J Sched (2007) 10: 375–386
DOI 10.1007/s10951-007-0043-7

Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem

Véronique Bouffard · Jacques A. Ferland

Published online: 20 October 2007

Abstract The purpose of this paper is to improve the simulated annealing method with a variable neighborhood search to solve the resource-constrained scheduling problem. We also compare numerically this method with other neighborhood search (local search) techniques: threshold accepting methods and tabu search. Furthermore, we combine these techniques with multistart diversification strategies and with the variable neighborhood search technique. A thorough numerical study is completed to set the parameters of the different methods and to compare the quality of the solutions that they generate. The numerical results indicate that the simulated annealing method improved with a variable neighborhood search technique is indeed the best solution method.

Keywords Simulated annealing · Threshold accepting · Tabu search · Diversification · Variable neighborhood search · Resource-constrained scheduling

1 Introduction

The purpose of this paper is to improve the simulated annealing method introduced in Jeffcoat and Bulfin (1993) with a variable neighborhood search technique to solve the resource-constrained scheduling problem (RCSP). We also complete a numerical study to verify that this method generates better solutions than other neighborhood search techniques like threshold accepting methods and tabu search even if they are combined with multistart diversification strategies or with the variable neighborhood search technique.

The resource-constrained scheduling problem (RCSP) formulated in Jeffcoat and Bulfin (1993) is to determine a schedule for n tasks ($i = 1, 2, \ldots, n$) requiring resources to be completed. Each task i requires a specified number of periods to be completed, and a specified number of units of each resource during each period of its completion. Furthermore, each task i cannot be initiated before an earliest period (ready time of the task) and must be completed within a latest period (deadline of completion). For each resource, the number of units available during each period is also specified.

Two important assumptions simplify the problem. First, there is no preemption in the completion of any task. Second, there is no precedence relationship among the tasks. Nevertheless, the problem is Np-hard (Blazewicz 1978). We use the following notation to formulate the problem:

- r_i: ready time of task $i = 1, 2, \ldots, n$; i.e., the earliest period when i can be initiated.
- f_i: deadline for completion of task $i = 1, 2, \ldots, n$, i.e., the last period when i can be completed.
- d_i: duration of task $i = 1, 2, \ldots, n$; i.e., the number of periods required to complete it.
- $a_{i\ell}$: number of units of resource $\ell = 1, 2, \ldots, L$ required during each period of completion of task $i = 1, 2, \ldots, n$.
- $B_{\ell j}$: number of units of resource $\ell = 1, 2, \ldots, L$ available during period j of the horizon J.

This research was supported by NSERC grant (OGP 0008312) the first author received a FCAR fellowship during her M.Sc. studies.

V. Bouffard · J.A. Ferland (✉)
University of Montreal, CP 6128, Succ Centre-Ville, Montreal, PQ H3C 3J7, Canada
e-mail: ferland@iro.umontreal.ca

V. Bouffard
e-mail: bouffarv@iro.umontreal.ca

J: the horizon is the set of period indices j

$$J = \left\{ j : 1 \leq j \leq \max_{1 \leq i \leq n} \{f_i\} \right\}.$$

The decision variable x_i denotes the starting period of task $i = 1, 2, \ldots, n$, and the vector x

$$x = [x_1, x_2, \ldots, x_n],$$

denotes a schedule of the n tasks. A schedule x is said *admissible* if each task i is initiated at one of its admissible starting period J_i; i.e., $x_i \in J_i$ where

$$J_i = \{j : r_i \leq j \leq f_i - d_i + 1\}.$$

For a schedule x, let $E_j(x)$ denote the subset of tasks being executed during period $j \in J$; i.e.,

$$E_j(x) = \{i : x_i \leq j \leq x_i + d_i - 1\}.$$

An admissible schedule x is *feasible* if for $\ell = 1, 2, \ldots, L$ and $j \in J$

$$\sum_{i \in E_j(x)} a_{i\ell} \leq B_{\ell j}.$$

Now, to determine a *best admissible* schedule we use a penalty approach where resource constraints violations are minimized. This problem can be formulated as follows:

$$(P) \quad \min \; P(x) = \sum_{j \in J} \sum_{\ell=1}^{L} \max \left\{ 0, \sum_{i \in E_j(x)} a_{i\ell} - B_{\ell j} \right\}$$
$$\text{Subject to } x_i \in J_i, \quad i = 1, 2, \ldots, n.$$

Of course, whenever $P(x) = 0$, then the admissible schedule x is feasible. But if there is no admissible schedule satisfying all the resource constraints, then the optimal solution of (P) corresponds to an admissible solution minimizing the *shortages of resources*.

Several different resource-constrained scheduling problems have been studied in the literature. The problem (RCSP) studied in the paper has been introduced by Jeffcoat and Bulfin (1993). The Cumulative Scheduling Problem (CUSP) studied by Baptiste et al. (1999) is closely related. Each task is characterized as in the (RCSP), but only one resource is considered. The objective is to decide if there exists a schedule of the tasks verifying the capacity constraint on the number of units of the resource available during each period and all the time constraints of the tasks. The variant of this problem where more than one resource is required has been studied by Carlier and Pinson (1998).

In the Resource-Constrained Scheduling (RCS) problem described by Srivastav and Stangier (1997), each task is characterized as in the (RCSP) above except that no deadline for completion has to be satisfied. The objective in this problem is to schedule the starting time of the tasks in order to minimize the latest completion time (or makespan). Zhang et al. (2004) study the special case where there is no ready time specified for the tasks and where 2 resources are used.

The Resource-Constrained Project Scheduling Problem (RCPSP) is probably the most widely studied problem of this type. Here, the tasks are part of a project. Instead of having ready time and deadline for completion, the tasks are related to each other through precedence relationships. Several variants of this problem exist according to the different types of resources used, according to the number of modes available to process the tasks, according to the objective function used, etc. A complete classification of the different variants can be found in Brucker et al. (1999).

Several problems can be formulated as (RCSP). Jeffcoat and Bulfin (1993) mention a foundry scheduling problem where the tasks correspond to lots to be produced. Each lot is using one of several parallel production lines and requires different raw materials and different kinds of manpower available in limited quantities. Reduced inventory space and shipping commitments induce ready times and deadlines for the tasks. Brucker and Knust (2001) show that several timetabling problems can be formulated as (RCPSP). But in some of these timetabling problems, there is no precedence relationship between tasks, and hence they can be formulated as (RCSP). In the formulation of the basic high-school timetabling problem (Schaerf 1999b; de Werra 1985), the lectures correspond to the tasks, and the classrooms and the teachers to the resources. Similarly, in the university course timetabling problem (Aubin and Ferland 1989; Burke and Petrovic 2002; Petrovic and Burke 2004; Schaerf 2006; Schaerf 1999a; de Werra 1985), the lectures correspond also to the tasks, and the classrooms, the teachers, and the students to the resources. This formulation allows simulating the conflicting situations where some teachers would have to teach or where some students would have to take more than one lecture simultaneously. Note then in these two examples of timetabling problems there is no ready time or deadline for any task.

In Sect. 2, we characterize the neighborhood structure introduced in Jeffcoat and Bulfin (1993) that we use to implement the neighborhood search (or local search) techniques. Then we summarize the simulated annealing method of Jeffcoat and Bulfin (1993), three threshold accepting methods and a tabu search to solve the (RCSP). Finally, a technique to generate an initial solution is introduced. In Sect. 3, we give three improving strategies: two different multistart diversification strategies and the variable neighborhood search technique. The numerical efficiency of the methods is analyzed in Sect. 4 where we use three different sets of randomly generated problems; the first set is used to fix the parameters of the methods, the second one, to select the best improving strategy for each method, and the third one, to compare

the quality of the solutions generated by the methods and their variants when combined with an improving strategy. Finally, concluding remarks are included in the conclusion. The numerical results indicate that the simulated annealing method in Jeffcoat and Bulfin (1993) improved with a variable neighborhood search is the best solution method.

2 Neighborhood search techniques (NST)

A *neighborhood search technique* (NST)(or local search technique) (Ferland et al. 1996; Reeves 1996) is an iterative procedure to move from an admissible schedule x to a new one x' in its neighborhood $N(x)$ until some admissible schedule x^* satisfying some stopping criterion is found. In general, the neighboring solutions $x' \in N(x)$ are generated by slightly modifying x. In our implementation we use the neighborhood introduced by Jeffcoad and Bulfin (1993).

Denote by $x(i, \ell)$ and $x(i, r)$ the schedules derived from x as follows:

$$x(i, \ell) = [x_1, x_2, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_n],$$

$$x(i, r) = [x_1, x_2, \ldots, x_{i-1}, x_i + 1, x_{i+1}, \ldots, x_n];$$

i.e., task i is started one period earlier and one period later in $x(i, \ell)$ and $x(i, r)$, respectively. If $x_i - 1 \geq r_i$ (i.e., if $x(i, \ell)$ is admissible), then $x(i, \ell)$ is included in $N(x)$. Similarly, if $x_i + 1 \leq f_i - d_i + 1$, $x(i, r)$ is included in $N(x)$. Thus the size of the neighborhood $N(x)$ is at most equal to $2n$. To generate $N(x)$, we consider each task i sequentially to obtain $x(i, \ell)$ and $x(i, r)$. Note that additional neighborhood structures are introduced in Sect. 3.3 to specify the variable neighborhood search approach.

The most straightforward NST is the *descent method*. At each iteration, the best admissible schedule $x' \in N(x)$ (i.e., $x' = \arg\min_{z \in N(x)} P(z)$) is selected as the current solution of the next iteration. The procedure stops whenever $P(x) = P(x')$; i.e., whenever a first local minimum x is reached. Other neighborhood search techniques have been developed to avoid being trapped in a first local minimum. Variants of some of these techniques are introduced next to solve problem (P).

2.1 Simulated annealing

Simulated annealing allows nondescent modification to avoid being trapped in a local minimum. Originally the approach was used to simulate the evolution of an unstable physical system toward a thermodynamic stable equilibrium point at a fixed temperature. Kirkpatrick et al. (1983) and Cerny (1985) were the first using it to solve optimization problems.

At each iteration of this probabilistic technique, an admissible schedule x' is selected randomly in the neighborhood $N(x)$ of the current schedule x. This schedule x' replaces x as the current solution if $\Delta P = P(x') - P(x) < 0$. But x' can replace x even if $\Delta P \geq 0$ (i.e., no improvement of P is induced by moving from x to x') according to a probability decreasing with the value of ΔP and with the number of iterations already completed. More specifically, x' replaces x with probability $e^{-\Delta P/T}$ where the parameter T (referred to as the temperature factor) decreases with the number of iterations completed.

We use a more deterministic implementation proposed by Jeffcoat and Bulfin (1993). For a given value T of the temperature, instead of selecting randomly the trial solutions x' in $N(x)$, we rather try to modify sequentially the starting time x_i of each task i. Each time we modify the value T of the temperature, we select a different random permutation of the indices $\{1, 2, \ldots, n\}$ corresponding to the order in which the tasks are considered sequentially.

A parameter $R \in (0, 1)$ is used to modify the temperature ($T := T \cdot R$). Even if this cooling schedule cools more rapidly than the logarithmic schedule proposed by Van Laarhoven et al. (1992), the numerical results in Jeffcoat and Bulfin (1993) indicate that it performs well. The stopping criterion is specified in terms of a maximum number (*itermax*) of different values of the temperature and in terms of a maximum number (*countmax*) of successive values of the temperature where the value PC of the current solution before cycling through all tasks i is not improved during the cycle. For the sake of completeness, the procedure is summarized in Fig. 1.

2.2 Threshold accepting methods

Threshold accepting methods (Dueck 1993; Dueck and Scheuer 1990) are also iterative procedures that can be seen, in some sense, as deterministic variants of simulated annealing. At each iteration, an admissible schedule x' is selected randomly in the neighborhood $N(x)$ of the current solution x. Then we rely on the value of an auxiliary function $\gamma(x, x')$ and on a threshold value dr to decide if x' replaces x as the current solution; i.e., x' replaces x if $\gamma(x, x') \leq dr$. Several variants can be specified according to different ways of specifying $\gamma(x, x')$ and dr.

Our implementation of the threshold accepting approach is a straightforward modification of the simulated annealing procedure of Fig. 1 where the probabilistic test in 4, and 5 of Step 1 is replaced by a threshold test.

In this paper we consider three different variants introduced in Dueck (1993); Dueck and Scheuer (1990). In the "*standard threshold accepting method*", the threshold value dr is updated as the temperature value T is modified in the simulated annealing; i.e.,

$$dr := a \cdot dr$$

Initialization

Let x^o be an initial admissible schedule.
Let $x^* := x := x^o$.
{x and x^* denote the current and the best known solutions, respectively}.
Let T^o be an initial temperature.
Let $T := T^o$; iter := count := 0.

Step 1 (For a given value T of the temperature)

$PC := P(x)$.
Generate a permutation Γ of the indices $\{1, 2, \ldots, n\}$.
For each task i (considered in the order specified by Γ)

1. **If** $P(x^*) = 0$, **then** Stop
2. **If** $x(i, \ell)$ is admissible (i.e., if $x_i - 1 \geq r_i$), **and**
   If $\Delta P = P(x(i, \ell)) - P(x) < 0$, **then**
   $x := x(i, r)$
   go to 6
3. **If** $x(i, r)$ is admissible (i.e., if $x_i + 1 \leq f_i - d_i + 1$), **and**
   If $\Delta P = P(x(i, r)) - P(x) < 0$, **then**
   $x := x(i, r)$
   go to 6
4. **If** $x(i, \ell)$ is admissible (i.e., if $x_i - 1 \geq r_i$), **then**
   let $\Delta P = P(x(i, \ell)) - P(x)$
   select $r \in (0, 1)$ randomly
   If $r < e^{-\Delta P/T}$, **then**
   $x := x(i, \ell)$
   go to 6
5. **If** $x(i, r)$ is admissible (i.e., if $x_i + 1 \leq f_i - d_i + 1$), **then**
   let $\Delta P = P(x(i, r)) - P(x)$
   select $r \in (0, 1)$ randomly
   If $r < e^{-\Delta P/T}$, **then**
   $x := x(i, r)$
6. **If** $P(x) < P(x^*)$, $x^* := x$

Step 2

$T := T \cdot R$
If $P(x) < PC$, **then** count := 0
iter := iter +1
count := count +1

Step 3 (Stopping criteria)

If iter > *itermax* **or** count > *countmax*, **then** Stop.
Otherwise, repeat Step 1.

Fig. 1 Simulated annealing procedure

where $a \in (0, 1)$ is a parameter of the procedure. Furthermore, the threshold function $\gamma(x, x')$ is

$$\gamma(x, x') = P(x') - P(x).$$

Hence x' replaces x if it is not deteriorating the objective function by more than the threshold value dr.

In the "*great deluge method*", the threshold value dr can be seen as the level of water, and x' replaces x if its value $P(x')$ is below the level of water, independently of the value $P(x)$. Hence

$$\gamma(x, x') = P(x').$$

The initial value dr^o of the threshold is equal to $P(x^o)$, and the threshold value dr is updated linearly according to the parameter δ; i.e.,

$$dr^o := P\big(x^o\big),$$
$$dr := dr - \delta.$$

Finally, in the "*maximal deterioration method*", x' replaces x if and only if its value $P(x')$ does not deteriorate the value $P(x^*)$ of the best known admissible solution by more than a threshold value μ. Hence

$$\gamma(x, x') = P(x')$$

and

$$dr := P(x^*) + \mu$$

where μ is a parameter of the procedure.

2.3 Tabu search

This solution procedure was developed independently by Glover (1986) and Hansen (1986). At each iteration, the best admissible schedule $x' \in N(x)$ is selected to be the current solution of the next iteration. As long as $P(x') < P(x)$, the procedure behaves like the descent method. But if $P(x') \geq P(x)$, then moving from x to x' induces no improvement or even a deterioration of the objective function $P(x)$. Nevertheless, this allows to move out of a local minimum in a different way than the simulated annealing approach. It is then interesting to compare numerically the two approaches for solving problem (P).

Now, since the value of the objective function $P(x)$ is not necessarily monotone decreasing during the resolution, a safeguard against cycling is required. This is provided through a memory mechanism forbidding to return to a solution already visited. Hence a short term *tabu list* is used to keep the modifications recently used to generate the sequence of current solutions. More specifically, suppose that the element $x' \in N(x)$ selected as the current solution for the next iteration is $x(i, \ell)$ or $x(i, r)$. Then, in the new schedule, i is initiated at $x_i - 1$ or $x_i + 1$ rather than at x_i. To provide a safeguard against cycling we forbid task i to be initiated at x_i in any current solution of the next $|TL|$ iterations, where $|TL|$ is a fixed scalar. The mechanism is implemented by using a cyclic list TL that is updated at each iteration by including the pair (i, x_i) corresponding to the task i modified and its current starting period x_i. Since the length $|TL|$ of the tabu list is fixed, if the list is full when (i, x_i) is put into the list, then the oldest element is eliminated from it. An admissible schedule $z \in N(x)$ is said tabu (and cannot be selected as the current solution for the next iteration) if $z = x(i, \ell)$ and the pair $(i, x_i - 1) \in LT$ or if $z = x(i, r)$ and the pair $(i, x_i + 1) \in LT$.

The mechanism has the drawback of making tabu other admissible solutions that were not generated before. Now, since some of these tabu solutions might be very good, we introduce an *aspiration criterion* to bypass the tabu status of such solutions. Hence even if $z \in N(x)$ is tabu, whenever its value $P(z)$ is smaller than $P(x^*)$, the value of the best solution x^* generated so far (i.e., $P(z) < P(x^*)$), z becomes the best solution (replacing x^*) and the current solution (replacing x).

Finally, the stopping criteria are specified in terms of a maximum number (*itermax*) of iterations and in terms of a maximum number (*countmax*) of consecutive iterations where the shortages of resources in the objective function $P(x)$ do not improve.

In our numerical experimentation we compare the strategy of using the whole neighborhood to identify the current solution for the next iteration against the one using only a subset $N_v(x) \subset N(x)$. Here v denotes a sequence of indices of a subset of tasks, and these indices are used sequentially to generate the elements of $N_v(x)$. We consider subsets of different size including $n'(n' \leq n)$ tasks selected randomly, and the sequence v is a random permutation of the indices of these tasks.

We also compare numerically a "standard" variant where we generate completely the neighborhood $N_v(x)$ before selecting the current solution for the next iteration with a more "aggressive" variant where the generation of $N_v(x)$ is interrupted whenever the aspiration criterion is satisfied, in which case this solution better than the current best is used as the current solution for the next iteration. Note that in the aggressive variant, if the iteration is interrupted, we continue to run through the indices in v during the next iteration before generating a new sequence v.

2.4 Initial solution

To generate randomly the initial solution for the different NST, we use the following straightforward constructive method. First we order the tasks according to some criterion, and then we schedule the tasks sequentially in order to use efficiently the resources still available.

The ordering criterion is specified in terms of the resources. Denote by c_i the total number of units of resources required to complete task i; i.e., for each $i = 1, 2, \ldots, m$,

$$c_i = d_i \sum_{\ell=1}^{i} a_{i\ell}.$$

The tasks are scheduled sequentially in decreasing order of their values c_i (i.e., the tasks requiring larger numbers of units are scheduled first). Each task i is completed during

its admissible period where the largest number of units of resources is still available; i.e., the task i is initiated at period $s \in J_i$ such that

$$s = \arg\max_{j \in J_i} \left\{ \sum_{t=j}^{j+d_{i-1}} \sum_{\ell=1}^{L} ba_{\ell t} \right\}$$

where $ba_{\ell t}$ denotes the number of units of resource ℓ still available in period t before scheduling task i.

3 Improving strategies

Several strategies have been proposed to improve the efficiency of neighborhood search techniques (NST). In this paper we analyze three different strategies to search more extensively the domain of admissible schedules by leading the search to unexplored regions of the domain. These strategies are diversification tools.

3.1 Random diversification (RD)

In this approach, the NST is executed several times using different initial solutions to solve a problem. The different initial solutions are generated randomly. In our implementation, we use a variant of the procedure in Sect. 2.4. In order to generate several initial solutions, instead of scheduling the tasks i in decreasing order of their c_i, we select the next task to be scheduled randomly using a distribution where the probability of selecting task i is proportional to its value c_i (*roulette whell*).

3.2 First order diversification (FOD)

The *first order diversification* (FOD) strategy was first introduced in Kelly et al. (1994) for the quadratic assignment problem. It is also a multistart procedure using the current local minimum x^* to generate a new initial solution. The strategy is to move away from the current admissible schedule x^* by modifying sequentially the current starting period of some tasks i to $x_i^* - 1$ or $x_i^* + 1$. At each iteration of the process, we select the task i for which modifying its current starting period induces the smallest deterioration or the largest improvement of the objective function P. The admissible schedule generated after modifying the starting time of a fixed percentage of the tasks become the new initial solution.

3.3 Variable neighborhood search approach

The *variable neighborhood search* (VNS) approach was introduced by Hansen and Mladenović (2001, 1997). The VNS is also a multistart approach, but it explores different neighborhoods of the best known solution x^* instead of using different initial solutions to apply a NST several times.

A set of neighborhood structures N^k, $k = 1, 2, \ldots, K$, have to be specified a priori. At each iteration, a local minimum x'' is generated using a NST where the initial solution x' is selected randomly in $N^k(x^*)$, and where the neighborhood structure N^k is used. If the local minimum x'' is better than x^* (i.e., if $P(x'') < P(x^*)$), then x'' replaces x^*, and the neighborhood structure N^1 is used for the next iteration. Otherwise, the neighborhood structure N^{k+1} is used for the next iteration unless $k = K$, in which case N^1 is used for the next iteration. Note that we return to structure N^1 whenever x'' is better than x^* because, in general, the complexity of using any neighborhood structure increases with the value of the index k.

3.3.1 *Neighborhood structures*

In our implementation we consider four different structures. The first three are nested ($N^1 \subset N^2 \subset N^3$) and the fourth one is specified differently.

The structure N^1 is the neighborhood structure described in Sect. 2. N^2 and N^3 are straightforward extensions of N^1:

$$N^2(x) = N^1(x) \cup \{\text{admissible schedules } x(i, \ell^2) \text{ and } x(i, r^2):\ i = 1, 2, \ldots, n\}$$

where

$$x(i, \ell^2) = [x_1, x_2, \ldots, x_{i-1}, x_i - 2, x_{i+1}, \ldots, x_n]$$

and

$$x(i, r^2) = [x_1, x_2, \ldots, x_{i-1}, x_i + 2, x_{i+1}, \ldots, x_n].$$

Similarly,

$$N^3(x) = N^2(x) \cup \{\text{admissible schedules } x(i, \ell^3) \text{ and } x(i, r^3):\ i = 1, 2, \ldots, n\}$$

where

$$x(i, \ell^3) = [x_1, x_2, \ldots, x_{i-1}, x_i - 3, x_{i+1}, \ldots, x_n]$$

and

$$x(i, r^3) = [x_1, x_2, \ldots, x_{i-1}, x_i + 3, x_{i+1}, \ldots, x_n].$$

When we generate these neighborhoods, the neighbor schedules associated with task i are generated in the following order:

(i) for $N^2(x) : x(i, \ell^2), x(i, r^2), x(i, \ell), x(i, r)$,
(ii) for $N^3(x) : x(i, \ell^3), x(i, r^3), x(i, \ell^2), x(i, r^2), x(i, \ell), x(i, r)$.

The elements of the structure $N^4(x)$ are generated by "switching" the starting periods of two tasks. Hence, for all pair (i_1, i_2) of tasks $(1 \le i_i, i_2 \le n)$, denote

$$x(i_1, i_2) = [x_1, x_2, \ldots, x_{i_1-1}, x_{i_2}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_1}, x_{i_2+1}, \ldots, x_n],$$

and $x(i_1, i_2) \in N^4(x)$ if it is an admissible schedule (i.e., $x_{i_2} \in J_{i_1}$ and $x_{i_1} \in J_{i_2}$). This neighborhood is generated according to a sequence of pairs of tasks rather than according to a sequence of tasks as for N^1, N^2 and N^3 (i.e., the sequence v in the tabu search and Γ in the other methods include a sequence of pairs of tasks). Furthermore, for the tabu search, the tabu list is modified by including the pair of elements (i_1, x_{i_1}) and (i_2, x_{i_2}) whenever $x(i_1, i_2)$ is selected to replace the current solution x. Also, any solution $x(i_1, i_2)$ is tabu if (i_1, x_{i_2}) or (i_2, x_{i_1}) is an element of the tabu list.

In our numerical experimentation we consider two different implementations: the "nested" variant (VNSN) using the three nested structures N^1, N^2, and N^3, and the "switching" variant (VNSS) using the structures N^1, N^2, and N^4. The purpose is to see if it is better to increase the number of nested structures (adding N^3) or to diversify with a different kinds of neighborhood like N^4. We are not trying to identify the best number of different neighborhood structures to use. This would require additional numerical tests.

4 Numerical results

Three different sets of problems are used to complete the numerical experimentation. We determine the proper values of the parameters for each method with a first set of 50 problems. Then a second set of 50 other problems is used to select the best improving strategy for each method. Finally we compare numerically all the methods and all the methods combined with their best strategy with a third set of 100 other problems. The problems in each of these sets are generated randomly using a variant of the approach proposed in Jeffcoat and Bulfin (1993) that can be summarized as follows.

Each problem has $n = 100$ tasks, uses $L = 4$ types of resources over an horizon of $|J| = 100$ periods. For each task i, the duration d_i is a random integer numbers in $[1, 20]$, and the resources required are $a_{i1} = 1$, and random integer numbers in the intervals $[0, 5]$, $[0, 10]$, and $[0, 15]$ for resource types 2, 3, and 4, respectively.

The ready time r_i and the deadline for completion f_i of each task i together with the number of units $B_{\ell j}$ of each resource ℓ available during each period j are determined in such a way that the following schedule s is admissible and feasible (i.e., $P(s) = 0$). This schedule is generated as follows. The first task starts in period 1 (i.e., $s_1 = 1$), the second task starts in the period following completion of the first task (i.e., $x_2 = s_1 + d_1$), and so on. Now, if scheduling task i after completing task $(i-1)$ would induce that task i finishes later than period 100 (i.e., if $(s_{i-1} + d_{i-1}) + d_i > 100$), then task i starts in period 1 (i.e., $s_i = 1$), and the process continues until all tasks are scheduled. Then for each task i,

$$r_i = \max\{1, s_i - u_i\},$$
$$f_i = \min\{100, s_i + d_i + v_i\}$$

where u_i and v_i are random integers in $[1, 10]$. Furthermore, to guarantee that s is feasible, the resource availabilities are specified as follows:

$$B_{\ell j} = \sum_{i \in E_j(s)} a_{i\ell} + w_{\ell j}$$

where $w_{\ell j}$ is a random integer number in the intervals $[0, 1]$, $[0, 5]$, $[0, 10]$, $[0, 15]$ for $\ell = 1, 2, 3$, and 4, respectively.

Note that the problems are generated as in Jeffcoat and Bulfin (1993) except for the values of the resource availabilities. Indeed in Jeffcoat and Bulfin (1993)

$$B_{\ell j} = \max_{t \in J} \left\{ \sum_{i \in E_t(s)} a_{i\ell} \right\}$$

inducing easier problems to solve in general. Furthermore, we had to complete our numerical experimentation using randomly generated problems since we are not aware of real data available.

All the tests were completed on a Pentium II having an AMD Athlon 750 MHz processor.

4.1 Parameters setting for the method

Each problem of the first set of 50 problems is solved once. The details of this thorough analysis to fix the parameters of the methods can be found in (Bouffard 2003), but in this paper we limit the presentation to a summary of the approach used and of the values selected for the parameters.

In order to complete a fair comparison of the methods, we determine the values of the parameters *itermax* and *countmax* of the stopping criteria such that they run for the same length of time. These values are determined for a short (3 s), a medium (6 s), and a long (30 s) length of time. Furthermore, different values of the other parameters are also selected for each length of time. Finally, to measure the performance of each selection of parameters, we use the average value of the objective function and the number of problems where the best value is achieved with respect to the others.

For the simulated annealing, we compare the efficiency of several pairs of values for the initial temperature T^o and for the parameter R to modify the temperature. The values selected for the different length of time are summarized in

Table 1 Values of the initial temperature T^o and of the parameter R for simulated annealing

Length of time (s)	T^o	R	itermax	countmax
3	5	0.994	600	20
6	7	0.997	1200	15
30	10	0.99925	10 000	150

Table 2 Values of the initial threshold d_{r_o} and the parameter a for the standard threshold accepting

Length of time (s)	d_{r_o}	a	itermax	countmax
3	10	0.9966	750	15
6	10	0.9983	1450	20
30	25	0.99952	9000	550

Table 3 Values of the initial threshold d_{r_o} and the parameter a for the great deluge method

Length of time (s)	δ	itermax	countmax
3	0.80	750	200
6	0.40	1500	400
30	0.085	10 000	1975

Table 4 Values of the parameter μ for the maximal deterioration method

Length of time (s)	μ	itermax	countmax
3	7	600	500
6	7	1200	1000
30	7	6000	200

Table 5 Length of the tabu list $|TL|$ for different average running times

Length of time (s)	$\|TL\|$	itermax	countmax
3	70	1500	450
6	70	3000	1000
30	80	17 000	5750

Table 1. A similar approach is used to select the values of the parameters of the different variants of the threshold accepting methods. They are summarized in Tables 2, 3 and 4.

The tabu search method is more difficult to calibrate because the number of parameters is larger. This leads us to set these sequentially. First the size $|TL|$ of the tabu list is selected. The results are summarized in Table 5 for three different lengths of time. Then, comparing different sizes for the set v, the results indicate that using 30% of the tasks to specify the set v is the best approach.

In (Bouffard 2003) we also compare the efficiency of using a fixed length tabu list $|TL|$ with that of using a "variable length tabu list" (Taillard 1991) where only the $|\overline{TL}|$ last elements of the list are used to verify the tabu status of a schedule. The value of $|\overline{TL}|$ is a random integer value satisfying the relation

$$t_{\min} \leq |\overline{TL}| \leq |TL|$$

where $t_{\min} = \lfloor 0.8|TL| \rfloor$. Using the first set of 50 problems, the results show that it is better to use a "variable length tabu list".

Finally, the results in Bouffard (2003) also show that the "aggressive" variant is more efficient in general than the "standard" one. Hence, we continue the numerical experimentation with the "aggressive" variant of the tabu search using 30% of the tasks to specify the subset v, and a "variable length tabu list".

4.2 Selecting the improving strategies

Each of the 50 problems of the second set is solved once to determine the best improving strategy (RD, FOD, VNSN, or VNSS) to use with each method having its parameters set as in Sect. 4.1. For the first order diversification (FOD) strategy, we modify the starting time of 75% of the tasks to generate the new initial solution.

A table is associated with each method where, for each strategy, we indicate the average value $\overline{P(x^*)}$ of the best schedules generated for this set of problems, and the number #BS of problems for which the strategy generates the best schedule with respect to the others, and the average number #RC of times that the corresponding method is restarted (for RD and FOD) or the number of times that the neighborhood structure is changed (for VNSN and VNSS). Furthermore, each strategy is used for a period of 30 seconds to solve each problem.

Whenever an improving strategy is combined with a NST, we have to specify the length of time that the NST is applied each time it is restarted or each time the neighborhood structure is changed. This has an impact on the number of times (intensity level) that the strategy is applied. It seems interesting to analyse if it is more efficient to use an improving strategy with a higher or a lower intensity level. Here we consider two different intensity levels: *a low level or a high level* where the NST is applied for a length of 6 or 3 seconds each time it is restarted or each time the neighborhood structure is changed, respectively.

The best improving strategy for the different methods are identified according to the results in Tables 6, 7, 8, 9 and 10. The results summarized in Table 11 indicate that there is no uniformly dominating improving strategies. The two strategies VNSS and RD are not selected as best for any method. It is interesting to note that the nested variant (VNSN) of the

Table 6 Simulated annealing: average value $\overline{P(x^*)}$, number of problems $\#BS$ where the best schedule is achieved, and number of restarts $\#RC$ according to the improving strategy and the intensity level

Improving strategy	Intensity level	$\overline{P(x^*)}$	#BS	#RC
RD	low	10.9	2	4.6
	high	12.3	0	9.8
FOD	low	10.9	4	4.0
	high	12.7	4	8.9
VNSN	low	7.3	18	3.2
	high	**6.5**	**29**	6.1
VNSS	low	15.4	0	2.0
	high	10.6	7	3.3

Table 7 Standard threshold accepting method: average value $\overline{P(x^*)}$, number of problems $\#BS$ where the best schedule is achieved, and number of restarts $\#RC$ according to the improving strategy and the intensity level

Improving strategy	Intensity level	$\overline{P(x^*)}$	#BS	#RC
RD	low	19.7	11	4.1
	high	19.5	6	9.0
FOD	low	**18.3**	**14**	4.0
	high	18.8	**14**	8.8
VNSN	low	21.7	7	2.0
	high	19.0	10	3.9
VNSS	low	21.7	7	2.0
	high	19.0	10	4.0

Table 8 Great deluge method: average value $\overline{P(x^*)}$, number of problems $\#BS$ where the best schedule is achieved, and number of restarts $\#RC$ according to the improving strategy and the intensity level

Improving strategy	Intensity level	$\overline{P(x^*)}$	#BS	#RC
RD	low	29.2	1	11.0
	high	40.3	0	23.6
FOD	low	27.0	6	9.5
	high	35.6	3	18.7
VNSN	low	**21.5**	**23**	7.1
	high	**21.5**	17	14.0
VNSS	low	23.8	10	6.7
	high	25.2	6	13.5

variable neighborhood search strategy is more efficient than (or at least as efficient) the switching variant (VNSS) for all the neighborhood search techniques.

4.3 Comparing the methods and their improving strategies

Now we use the third set of 100 problems to compare the methods and their combinations with their best improving strategies as identified in Sect. 4.2. Here also, each problem is solved once.

Table 9 Maximal deterioration method: average value $\overline{P(x^*)}$, number of problems $\#BS$ where the best schedule is achieved, and number of restarts $\#RC$ according to the improving strategy and the intensity level

Improving strategy	Intensity level	$\overline{P(x^*)}$	#BS	#RC
RD	low	22.8	5	4.1
	high	25.1	5	9.1
FOD	low	20.3	13	4.0
	high	**20.2**	**17**	9.0
VNSN	low	22.8	7	3.8
	high	21.8	13	7.3
VNSS	low	22.8	7	3.8
	high	23.5	8	7.0

Table 10 Tabu search: average value $\overline{P(x^*)}$, number of problems $\#BS$ where the best schedule is achieved, and number of restarts $\#RC$ according to the improving strategy and the intensity level

Improving strategy	Intensity level	$\overline{P(x^*)}$	#BS	#RC
RD	low	34.6	7	4.4
	high	42.7	3	10.0
FOD	low	**27.6**	**23**	7.1
	high	31.1	8	15.5
VNSN	low	34.5	8	3.1
	high	34.2	5	8.4
VNSS	low	35.0	7	3.1
	high	36.0	4	9.1

Table 11 Methods and their improving strategy selected

Method	Improving strategy	Intensity level
simulated annealing (SA)	VNSN	high
standard threshold accepting (STA)	FOD	low
great deluge (GD)	VNSN	low
maximal deterioration (MD)	FOD	high
tabu search (TS)	FOD	low

In Table 12 we compare the methods globally in terms of the average value $\overline{P(x^*)}$ of the best schedules generated and the number #BS of problems for which the method generates the best solution with respect to the others. The results in Table 12 indicate a clear prevalence of the simulated annealing method mostly when combined with the nested variable neighborhood search (VNSN).

For any solution procedure, if the value $P(x^*)$ of the best solutions for the 100 problems would be normally distrib-

Table 12 Global comparison of the methods

Procedures	$\overline{P(x*)}$	SD	INT	#BS
SA	9.6	7.8	[1.8, 17.4]	36
SA & VNSN	7.0	5.0	[2.0, 12,0]	69
STA	21.7	7.4	[14.3, 29.1]	0
STA & FOD	20.1	7.9	[12.2, 28.0]	1
GD	20.3	7.5	[12.8, 27.8]	1
GD & VNSN	20.2	8.7	[11.5, 28.9]	1
MD	21.3	8.0	[13.3, 29.3]	0
MD & FOD	20.4	8.3	[12.1, 28.7]	1
TS	26.5	8.9	[17.6, 35.4]	0
TS & FOD	26.1	10.1	[16.0, 26.2]	0

uted, then it is well known that the probability

$$P(x^*) \in \mathrm{INT} = \left[\overline{P(x^*)} - SD, \overline{P(x^*)} + SD\right],$$

is equal to 0.683 (where SD denotes the standard deviation of the distribution of $P(x^*)$). Hence, if we use INT to compare the behavior of the different solution procedures, the results in Table 12 illustrate even more strongly the prevalence of the SA & VNSN method. Indeed, for all methods other than SA (where INT(SA & VNSN) $\subset$ INT(SA)) and GD & VNSN (where INT(GD & VNSN) slightly intersect with INT(SA & VNSN)), their interval INT is completely on the right hand side of INT(SA & VNSN).

In addition we compare the procedures by pairs in Table 13. The entry (i, j) is equal to the difference between the number of problems where procedure i generates a solution better or as good as procedure j, and the number of problems where procedure j generates a solution better as good as procedure i. Hence, whenever the symmetric entries (i, j) and (j, i) have absolute value close to 0, then the efficiency of the two procedures is quite similar. If the value of entry (i, j) is positive, then according to this criterion, the efficiency of the procedure associated with row i over the one associated with column j increases with the value of entry (i, j).

The results in Table 13 indicate that the simulated annealing method combined with the nested variable neighborhood search is also the most efficient procedure according to this criterion. Furthermore, the simulated annealing method is neatly more efficient than all the other neighborhood search techniques even if they are combined with any improving strategy.

The second group of more efficient procedures are the threshold accepting methods combined with their improving strategies. Among these, the great deluge method combined with the nested variable neighborhood search is the most efficient. Now, since the threshold accepting methods can be seen as some kind of deterministic variants of the simulated annealing method, we may conjecture that the probabilistic behavior of the simulated annealing method has a positive impact on the efficiency of the method.

Finally, our implementation of the tabu search method even combined with the first order diversification is far less efficient than any of the threshold accepting methods. It is worth noticing that simulated annealing methods seem also to generate better results than tabu search methods for the resource-constrained project scheduling problem (RCPSP) closely related to our problem (P). Indeed, Alcaraz and Maroto (2001) report that the Bouleimen and Lecoq simulated annealing method (Bouleimen and Lecocq 1998) is the third best method after their genetic algorithm and Hartman genetic algorithm (Hartmann 1998) to solve RCPS. For the problems in the Project Scheduling LIBrary (PSBLIB) (Kolisch et al. 1995) having 30 and 60 tasks, Bouleimen and Lecoq simulated annealing method is two to three times better than Baar et al. tabu search method (Baar et al. 1998) with respect to the criteria based on the average deviation from the optimal or the best known solution.

Table 13 Comparing the procedures by pairs

	SA	SA & VNSN	STA	STA & FOD	GD	GD & VNSN	MD	MD & FOD	TS	TS & FOD
SA	0	−34	86	80	85	82	91	85	96	93
SA & VNSN	34	0	96	97	93	90	97	98	99	100
STA	−86	−96	0	−16	−14	−15	−12	−16	45	32
STA & FOD	−80	−97	16	0	0	1	13	−11	43	40
GD	−85	−93	14	0	0	−7	8	1	50	33
GD & VNSN	−82	−90	15	1	7	0	22	16	45	53
MD	−91	−97	12	−13	−8	−22	0	−8	53	50
MD & FOD	−85	−98	16	11	−1	−16	8	0	53	51
TS	−96	−99	−45	−43	−50	−45	−53	−53	0	−17
TS & FOD	−93	−100	−32	−40	−33	−53	−50	−51	17	0

5 Conclusion

The numerical results indicate that Jeffcoat and Bulfin simulated annealing method (Jeffcoat and Bulfin 1993) generates better solutions than other local search techniques like threshold accepting methods and the tabu search. This is consistent with the fact that the simulated annealing approach performs better than the tabu search approach for RCPSP (Alcaraz and Maroto 2001). Furthermore, the performance of the simulated annealing method can be improved with a variable neighborhood search approach. Finally, even if we combine the threshold accepting methods and the tabu search method with improving strategies like random diversification, first order diversification, or variable neighborhood search, the numerical results indicate that their performance remains inferior to that of the simulated annealing method.

Acknowledgements The authors are grateful to the referees for their comments to improve a previous version of the paper.

References

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

Aubin, J., & Ferland, J. A. (1989). A large scale timetabling problem. *Computers and Operations Research*, *16*, 67–77.

Baar, T., Brucker, P., & Knust, S. (1998). Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem. In S. Voss, S. Martello, I. Osman, & C. Roucairol (Eds.), *Meta-heuristics: advances and trends in local search paradigms for optimization* (pp. 1–18). Dordrecht: Kluwer Academic.

Baptiste, P., Le Pape, C., & Nuijten, W. (1999). Satisfiability and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, *92*, 305–333.

Blazewicz, J. (1978). Complexity of computer scheduling algorithms under resources constraints. In *Proceedings first meeting AFCET-SMF on applied mathematics* (pp. 169–178), Palaiseau, Poland.

Bouffard, V. (2003). *Méthodes heuristiques pour un problème d'ordonnancement avec contraintes sur les ressources*. Master thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada, April 2003.

Bouleimen, K., & Lecocq, H. (1998). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem. In G. Barbasoğlu, S. Karabati, L. Ozdamar, & G. Ulusoy (Eds.), *Proceedings of the sixth international workshop on project management and scheduling* (pp. 19–22). Istanbul: Boğaziçi University Pringing Office.

Brucker, P., & Knust, S. (2001). Resource-constrained project scheduling and timetabling. In E. Burke & W. Erben (Eds.), *Lecture notes in computer science: Vol. 2079. PATAT III* (pp. 277–293). Berlin: Springer.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research*, *112*, 3–41.

Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, *140*, 266–280.

Carlier, J., & Pinson, E. (1998). Jackson's pseudo-preemptive schedule for the $P_m/r_i, q_i/C_{\max}$ scheduling problem. *Annals of Operations Research*, *83*, 41–58.

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*, 41–51.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, *19*, 151–162.

Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of computational Physics*, *104*, 86–92.

Dueck, G., & Scheuer, T. (1990). Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, *90*, 161–175.

Ferland, J. A., Hertz, A., & Lavoie, A. (1996). An object-oriented methodology for solving assignment-type problems with neighborhood search techniques. *Operations Research*, *44*, 347–359.

Glover, F. (1986). Future path for integer programming and links to artificial intelligence. *Computers and Operations Research*, *13*, 533–549.

Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming. In: *Congress on numerical methods in combinatorial optimization*, Capri, Italy.

Hansen, P., & Mladonović, N. (2001). Variable neighborhood search: principles and applications. *European Journal of Operations Research*, *130*, 449–467.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, *45*, 733–750.

Jeffcoat, D. E., & Bulfin, R. L. (1993). Simulated-annealing for resource-constrained scheduling. *European Journal of Operational Research*, *70*, 43–51.

Kelly, J. P., Laguna, M., & Glover, F. (1994). A study of diversification strategies for the quadratic assignment problem. *Computers and Operations Research*, *21*, 885–893.

Kirkpatric, S., Gelatt, C. D. Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

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

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, *24*(11), 1097–1100.

Petrovic, S., & Burke, E. K. (2004). University timetabling. In J. Y. -T. Leung (Ed.), *Handbook of scheduling: algorithms, models, and performance analysis*. Boca Raton: CRC Press, Chap. 45.

Reeves, C. R. (1996). Modern heuristic techniques. In V. J. R. Smith, I. H. Osman, C. R. Reeves, & G. D. Smith (Eds.), *Modern heuristic search methods* (pp. 1–25). New York: Wiley.

Schaerf, A. (1999a). A survey of automated timetabling. *Artificial Intelligence Review*, *13*, 87–127.

Schaerf, A. (1999b). Local search techniques for large high school timetabling problems. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, *29*, 368–377.

Schaerf, A. (2006). Measurability and reproducibility in timetabling. In E. K. Burke & H. Rudova (Eds.), *Proceedings of the 6th international conference on the practice and theory of automated timebabling* (pp. 53–62). Brno: Masaryk University.

Srivastav, A., & Stangier, P. (1997). Tight approximations for resource constrained scheduling and bin packing. *Discrete Applied Mathematics*, *79*, 223–245.

Taillard, E. (1991). Robust tabu search for the quadratic assignment problem. *Parallel Computing*, *17*, 443–455.

Van Laarhoven, P. J. M., Aarts, E. H. L., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, *40*, 113–125.

Zhang, G., Cai, X., & Wong, C. K. (2004). Some results on resource constrained scheduling. *IIE Transactions*, *36*, 1–9.