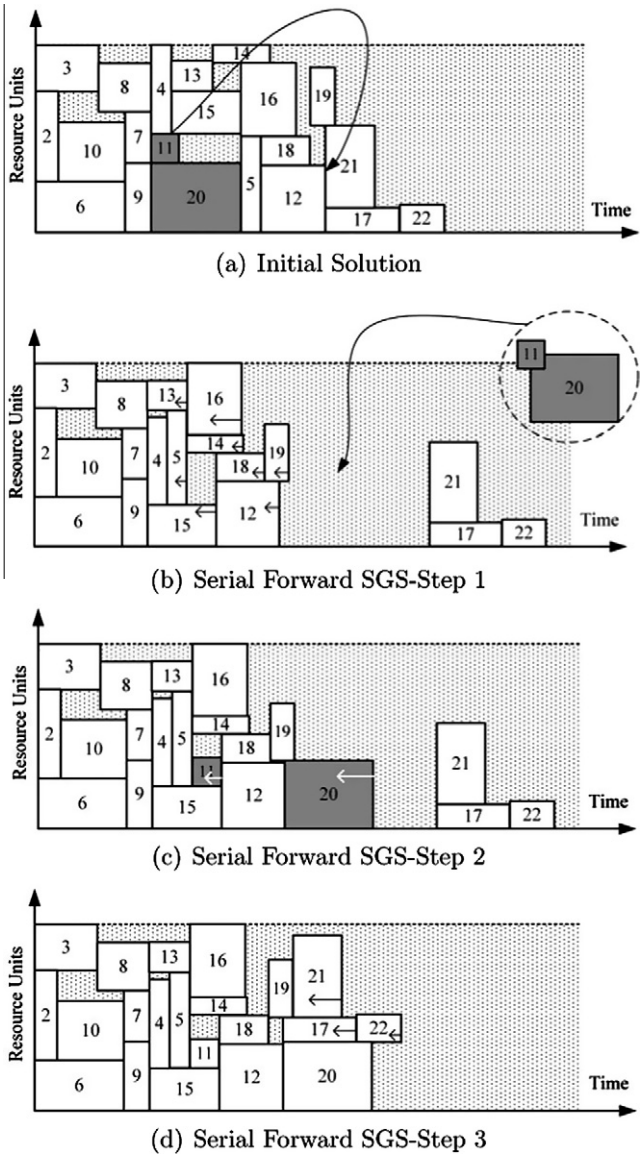

Fig. 2. Neighborhood operator-event relocate.

solution by scheduling all of its activities independently as early as possible. As mentioned above, despite the fact that local search considers events for relocations, at the scheduling procedure each activity is rescheduled individually. Finally, the remaining activities are scheduled as early as possible and the resulting schedule is given in Fig. 2d.

4.5.3. Neighborhood evaluation mechanism

AILS examines only a subset of all the neighbors, in order to speed up the neighborhood evaluation procedure. Since the most computationally expensive function, at this phase, is SGS, an effort is given to utilize it as less as possible. In particular, as soon as a solution is subject for improvement through local search, the finish times of the activities are stored, namely *elite times*. Every time a better neighbor is met the *elite times* are updated. If at some step of SGS there are scheduled activities for which the finish times are *all larger* than their elite values, *none* of their successors has been scheduled so far, and additionally the remaining scheduling sequence of the activities is identical with this of the elite neighbor, SGS terminates, since the resulting neighbor will be of lower quality than an already visited neighbor.

Let s and s' be two different solutions of RCPSP. Let l_{si} be the finish time of activity i that belongs to the solution s, and $l_{s'i}$ be the finish time of the same activity i that belongs to the solution s'. Let S be a set, that comprises activities of s' that have been scheduled and none of their successors has been scheduled so far. Then, given that the non preemptable duration d_i of an activity is deterministic parameter, and additionally the remaining scheduling sequence of the activities is identical, if $l_{s'i} > l_{si}, \forall i \in S$ then the resulting neighbor is assumed to be less valuable in the search process, and it is thus ignored.

In the best case, where the best neighbor of a solution neighborhood is the first neighbor evaluated, all the subsequent neighbors will not be completely examined, while in the worst case, where the neighbors are examined in a decreasing cost order, all neighbors will be evaluated. Nevertheless, in the general case the activation of this mechanism boosts the neighborhood evaluation procedure by significantly reducing the number of solutions examined.

It is worth mentioning that the above neighborhood evaluation mechanism is described for the forward scheduling. Nevertheless, it can be adjusted for the backward scheduling as well. At that case, if at some step of the backward SGS there are scheduled activities for which the *starting* times are *all smaller* than their elite values, *none* of their *predecessors* has been scheduled so far, and additionally the remaining scheduling sequence of the activities is identical with this of the elite neighbor, SGS terminates, since the resulting neighbor will be of lower quality than an already visited neighbor.

4.5.4. Adaptive perturbation strategy

The main focus of the perturbation method is on partially rebuilding the current solution, such that to retain the information of the local optimum as well as producing diversified solution structures. In this paper, an adaptive perturbation is employed to perturb the current solution according to information gathered by the search history. During the local search procedure duplicates of vectors $\vec{g}$ and $\vec{h}$ (see Section 4.5.1 for details), keep track of the times each activity has taken part into the local moves and the times a relocated activity that is scheduled in a particular time bucket. These duplicates hold the local search history and are reinitialized each time a new perturbation is performed. The more times an activity is relocated, the more "attractive" this activity is to the local search procedure. Similarly, the more times a particular time bucket has participated into a local move, the more "attractive" this time bucket is.

The proposed perturbation strategy reschedules the "less attractive" activities (obtained by the local search procedure) to the "less attractive" time buckets in an effort to construct diversified solution structures. Thus, local moves that were not chosen during the local search procedure, are now performed in an effort to spread the search over different regions of the solution space. The number of the rescheduled activities of the current solution is determined by a self-adapted "length". The perturbation length is depicted by parameter ϑ and it is reactively tuned according to the search progress. In particular, ϑ is initialized to one, and it is gradually incremented as the search does not find a better solution, until ϑ_{max} is reached. Every time a better solution is found, ϑ is reinitialized to one. At first, $\frac{\vartheta}{\vartheta_{max}}$% of the solution is perturbed. Subsequently, local search gives an effort to improve the perturbed solution. If local search fails to improve this solution, ϑ increments by one and the new perturbed solution will be $\frac{\vartheta+1}{\vartheta_{max}}$% modified. If no improvement is conducted and ϑ has reached ϑ_{max}, AILS returns the best solution found.

5. Computational results

5.1. Data sets & parameter settings

For the evaluation of the proposed solution framework, various numerical experiments have been conducted on the benchmarks of Kolisch, Sprecher, and Drexl (1995). In particular, the sets J30, J60, and J120 of problems instances were the test bed for the computational analysis. These sets consist of 30, 60, and 120 activities and contain 480 instances, 480 instances and 600 instances for each benchmark set, respectively. The latter have been widely used in literature and they are available on the Internet (PSPLIB, Kolisch & Sprecher, 1996). The proposed method was implemented in Visual C++ 2008, and run on a PIV 2.8 GHz PC.

The proposed solution method, namely SAILS, introduces five parameters; the number of initial solutions λ considered to build the initial reference set of solutions, the reference set size μ, the maximum local search iterations-schedules examined that an improvement has not been observed δ, the levels of the perturbation ϑ_{max}, and the number of solutions to be recombined by scatter search κ. The parameter λ seemed not to have important impact into the computational results. In particular, parameter λ was set equal to 100 if the maximum number of schedules examined *maxSch* was set equal to 1000, $\lambda = 200$ if *maxSch* = 5000, and $\lambda = 500$ if *maxSch* = 50,000. A sensitivity analysis indicates that the change in the quality of the solutions with respect to the parameter choices, is quite small. The parameter settings are given in Table 1. Fifteen problems instances of the benchmarks of Kolisch et al. (1995) are chosen intuitively from J120 set for the sensitivity analysis. More specifically, these problems include X–6–2, X–7–3, X–11–4, X–12–3, X–17–5, X–26–7, X–31–5, X–32–7, X–36–2, X–46–2, X–47–8, X–51–1, X–52–2, X–56–2, and X–56–3, which can be found in PSPLIB (Kolisch & Sprecher, 1996).

Three values were used on μ, that are, 12, 16 and 20. Parameter κ is set equal to 3, in order to keep the number of the possible solution combinations high. Parameters δ and ϑ_{max} are interrelated in a way to keep the total number of Local Search (LS) iterations-schedules examined around a specific level ($\approx$100 LS iterations-schedules examined with no improvement). In particular, ϑ_{max}, i.e., the number of the perturbation levels, was set equal to 2, 3, and 4. Preliminary experiments indicated that larger values than the value ranges given above, deteriorated the solution quality, and thus, are not reported herein. In an effort to maintain a balance between the number of perturbations and the local search oscillations, δ is defined according to ϑ_{max} values. In a similar manner, the rest settings are given in Table 1. Note that, column *Sum* of Table 1 refers to the sum of the makespan for all the fifteen problem instances that participated into the sensitivity analysis. The parameters' values that were chosen for the production of the best results reported herein were $\mu = 16$, $\kappa = 3$, $\delta = 40$, and $\vartheta_{max} = 3$.

Given that the termination criterion of the solution approaches in literature (Kolisch et al., 1995), is 50,000 schedules examined, *maxSch* is set equal to 50,000. However, in order to increase the probability of finding higher quality solutions, the proposed solution framework was allowed to oscillate further, i.e., until 200,000 schedules examined. Nevertheless, the proposed solution method, most of the times, needed to examine much fewer schedules than the threshold of 50,000, to produce the best results reported in this paper. In particular, for J30 set 65 on average schedules were examined, for J60 set 2405 on average schedules were met, while for J120 set an average of 13,234 schedules were produced.

Table 1
Sensitivity analysis for the parameters μ, κ, δ and ϑ_{max}.

μ	κ	δ	ϑ_{max}	*Sum*
		50	2	2625
12		40	3	2618
		30	4	2622
		50	2	2619
16	3	40	3	2612
		30	4	2616
		50	2	2621
20		40	3	2624
		30	4	2620

5.2. Comparative analysis

In this paper, AILS is considered as one of the main components of the proposed evolutionary algorithm, namely SAILS, as well as an autonomous solution method. To investigate thus on AILS' performance, numerical experiments were conducted, and comparisons were made to other local search methodologies that have been proposed in literature for solving the RCPSP. In particular, the parameter settings, that were used for the numerical experiments, were $\delta = 40$ and $\vartheta_{max} = 4$. Note that, a random multi-restart framework is considered and the maximum number of schedules examined *maxSch* = 50,000, was the termination criterion.

Table 2 summarizes the results of AILS compared to local search methodologies. The second column "Algorithm" reports the abbreviations of the local search algorithms considered (AILS–Adaptive Iterated Local Search, SA–Simulated Annealing, VNS–Variable Neighborhood Search, TS–Tabu Search, LS–Local Search). Column RS stands for the Representation Scheme (AL–Activity List, EL–Event List). The third column refers to the respective references, while the last three columns refer to the average deviation % from the optimum for J30 set of problems and denote the average deviation % from the critical path lower bound for J60 and J120 sets of problems, respectively. Moreover, the results are obtained examining 1000, 5000, and 50,000 schedules.

It is rather obvious that there are few studies that utilize a local search method in a single solution framework, while all of them use the AL representation. Note that, Thomas and Salhi (1998) developed a tabu search algorithm that belongs to local search algorithms, but their paper has not been included into the comparisons herein, since they did not used the benchmarks of Kolisch et al. (1995) as the test bed for their computational experiments. To the other end, despite the fact that Fleszar and Hindi (2004) developed a variable neighborhood search for solving RCPSP on the benchmarks of Kolisch et al. (1995), their VNS was not included in Table 2 but in Tables 4 and 5, since the authors followed a different approach for the schedules considered. Lastly, Ranjbar (2008) cannot be included in the comparisons presented in this subsection (but included in Table 5), since the author does not report the number of schedules examined to obtain the results, but only computational times.

Table 2 illustrates that AILS produces consistently high quality solutions for all benchmark sets. In particular, AILS achieves 0.00 (%) average deviation from the optimum for J30 set, 10.91% average deviation from the CPM lower bounds for J60 set, and 32.66% for J120 set of problem instances. It is worth noticing that there is not great difference between the results obtained by examining 50,000 schedules to the results obtained by 5000 schedules for J120 set of problems. This indicates that AILS is capable of producing high quality solutions examining few schedules, and thus, encourages its integration into hybrid algorithms as a post optimization procedure.

Having tested AILS's performance, the computational analysis proceeds by evaluating SAILS' performance. To this end, numerical experiments on the benchmarks of Kolisch et al. (1995) were conducted and comparisons were made to all competitive metaheuristics proposed for solving the RCPSP.

Table 3 summarizes the results obtained by all metaheuristic algorithms of literature, including population based approaches

Table 2
Computational comparisons among Local Search metaheuristics on the benchmarks sets J30, J60, and J120 (Kolisch et al., 1995).

Pr. set	Algorithm	RS	Reference	% Dev		
				1000	5000	50000
J30	AILS	EL	This paper	0,05	0,01	0,00
	SA	AL	Bouleimen and Lecocq (2003)	0,38	0,23	n/a
	TS	AL	Nonobe and Ibaraki (2002)	0,46	0,16	0,05
	TS-insertion rules	AL	Artigues et al. (2003)	n/a	n/a	n/a
	Hybrid LS techniques	AL	Pesek et al. (2007)	n/a	n/a	n/a
J60	AILS	EL	This paper	11,34	11,10	10,91
	Hybrid LS techniques	AL	Pesek et al. (2007)	n/a	n/a	11,10
	SA	AL	Bouleimen and Lecocq (2003)	12,75	11,90	n/a
	TS	AL	Nonobe and Ibaraki (2002)	12,97	12,18	11,58
	TS-insertion rules	AL	Artigues et al. (2003)	n/a	12,05	n/a
J120	AILS	EL	This paper	34,38	32,67	32,66
	TS	AL	Nonobe and Ibaraki (2002)	40,86	37,88	35,85
	SA	AL	Bouleimen and Lecocq (2003)	42,81	37,68	n/a
	Hybrid LS techniques	AL	Pesek et al. (2007)	n/a	n/a	n/a
	TS-insertion rules	AL	Artigues et al. (2003)	n/a	n/a	n/a

and hybrids, on J30, J60 and J120 problem sets. The first column "Algorithm", reports the abbreviations of the algorithms considered (SAILS–Scatter search-Adaptive Iterated Local Search, GA–Genetic Algorithm, ACOSS–Ant Colony Optimization Scatter Search, SS–Scatter Search, PR–Path Relinking, GAPS–Genetic Algorithm for Project Scheduling, FBI–Forward Backward Improvement, TS–Tabu Search, ANGEL–ANt colony optimization GEnetic algorithm Local search). Column RS stands for Representation Scheme (AL–Activity List, RK–Random Key, TO-RK–Topological Order Random Key, EL–Event List). The third column reports the references and the last four columns refer to the average deviation % from the optimum for J30 set of problems, while they denote the average deviation % from the critical path lower bound for J60 and J120 sets of problems, examining 1000, 5000, 50,000, and >50,000 schedules, respectively. The eighth column illustrates the performance of algorithms in literature that were permitted to examine more schedules than 50,000. In particular, the numbers in the parenthesis of the eighth column report the maximum number of schedules

Table 3
Computational comparisons among all competitive metaheuristics on the benchmarks sets J30, J60, and J120 (Kolisch et al., 1995).

Pr. set	Algorithm	RS	Reference	% Dev.			
				1000	5000	50000	>50000
J30	SAILS	EL	This paper	0,03	0,01	0,00	n/a
	AILS	EL	This paper	0,05	0,01	0,00	n/a
	GA, TS-PR	AL	Kochetov and Stolyar (2003)	0,10	0,04	0,00	n/a
	SS-PR	AL	Mahdi Mobini et al. (2009)	0,05	0,02	0,01	n/a
	GAPS	RK	Mendes et al. (2009)	0,06	0,02	0,01	n/a
	ACOSS	AL	Chen et al. (2010)	0.14	0,06	0,01	n/a
	SS-FBI	RK	Debels et al. (2006)	0,27	0,11	0,01	$0{,}01(5{\cdot}10^5)$
	GA	TO-RK	Debels and Vanhoucke (2007)	0,15	0,04	0,02	n/a
	GA-hybrid FBI	AL	Valls et al. (2008)	0,27	0,06	0,02	n/a
	TS	AL	Nonobe and Ibaraki (2002)	0,46	0,16	0,05	n/a
	GA	AL	Hartmann (2002)	0,38	0,22	0,08	n/a
	ANGEL	AL	Tseng and Chen (2006)	0,22	0,09	n/a	n/a
J60	SAILS	EL	This paper	11,05	10,72	10,54	$10{,}46(7{\cdot}10^4)$
	SS-PR	AL	Mahdi Mobini et al. (2009)	11,12	10,74	10,57	n/a
	GAPS	RK	Mendes et al. (2009)	11,72	11,04	10,67	10,67(63.546)
	ACOSS	AL	Chen et al. (2010)	11,72	10,98	10,67	n/a
	GA	TO-RK	Debels and Vanhoucke (2007)	11,45	10,95	10,68	n/a
	SS-FBI	RK	Debels et al. (2006)	11,73	11,10	10,71	$10{,}53(5{\cdot}10^5)$
	GA-hybrid FBI	AL	Valls et al. (2008)	11,56	11,10	10,73	n/a
	GA, TS-PR	AL	Kochetov and Stolyar (2003)	11,71	11,17	10,74	n/a
	AILS	EL	This paper	11,34	11,10	10,91	n/a
	GA	AL	Hartmann (2002)	12,21	11,70	11,21	n/a
	ANGEL	AL	Tseng and Chen (2006)	11,94	11,27	n/a	n/a
	TS	AL	Nonobe and Ibaraki (2002)	12,97	12,18	11,58	n/a
J120	ACOSS	AL	Chen et al. (2010)	35,19	32,48	30,56	n/a
	SAILS	EL	This paper	33,32	32,12	30,78	$30{,}39(2{\cdot}10^5)$
	GA	TO-RK	Debels and Vanhoucke (2007)	34,19	32,34	30,82	n/a
	GA-hybrid FBI	AL	Valls et al. (2008)	34,07	32.54	31,24	n/a
	GAPS	RK	Mendes et al. (2009)	35,87	33,03	31,44	31,20(127.341)
	SS-PR	AL	Mahdi Mobini et al. (2009)	34,51	32,61	31,37	n/a
	SS-FBI	RK	Debels et al. (2006)	35,22	33,10	31,57	$30{,}48(5{\cdot}10^5)$
	GA, TS-PR	AL	Kochetov and Stolyar (2003)	34,74	33,36	32,06	n/a
	AILS	EL	This paper	34,38	32,67	32,66	n/a
	GA	AL	Hartmann (2002)	37,19	35,39	33,21	n/a
	ANGEL	AL	Tseng and Chen (2006)	36,39	34,49	n/a	n/a
	TS	AL	Nonobe and Ibaraki (2002)	40,86	37,88	35,85	n/a

examined, regarding the solution methodologies proposed by Mendes et al. (2009) and Debels et al. (2006). In this paper, SAILS examined 70,000 schedules for J60 set, and 200,000 for J120 set, and the results are summarized in the eighth column of Table 3. It must be noticed, that in Table 3 the Decomposition-based GA (DBGA) of Debels and Vanhoucke (2007) is not included, since for the DBGA the number of schedules cannot be accurately defined. The authors have thus used a time equivalant to their GA computational time (at each performance mode) as a stopping criterion. To this end, DBGA's performance is considered in the following tables (Tables 4 and 5) where the computational times are discussed.

For the sake of objectiveness all the comparisons were made according to the 50,000 performance mode. As shown in Table 3, SAILS produces solutions that deviate by 0.00% on average from the optimum for J30 set. Regarding the J60 set, SAILS produces consistently the best results, regarding the 1000, the 5000, the 50,000

Table 4
Computational times for 5000 and lower schedules examined.

Pr. Set	Algorithm	Reference	% Dev.	Av. CPU(s)	Max CPU(s)	No sch. (∗1000)	CPU freq.
J30	SAILS	This paper	0,01	0,15	1,60	5,0	2,8 GHz
	ACOSS	Chen et al. (2010)	0.06	0,13	3,29	5,0	1,86 GHz
	SS-PR	Mahdi Mobini et al. (2009)	0,02	0,07	0,17	5,0	3 GHz
	DBGA	Debels and Vanhoucke (2007)	0,04	0,06	n/a	5,0	1,8 GHz
	SS-FBI	Debels et al. (2006)	0,11	0,06	0,12	5,0	1,8 GHz
	ANGEL	Tseng and Chen (2006)	0,09	0,11	n/a	5,0	1 GHz
	LSSPER	Palpant et al. (2004)	0,00	10,26	123,00	1,1	2,3 GHz
	TS	Nonobe and Ibaraki (2002)	0,16	9,07	n/a	5,0	300 MHz
J60	SAILS	This paper	10,72	5,44	47,56	5,0	2,8 GHz
	ACOSS	Chen et al. (2010)	10,98	0,72	13,80	5,0	1,86 GHz
	SS-PR	Mahdi Mobini et al. (2009)	10,74	0,23	0,37	5,0	3 GHz
	DBGA	Debels and Vanhoucke (2007)	10,95	0,11	n/a	5,0	1,8 GHz
	SS-FBI	Debels and Vanhoucke (2007)	11,10	0,18	0,27	5,0	1,8 GHz
	ANGEL	Tseng and Chen (2006)	11,27	0,76	n/a	5,0	1 GHz
	LSSPER	Palpant et al. (2004)	10,81	38,78	223,00	2,2	2,3 GHz
	TS	Nonobe and Ibaraki (2002)	12,18	24,49	n/a	5,0	300 MHz
J120	SAILS	This paper	32,12	43,45	137,12	5,0	2,8 GHz
	ACOSS	Chen et al. (2010)	32,48	3,80	39,80	5,0	1,86 GHz
	SS-PR	Mahdi Mobini et al. (2009)	32,61	0,71	1,08	5,0	3 GHz
	DBGA	Debels and Vanhoucke (2007)	32,18	0,27	n/a	5,0	1,8 GHz
	SS-FBI	Debels et al. (2006)	33,10	0,65	0,93	5,0	1,8 GHz
	ANGEL	Tseng and Chen (2006)	34,49	4,79	n/a	5,0	1 GHz
	LSSPER	Palpant et al. (2004)	32,41	207,93	501,00	5,0	2,3 GHz
	TS	Nonobe and Ibaraki (2002)	34,88	645.33	n/a	5,0	300 MHz

Table 5
Computational times for non reported schedules, for 50,000 and more schedules.

Pr. set	Algorithm	Reference	% Dev.	Av. CPU (s)	Max CPU (s)	No sch. *1000	CPU freq.
J30	SAILS	This paper	0.00	0.19	2.88	50	2.8 GHz
	SS-PR	Mahdi Mobini et al. (2009)	0.01	0.81	1.41	50	3 GHz
	SS-PR (fast)	Mahdi Mobini et al. (2009)	0.02	0.67	1.41	n/a	3 GHz
	GAPS	Mendes et al. (2009)	0.01	5.02	n/a	50	1.33 GHz
	FF	Ranjbar (2008)	0.00	5.00	5.00	n/a	3 GHz
	DBGA	Debels and Vanhoucke (2007)	0.02	0.52	n/a	50	1.8 GHz
	SS-FBI	Debels et al. (2006)	0.01	0.69	1.27	50	1.8 GHz
	PBA	Valls et al. (2004)	0.10	1.16	5.49	n/a	400 MHz
	VNS	Fleszar and Hindi (2004)	0.01	0.64	5.86	n/a	1 GHz
	TS-FBI	Valls et al. (2003)	0.06	1.61	6.15	n/a	400 MHz
J60	SAILS	This paper	10.54	16.31	89.18	40	2.8 GHz
	SS-PR	Mahdi Mobini et al. (2009)	10.57	2.01	3.04	50	3 GHz
	SS-PR (fast)	Mahdi Mobini et al. (2009)	10.91	1.06	n/a	n/a	3 GHz
	GAPS	Mendes et al. (2009)	10.67	20.11	n/a	50	1.33 GHz
	FF	Ranjbar (2008)	10.56	5.00	5.00	n/a	3 GHz
	DBGA	Debels and Vanhoucke (2007)	10.68	1.11	n/a	50	1.8 GHz
	SS-FBI	Debels et al. (2006)	10.71	1.88	2.64	50	1.8 GHz
	PBA	Valls et al. (2004)	10.89	3.65	22.60	n/a	400 MHz
	VNS	Fleszar and Hindi (2004)	10.94	8.89	80.70	1653	1 GHz
	TS-FBI	Valls et al. (2003)	11.12	2.76	14.61	n/a	400 MHz
J120	SAILS	This paper	30.78	123.45	551.00	50	2.8 GHz
	SS-PR	Mahdi Mobini et al. (2009)	31.37	7.06	10.10	50	3 GHz
	SS-PR (fast)	Mahdi Mobini et al. (2009)	32.27	6.16	n/a	n/a	3 GHz
	GAPS	Mendes et al. (2009)	31.44	112.46	n/a	50	1.33 GHz
	FF	Ranjbar (2008)	31.42	5.00	5.00	n/a	3 GHz
	DBGA	Debels and Vanhoucke (2007)	30.69	2.99	n/a	50	1.8 GHz
	SS-FBI	Debels et al. (2006)	31.57	6.66	9.22	50	1.8 GHz
	PBA	Valls et al. (2004)	31.58	59.43	263.97	n/a	400 MHz
	VNS	Fleszar and Hindi (2004)	33.10	219.86	1126.97	10778	1 GHz
	TS-FBI	Valls et al. (2003)	34.53	17.00	43.94	n/a	400 MHz

and the 70,000 performance modes. In particular, it achieves 11.05%, 10.72%, 10.54% and 10.46% average deviation from CPM lower bounds at 1000, 5000, 50,000 and 70,000 schedules, respectively. Note that, AILS remains competitive and performs even better than more advanced solution methodologies. SAILS produces the best results regarding the 1000, the 5000 and the > 50,000 performance modes, for the J120 problem set. Considering the 50,000 schedules SAILS remains one of the most efficient algorithms for solving RCPSP achieving a 30.78% average deviation from CPM lower bounds. Regarding the performance of SAILS for more than 50,000 schedules, the best results are produced achieving 30.39% average deviation from CPM lower bound for J120 set.

In order to provide the basis of comparisons among heuristic methodologies for the RCPSP, Kolisch et al. (1995) limited the number of schedules examined and they did not focus on computational times. In this way, the tests are independent of compilers or authors' programming skills and evaluate the design of an algorithm and not a program code. Nevertheless, some authors report computational times, which are all listed in Tables 4 and 5 in a descending order according to the year of publication.

In particular, Table 4 shows the computational times at 5000 (and lower) schedules, on J30, J60 and J120 problem sets. The second column reports the abbreviations of the algorithms considered (PR-Path Relinking). The third column reports the references and the fourth column shows the average deviation % from the optimum for J30, while it denotes the average deviation % from the critical path lower bound for J60 and J120. The fifth and the sixth column denote the average and the maximum CPU time in seconds. In order to have an insight of the computer strength, the last column reports the CPU frequency. However, the latter is not adequate for deriving the right conclusions about the effectiveness of the solution approaches, since the computer architecture must be thoroughly examined. Similarly, Table 5 reports the computational times for algorithmic approaches that do not mention the number of schedules examined or the number of schedules iterations is unlimited. In the same table the computational times, for those who report 50,000 schedules (and more), are given. As shown in Tables 4 and 5 the computational times of SAILS are reasonable and even better than these of other algorithms, considering the different CPU frequencies. Note that, the computational times of AILS were almost the same with these of SAILS for each different performance mode, and thus are not reported herein. Moreover, it is worth mentioning that DBGA produces lower quality results for J30, J60 sets and higher quality for J120 set compared to these of SAILS, in remarkably shorter computational times. Finally, the Filter and Fan method of Ranjbar (2008) presents highly competitive results in very short computational times.

6. Conclusions and further research

This paper proposes a new solution representation and an evolutionary algorithm for solving the Resource Constrained Project Scheduling Problem (RCPSP). The proposed representation is a list of events, that depicts efficiently the problem's solution. The proposed solution method, namely SAILS, relies on a scatter search framework and consists of an initial phase and a scatter search phase. At the initial phase a construction heuristic is employed to produce distant solutions. At the scatter search phase, an efficient solution combination is proposed and an Adaptive Iterated Local Search (AILS) is introduced as an improvement method. In particular, AILS utilizes new enriched neighborhoods, guide the search via a frequency based short and long term memory, and incorporates an efficient perturbation strategy to escape from local optima. Emphasis was given on the simplicity and the effectiveness of AILS such that to encourage its integration into the scatter search framework proposed herein. Furthermore, the proposed solution combination is based on the event-list representation and relies on the scatter search principles.

Computational experiments on the benchmarks of Kolisch et al. (1995) show that both AILS and SAILS, produce consistently high quality solutions. In particular, AILS is highly competitive and performs even better than more advanced solution methodologies. To the other end, SAILS produces the best results for J30 and J60 problem sets, for all performance modes (i.e.,1000, 5000, 50,000, >50,000 schedules). Moreover, SAILS produces the best results for J120 problem set, considering 1000, 5000 and >50,000 schedules, while it remains one of the most efficient algorithms for solving RCPSP at 50,000 schedules.

In terms of further research, a worth pursuing direction is on the adjustment of the proposed representation scheme for depicting the solutions of other popular RCPSP variants as well as for depicting solutions of well known combinatorial optimization problems that share common characteristics to the RCPSP. Furthermore, there comes up a potential on the design of efficient solution methodologies that operate on the proposed event-based solution representation and produce high quality solutions to the RCPSP and its variants.

References

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource-constrained project scheduling problem. *Computers and Operations Research, 38*(1), 44–50.

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research, 149*, 249–267.

Blazewicz, J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics, 5*, 11–24.

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

Bräysy, O., Porkka, P., Dullaert, W., Repoussis, P. P., & Tarantilis, C. D. (2009). A well-scalable metaheuristic for the fleet size and mix vehicle routing problem with time windows. *Expert Systems with Applications, 36*(4), 8460–8475.

Chen, R. M., Wub, C. L., Wang, C. M., & Lo, S. T. (2010). Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert Systems with Applications, 37*, 1899–1910.

Chen, W., Shi, Y. J., Teng, H. F., Lan, X. P., & Hu, L. C. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences, 180*, 1031–1039.

Czogalla, J., & Finik, A. (2009). Fitness landscape analysis for the resource constrained project scheduling problem. *LNCS, 5851*, 104–118.

Debels, D., Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta heuristic for project scheduling. *European Journal of Operational Research, 169*, 638–653.

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

Fleszar, K., & Hindi, K. (2004). Solving the resource-constrained project scheduling problem by a variable neighborhood search. *European Journal of Operational Research, 155*, 402–413.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences, 8*, 156–166.

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. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics, 49*, 433–448.

Hsu, C. C., de Blois, Y., & Pyle, M. J. (2004). General motors optimizes its scheduling of cold-weather tests. *Interfaces, 34*(5), 334–341.

Kochetov, Y., & Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *Proceedings of the third international workshop of computer science and information technologies, Russia*.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for the resource–constrained project scheduling problem: Classification and computational analysis. In J. W. Eglarz, (Ed.), *Project scheduling, recent models, algorithms and applications* (pp. 147–178). Boston: Kluwer Academic Publishers.

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., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science, 41*, 1693–1703.

Kolisch, R., & Sprecher, A. (1996). PSPLIB – A project scheduling problem library. *European Journal of Operational Research, 96*, 205–216. <http://129.187.106.231/psplib/>.

Kone, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers and Operations Research, 38*, 3–13.

Mahdi Mobini, M. D., Rabbani, M., Amalnik, M. S., Razmi, J., & Rahimi-Vahed, A. R. (2009). Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. *Soft Computing, 13*, 597–610.

Martin, O., Otto, S. W., & Felten, E. W. (1991). Large-step Markov chains for the traveling salesman problem. *Complex Systems, 5*(3), 299–326.

Mendes, J. J. M., Goncalves, J. F., & Resende, M. G. C. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers and Operations Research, 36*, 92–109.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science, 44*, 714–729.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*, 333–346.

Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In C. C. Ribeiro & P. Hansen (Eds.), *Essays and surveys in metaheuristics* (pp. 557–588). Kluwer Academic Publishers.

Palpant, M., Artigues, C., & Michelon, P. (2004). LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research, 131*, 237–257.

Patterson, J. H. (1984). A comparison of exact approaches for solving the multiple constrained resources project scheduling problem. *Management Science, 30*(7), 854–867.

Pesek, I., Schaerf, A., & Zerovnik, J. (2007). Hybrid local search techniques for the resource-constrained project scheduling problem. *LNCS, 4771*, 57–68.

Ranjbar, M. (2008). Solving the resource constrained project scheduling problem using filter-and-fan approach. *Applied Mathematics and Computation, 201*, 313–318.

Ranjbar, M., & Kianfar, F. (2009). A hybrid scatter search for the RCPSP. *Transaction E: Industrial Engineering, 16*(1), 11–18.

Tarantilis, C. D., Zachariadis, E. E., & Kiranoudis, C. T. (2008). A hybrid guided local search for the vehicle routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing, 20*(1), 154–168.

Thomas, P. R., & Salhi, S. (1998). A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics, 4*, 123–139.

Tseng, L. Y., & Chen, S. C. (2006). A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research, 175*, 707–721.

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

Valls, V., Ballestin, F., & Quintanilla, S. (2004). A population-based approach to the resource-constrained project scheduling problem. *Annals of Operations Research, 131*, 305–324.

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

Wu, S., Wan, H. D., Shukla, S. K., & Li, B. (2011). Chaos-based improved immune algorithm (CBIIA) for resource-constrained project scheduling problems. *Expert Systems with Applications, 38*, 3387–3395.

Zhou, X., & Zhong, M. (2007). Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B, 41*, 320–341.