Int J Adv Manuf Technol (2009) 41:358–366
DOI 10.1007/s00170-008-1486-5

ORIGINAL ARTICLE

Hybrid-directional planning: improving improvement heuristics for scheduling resource-constrained projects

Kuo-Ching Ying · Shih-Wei Lin · Zne-Jung Lee

Received: 28 May 2007 / Accepted: 19 March 2008 / Published online: 7 May 2008
© Springer-Verlag London Limited 2008

Abstract The resource-constrained project scheduling problem (RCPSP) has been of a continuing interest and challenge for researchers and practitioners since its advent. The formidable computational requirements of the RCPSP have resulted in numerous attempts to develop heuristic procedures, leading to the interest in improvement heuristics. Traditionally, such heuristics constructed a schedule by the scheme of forward, backward, or bidirectional planning directions. In this paper, we introduce a hybrid-directional planning that can make all improvement heuristics (e.g., meta-heuristics) more effective in solving the RCPSP. To validate its effectiveness, the proposed scheme is incorporated into three popular meta-heuristics, including genetic algorithm, simulated annealing, and Tabu search. A comprehensive numerical investigation shows that the performance of such meta-heuristics is significantly increased by using the hybrid-directional planning, which indicates that such a hybrid planning direction will hopefully encourage researchers and practitioners to apply it to different improvement heuristics for solving the RCPSP.

Keywords Scheduling · Project management · Improvement heuristics · Meta-heuristics

K.-C. Ying (✉)
Department of Industrial Engineering and Management Information, Huafan University,
Taipei, Taiwan
e-mail: kcying@huafan.hfu.edu.tw

S.-W. Lin
Department of Information Management, Chang Gung University,
Tao-Yuan, Taiwan

Z.-J. Lee
Department of Information Management, Huafan University,
Taipei, Taiwan

1 Introduction

The use of project management continues to grow rapidly today. As one of the most challenging problems in project management, scheduling has been receiving considerable attention not only from researchers but also from practitioners. In the past few decades, the field of project management theory and practice has made tremendous strides forward. However, as projects grow in complexity, finding realizable schedules that efficiently exploit scarce resources continues to be a challenging task within project management.

The resource-constrained project scheduling problem (RCPSP) has captured the attention of many researchers and practitioners since it was first proposed by Kelley [1]. Many attempts have been made on the development of different advanced methods for the RCPSP. However, scheduling the RCPSP is an unenviable task since it belongs to the class of *NP*-hard combinatorial optimization problems [2]. Although optimal solutions of RCPSP can be obtained via exact procedures, such as linear programming [3, 4] and branch-and-bound method [5, 6], these methods may still take a prohibitive amount of computation, even for a problem with a moderate size. For practical purposes, it is more appropriate to look for heuristic procedures that generate a near-optimal solution at a relatively minor computational expense.

Excellent recent overviews of heuristic procedures for the RCPSP can be found in the investigation by Hartmann and Kolisch [7]. Currently available heuristic procedures for solving this problem in literature can be broadly classified into two categories, namely constructive heuristics and improvement heuristics. Constructive heuristics start from an empty schedule and add then activities one by one until a feasible schedule is achieved. To that purpose, the activities are usually ranked by using the priority rule, which determines the order that the activities are added to the schedule. Hundreds of constructive heuristics have been studied in the literature [8]. However, a major drawback of constructive heuristics is the non-robustness of their solutions. There exists no constructive heuristic that outperforms all other constructive heuristics given a variety of different performance criteria and manufacturing environments. In addition to priority rules, another major component of constructive heuristics is the schedule generation scheme (SGS), which determines the way in which a feasible schedule is constructed by assigning starting times to the different activities. The serial SGS and parallel SGS constitute the two basic schedule generation schemes. The serial SGS, which dates back to a paper by Kelley [1], sequentially adds activities to the schedule until a feasible complete schedule is obtained. Contrary to the serial SGS, the parallel SGS [6] iterates over the different decision points at which activities can be added to the schedule. It is well known from the literature that the parallel SGS generates non-delay schedules which do not always contain a schedule with minimum makespan. Hence, in this study, we use the serial SGS for mapping an activity list into a schedule.

On the other hand, improvement heuristics start from an initial feasible schedule and then provide operations for repeatedly transforming a schedule into an improved one [9]. These operations are repeated until a locally optimal solution is obtained. Improvement heuristics usually yield better solutions than construction heuristics [10]. Recently, some improvement heuristics, namely meta-heuristics, try to avoid getting stuck in a locally optimal solution by allowing an intermediate deterioration of the project duration. The meta-heuristic algorithmic framework can be applied to various optimization problems with slight modifications [11]. Methods of this type include genetic algorithm (GA) [12–14], simulated annealing (SA) [15–17], Tabu search (TS) [18], and the artificial immune system (AIS) [19]. Literature shows that these methods can obtain very good results for the RCPSP.

Although a number of studies have been done in the RCPSP to seek for an efficient and effective algorithm, little attention has been placed on the planning direction. The majority scheduling algorithms applied the forward planning to construct a feasible schedule. However, different scheduling algorithms can benefit form the backward and bidirectional planning schemes. The purpose of the present study is to ascertain the effect of incorporating the hybrid-directional planning (i.e., to construct schedules by mixing forward, backward, and bidirectional schemes) as compared to incorporating the sole planning direction into improvement heuristics for solving the RCPSP.

For this objective to be achieved, the article is structured as follows. In the next section, the RCPSPs to be addressed in this paper are defined. Section 3 describes the proposed hybrid-directional planning. Section 4 provides a brief introduction to the combinations of the hybrid-directional planning and three popular meta-heuristics, namely GA, SA, and TS. The computational results of the proposed approaches are provided, and performances are compared with other planning directions in Sect. 5. Finally, this study concludes with suggestions for future research.

2 Problem definition

The RCPSPs to be addressed in this paper are defined as follows: A project consists of a set of activities $J=\{1,\ldots,n\}$ where each activity j has a given duration d_j in a number of periods. Each activity has to be processed without interruption to complete the project (i.e., non-preemptive). Due to technological restrictions, precedence constraints between the activities may exist. Furthermore, resource constraints have to be observed. There are K renewable resource types. The availability of each resource type $k=1,\ldots,K$ in each time period is R_k unit. Each activity j requires r_{jk} units of resource k during each period of its duration where $k=1,\ldots,K$. Whenever the availability of a resource type is not sufficient to satisfy the requirements, those activities must not be executed simultaneously. All parameters are assumed to be non-negative integer valued. With these definitions, the objective is to find a non-preemptive schedule of the activities so that the precedence and resource constraints are satisfied and the makespan (i.e., project duration) is minimized.

3 The proposed hybrid-directional planning

Forward planning considers the normal time direction in scheduling the activities over time. When constructing a feasible schedule, we start with the dummy start activity and gradually schedule all activities until the dummy end activity has been assigned a starting time. In backward planning [20], the activities of any RCPSP instances could equally be scheduled in reverse time direction (i.e., starting with the dummy end activity and gradually scheduling all activities until the dummy start activity has been assigned a starting time). This can easily be obtained by reversing all precedence relations and using the reverse priority list on the resulting network. The resulting starting times can then easily be adjusted so that the starting time of the dummy start activity equals 0. Finally, the application of all possible left shifts will result in an active schedule for the original problem instance.

Bidirectional planning [21] constructs schedules in the forward and backward directions simultaneously by combining both the forward and backward scheduling schemes. To that purpose, both forward and backward priority lists should be used in the bidirectional planning. During each iteration of the bidirectional parallel scheduling scheme, the decision point of each direction with the smallest difference is considered. In the case of a tie, both decision points are considered. At each decision point, the activities are scheduled in the order of forward or backward priority list. The activities are assigned start or completion times that coincide with the decision point in the corresponding direction. This is continued until no more activities can be scheduled to start or finish at these decision points. Then the decision point(s) is (are) updated and a new iteration takes place until all activities have been scheduled. Finally, the bidirectional schedule can be translated into an active schedule by shifting the backward scheduled activities to the left in the order of their start times.

Besides performing the forward, backward and bidirectional planning, the scheduling schemes can be extended to the hybrid-directional planning (i.e., to construct schedules by mixing the forward, backward and bidirectional schemes). This is achieved by adding a planning direction (*PD*) code behind the schedule list as follows:

$$PD = \begin{cases} 1, & \text{forward directional planning} \\ 2, & \text{backward directional planning} \\ 3, & \text{bidirectional planning} \end{cases}$$

where PD=1, 2 or 3 is selected by the uniform distribution with equal probability.

For instance, the forward, backward, and bidirectional planning sequences of five-job schedules 3-1-5-2-4 can be simply represented as (3, 1, 5, 2, 4 | 1), (4, 2, 5, 1, 3 | 2), and (3, 1, 5, | 4, 2 | 3), respectively.

4 Combinations of hybrid-directional planning and meta-heuristics

This study proposes a hybrid-directional planning to improve improvement heuristics to solve the RCPSP. This section elaborates the distinct features of combining the hybrid-directional planning and three meta-heuristics. Subsection 4.1 explains the GA approach, Subsection 4.2 mentions the SA approach, and Subsection 4.3 discusses the TS approach.

4.1 Genetic algorithm approach

The genetic algorithm (GA) is based on the mechanisms of biological evolution and natural genetics. The approach was first proposed by Holland [22] and tried to implement the idea of survival of the fittest in the field of combinatorial optimization. A genetic algorithm starts with a population of $P(g)$ random schedules. New schedules are created by combining two existing schedules and by applying a mutation on a newly created schedule. A mutation typically consists of a unary neighborhood operator that is applied with a small probability. After creating a number of new schedules, the new population consists of the G_{Max} best schedules out of the newly created and /or existing schedules and the algorithm continues on the current population. This process typically continues until a number of populations have been created and evaluated. Following the termination of GA, the schedule generated by the algorithm can be subsequently derived by choosing the best individual found during the GA process.

For the application of the GA approach to solve the RCPSP by incorporating into the hybrid-directional planning, the chromosome presentation, the initial population, the reproductive strategy, the crossover operator, the mutation operator, and the termination condition are described as follows.

- **Chromosome representation and the initial population-** The search space for this chromosome representation is the permutation of n numbers in the set {1, 2, …, n}, where the i^{th} number in the permutation denotes the activity is the i^{th} activity to be processed. In addition, we add a genetic code behind the permutation of n numbers which represents the planning direction (*PD*). For example, a forward planning sequence of five-job schedules 3-1-5-2-4 is simply represented as (3, 1, 5, 2, 4, 1). Each individual in the initial population is generated randomly with a *PD* code that is selected from the forward, backward, or bidirectional planning directions with equal probability. A collection of *Pop_size* individuals form a population. We set *Pop_size*=6 in the experiments of this study.
- **Reproductive Strategy**-The likelihood that an individual will be selected as parent increases with the individual's fitness value. The objective function of this study is the minimization of the makespan. Therefore, the smaller the makespan an individual has, the higher the fitness value of the individual. Given a population P and an objective function value v_j for each $j \in P$, the fitness value f_j for each $j \in P$ is defined as $f_j = 1/(1+v_j)$ and thus the probability of j's selection is $f_j / \sum_{k \in P} f_k$
- **Crossover**-Two parents have a probability p_c of undergoing crossover. The newly generated individual has chromosomes consisting of genes from either of its parents. This paper uses the Order Crossover (OX) proposed by David [23], and set p_c=0.8 in the experiments of this study. When crossover occurs, the *PD* code is selected from either of its parents with equal probability.

– **Mutation**-New offspring has a probability p_m=0.3 of undergoing mutation. The mutation starts by selecting one job and comparing swap and insertion options. The selected job is paired with all other jobs individually to identify the swap or insertion operation leading to the largest improvement of the objective function value. The operation that yields the largest improvement of the objective function is chosen and then performed. When mutation occurs, the *PD* code is changed to the other two planning directions with equal probability.
– **Termination condition**- Two termination conditions are applied in this paper. One is that the process is terminated if the best solution equals the solution gotten by critical path method without resource constraints observed, and the other is that the process may be executed for a fixed number of maximum number of total generations (possible schedules) $S_{\max}$={1,000, 5,000, 50,000}.

Based on the above, a step-by-step procedure of the proposed GA can be described as follows:

```
Algorithm GA ($G_{Max}$, Pop_size, $p_c$, $p_m$)
{
    g←0; $F_{best}$←∞;
    initialize population P(g);
    while (g < $G_{Max}$) {
        recombine (reproductive, crossover and mutation) P(g) to yield C(g);
        evaluate C(g);.
        select P(g + 1) from P(g) and C(g);
        g←g + 1;
    }
}
```

The GA begins with four parameters, G_{Max}, *Pop_size*, p_c and p_m, where G_{Max} denotes the maximum number of generations in GA evolution, *Pop_size* is the population size, p_c represents the probability of crossover, and p_m represents the probability of mutation. The number of generations already evolved (g) and the objective function value of the best individual (F_{best}) are set to zero and ∞, respectively. Let $P(g)$ denote the population at generation g. The population of the initial generation $P(0)$ is produced randomly. The evolution continues until the number of generations equals to G_{Max}. In each generation, the production of the offspring is repeated *Pop_size* times. Parents with smaller objective function values have a higher probability of being chosen to produce the offspring. Each reproduction with a chosen pair of parents underscores the crossover operator with the probability p_c, and underscores the mutation operator with probability p_m. After repeating the reproduction *Pop_size* times, better offspring $C(g)$ will replace parents with worse fitness values to form the population of the next generation.

4.2 Simulated annealing approach

The ideas of simulated annealing (SA) date back to a paper by Metropolis et al. [24], who described an algorithm to simulate the annealing process of material. Then the approach was extended by Kirkpatrick et al. [25] to SA. Annealing is the process through which slow cooling of metal produces good and low-energy-state crystallization, whereas fast cooling produces poor crystallization. The optimization procedure of simulated annealing reaching a (near) global minimum mimics the crystallization cooling procedure.

In essence, SA draws an initial random schedule to start its search. A new schedule is created in the neighborhood of the current schedule and it is accepted if it is better than the current schedule. If the new schedule does not improve upon the current schedule, it is accepted with a probability that depends on the magnitude of the deterioration and on a temperature parameter. This temperature typically starts at a relatively large value and is reduced during the SA procedure to decrease the probability that non-improving schedules are accepted. This process is repeated until time runs out, and a number of schedules have been created or no new solution is accepted for a certain time.

The proposed SA approach is described as follows. The solution presentation is the permutation of n numbers in the set {1, 2, …, n}, where the ith number in the permutation denotes the job is the ith job to be processed. In addition, we add a *PD* code behind the permutation of n numbers. Let S be the set of feasible schedules and X be the current schedule, where $X \in S$. The ObjFun(X) denotes the calculation of the objective function value of X. The set $N(X)$ is the set of schedules neighboring X. Neighborhood is sampled either by swap or insertion with equal probability. The swap is done by randomly selecting the ith and jth number of X and then swapping the values of these two numbers directly. The insertion is done by randomly selecting the ith number of X and inserting it into the position immediately preceding the jth number of X. A step-by-step procedure of the proposed SA approach can be described as follows:

```
Algorithm SA ($S_{max}$, $C_{iter}$, $T_0$, $T_F$, α)
    $T \leftarrow T_0$; $c \leftarrow 1$;
    Select a feasible solution $X$ randomly, $X \leftarrow [x_1, x_2, \ldots, x_n]$;
    $X_{best} \leftarrow X$; $F_X \leftarrow$ objFun($X$); $F_{best} \leftarrow F_X$;
    While ($T \geq T_F$ or $c \leq S_{\max}$) {
        Repeat $C_{iter}$ times {
            $Y \leftarrow X$;
Let $i$ and $j$ be random integers between 1 and $n$ ($i \neq j$);
Generate random variable $r$ uniformly distributed in (0, 1);
If ($r<0.5$) {
    The $i^{th}$ number is swapped with the $j^{th}$ number in $Y$;
}
Else {
    The $i^{th}$ number in $Y$ is selected and inserted into the position immediately preceding the $j^{th}$ number in $Y$;
}
$F_{cur}$=objFun($Y$);
If ($F_{cur}<F_x$) {
    $P_{Accept}=1$;
}
Else {
    $\Delta = F_{cur} - F_x$; $P_{Accept}=exp(-\Delta/T)$;
}
Generate random variable $r$ uniformly distributed in (0, 1);
If ($r<P_{Accept}$) {
    $X \leftarrow Y$; $F_x=F_{cur}$;
}
If ($F_{cur}<F_{best}$) {
    $X_{best}=Y$; $F_{best}=F_{cur}$;
}
}
$T \leftarrow \alpha T$; $c \leftarrow c1$;
}
```

The SA in this study begins with five parameters, namely S_{max}, C_{iter}, T_0, T_F and α, where $S_{\max}$={1,000, 5,000, 50,000} is the maximum number of total iterations, $C_{iter} = (10 \times \text{activities numbers})$ denotes the number of iterations the search proceeds with a particular temperature, $T_0=100$ represents the initial temperature, $T_F=1$ represents the final temperature that stops the SA procedure if current temperature is lower than T_F, and $\alpha=0.95$ is the coefficient controlling the cooling schedule. First, the current temperature T is set to be the same as $T_0=100$. Next, an initial feasible schedule X is randomly generated with a *PD* code which is selected from the forward, backward or bidirectional planning direction with equal probability. The current best schedule X_{best} is set equal to X, and the best objective function value obtained so far F_{best} is set equal to objective function value of X (i.e. F_X). T is decreased once after running C_{iter} iterations from the previous decrease according to a formula $T \leftarrow \alpha T$, where $0<\alpha<1$ For each iteration, the next feasible schedule Y is generated from X either by swapping or by insertion with equal probability. When generating the next feasible schedule, the *PD* code is selected from the forward, backward or bidirectional planning directions with equal probability.

Let Δ denote the difference between objective function value of the feasible schedule X and Y; that is Δ=objFun(Y)-objFun(X). At any stage in the algorithm, the probability of replacing X with Y, where X is the current solution and Y is the next solution, given that $\Delta>0$, is $e^{-\Delta/T}$. This is accomplished by generating a random number $r \in [0,1]$ and replacing the feasible schedule X with Y if $r < e^{-\Delta/T}$. Meanwhile, if $\Delta<0$, the probability of replacing X with Y is 1. After T is lower than T_F or the process has been executed for a fixed number of maximum number of total iterations (S_{max}) the algorithm is terminated. The X_{best} records the best schedule as the algorithm progresses. Following the termination of SA, the optimal schedule can thus be derived by X_{best}.

4.3 Tabu search approach

Glover [26, 27] extended the steepest descent approach to the Tabu search (TS), which is an iterative improvement approach designed for getting global optimum solutions to combinatorial optimization problems. As in the steepest descent method, all schedules in the neighborhood of the current schedule are evaluated and the schedule with the smallest objective function value is chosen. However, TS is continuing the search, even though the new schedule does not improve the previous one. This characteristic of TS makes it possible that the series of schedules includes cycles (i.e., a sub-series of schedules is repeated time and time again). To avoid cycling to some extent, moves which would bring us back to a recently visited schedule should be forbidden or declared tabu for a certain number of iterations. This is accomplished by keeping the attributes of the forbidden moves in a list, called a tabu list. Any neighborhood move that belongs to the tabu list is therefore forbidden, except if the move would lead to a new best schedule. This phenomenon is called an aspiration criterion. The size of the tabu list must be large enough to prevent cycling but small enough not to forbid too many moves. More refined versions and a large number of successful applications of TS can be found in Glover [28].

The remaining part of this subsection describes how to apply the TS approach to solve the RCPSP by incorporating into the hybrid-directional planning. First, schedule representation and the initial schedule for the problem under consideration are discussed. Next, the neighborhood of the current schedule is defined and then tabu restriction is explained. The last part of this subsection deals with the aspiration criteria for the TS.

– **Initial schedule**-The initial feasible schedule X is randomly generated with a *PD* code, which is selected from the forward, backward, or bidirectional planning directions with equal probability.

– **Neighborhood**-The TS approach assumes that all schedules that can be reached from any current schedule can be classified as the neighborhood. If one alters the current schedule by swapping the sequence of two jobs or inserting a job into another position, the result is a member of the current solution's neighborhood. In this paper, we randomly choose i^{th} number of the current solution X, where the neighborhood of X is the set of all possible swaps and insertions of the i^{th} number with the other numbers. When generating the neighborhood of the current solution X, the *PD* code is selected from the forward, backward, or bidirectional planning directions with equal probability.
– **Tabu restrictions**: The most distinctive feature of the Tabu search is that recent moves are not allowed to be reversed; recent moves are forbidden (tabu). The search remembers by forming a list called the tabu list, consisting of all of these tabu moves. The tabu list is implemented as a FIFO (first in, first out) list of fixed length L in this paper.
– **Aspiration criterion**: In our implementation, the following commonly used aspiration criterion is employed to cancel the effect of a tabu status on a move: if a certain move is forbidden by tabu restrictions and the aspiration criterion is satisfied, this move is allowed to override the tabu restriction (i.e., if a tabu move is better than the best solution obtained so far by the search, then one can take this move even though it is tabu).

Based on the above, a step-by-step procedure of the proposed TS approach can be described as follows:

```
Algorithm TS ($S_{max}$, $L$)
{
$c \leftarrow 1$;
Select a feasible solution $X$ randomly, $X \leftarrow [x_1, x_2, \ldots, x_n]$;
While $c \leq S_{max}$ {
    Select $i^{th}$ number in $X$ randomly, and find the neighborhood of $X$.
    Choose the best improving and non-tabu solution $Y$ in $N$(X) (if none exist, choose the least worsening solution) or a tabu solution $Y$ in $N$(X) which is better than the best solution obtained so far (Aspiration criterion);
    Update the tabu list based on $Y$;
    $X \leftarrow Y$;
    $c \leftarrow c1$;
}
}
```

The TS begins with two parameters, S_{max} and L, where S_{max}={1,000, 5,000, 50,000} denotes the maximum number of total iterations (possible schedules) and L=12 is the length of tabu list. After the initial schedule X is produced, the TS continues until the maximum number of iterations already run (i.e., c=S_{max}). In each iteration, the i^{th} number of X is chosen randomly. Then all possible swaps and insertions with other numbers of X are the neighborhood of X. At the same time, the *PD* code is selected from the forward, backward, or bidirectional planning directions with equal probability. The set N(X), the non-tabu neighborhood of X, is the best set of schedules obtained so far. Choose a best improving schedule Y in N(X), update the tabu list based on Y, and set the current schedule X equal to Y. Before the end of each iteration, the variable c is increased by 1. Following the termination of TS, the schedule generated by the algorithm can be subsequently derived by choosing the best solution found during the TS process.

5 Computational results and discussion

In the following, we report on results of comprehensive computational experiments. They have been performed on a computer with Xeon 3.40 GHz clock pulse and 4GB RAM. All procedures have been coded by means of Microsoft Visual C++ 6.0.

5.1 Test problem

The effectiveness of applying the proposed hybrid-directional planning to meta-heuristics for solving the RCPSP was verified by performing computational experiments on a well-known benchmark problem set, namely single-mode resource-constrained project scheduling problems (SRCPSP), established by Kolisch and Sprecher [29]. Each of the activities of the SRCPSP has to be performed in one prescribed way (mode) using specified amounts of the resources provided.

The SRCPSP contains four subsets, namely J30, J60, J90, and J120, in which the numbers after J represent the amounts of activities of these subsets' problems. The J30, J60, J90, and J120 subsets have 480, 480, 480, and 600 numbers of instances, respectively. These instances were systematically generated by the standard project generator (ProGen). The entire benchmark set including its detailed characterization is available on a public ftp-site, namely, PSPLIB at http://129.187.106.231/psplib/.

5.2 Computational results

For each case of instance and each combination of planning directions with meta-heuristics, ten times (trials) of independent simulations were run. Then, the following four performance indexes were calculated:

- **Av. # Opt.**: The average number of optimal solutions for each benchmark problem subset, i.e., $\frac{\text{\# of optimal solutions of all trails}}{10}$.
- **Max. # Opt.**: The maximum number of optimal solutions achieved from each instance's best trail for each benchmark problem subset.
- **Av. Dev.**: The average relative deviation from optimal solutions for each benchmark problem subset, i.e., $\frac{\sum \text{deviations from optimal solutions of each trials}}{\text{\# of instances of the subset} \times 10} \times 100\%$.
- **Av. Max. Dev.**: The average relative maximum deviation of each instance's worst trail solution from optimal solution for each benchmark problem subset, i.e., $\frac{\sum \text{deviations of each instance's worst trail solution from optimal solution}}{\text{\# of instances of the subset}} \times 100\%$.

In the above performance indexes, except benchmark problem subset J30, the optimal solutions of instances are not provided by the PSPLIB. Hence, we set the critical path value which is solved by the critical path method (CPM) without resource constraints observed as an estimated optimal solution of each instance for J60, J90, and J120.

The final results are listed in Tables 1, 2, and 3. On the whole, the proposed hybrid-directional planning can dramatically improve the performance of various metaheuristics for each combination of problem sizes and maximum numbers of total iterations. As Tables 1 and 2 show, the GA and SA yield solutions, by planning in hybrid-directional, are all better than by planning in the forward, backward, and bidirectional on the performance indexes of Av. # Opt., Max. # Opt., and Av. Dev.; Meanwhile, by planning in the hybrid-directional, the GA and SA obtain lower Av. Max. Dev. in 11 out of 12 and 10 out of 12 combination problems, respectively.

As revealed in Table 3, the TS yield solutions by planning in hybrid-directional are all better than by planning in the other three directions on the performance indexes of Av. # Opt. and Max. # Opt.; Moreover, by planning in the hybrid-directional, the TS gets lower Av. Dev. and Av. Max. Dev. in 11 out of 12 and 8 out of 12 combination problems, respectively.

6 Conclusions

The purpose of this study is to make a step towards establishing an effective direction planning to solve the RCPSP by improvement heuristics. In this study, we have proposed a hybrid–directional planning scheme that can be applied in any improvement heuristics to solve the RCPSP. In order to evaluate the effectiveness of the proposed scheme, different planning directions are incorporated into

Table 1 Computational results for the GA approach

S_{max}	Problem set	Av. # Opt.				Max. # Opt.				Av. Dev.				Av. Max. Dev.			
		Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid
1000	J30	242.7	217.9	238.5	278.2	248	222	242	284	2.31	3.12	2.56	1.81	17.50	23.26	16.67	13.43
	J60	217.8	179.9	219.2	237.3	221	184	224	242	16.13	16.35	15.39	15.02	123.38	124.68	119.48	119.48
	J90	210.1	197.1	222.6	235.1	213	203	225	240	16.00	15.88	15.09	14.83	120.34	118.64	120.34	118.64
	J120	19.5	23	31.2	40	24	26	36	42	47.62	45.74	44.15	43.13	254.55	231.68	232.32	237.37
5000	J30	266.2	240.4	258.8	306.9	272	244	262	317	1.79	2.46	1.98	1.31	13.64	22.81	16.67	11.36
	J60	232.8	200.9	231.2	260.1	236	208	236	265	15.03	15.19	14.30	13.92	122.08	116.88	120.78	115.58
	J90	230.7	218	241.6	264.3	235	225	245	269	14.85	14.66	13.83	13.55	120.34	118.64	120.34	118.64
	J120	31.6	33.4	52	55.5	36	37	60	69	44.65	42.83	41.12	40.50	239.39	231.31	230.69	222.22
50000	J30	300	263	274.8	341.4	305	269	284	348	1.76	1.93	1.57	0.80	13.64	22.81	16.67	11.11
	J60	256.3	221.1	248.1	278.9	260	229	251	282	13.50	14.05	13.36	12.93	116.88	119.48	115.58	115.58
	J90	256.1	240.3	259.1	284.3	262	247	262	290	13.54	13.67	12.93	12.64	115.25	113.56	111.86	111.86
	J120	51.1	49.2	75.7	83.1	55	53	78	86	41.32	40.40	39.00	38.54	222.22	221.78	219.80	219.19

Table 2 Computational results for the SA approach

S_{max}	Problem set	Av. # Opt.				Max. # Opt.				Av. Dev.				Av. Max. Dev.			
		Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid
1000	J30	257.2	235.2	246.1	321.6	262	241	251	328	2.02	2.62	2.28	1.16	13.64	22.81	16.67	10.2
	J60	228.6	200.6	223.9	269.7	232	205	231	275	15.55	15.59	15.29	14.27	128.57	124.68	124.68	120.78
	J90	223	219.5	230.2	271	227	223	233	274	15.49	15.16	14.78	14.11	120.34	118.64	120.34	118.64
	J120	25.8	30.8	36.1	55	28	33	38	59	64.13	60.69	61.48	60.17	239.60	233.33	237.62	232.32
5000	J30	275	251.7	260.3	347.4	278	255	262	351	1.58	2.17	1.93	0.78	13.64	22.81	16.67	8.62
	J60	244.1	215.3	239	284.4	247	220	243	287	14.65	14.76	14.47	13.57	120.78	119.48	118.18	115.58
	J90	246	236.6	248.5	288.8	249	239	253	293	14.66	14.30	13.91	13.43	120.34	113.56	115.25	113.56
	J120	44.9	44.7	56.9	73.8	50	51	59	80	61.93	60.80	59.06	58.35	237.62	231.68	232.32	231.31
50000	J30	297.2	269.7	278.9	373	301	273	283	377	1.30	1.85	1.59	0.53	13.64	22.81	16.67	5.95
	J60	262.4	226.5	250.8	298.5	265	228	252	302	13.82	14.02	13.71	12.91	119.48	114.29	119.48	116.88
	J90	263.3	253	260.6	302.5	264	257	263	305	13.74	13.50	13.27	12.77	116.95	113.56	115.25	113.56
	J120	61.3	64.3	74.7	98.6	66	70	77	102	59.43	57.31	57.42	56.64	228.28	222.22	230.30	227.27

Table 3 Computational results for the TS approach

S_{max}	Problem set	Av. # Opt.				Max. # Opt.				Av. Dev.				Av. Max. Dev.			
		Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid	Fwd.	Bwd.	Bid.	Hybrid
1000	J30	252.5	218.3	243.2	312.3	258	225	247	321	2.07	2.98	2.37	1.21	16.28	22.81	16.67	9.28
	J60	218.7	181.4	220	252.4	224	185	226	257	15.70	16.14	14.98	14.51	125.97	119.48	125.97	111.69
	J90	207.9	191.3	223.2	252.6	212	195	228	256	15.67	15.54	14.60	14.34	127.12	120.34	118.64	118.64
	J120	20.3	21.7	34.7	37	24	26	40	39	46.73	44.99	42.92	43.73	244.44	224.24	229.70	231.68
5000	J30	280.3	246.1	262.5	352.1	285	251	266	359	1.57	2.40	1.92	0.72	16.05	22.81	16.67	8.33
	J60	239.4	205.9	238.7	282.5	246	213	243	286	14.37	14.77	13.98	13.25	111.69	115.58	119.48	118.18
	J90	232.4	221.9	244.6	286.2	241	229	250	293	14.60	14.39	13.52	13.16	115.25	113.56	116.95	113.56
	J120	32.8	35.1	54.6	71.6	39	39	61	75	43.92	42.37	40.60	40.29	233.33	226.26	233.33	222.22
50000	J30	298.2	264.9	283	384.5	303	272	285	389	1.37	1.91	1.66	0.45	13.64	22.81	16.67	8.33
	J60	263	227.8	253.8	303.1	265	230	257	306	13.12	13.56	12.99	12.38	111.69	114.29	119.48	110.39
	J90	262.8	247.2	264	308	267	251	268	311	13.20	13.34	12.58	12.15	108.48	113.56	113.56	111.86
	J120	62.2	58.3	90.5	116.6	68	63	93	120	40.24	39.82	37.92	37.36	217.82	268.85	216.16	217.17

some meta-heuristics, including GA, SA, and TS. The results show a striking effect of scheduling direction on the performance of improvement heuristics to solve the RCPSP.

First of all, the experiments indicate that the planning direction has a considerable influence on the performance of meta-heuristics to solve the RCPSP. For most instances, the proposed hybrid–directional planning scheme can outperform the other planning directions. These analytical results strongly reveal that the hybrid-directional planning should also be included in other improvement heuristics such as the descent approach, the neighborhood search and other meta-heuristic-based approaches.

Furthermore, there has been no such planning direction of the forward, backward, and bidirectional that works well for all RCPSPs. This clearly suggests that such a hybrid-directional planning is worth exploring in the context of solving the RCPSP. It is worth pointing out that the proposed hybrid-directional planning is a general scheme for different improvement heuristics. The results obtained by this study will hopefully encourage practitioners to apply it to the real-world RCPSP.

Future research can extend this study in several possible ways. First, the developed hybrid-directional planning can be extended to multi-mode RCPSP. Second, research into this problem may be continued by applying the proposed hybrid-directional planning to other state-of-the-art improvement heuristics. Finally, using the hybrid-directional planning to the RCPSP with different performance criteria are worth examining further.

Acknowledgments The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financial support of this research under Contract No. NSC 95-2221-E-211-009.

References

1. Kelley JE (1963) The critical path method: resources planning and scheduling. In: Muth JF, Thompson GL (eds) Industrial scheduling. Prentice-Hall, Upper Saddle River, NJ
2. Błażewicz J, Lenstra J, Rinnooy KA (1983) Scheduling subject to resource constraints: Classification and complexity. Discrete Appl Math 5:11–24
3. Alvarez-Valdés R, Tamarit JM (1993) The project scheduling polyhedron: dimension, facets and lifting theorems. Eur J Oper Res 67:204–220
4. 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. Manage Sci 44:715–729
5. Stinson JP, Davis EW, Khumawala BM (1978) Multiple resource-constrained scheduling using branch and bound. AIIE Trans 10:252–259
6. Brucker P, Knust S, Schoo A, Thiele O (1998) A branch-and-bound algorithm for the resource-constrained project scheduling problem. Eur J Oper Res 107:272–288
7. Hartmann S, Kolisch R (2000) Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. Eur J Oper Res 127:394–407
8. Klein R (2000) Scheduling of resource-constrained projects. Kluwer, Dordrecht
9. Ying KC (2008) Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. Int J Adv Manuf Tech, DOI 10.1007/s00170-007-1104-y
10. Lin SW, Ying KC (2007) Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics. Int J Adv Manuf Tech 34:1183–1190
11. Ying KC, Lin SW (2007) Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems. Int J Adv Manuf Tech 33:793–802
12. Reddy JP, Kumanan S, Chetty OVK (2001) Application of Petri nets and a genetic algorithm to multi-mode multi-resource constrained project scheduling. Int J Adv Manuf Tech 17:305–314
13. Hartmann S (2002) A self-adapting genetic algorithm for project scheduling under resource constraints. Nav Res Log 49:433–448
14. Kumanan S, Jose GJ, Raja K (2006) Multi-project scheduling using a heuristic and a genetic algorithm. Int J Adv Manuf Tech 31:360–366
15. Cho J, Kim YD (1997) A simulated annealing algorithm for resource-constrained project scheduling problems. J Oper Res Soc 48:736–744
16. Bouleimen K, Lecocq H (2003) A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple modes version. Eur J Oper Res 149:268–281
17. Shukla SK, Son YJ, Tiwari MK (2008) Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling. Int J Adv Manuf Tech, DOI 10.1007/s00170-006- 0907-6
18. Thomas PR, Salhi S (1998) A tabu search approach for the resource constrained project scheduling problem. J Heuristics 4:123–139
19. Agarwal R, Tiwari MK, Mukherjee SK (2007) Artificial immune system based approach for solving resource constraint project scheduling problem. Int J Adv Manuf Tech, DOI 10.1007/s00170- 006-0631-2
20. Li KY, Willis RJ (1992) An iterative scheduling technique for resource-constrained project scheduling. Eur J Oper Res 56:370–379
21. Klein R (2000) Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects. Eur J Oper Res 127:619–638
22. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
23. David L (1985) Applying adaptive algorithms to epistatic domains. Proceedings of the 9th International Joint Conference on Artificial Intelligence. IEEE Computer Society Press, Los Angeles, California
24. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087–1092
25. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
26. Glover F (1989) Tabu search-Part I. ORSA J Comput 1:190–206
27. Glover F (1990) Tabu search-Part II. ORSA J Comput 2:4–32
28. Glover F, Laguna M (1997) Tabu search. Kluwer, Dordrecht
29. Kolisch R, Sprecher A (1996) PSLIB - A project scheduling problem library. Eur J Oper Res 96:205–216