J Math Model Algor (2010) 9:357–373
DOI 10.1007/s10852-010-9132-2

Tri-directional Scheduling Scheme: Theory and Computation

H. R. Yoosefzadeh · Hamed R. Tareghian · M. H. Farahi

Received: 24 October 2009 / Accepted: 7 April 2010 / Published online: 22 May 2010

Abstract In this paper we introduce a new scheduling scheme based on so called tri-directional scheduling strategy to solve the well known resource constrained project scheduling problem. In order to demonstrate the effectiveness of tri-directional scheduling scheme, it is incorporated into a priority rule based parallel scheduling scheme. Theoretical and numerical investigations show that the tri-directional scheduling scheme outperforms forward, backward and even bidirectional schemes depending on the problem structure and the priority rule used. Based on empirical evidence, it seems that as the number of activities are increased, the tri-directional scheduling scheme performs better irrespective of the priority rule used. This suggests that tri-directional scheme should also be applied within the category of heuristic methods.

Keywords Project management · Scheduling schemes · Heuristic algorithms · Bidirectional scheduling scheme

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is a generalization of the job shop problem and as such belongs to the class of NP-hard problems [2]. In a

H. R. Yoosefzadeh (✉) · H. R. Tareghian · M. H. Farahi
Faculty of Mathematical Sciences, Ferdowsi University of Mashhad,
P.O. Box 1159-91775, Mashhad, Iran
e-mail: ha_us346@stu-mail.um.ac.ir

H. R. Tareghian
e-mail: taregian@ferdowsi.um.ac.ir

M. H. Farahi
e-mail: farahi@math.um.ac.ir

typical RCPSP there are $j = 1, \ldots, N$ ($N = |\mathbf{J}|$) activities that are usually processed without preemption. Each activity $j \in \mathbf{J}$ has a fixed duration d_j. The execution of activities is governed by the *precedence constraints* that usually exist between them. Precedence constraints between activities can be represented by *minimum time lags* $\lambda_{ij} \geq 0$ between the starts of activities i and $j \in \mathbf{J}$. Notice that when $\lambda_{ij} = d_i$, the usual finish to start precedence is present between activities i and $j \in \mathbf{J}$. Overlapping of activities is possible, for instance when $0 \leq \lambda_{ij} < d_i$, i.e., activity j can start before activity i finishes, or when $\lambda_{ij} > d_i$, i.e., activity j must wait for $\lambda_{ij} - d_i$ periods after activity i finishes, before it can start.

Processing of activities is subject to the availability of resources with limited capacities. While being processed, activity $i \in \mathbf{J}$ uses a constant amount of u_{ir} units of resource type $r \in \mathbf{R}$, where $\mathbf{R} = \{1, \ldots, |\mathbf{R}|\}$. The availability of resources is constant during the planning horizon $t = 1, \ldots, T$, the availability of a type $r \in \mathbf{R}$ resource is restricted to a constant amount a_r. As such, *resource constraints* are also present. Therefore, activities which may be performed concurrently compete for scarce resources. Hence, whenever the availability of a resource type is not sufficient to satisfy the total requirements, those activities can not be executed simultaneously and resource constraints have to be observed. It can be easily verified that an RCPSP with $|\mathbf{R}|$ renewable resources each of which has an availability of one unit per period corresponds to a job shop problem with $M = |\mathbf{R}|$ machines (see [19]). The most common considered objective of the RCPSP is to schedule (allocation of the given resources to activities in order to determine the start, s_i and completion, f_i of all activities) the project's activities such that both their precedence and resource constraints are satisfied and the project's makespan is minimized.

The structure of a project is usually depicted as an acyclic activity-on-node network (AON) with nodes depicting activities and arcs representing the precedence constraints. The dummy activities 1 and N ($d_1 = d_N = 0$) in such a network represent the beginning and the end of the project.

The RCPSP which is frequently studied as an optimization problem was originally proposed in [17]. Since then various exact, heuristic and metaheuristic methods have been proposed to solve this problem. Being classified as NP-hard, only small-sized RCPSP instances with up to 100 activities can be solved exactly in a satisfactory manner [23]. Therefore, many heuristic procedures are applied to study large and/or complicated RCPSPs.

Optimal methods for the RCPSP basically involve the use of *mathematical programming* (e.g. [7]), and *implicit enumeration*; i.e. dynamic programming (e.g. [16]) and branch and bound (e.g. [4]). As mentioned previously, optimal methods are not really suitable for the real world RCPSPs.

Many heuristics (single-pass, multi-pass and sampling methods) and metaheuristics (e.g. tabu search, genetic algorithm) have been proposed for the RCPSP. In 2006, Kolisch and Hartman categorized a number of heuristic and metaheuristic methods to study and compare their performances based on standardized experimental design [13]. Most of the heuristic methods presented to solve the RCPSP so far can be broadly categorized as being *constructive* or *improvement* methods.

In constructive methods schedules are built gradually from scratch. Priority rules and scheduling generation schemes (SGS) are the two main components of constructive methods. SGS determines the way in which a feasible schedule is constructed by assigning starting times to activities. Based on the selected priority rule and the

SGS, activities are taken from a list of eligible activities and added to the partial schedule one by one. Eligible activities are those that satisfy both precedence and resource constraints. This is repeated until all activities are considered and the complete schedule is obtained. Many constructive methods have been presented and studied in the literature [9]. However, Kuo-Ching Ying et. al, states: "there is no single constructive heuristic that outperforms all other constructive heuristics given different performance criteria and planning environments" [15].

There are two SGSs, namely the *serial scheme* and the *parallel scheme*. Both generate feasible schedules (for more details see [11]). The serial SGS adds eligible activities to the partial schedule one by one. This is repeated until a complete schedule is obtained. In contrast, the parallel SGS adds one or more eligible activities to the partial schedule whenever state changes occur (ongoing activity (ies) terminates or some resources become available). This process is continued until the complete schedule is obtained.

In improvement methods the process of generating a schedule starts by improving the value of the objective function of an initial feasible solution. This solution is somehow transformed to a better one. This is done by applying a number of operations either to an individual or a subset of solutions. Operations that transform one solution into another usually consist of changing the ranking of the activities. In each step, only operations that lead to solutions improving the best known objective function value are allowed. In order to escape local optima, meta-heuristics allow intermediate deterioration of the objective function value. Again in order to generate a solution, activities are ranked and the SGS is applied. Metaheuristic methods include genetic algorithms (e.g. [6, 14], and [18]), simulated annealing (e.g. [3, 5] and [20]), tabu search (e.g. [21]) and the artificial immune system (e.g. [1]).

Constructive methods generate acceptable solutions using moderate processing efforts. However, improvement algorithms usually provide better quality solutions albeit with higher processing efforts [11]. As described, scheduling generation schemes are a fundamental part of both constructive and improvement methods.

In the literature, papers studying the effect of the scheduling direction are relatively scarce. Although forward scheduling is the most common scheme used in schedule generations, some recent studies indicate that backward and bidirectional schemes can improve the performance of the SGSs. The effect of incorporating hybrid-directional scheduling (i.e., to construct schedules by mixing forward, backward, and bidirectional schemes) as compared to incorporating only one planning direction was recently studied and it was shown that the performance of such meta-heuristics is significantly increased by using hybrid-directional planning [15]. In this paper, we describe how the performance of the schedule generation schemes can still be increased by tri-directional scheduling.

The rest of the paper is structured as follows. In Section 2, we present the mathematical model of the RCPSP. In Section 3 we briefly describe the parallel SGS when used within forward or bidirectional scheduling schemes. In Section 4, the new tri-directional (trd.) scheduling scheme is introduced. In order to evaluate different scheduling schemes under various directions (forward, bidirectional and tri-directional) most important priority rules are integrated with scheduling schemes and theoretical as well as computational experiments are presented and discussed in Section 5. Section 6 contains a summary and conclusions.

2 The Problem

A conceptual model for the general RCPSP (overlapping of activities is allowed) as described in Section 1 is given below:

$$\begin{aligned} & Min\ f_N \\ subject\ to & \\ & f_1 \geq 0 \\ & f_j - f_i \geq d_j - d_i + \lambda_{ij}\ \forall\ (i,\ j) \in \mathbf{H} \\ & \sum_{i \in \mathbf{Q}_t} u_{ir} \leq a_r \qquad t = 1,\ 2, \ldots,\ f_N;\ r = 1,\ 2,\ \ldots,\ |R| \end{aligned} \tag{1}$$

where $\mathbf{H}$ denotes the set of pairs of activities indicating precedence constraints, f_i is finish time of activity i ($i = 1, \ldots, N$) and $\mathbf{Q}_t$ denotes the set of activities in progress in time interval $]t-1, t]$: $\mathbf{Q}_t = \{i | f_i - d_i \leq t < f_i\}$. The underlying assumptions of model (1) are: (a) activities are numbered from 1 to N, where activities 1 and N are dummies; (b) no preemption of activities are allowed; (c) there are $|R|$ renewable resource types with u_{ir} the constant requirement of activity i for resource r, and a_r the constant availability of resource type r ($1 \leq i \leq N$, $1 \leq r \leq |R|$). Note that in this paper we have considered the usual RCPSP, i.e. $\lambda_{ij} = d_i$.

3 Scheduling Schemes

In this section, we briefly describe the scheduling scheme that we have adopted in our new tri-directional scheduling scheme, namely parallel scheduling scheme (PSS). Information about the other scheduling schemes can be found in [11] among others. As seen in Fig. 1, the parallel scheduling scheme iterates over some decision points that correspond to the completion times of already scheduled activities. Therefore, at most $N = |\mathbf{J}|$ decision points are considered. At each decision point, the unscheduled activities whose predecessors have completed are considered and are scheduled subject to no resource conflict occurring at that time. To illustrate the parallel scheduling scheme, we use the following notations and the algorithm of Fig. 1 which is adopted from Kolisch [11].

$\mathbf{R}$	Set of available resources
$\mathbf{C}_n$	Set of completed activities at stage n
$\mathbf{D}_n$	Set of eligible activities at stage n
πa_r	Remaining capacity of resource r at any stage
$\mathbf{P}_i$	Set of predecessors of activity i
$\upsilon(i)$	Priority value of activity i
t_n	Scheduling time at stage n
$\mathbf{J}$	Set of project activities
$\mathbf{A}_n$	Set of ongoing activities at stage n

It was observed that PSS searches in a smaller solution space than serial scheduling scheme (SSS) and with this drawback the solution space of PSS might not contain an optimal schedule in contrast to SSS [11]. However in [11], Kolisch remarked: *the parallel scheduling scheme performs better for "hard" problems* (with a high resource factor and/or low resource strength) *while the serial scheduling scheme is*

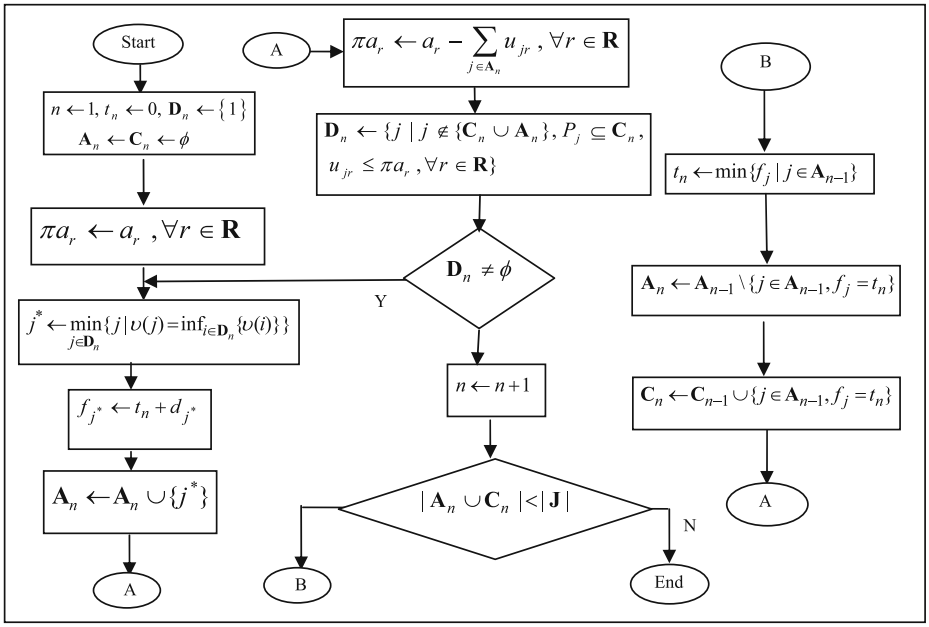

Fig. 1 Parallel scheduling scheme (PSS)

better for "easy" problems (with a low resource factor and/or high resource strength). In addition, Klein showed that using different scheduling directions will overcome the disadvantage of PSS in yielding *non-delay* schedules that for some instances may not include the optimal schedule (see [8]). As we believe that higher dimensional scheduling schemes demonstrate their capabilities in harder instances, we only adopt PSS in our proposed scheduling scheme.

4 Multiple Scheduling Directions

4.1 Bidirectional Parallel Scheduling Scheme

Schedules are constructed either in a manner in which an activity is scheduled when all its predecessors are scheduled (forward) or in a manner in which an activity is scheduled when all its successors are scheduled (backward). Forward and backward scheduling schemes are combined in bidirectional scheduling in order to construct schedules in both directions simultaneously. There are two lists (forward and backward) of activities, one for each scheduling direction, that are sorted according to some priority rule. Activities that either have no predecessors or their predecessors have already been scheduled are added to the forward list. Activities of the backward list are those that either have no followers or their followers have been scheduled in backward direction. In addition, there are two decision points, one for each direction (forward-τ_f and backward-τ_b). Two sets $ES_f\ (\tau_f)$ and $ES_b\ (\tau_b)$ contain the forward and backward eligible activities at τ_f and τ_b respectively. Activities from $ES_f\ (\tau_f)$ are assigned starting time at τ_f and activities from $ES_b\ (\tau_b)$ are assigned finishing

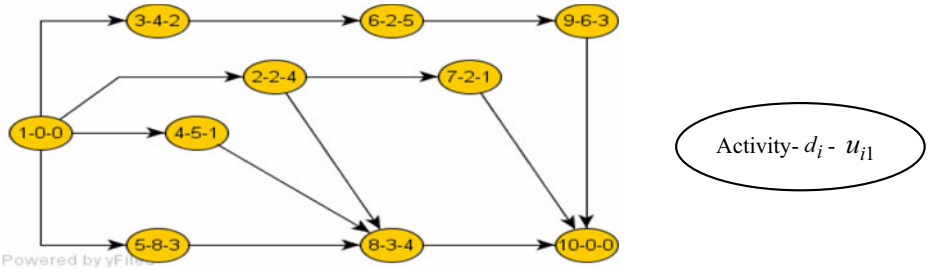

Fig. 2 An example of a project network

times at τ_b. This process is continued until no more activities can be scheduled at current τ_f and τ_b. Therefore, decision points are updated and iteration continues. Iterations stop when there are no more activities to schedule. For an activity that can be scheduled in either direction, the following *tie-breaking rule* is used to determine the direction that the activity is scheduled. If the difference between its start time in forward scheduling and the beginning of the project is smaller than the difference between its finish time in backward scheduling and project completion time, the activity is forward scheduled, otherwise it is backward scheduled. The complete schedule is obtained by left shifting the activities that are backward scheduled in the order of their starting times.

Example 1 Bidirectional PSS was applied to the project network shown in Fig. 2. In this project it is assumed that the per unit time availability of a single resource is 7 (see Figs. 3 and 4). After scheduling the dummy start and end activities, $\tau_f = 0$ and $\tau_b = 32$ (32 is the sum of all activities durations). Eligible sets are $ES_f\ (\tau_f = 0) = \{2,3,4,5\}$, and $ES_b\ (\tau_b = 32) = \{7,8,9\}$. Using shortest processing time (SPT), activity 2 is added to the forward partial schedule (FS) after which activity 7 and then activity 8 are added to the backward partial schedule (BS) respectively. No more activities can be scheduled at current τ_b, therefore $ES_b\ (\tau_b = 30) = \{9\}$. Between activities 3, 4, 5 and 9, activities 3 and 4 are forward scheduled. As no more activities can be scheduled at $\tau_f = 0$, we update the eligible set, i.e. $ES_f\ (\tau_f = 2) = \{5\}$. Among eligible activities, namely activity 5 and 9, activity 9 is backward scheduled. Backward eligible set is updated. Thus $ES_b\ (\tau_b = 29) = \{5\}$. Using the tie breaking rule, activity

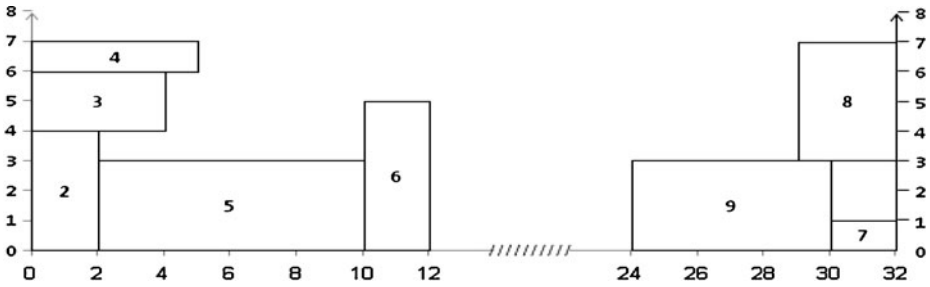

Fig. 3 Forward and backward schedules

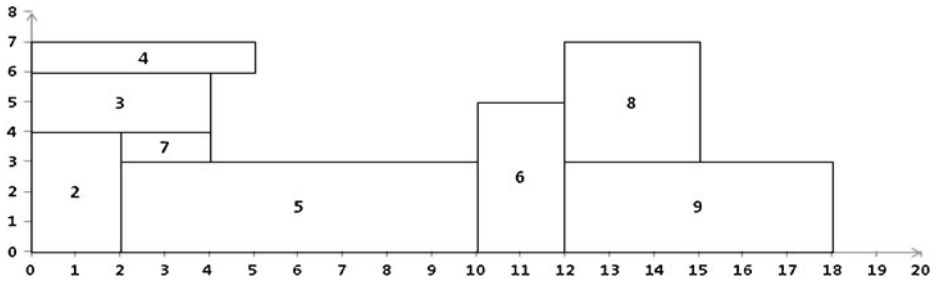

Fig. 4 Schedule obtained by bidirectional and forward scheduling scheme

5 is forward scheduled. Forward eligible set is updated, i.e. $ES_f\ (\tau_f = 4) = \{\}$. As forward eligible set is empty at this time, we extend the time until at least one activity becomes eligible. So $ES_f\ (\tau_f = 10) = \{6\}$ and $ES_b\ (\tau_b = 24) = \{6\}$. Using the tie breaking rule, activity 6 is forward scheduled. There is no more activity to be scheduled. The partial forward and backward schedules are shown in Fig. 3. In order to link FS and BS, we sort activities in BS in ascending order, by their starting times. We then shift the sorted activities to the left as far as the precedence and the resource constraints are not violated. The complete schedule is shown in Fig. 4 and as it can be seen the makespan is 18.

The example was also solved using forward scheduling scheme. The schedule obtained is the same as the schedule obtained by bidirectional scheme, i.e. Fig. 4.

4.2 Tri-directional Parallel Scheduling Scheme

Klein showed that scheduling direction has a considerable influence on the performance of constructive methods [8]. In order to benefit even more from the advantages of scheduling directions, in this paper we propose a new scheduling scheme in which activities are either forward scheduled (FS), backward scheduled (BS) or midway scheduled (MS). The complete schedule is obtained by interlinking FS, BS and MS.

The main idea of tri-directional scheduling is to allow more activities to be able to get left shifted in order to benefit from the possibility of filling the existing gaps of the forward schedule by the shifted activities. Scheduling activities in tri-directional scheme is similar to bidirectional scheme with the difference that those activities that can be scheduled in either direction are now scheduled in midway direction. To illustrate, if activity j is only available in either forward or backward direction, it is scheduled in the direction that is available. On the other hand, if it is available in both directions, it is scheduled in midway. Now, having obtained a forward, backward, and a midway schedule, for interlinking these three schedules into a single one, activities from midway schedule and backward schedule are respectively considered for left shifting. For so doing, at first midway activities are considered in non-increasing order of their scheduled starting times and then the backward activities are considered in non-decreasing order of their scheduled starting times. Left shifting the midway schedule, using an ascending order of the starting times would be similar to the bidirectional scheme. However, it is not exactly equivalent

to the bidirectional scheme, because in many cases in tri-directional scheme more activities get the opportunity to be moved (note that the number of activities that are being moved in tri-directional scheme is *at least* equal to the number of activities that are being moved in bidirectional scheme). If there is more than one activity with identical starting times, then the first activity that is left shifted is the one that was scheduled last. Following these orders the activities are successively left shifted into FS such that no violation of precedence and resource constraints occurs.

In the following lemmas and theorem we try to state the sufficient conditions for the proposed tri-directional scheduling scheme in order to perform better than unidirectional and/or bidirectional scheduling schemes. However, the computational experiments carried out in this study clearly demonstrate the superiority of tri-directional scheme in almost all the problem instances that were solved.

Remark 1 In contrast to the bidirectional scheduling scheme in tri-directional scheme there is no need to use tie-breaking rules.

Lemma 1 *Let P be an instance of the RCPSP with N activities. Let* **S** *be a feasible schedule with respect to both precedence and resource constraints of P where preemption is not allowed. Let T be the makespan of schedule* **S**. *Let* **S**′ *be the schedule obtained by applying preemption of activities to* **S**. *If T' is the makespan of S', then $T' \leq T$.*

Proof See Appendix.

Lemma 2 *Let P be a project as described in Lemma 1 with no preemption. Let T_1 and T_2 be the makespans of the schedules by using forward and bidirectional scheme, respectively. Let PS_f and PS_b are left and right sub-schedules of bidirectional scheme respectively. In order to obtain a complete schedule, suppose we shift PS_b as a block to left such that the largest finish time of activities in PS_f coincide with the smallest start time of activities in PS_b. If the makespan of such a complete schedule is T_1, then $T_2 \leq T_1$.*

Proof See Appendix.

Lemma 3 *Let T_2 be as defined in Lemma 2. Let T_3 be the makespan of tri-directional scheduling scheme and PS_f, PS_m and PS_b be the sets of left, middle and right sub-schedules in the tri-directional scheduling scheme, respectively. If PS_m and PS_b are linked to the PS_f as independent blocks of activities and if the resulting makespan is equal to T_2, then $T_3 \leq T_2$.*

Lemmas 2 and 3 yield the following theorem.

Theorem 1 *Let T_1, T_2 and T_3 are the makespans of the schedules obtained by using forward, bidirectional and tri-directional scheduling scheme, respectively. If the conditions of Lemma 2 and 3 are satisfied then $T_3 \leq T_2 \leq T_1$.*

Example 2 By applying the tri-directional PSS in combination with the SPT rule, we obtain forward, backward and midway schedules. In the first two iterations,

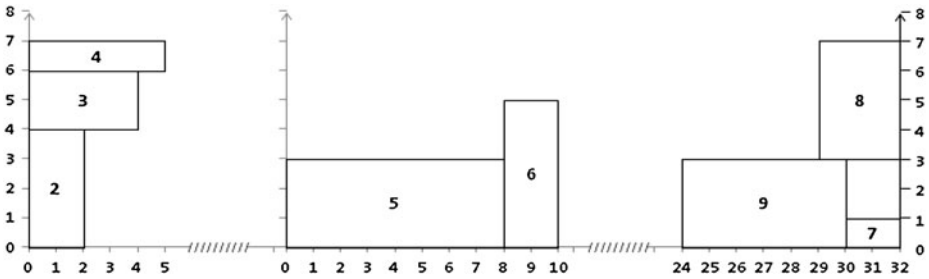

Fig. 5 Forward, midway and backward schedules

the dummy activities are scheduled and we obtain ES_f $(\tau_f = 0) = \{2,3,4,5\}$, ES_m $(\tau_m = 0) = \{\}$ and ES_b $(\tau_b = 32) = \{7,8,9\}$. For the sake of orderliness, we schedule the midway activities in a separate horizontal coordinates. Activity 2 is forward scheduled, while activities 7 and 8 are backward scheduled. Subsequently, activities 3 and 4 are also forward scheduled. As no more activities can be scheduled at current time, by forwarding the scheduling time, we update the eligible sets. Therefore, we have: ES_b $(\tau_b = 30) = \{9\}$ and ES_f $(\tau_f = 2) = \{5\}$. In the next step, activity 9 is backward scheduled. Now ES_b $(\tau_b = 30) = \{\}$, so $\tau_b = 29$ and ES_b $(\tau_b = 29) = \{5\}$. Since activity 5 can be scheduled in either direction we have ES_m $(\tau_m = 0) = \{5\}$. Therefore, activity 5 is midway scheduled. In the last iteration, the only remaining eligible activity 6 must be scheduled. It is available in either direction, so ES_m $(\tau_m = 8) = \{6\}$. All the activities are scheduled, and the corresponding schedules are shown in Fig. 5. We now must link the three schedules together. At first, midway scheduled activities are considered in decreasing order of their scheduled starting times and then left shifted to link with the forward scheduled activities. In so doing activity 6 is left shifted to start at 4 and activity 5 is left shifted to start at 6. Now the backward scheduled activities are considered in non-decreasing order of their scheduled starting times, and then left shifted. As the result activities 7, 8 and 9 are left shifted to start at 2, 14 and 6 respectively. As it can be seen from Fig. 6, the complete schedule makespan is 17.

The outline of the proposed method is given in Fig. 7. The notations used are as in Fig. 1 and Example 2, but indices f, b, m refer to forward and backward and midway directions, respectively.

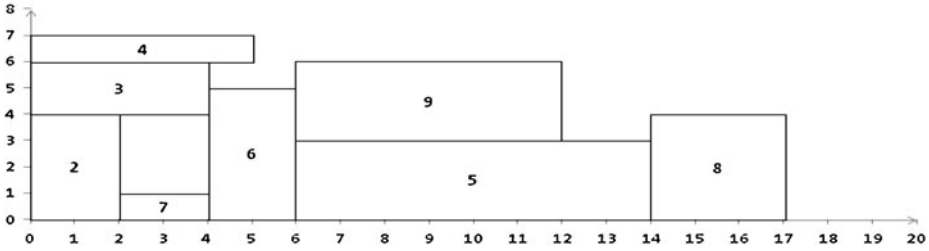

Fig. 6 Complete schedule obtained by tri-directional parallel scheduling scheme

Initialize

$\mathbf{A}_f = \{\}$, $\mathbf{C}_f = \{1\}, t_f = 0$ and $ES_f(t_f = 0) = \{\}$,

$\mathbf{A}_m = \mathbf{C}_m = \{\}, t_m = 0$ and $ES_m(t_m = 0) = \{\}$,

$\mathbf{A}_b = \{\}$, $\mathbf{C}_b = \{N\}, t_b = ub$ ($ub = \sum_{j=1}^{N} d_j$ is an upper bound) and $ES_b(t_b = ub) = \{\}$.

Update $ES_f(t_f = 0)$ and $ES_b(t_b = ub)$ w.r.t. resource and precedence constraints.

While $\left|(\mathbf{A}_f \cup \mathbf{C}_f) \cup (\mathbf{A}_b \cup \mathbf{C}_b) \cup (\mathbf{A}_m \cup \mathbf{C}_m)\right| < N$

Begin

Update $ES_m(t_m)$ as: $ES_m(t_m) = \{\, j \mid j \in ES_f(t_f) \cap ES_b(t_b) \,\} \Rightarrow$

$ES_f(t_f) = ES_f(t_f) - ES_m(t_m)$ and $ES_b(t_b) = ES_b(t_b) - ES_m(t_m)$

If none of activities in $ES_m(t_m) \neq \phi$ can be schedule at t_m then update t_m, $\mathbf{A}_m$ and $\mathbf{C}_m$.

While $ES_m(t_m) \neq \phi$ do

Begin

Select $j^* = \min_{j \in ES_m(t_m)} \{j \mid \upsilon(j) = \inf_{i \in ES_m(t_m)} \{\upsilon(i)\}\}$

Schedule j^* at t_m w.r.t. resource constraint and Set $ES_m(t_m) = ES_m(t_m) - \{j^*\}$.

If there is no enough resource to schedule j^* then $ES_m(t_m) = ES_m(t_m) - \{j^*\}$.

End

While $ES_f(t_f) \neq \phi$ && $ES_b(t_b) \neq \phi$

Begin

Select $j^* = \min_{j \in ES_f(t_f) \cup ES_b(t_b)} \{j \mid \upsilon(j) = \inf_{i \in ES_f(t_f) \cup ES_b(t_b)} \{\upsilon(i)\}\}$

If $j^* \in ES_f(t_f)$ schedule j^* in forward direction at t_f w.r.t. resource constraint and then Set $ES_f(t_f) = ES_f(t_f) - \{j^*\}$.

Else schedule j^* in backward direction at t_b w.r.t. resource constraint and then Set $ES_b(t_b) = ES_b(t_b) - \{j^*\}$.

End

If $ES_f(t_f) = \{\}$ update t_f and $ES_f(t_f)$, $\mathbf{A}_f$ and $\mathbf{C}_f$.

If $ES_b(t_b) = \{\}$ update t_b and $ES_b(t_b)$, $\mathbf{A}_b$ and $\mathbf{C}_b$.

End

Interlink

Shift the activities in $\mathbf{A}_m \cup \mathbf{C}_m$ to the forward sub-schedule (in non-increasing order of their starting times) and obtain the new sub-schedule S'.

Shift the activities in $\mathbf{A}_b \cup \mathbf{C}_b$ to the S' (in non-decreasing order of their starting times).

Fig. 7 Outline of tri-directional scheduling scheme

5 Computational Experiments

5.1 Priority Rules

The performance of the tri-directional parallel scheduling scheme is evaluated using different priority rules. Priority rules are used for selecting the activity which has to be scheduled next. Many priority rules have been designed by researchers and used in their studies of the RCPSP and related problems. These have been classified differently. One such classification categorizes rules according to the information that priority rules use [10]. As such, they have been categorized as *network-based rules* (e.g. most total successors), *critical path-based rules* (e.g. latest starting time), *resource-based rules* (e.g. greatest resource demand) and *composite rules* (e.g. activity

resource controls). In this paper, we have adopted the complete list of priority rules used in [8]. We refer the interested reader to the aforementioned paper for a detailed description of these priority rules.

5.2 Combinations of Scheduling Schemes and Priority Rules

In order to demonstrate the superiority of the tri-directional parallel planning scheme, we compare the effectiveness of the forward, bidirectional and tri-directional scheduling schemes in combination with the different priority rules. The result obtained by each combination is an upper bound (UB) on the optimal project duration. In our evaluations of different planning schemes, we have used the number of instances solved to optimality; the best results obtained; average relative deviation from the optimal solution ($\sum_{i=1}^{q}(\frac{UB_i - OPT_i}{OPT_i}) \times 100/q$) in which OPT_i is the optimal project completion time for instance i, and q is the number of instances to be evaluated; and the maximum relative deviation from the optimum solution ($\underset{i=1}{\overset{q}{Max}}(\frac{UB_i - OPT_i}{OPT_i}) \times 100$) as evaluation criterion.

All the experiments that are reported in this paper are performed on a PC Pentium 4 with CPU 1.8 Intel and 512 MB. All procedures are coded by means of MATLAB R2008a. The benchmark SMFF of Kolisch's benchmarks are used in the evaluation processes. This set contains 480 instances with 30 activities and with $RS = 0.2, 0.5, 0.7, 1$. As in [8] we excluded 120 easy instances (i.e. those with

Table 1 No. of instances solved to optimality and the best solution obtained

Priority rules	No. of instances solved to optimality					No. of instances with best solutions				
	1 fwd.	2 bwd.	3 bid.	4 trd.	5 %±	6 fwd.	7 bwd.	8 bid.	9 trd.	10 %±
SPT	32	36	71	72	+1.41	172	143	**253**	**254***	+0.4
MIS	57	46	83	86	+3.61	185	115	224	244	+8.93
MTS	70	73	106	**124***	+16.98	183	140	219	238	+8.67
GRPW	59	57	81	92	+13.58	181	169	222	242	+9.0
GRPW*	**81**	**77**	**110**	123	+11.82	**203**	**179**	216	233	+7.87
EST	35	37	75	76	+1.33	290	201	**319**	**327***	+2.51
EFT	30	39	68	67	−1.47	284	212	314	311	−0.96
LST	**93**	80	117	115	−1.7	**321**	254	294	294	0.0
LFT	86	**82**	**133**	**138***	+3.76	217	199	290	310	+6.9
MSLK	84	68	104	110	+5.77	315	**268**	289	310	+7.27
GRD	39	46	69	71	+2.9	**173**	**172**	**241**	**271***	+12.45
WRUP	**49**	**51**	**71**	**79***	+11.27	152	150	220	251	+14.09
ACTRES	81	82	113	126	+11.5	158	141	187	212	+13.37
ACROS	67	73	110	120	+9.09	164	123	**217**	**250***	+15.21
TIMRES	85	82	**129**	**140***	+8.53	**183**	**158**	195	216	+10.77
TIMROS	**86**	**87**	125	127	+1.6	179	**159**	203	231	+13.79
IRSM	91	76	125	131	+4.8	196	**186**	202	215	+6.44
WCS	**97**	**85**	**132***	**131**	−0.76	**248**	110	**235***	**234**	−0.43
ACS	92	85	124	130	+4.84	201	113	191	201	+5.24
RAND	38	36	67	**77***	+14.93	125	120	229	**270***	+17.9

Table 2 Average and maximum relative deviation from optimality

Priority rules	Average relative deviation from optimality					Maximum relative deviation from optimality				
	fwd.	bwd.	bid.	trd.	%±	fwd.	bwd	bid.	trd.	%±
SPT	13.09	14.39	8.66	8.21	−5.2	44.28	43.644	36.49	36.49	0
MIS	9.14	13.77	7.77	7.5	−3.47	41.86	54.17	33.33	33.33	0
MTS	6.66	7.39	5.22	5.07	−2.87	**25**	34.72	24.49	26.38	+7.72
GRPW	10.47	12.18	8.59	8.17	−4.89	41.86	48.61	34.21	39.04	+14.12
GRPW*	**6.2**	**6.71**	**4.78***	**4.82**	+0.8	32.81	**30.19**	**23.68**	**22.49***	−5.03
EST	10.49	12.77	7.51	7.35	−2.13	41.86	35.36	33.33	33.33	0
EFT	11.23	12.49	7.41	7.54	+1.75	38.04	35.9	29.03	29.03	0
LST	6.04	6.94	5.25	5.2	−0.95	32.81	**27.69**	26.56	26.56	0
LFT	**5.86**	**6.78**	**4.49**	**4.04***	−10.02	**24.14**	31.71	**21.73***	**21.73***	0
MSLK	8.96	9.61	6.7	6.59	−1.64	44.21	55.55	37.93	37.93	0
GRD	12.33	14.19	9.48	8.91	−6.01	44.07	55.1	38.46	38.46	0
WRUP	**9.78**	**11.03**	**7.65**	**7.13***	−6.8	**38.78**	**39.13**	**33.33**	**31.58***	−5.25
ACTRES	6.92	7.47	5.55	4.72	−14.95	35.9	39.66	30.77	30.77	0
ACROS	6.98	7.44	5.12	**4.62***	−9.77	38.1	31.75	38.1	38.1	0
TIMRES	6.32	6.95	5.26	4.92	−6.46	**32.81**	34.7	23.53	**27.42**	+16.53
TIMROS	**6.09**	**6.39**	**4.86**	4.75	−2.26	**32.81**	**28.57**	**23.44***	36.79	+56.95
IRSM	6.56	7.74	5.23	5.23	0	33.78	35.05	28.81	28.81	0
WCS	**5.31**	**6.40**	**4.68**	4.68	0	**27.78**	**23.68**	**25***	**25***	0
ACS	5.51	6.70	4.69	**4.26***	−9.17	31.25	**23.68**	**25**	**25**	0
RAND	13.09	15.02	8.75	**8.08***	−7.66	48.21	50	**41.23***	**41.23***	0

$RS = 1$). We also used the set j60 of Kolisch's benchmarks with $RS = 0.2$. Instances with 101 activities are taken from Alvarez-Valdes & Tamarit benchmarks [12].

Table 1 displays the performance of forward (fwd.), backward (bwd.), bidirectional (bid.) and tri-directional (trd.) scheduling schemes using six categories of different priority rules. It can be seen that under different priority rules in both criterions: number of instances solved to optimality and the best results obtained the new tri-directional scheduling scheme outperforms the other scheduling schemes. Columns 5 and 10 of the Table display the percentage improvement of the proposed scheme when compared to the best available scheme. For instance when MTS is

Table 3 Average and maximum relative deviation from optimality ($RS = 0.2$, 30 activities)

Priority rules	Average relative deviation from optimality					Maximum relative deviation from optimality				
	1	2	3	4	5	6	7	8	9	10
	fwd.	bwd.	bid.	trd.	%±	fwd.	bwd.	bid.	trd.	%±
GRPW*	10.28	9.76	8.03	8.61	+7.22	32.81	30.19	23.68	22.49	−5.03
LST	10.80	10.88	9.58	9.73	+1.57	32.81	27.69	25.00	25.00	0
LFT	9.31	10.15	8.03	7.6	−5.35	24.14	28.81	21.27	21.27	0
WRUP	12.84	14.36	10.62	10.60	−0.19	36.91	38.46	27.63	27.63	0
TIMRES	11.01	10.56	8.92	9.13	+2.35	32.81	28.57	23.53	27.42	+16.53
TIMROS	10.32	9.56	8.43	9.20	+9.13	32.81	25.42	22.44	35.78	+59.45
WCS	9.18	10.13	8.65	8.82	+1.97	27.78	27.88	24.49	24.49	0

Table 4 Average and maximum relative deviation from optimality ($RS = 0.5$, 30 activities)

Priority rules	Average relative deviation from optimality					Maximum relative deviation from optimality				
	fwd.	bwd.	bid.	trd.	%±	fwd.	bwd.	bid.	trd.	%±
GRPW*	4.21	6.14	3.26	3.01	−7.67	18.37	23.81	11.90	4.00	−66.39
LST	4.93	5.91	3.80	3.56	−6.32	15.69	19.57	16.94	16.94	0.00
LFT	4.47	6.21	3.95	3.87	−2.03	21.95	31.71	12.25	11.12	−9.22
WRUP	4.83	10.72	3.20	3.40	+6.25	38.78	36.21	33.33	26.67	−19.98
TIMRES	9.84	5.85	7.47	6.67	−10.71	28.78	17.07	18.42	13.66	−25.84
TIMROS	4.90	5.7	4.35	3.82	−12.18	21.95	21.43	16.94	10.75	−36.54
WCS	5.08	5.64	4.06	3.38	−16.75	15.39	16.32	16.94	16.94	0.00

used as the priority rule, tri-directional scheme solves 16.98% more instances to optimality. In fact, performance improvement ranges from 1.33% to 16.98%.

Bold numbers in the Tables depict the best result obtained by the corresponding scheme under each category of priority rules. The asterisks indicate the best result in each category. In comparison to bidirectional scheme, the proposed scheme solves 7 more instances (140 vs. 133) albeit under different priority rule. For the instances that are not solved to optimality, the results obtained are also improved. For instance, under ACROS as priority rule, tri-directional scheduling scheme improves the best results obtained by 15.21%. With regards to the best results obtained, performance improvement ranges from 0.4% to 17.9%

Table 2 reveals that tri-directional scheduling scheme almost outperforms the bidirectional scheme with respect to all measures. With respect to average relative deviation, tri-directional scheme always yields smaller values when compared to forward and backward schemes. Furthermore, it solves more problem instances to optimality than unidirectional and bidirectional schemes.

In Table 2 when average and maximum relative deviations from optimality are displayed, once again tri-directional scheme performs better. It improves the average relative deviation by 10.02% (under LFT) and maximum deviation by 5.03% (under GRPW*). Apparently, tri-directional scheme with respect to two priority rules namely GRPW* and EFT, has not performed as well as bidirectional scheme. However, the difference in performance of the given schemes under the two priority rules does not seem to be significant.

Table 5 Average and maximum relative deviation from optimality ($RS = 0.7$, 30 activities)

Priority rules	Average relative deviation from optimality					Maximum relative deviation from optimality				
	fwd.	bwd.	bid.	trd.	%±	fwd.	bwd.	bid.	trd.	%±
GRPW*	3.33	4.14	2.44	1.92	−21.31	16.33	23.68	13.63	13.63	0.00
LST	2.78	3.96	2.15	2.10	−2.33	13.21	23.68	13.63	12.54	−8.00
LFT	3.37	3.87	2.22	2.17	−2.25	15.39	25.00	14.28	14.28	0.00
WRUP	6.52	7.89	4.74	3.9	−17.72	28.07	39.13	22.81	22.81	0.00
TIMRES	2.95	4.4	2.44	1.72	−29.51	11.36	28.57	13.63	13.63	0.00
TIMROS	2.77	3.82	1.99	1.59	−20.10	14.29	28.57	13.63	13.63	0.00
WCS	2.71	3.46	2.03	2.01	−0.99	12.25	15.68	13.63	13.63	0.00

Table 6 Average and maximum relative deviation from optimality ($RS = 0.2$, 60 activities)

Priority rules	Average relative deviation from optimality					Maximum relative deviation from optimality				
	fwd.	bwd.	bid.	trd.	%±	fwd.	bwd.	bid.	trd.	%±
GRPW*	13.22	13.2	12.01	11.67	−2.83	25	23.0	23.61	25.77	+9.15
LST	12.51	13.26	13.52	13.4	−0.89	28.74	28.75	28.74	28.74	0
LFT	12.77	15.34	13.0	12.74	−2.0	26.92	26.23	26.39	25.74	−2.46
WRUP	17.39	18.9	13.32	13.0	−2.4	45.95	43.1	27.03	25.68	−4.99
TIMRES	14.17	13.59	12.29	11.47	−6.67	35.14	33.69	22.99	22.97	−0.09
TIMROS	13.39	12.81	12.52	12.0	−4.15	27.59	25.64	24.32	26.08	+7.24

The research on the constructive methods shows that their performances to some extent depend on the structure of the problem instances. To study the performance of scheduling schemes including the new tri-directional scheme in more detail, we adopt as Klein [8] the complexity measure introduced in [10], i.e. the *resource strength*. For each resource, it is defined as $(a_r - u_r^{\max})/(k_r^{peak} - u_r^{\max})$, in which a_r is per unit of time availability of resource r, $u_r^{\max}$ is the maximum requirement of a single activity of resource r, and k_r^{peak} is the peak requirements for resource r when activities are scheduled at their earliest start time. From now on, we exclude from further investigation the inferior backward scheduling scheme and focus only on forward, bidirectional and tri-directional schemes.

High ($RS = 0.7$), medium ($RS = 0.5$) and low ($RS = 0.2$) resource strengths are examined. As the resource strength decreases, the average and maximum relative deviations from the optimal solution increase. Average relative deviation with tri-directional scheduling scheme and under different priority rules ranges from −5.35 to +9.13 when $RS = 0.2$ (see column 5 of Table 3), ranges from −16.75 to +6.25 when $RS = 0.5$ (see column 4 of Table 4) and finally ranges from −29.51 to −0.99 when $RS = 0.7$ (see column 4 of Table 5). This confirms the expectation that by increasing the complexity of the problem instances (low resource strength) the deviation from the optimal solution increases.

If low resource strength is combined with higher number of activities in a project, will the tri-directional scheduling scheme behaves similarly? Tables 6 and 7 present an answer to this question.

With reference to Tables 6 and 7, it is evident that the performance of tri-directional scheme improves as the *hardness* of the problem instances increases. For example, when the number of activities increases to 60, the average relative deviation

Table 7 Average and maximum relative deviation from optimality ($RS \leq 0.4$, 101 activities)

Priority rules	Average relative deviation from optimality					Maximum relative deviation from optimality				
	fwd.	bwd.	bid.	trd.	%±	fwd.	bwd.	bid.	trd.	%±
GRPW*	4.26	4.62	4.51	4.37	−3.1	32.27	28.49	19.81	20.75	+4.75
LST	5.05	5.26	5.26	5.23	−0.57	16.51	16.73	17.63	18.28	+3.68
LFT	4.58	4.63	5.73	5.09	−11.17	15.57	16.2	18.11	17.17	−5.19
WRUP	10.18	9.57	7.95	7.58	−4.65	54.25	41.23	34.96	23.29	−33.38
TIMRES	7.9	6.98	6.85	6.8	−0.73	29.21	29.01	28.32	25.65	−9.43
TIMROS	8.65	8.5	8.84	8.41	−4.86	21.7	21.68	20.75	20.87	+0.58

Table 8 Average relative deviation from optimality

Priority rules	30 Activities				60 Activities				101 Activities			
	bid.	bid.2	trd.	%±	bid.	bid.2	trd.	%±	bid.	bid2.	trd.	%±
SPT	8.66	8.5	8.21	**−3.41**	9.28	9.19	8.58	**−6.64**	10.66	10.69	10.38	**−2.9**
MIS	7.77	7.6	7.5	**−1.32**	7.06	7.25	7.04	**−2.9**	9.56	9.59	8.62	**−10.11**
MTS	5.22	5.00	5.07	+1.4	4.63	4.76	4.89	+2.73	4.73	4.87	4.67	**−4.11**
GRPW	8.59	8.47	8.17	**−3.54**	10.14	10.01	10.13	+1.2	8.88	8.57	8.15	**−4.9**
GRPW*	4.78	4.55	4.82	+5.93	4.63	4.54	4.18	**−7.93**	4.54	4.72	4.32	**−8.47**
EST	7.51	7.46	7.35	**−1.47**	11.24	11.24	11.24	**0**	10.38	10.21	9.68	**−5.19**
EFT	7.41	7.44	7.54	+1.34	10.14	10.00	10.14	+1.4	10.61	10.61	10.15	**−4.34**
LST	5.25	5.29	5.2	**−1.7**	6.18	6.03	6.01	**−0.33**	5.69	5.63	5.68	+0.89
LFT	4.49	4.49	4.04	**−10.02**	5.78	5.78	5.54	**−4.15**	5.77	5.00	5.03	+0.6
MSLK	6.7	6.75	6.59	**−2.37**	7.11	6.8	6.24	**−8.24**	7.21	6.95	7.12	+2.45
GRD	9.48	9.49	8.91	**−6.11**	12.22	12.25	11.74	**−4.16**	9.85	9.85	9.27	**−5.89**
WRUP	7.65	7.48	7.13	**−4.68**	5.96	5.71	5.58	**−2.28**	7.94	7.92	7.57	**−4.42**

from the optimal solution obtained by tri-directional schemes is always less than that obtained by bidirectional scheme (see column 5 of Table 6). The same behavior can be seen from the results of the Table 7 where the number of activities has increased to 101 (see column 5 of Table 7).

One more question remains to be answered. Is "tri-directional scheduling scheme equivalent to bidirectional scheduling scheme with tie-breaking rule such as *always schedule the activities of BS* ∩ *FS in BS*"? We addressed this question empirically as follows. We ran the previous test problems using bid., trd. and bidirectional scheduling scheme coupled with the given tie-breaking rule (bid.2). The resource strength ranges from 0.2 to 0.7. As it can be seen from Table 8, tri-directional scheduling scheme gives better results in almost all the test problems when compared to bid.2. The better performance of tri-directional is again more evident in projects with larger number of activities. It seems that the better performance of trd. is due to shifting more activities to left as well as dividing the activities into different directions.

6 Conclusions

In this paper, we have introduced tri-directional scheduling scheme in order to improve the performance of constructive scheduling schemes when applied to the RCPSP. We have studied and compared the performance of forward, backward, bidirectional and tri-directional planning schemes in the context of different priority rules. Computational experiments and some analytical remarks revealed that the tri-directional scheduling scheme outperforms the other mentioned scheduling schemes in almost all the standard benchmark instances used. It was also shown that in problem instances with higher degree of complexity, the tri-directional scheme performs better than existing schemes. Furthermore, in contrast to the bidirectional scheduling scheme in tri-directional scheduling scheme there is no need to use tie-breaking rules. It seems that incorporating double justification as proposed in [22] into multi-dimensional scheduling schemes, may improve the quality of the solution even more. The authors are investigating this question at present.

Appendix

Proof of Lemma 1 Without loss of generality, suppose that only one resource type is required by the activities of P. Let $\mathbf{A}_N$ be the set of activities of $\mathbf{S}$ whose finish time is T. Let u_{j1} is constant per unit of time resource requirements (resource 1) of activity j and $\mathbf{P}_j$ denotes the set of predecessors of activity j. Suppose that, there are w ($w \geq 1$) gaps in the feasible schedule $\mathbf{S}$. Gap is defined as a space in $\mathbf{S}$ where there is availability of resource such that part of some eligible activities can be executed. The amount of available resource is called depth of the gap.

For the *k-th* gap; Gap_k, $k = 1, \ldots, w$ define:

$$Active\,(\mathbf{A}_N, k) = \{i \in \mathbf{J} = \{1, ..., N\} \mid SG_k < F_i \leq \max\{S_i \mid i \in \mathbf{A}_N\}\}$$

in which S_i and F_i are the start time and the finish time of activity $i, i = 1, \ldots, N$ respectively in schedule $\mathbf{S}$, and Gap_k occurs in the time interval $[SG_k, SG_k + d'_k]$ where $SG_k + d'_k < T$. If there is a Gap_{i_0} subject to $\forall i \in Active\ (\mathbf{A}_N, i_o)$ such that $i \notin \mathbf{P}_j$; $\forall j \in \mathbf{A}_N$, and $\sum_{j \in \mathbf{A}_N} u_{j1} \leq g_{i_0}$ in which g_{i_0} is the depth of Gap_{i_0}, then by transferring a part of all activities that belong to $\mathbf{A}_N$ onto Gap_{i_0} we have $T' < T$. □

Proof of Lemma 2 By Lemma 1, when PS_f and PS_b are interlinked, the activities of PS_b that are left shifted towards PS_f may fill some or all of the possible gaps that exist in PS_f and PS_b. This may reduce the gaps in the complete schedule and will result in a schedule with smaller makespan, i.e. $T_2 \leq T_1$. □

Proof of Lemma 3 In tri-directional scheme, we have three sub-schedules, namely PS_f, PS_m and PS_b. When PS_m is interlinked to PS_f, according to Lemma 1, the makespan of the resulting schedule i.e. PS_{fm} can not be longer than the makespan of the schedule obtained when activities of PS_f and PS_m are linked together as two independent blocks of activities. The same argument can be applied to PS_{fm} and PS_b, resulting in a schedule, PS_{fmb}. Thus the makespan PS_{fmb} is not worse than T_2, i.e. $T_3 \leq T_2$. □

References

1. Agarwal, R., Tiwari, M.K.., Mukherjee, S.K.: Artificial immune system based approach for solving resource constraint project scheduling problem. Int. J. Adv. Manuf. Technol. **34**, 584–593 (2007). doi:10.1007/s00170-006-06312
2. Blazewicz, J., Lenstra, J., Rinnoy, K.A.: Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)
3. Bouleimen, K., Lecocq, H.: 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 (2003)
4. Brucker, P., Knust, S., Schoo, A., Thiele, O.: A branch and bound algorithm for the resource-constrained project scheduling problem. Eur. J. Oper. Res. **107**, 272–288 (1998)
5. Cho, J., Kim, Y.D.: A simulated annealing algorithm for resource-constrained project scheduling problems. J. Oper. Res. Soc. **48**, 736–744 (1997)
6. Hartmann, S.: A self-adapting genetic algorithm for project scheduling under resource constraints. Nav. Res. Logist. **49**, 433–448 (2002)
7. Icmeli, O., Rom, W.O.: Solving resource constrained project scheduling problem with optimization subroutine library. Comput. Oper. Res. **23**, 801–817 (1996)

8. Klein, R.: Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects. Eur. J. Oper. Res. **127**, 619–638 (2000)
9. Klein, R.: Scheduling of Resource Constrained Projects. Kluwer, Dordrecht (2000)
10. Kolisch, R.: Project Scheduling Under Resource Constraints Efficient Heuristics for Several Problem Classes. Physica, Heidelberg (1995)
11. Kolisch, R.: Serial and parallel resource-constrained project scheduling methods revisited—theory and computation. Eur. J. Oper. Res. **90**, 320–333 (1996)
12. Kolisch, R., Schwindt, C., Sprecher, A.: Benchmark instances for project scheduling problems. In: Weglarz, J. (ed.) Project Scheduling. Recent Methods, Algorithms and Applications. Kluwer's International Series (1998)
13. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: an update. Eur. J. Oper. Res. **174**, 23–37 (2006)
14. Kumanan, S., Jose, G.J., Raja, K.: Multi-project scheduling using a heuristic and a genetic algorithm. Int. J. Adv. Manuf. Technol. **31**, 360–366 (2006)
15. Kuo-Ching, Y., Shih-Wei, L., Zne-Jung, L.: Hybrid-directional planning: improving improvement heuristics for scheduling resource-constrained projects. Int. J. Adv. Manuf. Technol. **41**, 358–366 (2009)
16. Petrovic, R.: Optimization of resource allocation in project planning. Oper. Res. **16**, 559–586 (1968)
17. Pritsker, A., Watters, L., Wolfe, P.: Multi-project scheduling with limited resources: a zero-one programming approach. Manage. Sci. **16**, 93–107 (1969)
18. Reddy, J.P., Kumanan, S., Chetty, O.V.K.: Application of Petri nets and a genetic algorithm to multi-mode multi-resource constrained project scheduling. Int. J. Adv. Manuf. Technol. **17**, 305–314 (2001)
19. Schrage, L.: Solving resource-constrained network problems by implicit enumeration—nonpreemptive case. Oper. Res. **18**, 263–278 (1970)
20. Shukla, S.K., Son, Y.J., Tiwari, M.K.: Fuzzy-based adaptive samplesort simulated annealing for resource-constrained project scheduling. Int. J. Adv. Manuf. Technol. **36**, 982–995 (2008). doi:10.1007/s00170-006-0907-6
21. Thomas, P.R., Salhi, S.: A tabu search approach for the resource constrained project scheduling problem. J. Heuristics **4**, 123–139 (1998)
22. Valss, V., Ballestin, F., Quintanilla, S.: Justification and RCPSP: a technique that pays. Eur. J. Oper. Res. **165**, 375–386 (2005)
23. Zamani, M.R.: A high-performance exact method for the resource-constrained project scheduling problem. Comput. Oper. Res. **28**, 1387–1401 (2001)