Expert Systems with Applications 41 (2014) 5870–5881

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Improving RCPSP solutions quality with Stacking Justification – Application with particle swarm optimization

Amer Fahmy [a,*], Tarek M. Hassan [b,1], Hesham Bassioni [c,2]

[a] Loughborough University, UK
[b] School of Civil & Building Engineering, Loughborough University, UK
[c] Construction & Building Engineering Dept., Arab Academy for Science, Technology & Maritime Transport, Egypt

ARTICLE INFO

Keywords:
Schedule Justification
Resource-constrained project scheduling problem (RCPSP)
Particle swarm optimization
Priority rules

ABSTRACT

Justification is a simple technique which was presented for improving the quality of schedules generated with algorithms for solving RCPSPs. Few researches implemented justification with meta-heuristics and proved that algorithms optimization results using justification outperforms the performance of the same algorithms without justification. In this paper, Stacking Justification is presented for further improvement of generated schedules quality. For testing the performance of this new justification technique, the *multiple justification particle swarm optimization* (*MJPSO*) algorithm was developed. Test results show that implementing *Stacking Justification* in a multiple justification approach was having a significant effect on solutions quality, and the performance results were very effective when compared to best performing algorithms in literature. Additionally, combined priority rules (CPR), a suggested approach for proper spread of swarm particles over several good quality areas within search space during swarm initialization, showed a considerable improvement on results especially for small number of generated schedules which is more suitable for practical applications.

1. Introduction

RCPSP is a well-known standard problem in the context of project scheduling; due to its complexity, RCPSP became a very attractive field for researchers either in scheduling field or in operations research. Numerous numbers of researches were presented within the RCPSP context, which were summarized along with the problem's developments and extensions in the detailed surveys of Icmeli, Erenguc, and Zappe (1993), Herroelen, Reyck, and Demeulemeester (1998), Brucker, Drexl, Mohring, Neumann, and Pesch (1999), Kolisch and Padman (2001), Neumann, Schwindt, and Zimmermann (2003) and Hartmann and Briskorn (2010). RCPSP has been considered as NP-Hard in the strong sense (Blazewicz, Lenstra, & Rinooy Kan, 1983), and accordingly most researches within the last two decades concentrated on heuristics & meta-heuristics for solving RCPSP. Most heuristic approaches consists of 2 main elements: a *Priority Rule* to set the priority of each activity based on predefined criteria, and a *Schedule Generation Scheme* to create the schedule using the prioritized activity list. Recently, another element was added by Valls, Ballestín, and Quintanilla (2005), the *Justification* technique, for the improvement of generated schedules quality.

Several practical scheduling applications require high quality solutions to efficiently utilize the project resources with minimal or at least reasonable computational burden. Accordingly, the continuous improvement of RCPSP solving techniques to improve solutions quality and/or minimize the analysis time is essential for such applications which is the main motivation for this research. This paper involves the presentation of a new justification technique, Stacking Justification, and its implementation in the proposed *multiple justification PSO* (or *MJPSO*). Moreover, a simple approach was presented for applying *combined priority rules* (*CPR*) to enable proper spread of swarm particles over several good quality areas within search space during swarm initialization.

The experimental results detailed in Section 8 of this paper show that *Stacking Justification* outperformed the original *Justification* technique in some cases; but most importantly, results also show that the proposed approach for combination of both justification techniques was always more effective than using only one. Moreover, the performance results were very effective when compared to best performing algorithms in literature. Additionally, the implementation of *CPR* for swarm particles initialization was

* Corresponding author. Tel.: +44 797 9831402.
E-mail addresses: amer_mohey@yahoo.co.uk (A. Fahmy), T.Hassan@lboro.ac.uk (T.M. Hassan), hbassioni@aast.edu (H. Bassioni).
[1] Tel.: +44 780 8477551; fax: +44 1509 222602.
[2] +20 100 1240110.

http://dx.doi.org/10.1016/j.eswa.2014.03.027
0957-4174/

proven to improve the exploration capabilities of the *PSO* algorithm in most RCPSP cases. These proposed improvements can support any practical application in the scheduling field in the production of high quality solutions in a reasonable analysis time.

This paper is organized as follows: Section 2 is a brief introduction for RCPSP, section 3 explains the original justification technique, Section 4 presents the new Stacking Justification technique, Section 5 presents the proposed *combined priority rules* (*CPR*) approach, Section 6 introduces particle swarm optimization and its different variations and topologies, Section 7 presents the proposed *multiple justification PSO* (*MJPSO*), analysis results are detailed and discussed in Section 8, and finally conclusions & future work are stated in Section 9.

2. Resource-constrained project scheduling problem (RCPSP)

2.1. Overview

RCPSP is one of the most famous problem types in the scheduling context; it involves the determination of activities start times which achieve the minimum overall duration without violating any of the precedence or resource constraints.

For solving the RCPSP, three main approaches were followed within literature: exact (or deterministic), heuristics and meta-heuristics. Several Branch & Bound algorithms were developed and presented for solving RCPSPs deterministically (Brucker, Knust, & Thiele, 1998; Demeulemeester & Herroelen, 1992; Mingozzi et al., 1998; Klein & Scholl, 1999); these algorithms can get the exact optimum solutions, but the analysis time gets impractical with the increase in problem size. A deterministic oriented survey was presented by Kolisch and Padman (2001); the survey reviewed the vast literature of deterministic scheduling with a perspective that integrates models, data, and optimal and heuristic algorithms, for the major classes of project scheduling problems.

An extensive survey of all deterministic and heuristic procedures which are presented in the literature for the scheduling problems can be found in Chapters 6 and 8 of the Project Scheduling Research Handbook of Demeulemeester and Herroelen (2002). In simple terms, most heuristic approaches can be summarized as the process of generating & justifying a schedule from an ordered activity list; and accordingly these heuristics include 3 main elements: a *Priority Rule* to set the priority of each activity based on predefined criteria, a *Schedule Generation Scheme* to create the schedule using the prioritized activity list, and a *Justification* technique to improve the quality of generated schedule.

Priority rules function is to arrange the activities list in an order which will generate a good solution. Priority rules was initially presented in the pioneering work of Kelley, 1963, which was then followed by many other researches presenting new priority rules and testing their performance (summarized in Kolisch, 1996b). After activities are prioritized, the second step is to generate the schedule vide a Schedule Generation Scheme (SGS). There are two types of SGS: Serial (SSGS) presented by Kelly (1963) and Parallel (PSGS) which have two associated algorithms in literature Kelly (1963) and "Brooks Algorithm" (Bedworth & Bailey, 1982); refer to Kolisch (1996a) for detailed description of SGSs. Additionally, the forward–backward improvement (FBI) proposed by Li and Willis (1992) was found to improve the results, by applying SGS in a forward direction and performing another cycle in reverse order and backward scheduling (reversed precedence network). And finally, the schedule quality can be improved by the simple *Justification* scheme proposed by Valls et al. (2005). Justification involves two successive cycles of shifting the activities in the current project time frame (right shifting then left shifting) without violating the resource constraints; this guarantees that the project make-span will be either the same or shorter. In this paper, *Stacking Justification*, a new justification technique, is proposed with a variation to the original technique in the activities selection criteria in each justification cycle in a way to minimize the gaps within resources profiles.

The use of Meta-heuristics in RCPSP solving involves the generation of activities order list which can produce better solutions based on experience gained in previous generation cycles. Several meta-heuristic techniques were implemented in the RCPSP context, such as tabu search (TS) (Baar, Brucker, & Knust, 1998; Klein, 2000; Kochetov & Stolyar, 2003; Nonobe & Ibaraki, 2001; Thomas & Salhi, 1998), simulated annealing (SA) (Bouleimen & Lecocq, 2003; Rutenbar, 1989), genetic algorithm (GA) (Alcaraz & Maroto, 2001; Alcaraz, Maroto, & Ruiz, 2004; Valls, Ballestín, & Quintanilla, 2008; Valls et al., 2005), ant colony optimization (ACO) (Lo, Chen, Huang, & Wu, 2008; Merkle, Middendorf, & Schmeck, 2002), and particle swarm optimization (PSO) (Chen, 2011; Zhang, Li, & Tam, 2005; Zhang, Sun, Zhu, & Yang, 2008).

2.2. RCPSP model

A simple notation for the RCPSP can be stated as follows: The problem resembles a project with a set of activities $V = \{1, \ldots, n\}$ with two dummy activities (1 & n) at the start and end of the project. Each activity (i) is processed in a duration d_i and requires r_{ik} units from resource k during each period of its duration. Activities are logically tied with each other with finish-to-start relations without time lags, where P_i represents the direct predecessors of activity i. The objective is to obtain a schedule S (corresponding to a set of activities start times S_1 to S_n) with minimum time span T (where $T = S_n$) in which the total resource units required r_{tk} of resource k in time period t does not exceed the available resource units a_k. For detailed RCPSP notations and models refer to Brucker et al. (1999).

3. Justification

Valls et al. (2005) proposed a new simple technique named as "*Justification*" or "*Double Justification*" (abbreviated as *DJ* through the rest of the paper). DJ can be applied to almost any heuristic to improve the solutions quality. In brief, DJ adjusts the start time of activities with two cycles without violating resource constraints leading to a final schedule with a make-span either the same or shorter. Because *Justification* was recently introduced (Valls et al., 2005), only few researches has adopted this schedule quality improvement technique in their algorithms development, such as Peteghem and Vanhoucke (2008) [*GA*], Lova, Tormos, Cervantes, and Barber (2009) [*GA*], Ranjbar, De Reyck, and Kianfar (2009), Chen (2011) [*PSO*], Coelho and Vanhoucke (2011) [*GA*] and Zamani (2013) [*GA*].

The DJ involves two cycles, right justification cycle followed by a left justification cycle. In the right justification cycle, the schedule is to be turned into a *right active schedule* (a schedule where no activity can be started later than its current start without violating resource constraints or delaying another activity), where the start time S_i of each activity i (except activity n) is shifted to its latest possible time S'_i; the resulting schedule S' (or S^R) is the right justified schedule of original schedule S. Similarly, the next cycle it to turn S^R into a left active schedule (a schedule where no activity can be started earlier than its current start without violating resource constraints or delaying another activity), where the start time S_i of each activity i (except activity 1) is shifted to its earliest possible time S'_i; the resulting schedule S' (or S^L) is the left justified schedule of original schedule S. Accordingly, the $DJ(S) = (S^R)^L$ is the double justified schedule of S.

In addition to the original DJ concepts, Chen (2011) has added an important feature to DJ which is essential when meta-heuristics are used, *Mapping*. After the solution is improved by DJ, the activities priorities list is still with the original order as it was before DJ; which means that following meta-heuristic cycles or generations will not be able to acquire the activities order of the improved solution. Accordingly, it is necessary to perform a mapping (or synchronization) process to re-build the priorities list after solution improvement. This mapping process was proposed as $P_i = S_i^{-1}$, where P_i is the priority of activity i and its value is mapped as the inverse of the activity's start time. This gives the earlier activities (after mapping) a higher priority than latter ones, and accordingly the priorities are synchronized to the new improved solution. Beside the logic for the need for mapping, this process was tested in Chen (2011) and was found to be improving the overall optimization results; and this was confirmed also during the analysis of this research.

In this paper, a small modification was performed to the mapping process. In forward scheduling, applying $P_i = S_i^{-1}$ maps activities priorities correctly to the new justified schedule because activities are scheduled based on their earliest start; however, for backward scheduling, activities are scheduled based on their latest finish, which makes the previous mapping partially inaccurate due to the variance in activities durations. So, it is proposed in this paper to apply $P_i = F_i^{-1}$ (where F_i is the finish time of activity i) to map activities priorities after backward scheduling.

4. Stacking Justification

Stacking Justification (*SJ*) is another simple and efficient technique which leads to further improvement to solutions quality; but before going through its concept and process, the reason for proposing another justification technique needs to be clarified. *DJ* was proven by Valls, Ballestín, and Quintanilla (2005)) to be efficient with several algorithms, but what makes it suitable to almost all optimization algorithms and causing a steady improvement. The key behind this success is that DJ improves the solutions quality by changing the solution itself (i.e. the activities priorities list) to a better neighbouring solution, if a better neighbour exists; and since the same result will occur if the original priorities list was selected for analysis in next generations, then this original list is somehow eliminated from the search space, which makes *DJ's* effect to appear as minimizing the search space rather than improving solutions quality. This observation will be further clarified in the examples within this section.

If we considered that the *SGS* minimizes the original search space by eliminating the priorities lists which does not respect precedence, and that the *DJ* further minimizes the search space by directing few neighbouring solutions to their local optimum, then another justification technique which can further minimize the search space (above *SGS* & *DJ* effect) will definitely be efficient.

DJ works on justifying the solution by scheduling activities based on the order of their start/finish times (i.e. their priorities), then respecting the resources limits; while the concept of *SJ* is to respect first the resources limits by scheduling activities which will give more efficient stacking (minimal gaps in resources profiles), then will respect the activities priorities (if more than one activity are having the same stacking efficiency).

SJ also consists of two justification cycles (right & left), and its process can be detailed as follows:

(1) In the right (left) cycle, activities are prioritized in ascending (descending) order based on their latest finish time (earliest start time), so the higher (lower) the *LFT* (*EST*) the more priority the activity gets and the earlier it will be scheduled.
(2) Activities will then be scheduled in successive iterations, each corresponding to a time period t in the project time frame, starting from time period 0 (T) for right (left) cycle.
(3) During each cycle/iteration, a list of activities eligible for scheduling is prepared. An activity is considered as eligible for scheduling if it can or start (finish) in the current time period t without violating resources constraints throughout its total duration. Activities are sorted in the list based on their initial priority order, but if there exists an activity which must be scheduled in this time period (i.e. the activity's $LFT = t$ ($EST = t$)) or otherwise it will impact the project duration, it will be given the highest priority.
(4) The highest priority activity will be scheduled, and step 3 is to be repeated.
(5) If there is no eligible activities within this iteration, the current time period is advanced to the next (previous) time period $t + 1$ ($t - 1$) until all activities are scheduled.

4.1. Example 1

To illustrate how *SJ* works and compare it to the original *DJ*, let's consider the simple example shown in Fig. 1. The network consists of 4 activities (+2 dummies), and one renewable resource with 2 available units.

Table 1 was prepared for assessing the effect of the *SGS*, *DJ* and *SJ* on the search space size. Since the number of activities to be optimized is four, then the number of possibilities for priorities list is 4! = 24 options (2345,2354,...,etc.). The serial schedule generation scheme (*SSGS*) eliminated all options which do not respect precedence. For example, priorities list "2435" will be automatically converted to "2345" during *SGS*, as scheduling of activity 4 will have to wait until activity 3 is scheduled. And accordingly, as shown in Table 1, *SGS* will reduce the search space from 24 options down to only 6 options.

As shown in Fig. 2, after the application of *SGS*, priorities list options were reduced to six options: 2354, 2534, 2345, 3425, 3245, & 3254; out of which, three options (3425, 3245, & 3524) are achieving the optimum make-span of 3 time units.

When the *SSGS* is followed by a *DJ*, solution "2345" will be optimized and converted to "3425" during the right justification cycle. So, the search space was reduced into 5 options after *DJ*, 2 of which are non-optimum & 3 are optimum. On the other hand, when the *SGS* is followed by a *SJ*, solutions "2354", "2534", & "2345" and converted to "3425" during the right justification cycle; which leads to a reduction of the search space down to 3 options, all are optimum.

4.2. Example 2

To elaborate the observations of example 1, another larger example was considered as shown in Figs. 3 and 4. The network consists of 7 activities (+ 2 dummies), and one renewable resource with 4 available units.

The activities to be optimized are seven, so the number of possibilities for priorities list is 7! = 5040 options. *SGS* eliminated 4725 options which do not respect precedence, leaving only 315 valid options. A detailed analysis was performed to determine how much options were eliminated by each of *SJ* & *DJ*, as well as both together (*Stacking & Double Justification*, abbreviated as *SDJ* through the balance of the paper); and the results were listed in Table 2. The reduction in search space was higher for *DJ* than *SJ* in forward scheduling, while the reverse for backward scheduling; and finally for *SDJ*, the reduction was higher than both *DJ* & *SJ*. The extra search space reduced for *SDJ* above *DJ* is completely dependent on the problem's network structure and resources requirements, in

ID	d	r	Successors
1	0	-	2, 3
2	1	1	5
3	1	2	4
4	1	1	6
5	1	2	6
6	0	-	-

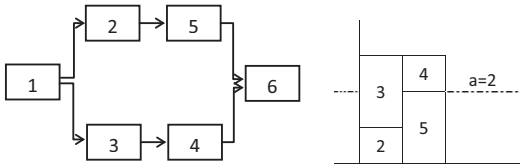


Fig. 1. Example 1 – Activities list, network and original resource profile.

Table 1
Effect of applying *SGS*, *DJ* and *SJ* on the search space of example 1.

Original	2354	2453	2534	2543	4253	4523	5234	5243	5423	2345	2435	4235	3425	3452	3542	4325	4352	4532	5342	5432	3245	3254	3524	5324
SGS	2354	2534								2345			**3425**								**3245**	**3254**		
DJ	2354	2534								**3425**											**3245**	**3254**		
SJ	**3425**																				**3245**	**3254**		

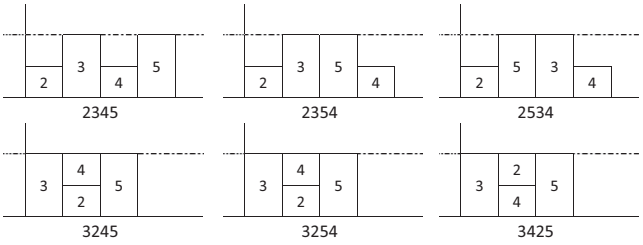


Fig. 2. Example 1 – Resource profiles for available options after SSGS.

ID	d	r	Successors
1	0	-	2,5,6
2	3	3	3,8
3	2	2	4
4	2	1	9
5	4	2	9
6	1	1	7
7	1	3	9
8	2	4	9
9	0	-	-

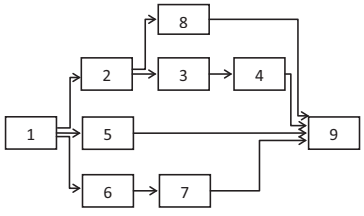


Fig. 3. Example 2 – Activities list & network.

example 2 this extra reduction is 1%, while it was more than 33% in example 1.

The effect of this reduction will appear clearly in the experimental results that it significantly improves the final solutions; but for the current example, Table 3 shows this impact on the amount of optima reached with only 1 iteration (note: *LFT* priority rule was used).

5. Combined priority rules (CPR)

The main function of using priority rules in scheduling is to arrange the activities list in an order which will lead to a good solution when the schedule is generated. Implementing a priority rule involves minor computational effort, and accordingly utilizing the proper rule can have a significant impact on the solution's quality.

In meta-heuristics, the use of priority rules mostly involves the initialization of the first population; and in particle swarm specifically, initializing the swarm particles with an ineffective priority rule will lead to placing the particles in a bad side of the search space, and accordingly will require large computational effort to reach the optimum solution, if not trapped away from it.

Each of the most commonly used priority rules in literature is having a different performance with difference problem types and sizes (for comparison of priority rules performance refer to Kolisch (1996b)). Regardless of which rule is having the highest performance, the facts that each rule is having a different

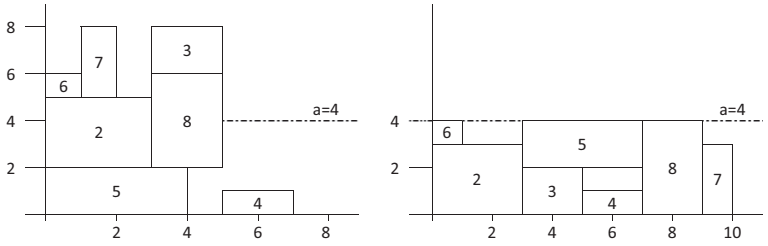


Fig. 4. Example 2 – Original resource profile and optimum resource profile.

Table 2
Example 2 – Search space size under different justification schemes.

	Search Space Size					Search Space reduction by SGS		
	Original	SSGS	DJ	SJ	SDJ	DJ (%)	SJ (%)	SDJ (%)
Forward scheduling	5040	315	215	225	215	31.7	28.6	31.7
Backward scheduling	5040	315	163	157	155	48.3	50.2	50.8
Forward/Backward	5040	630	378	382	370	40.0	39.4	41.3

Table 3
Example 2 - Optima reached under different justification schemes.

	Optima reached				Solution rate (from 315 options)			
	SSGS	DJ	SJ	SDJ	SSGS (%)	DJ (%)	SJ (%)	SDJ (%)
Forward Scheduling	165	242	247	256	52.4	76.8	78.4	81.3
Backward Scheduling	76	139	203	208	24.1	44.1	64.4	66.0
Forward/Backward	197	244	247	256	62.5	77.5	78.4	81.3

behaviour and producing different moderate to good quality solutions are the basis of the proposed *combined priority rules* (*CPR*) approach.

The concept of *CPR* is to make use of the differential behaviour of priority rules in order to achieve proper spread of initial population on the good areas of the search space. If we assumed that the search space consists of several good areas, each with a local optimum, and that each priority rule directs to one of these areas, then initializing each individual (or particle for PSO) in the initial population with a different priority rule will lead to proper diversity of population over several areas of the search space with high prospects of being close to the overall optimum, which consequently will lead to reaching the optimum quicker than normal initialization process. This assumption will be proven to be correct in the test results (section 8.1). And since applying priority rules involves very small computational effort and the particles were going to be initialized in all cases, then this process involves almost no additional computational burden.

6. Particle swarm optimization (PSO)

The particle swarm optimization (PSO) is an evolutionary multi-heuristic technique introduced by Kennedy and Eberhart (1995), inspired by the swarming behaviour of flocking birds, in which each individual movement is influenced by its own experience and the overall swarm experience. PSO resembles this behaviour through a population of swarm particles (or a potential solution) moving in space (i.e. search space; the velocity of each particle is calculated based on the experience of the particle (local optimum) as well as the overall swarm experience (global optimum).

Each swarm particle is presented with two characteristics, position and speed. Swarm particles corresponds to potential solutions, so each particle's position and velocity vectors will be consisting of n components corresponding to the number of components within the analysed problem. In each iteration, the position vector of each particle is updated using the velocity vector, the particle (or solution) fitness is calculated, and the local & global best solutions are updated. The velocity vector are updated in each iteration (or generation) based on the experience gained in the previous iterations.

6.1. The PSO model

Considering a swarm with M particles, and a problem with n components, then the particles are presented with the position vector $X_i = \{X_{i1},\ldots,X_{in}\}$, the velocity vector $V_i = \{V_{i1},\ldots,V_{in}\}$, and the particle's local best fitness f_{li} and the best solution position vector $L_i = \{L_{i1},\ldots,L_{in}\}$. And additionally, the swarm experience will be presented by the global best f_g and the global best solution position vector $G = \{G_1,\ldots,G_n\}$.

In the original PSO, presented by Kennedy and Eberhart (1995), the velocity and position vectors of component j of particle i at iteration t are to be updated using Eqs. (1) and (2); where V_{ij}^{t-1} and X_{ij}^{t-1} are the velocity and position vectors of the same component at iteration $t-1$, w is the inertia weight for controlling the influence of previous iteration's velocity to succeeding iteration, c_1 and c_2 are learning factors to adjust the approach behaviour of the particle to the local and global optima, and r_1 and r_2 are two random numbers between 0 and 1. Bratton and Kennedy (2007) suggested that a value of 0.73 for w and 2.05 for both c_1 & c_2 would result in efficient performance of the PSO, and were implemented accordingly in most of succeeding researches; and adopted accordingly in this paper.

$$V_{ij}^{t} = w \times V_{ij}^{t-1} + c_1 \times r_1 \times \left(L_{ij}^{t-1} - X_{ij}^{t-1}\right) + c_2 \times r_2 \times \left(G_j^{t-1} - X_{ij}^{t-1}\right) \quad (1)$$

$$X_{ij}^{t} = X_{ij}^{t-1} + V_{ij}^{t} \quad (2)$$

6.2. PSO variations

Several variations were proposed for the original PSO to enhance its performance and overcome its drawbacks, such as parameter dependency, loss of diversity and early convergence; the most famous of which is the "*Standard PSO*" by Bratton and Kennedy (2007), where the velocity update method was modified as shown in Eq. (3). The equation was modified from *original PSO* by the introduction of the *constriction factor* (γ) as a multiplier to the equation, in lieu of the inertia weight(w) which was applied only to previous iteration's velocity V_{ij}^{t-1}. The optimum value for γ was also suggested as 0.73.

$$V_{ij}^{t} = \gamma \times \left(V_{ij}^{t-1} + c_1 \times r_1 \times \left(L_{ij}^{t-1} - X_{ij}^{t-1}\right) + c_2 \times r_2 \times \left(G_j^{t-1} - X_{ij}^{t-1}\right)\right) \quad (3)$$

Reviewing PSO variations and their pros and cons are beyond the scope of this paper; but for examples of other PSO variations refer to the following publications: Chen and Li (2007), Higashi and Iba (2003), Janson and Middendorf (2005), Kiranyaz, Ince, Yildirim, and Gabbouj (2010), Kiranyaz, Pulkkien, and Gabbouj (2011), Krohling and Coelho, 2006, Liang and Qin (2006), Lovberg and

Krink (2002), Mendes, Kennedy, and Neves (2004), Ratnaweera, Halgamuge, and Watson (2002), (2003), Suganthan (1999), Van den Bergh and Engelbrecht (2002), (2004), Xie, Zhang, and Yang (2002a), (2002b), (2002c), and Zhang and Xie (2003).

6.3. PSO communication topologies

The PSO communication topology represents how gained experience is passed between successive iterations. The original PSO was based on the "*gbest*" topology (Fig. 5a), where velocity is updated as shown in Eq. (1); this topology is the most common in PSO researched. The "*lbest*" topology (c.f. Bratton & Kennedy, 2007) was introduced to minimize situations where the algorithm is trapped into local optima.

Also several variations were proposed for the PSO communication topology (refer to previous section); listing these variations are also beyond the scope of this paper, but for the sake of comparison of this paper's test results with the current best results in literature for PSO with RCPSP, obtained by Chen (2011), the neighbouring topology will be reviewed and used at the last stage of the testing process (refer to Section 8.4 of this paper).

Chen (2011) proposed the modification of the communication topology by the addition of a new parameter, the *gbest ratio* (*GR*), to manipulate the trade-off between *gbest* and *lbest* randomly during the optimization iterations with a predefined trade-off range corresponding to the *GR* value. Accordingly, the velocity vector will be updated using Eq. (4), where Y_j is the position vector resulting from the *gbest/lbest* trade-off as per Eq. (5). During each iteration, a random value *rand* is obtained [0, 1] and compared to the predefined *GR*, if the value is smaller than *GR* the *gbest* is used, otherwise the position vector of the best neighbour is used.

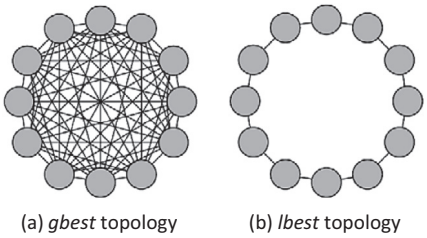
$$V_{ij}^{t} = \gamma \times \left(V_{ij}^{t-1} + c_1 \times r_1 \times \left(L_{ij}^{t-1} - X_{ij}^{t-1}\right) + c_2 \times r_2 \times \left(Y_j^{t} - X_{ij}^{t-1}\right)\right) \tag{4}$$

$$Y_j^{t} = \begin{cases} G_j^{t-1} & r \quad \text{and} \quad < GR \\ X_{kj}^{t-1} \text{ where } k \in \text{best neighbor}(i) & \text{otherwise} \end{cases} \tag{5}$$

6.4. PSO applications in RCPSP

Since the presentation of PSO by Kennedy and Eberhart (1995), and it attracted a lot of research and practical applications. Due to the complexity of RCPSPs, few researches applied PSO for solving RCPSPs; but, because PSO is relatively new, the number of applications in RCPSP was not large, especially when compared to other optimization techniques such as Genetic Algorithms.

PSO applications in RCPSP can be categorized under two categories corresponding to the main RCPSP's categories, single mode RCPSP (or SRCPSP) and multi-mode RCPSP (or MRCPSP). For SRCPSP, which is the focus of this research, Zhang et al. (2005) developed the first PSO scheme for RCPSP; this was followed by few other researches concentrating either on improvement of solution results (Chen, 2011) or extensions of PSO and applying on RCPSP (Chen, Wub, Wang, & Lo, 2010; Tchomté & Gourgand, 2009). The experimental results of these applications were reviewed, and the best results (Chen, 2011) were used for comparison with this study's results. Implementation of PSO was made in several practical applications such as power systems optimization (Paliwal, Patidar, & Nema, 2014), Patients scheduling (Kanaga & Valarmathi, 2012) and urban water resources planning (Zarghami & Hajykazemian, 2013).



(a) *gbest* topology (b) *lbest* topology

Fig. 5. PSO communication topologies [From Bratton and Kennedy (2007)].

While for MRCPSP, the applications are minimal. According to the review made in this research, and similarly as per the MRCPSP survey presented by Peteghem and Vanhoucke (2014), only two MRCPSP researches applied PSO: Zhang, Tam, and Li (2006) and Jarboui, Damak, Siarry, and Rebai (2008); other implementations of PSO in a multi-mode environment were directly applied to practical applications, such as optimization of quay crane time (Wang, Zheng, Zheng, Guo, & Liu, 2012).

7. Multiple justification particle swarm optimization (MJPSO) algorithm

The proposed *multiple justification particle swarm optimization* (*MJPSO*) algorithm is a combination of the techniques explained early in this paper. It is simply a *Standard PSO*, with the implementation of *FBI*, *SDJ* (with mapping) and *CPR*.

And accordingly, the pseudo code for the *MJPSO* can be summarized as shown in Table 4.

The selected priority can be either a single priority rule, or the *CPR*; and the justification scheme can be original justification, Stacking Justification, or a combination of both. The total generated schedules can be calculated using Eq. (6); where for each particle, each justification cycle (whether original or stacking, left or right) is considered as a generated schedule, in addition to the original schedule generated with selected *SGS*.

$$\textit{Total generated schedules} = (\text{No. of swarm particles}) \times (1 + \text{No. justification cycles}) \tag{6}$$

8. Testing results & analysis

The *Project Scheduling Problem Library* (*PSPLIB*)'s (Kolisch & Sprecher, 1997) SRCPSP j-30 (480 instances), j-60 (480 instances), j-90 (480 instances) & j-120 (600 instances) were used for this testing purpose. These problem sets were extensively used in literature which will enable proper comparability for the algorithms performance. Results were obtained using a personal laptop with Intel core processor i7 2.4 GHz. The computer was operated by

Table 4
MJPSO pseudo code.

```
While total generated schedules < schedules limit
  t = 1
  For each particle i in forward & backward swarms
    If t = 1 Initialize swarm particles using selected priority rule
    Update V_i^t & X_i^t
    Generate schedule using selected SGS
    Apply selected justification scheme
    Map justified solution into X_i^t
    Calculate particle's fitness
    Update local best L_i^t & global best G^t
  End for
  t = t + 1
End while
```

Microsoft Windows 7, and the algorithm was programmed in Visual C#.Net 4.0.

For j-60, j-90 & j-120, not all optimum values are known; so, the comparison was made using Eq. (8), where the *average deviation from critical path* (AD_{CP}) is the time increase due to the consideration of resources constraints between the best solution reached and the original problems' critical path; while for the j-30 instances, all optimum values are well known, so another measure is commonly used in literature as shown in Eq. (7), where the *average deviation from optimum* (AD_O) represents the variance between the analysis results and the instances optima.

$$AD_O = \frac{\left(\sum_{i=1}^{N} \frac{(BT_i - OT_i)}{OT_i}\right)}{N} \tag{7}$$

$$AD_{CP} = \frac{\left(\sum_{i=1}^{N} \frac{(BT_i - CP_i)}{CP_i}\right)}{N} \tag{8}$$

Where N is the total number of instances, BT_i is the best project time found for instance i, and OT_i is the optimum project time for instance i, CP_i is the total time of the critical path for instance i.

As stated in the previous section, the algorithm's stopping criteria was set to the number of schedules generated during the analysis, which enables fair comparison among algorithms regardless of the efficiency of the hardware or programming technique implemented; where 1000, 5000 & 50,000 schedules were generated through all testing process for the comparability with other test results in literature.

Standard PSO was adopted, with a *gbest* communication topology (except where indicated that *neighbouring* topology is used). *FBI* was implemented, with the number of forward and backward particles as indicated in headers of results tables. And finally, three of the well-known, and of proven high performance priority rules (c.f. Kolisch, 1996b), were used (LST, LFT & MTS), as well as the proposed combined priority rules (*CPR*) approach. Testing was performed using serial *SGS* because it was proven by Kolisch (1996a) that it is sometimes impossible to reach an optimal solution with parallel *SGS*.

8.1. Justification test results

For testing the new justification technique, five justification schemes were implemented: *NJ* (no justification), *DJ* (original double justification as per Valls, Ballestín, and Quintanilla (2005)), *SJ* (Stacking Justification as proposed in this paper), *SDJ* (*SJ* & *DJ*, where both techniques are applied to all particles during each iteration), and *ASDJ* (alternating *SJ* & *DJ*, where both techniques are used alternatively between iterations). The constriction factor value was set to 0.73 (as suggested by Bratton & Kennedy, 2007).

Generally, *SJ* outperformed *DJ* for small sized problems, while the case became gradually reversed with the increase in problem size. However, using combined justifications (or *SDJ*) was always showing better performance than both techniques separately. While the test results for *ASDJ* showed an unsteady good results for the 1000 schedules (mainly due to the reduction in number of justification cycles), but the performance was definitely lower than *SDJ* for larger sized problems and larger number of schedules. And accordingly, *SDJ* will be used for further detailed analysis in the following sections.

8.2. Improving test results

For the purpose of improving the test results of the *SDJ* justification scheme, detailed testing was performed using different values of constriction factor. Table 6 shows that the larger the problem size, the larger the need for lower constriction factor. Best results were achieved using $\gamma = 0.4$ to 0.6 for j-30, $\gamma = 0.3$ to 0.5 for j-60, and $\gamma = 0.2$ to 0.3 for both j-90 & j-120. Testing was also performed one time using serial *SGS*.

8.3. CPR test results

Results showed in Tables 5 and 6 shows clearly that *CPR* is having a significant improvement to the quality of achieved solutions when the stopping condition is set 1000 schedules. This improvement decreases with the increase of analysis time (or number

Table 5
Test results for different justification schemes.

Justification Scheme	Priority Rule	1,000 Schedules (10F/10B Particles)				5,000 Schedules (10F/10B Particles)				50,000 Schedules (20F/20B Particles)			
		AD_O	AD_{CP}			AD_O	AD_{CP}			AD_O	AD_{CP}		
		j-30	j-60	j-90	j-120	j-30	j-60	j-90	j-120	j-30	j-60	j-90	j-120
NJ	LST	1.34	14.81	14.95	45.02	0.70	13.79	14.25	43.59	0.36	12.77	13.16	41.00
	LFT	1.33	14.98	15.11	45.62	0.66	13.90	14.21	43.87	0.26	12.57	12.91	40.63
	MTS	1.18	14.41	14.46	44.03	0.57	13.65	13.95	42.75	0.21	12.50	12.79	39.74
	CPR	0.90	14.54	14.65	44.03	0.63	13.57	13.98	43.22	0.30	12.49	12.78	39.92
DJ	LST	0.63	12.98	12.60	38.66	0.21	12.06	11.97	37.13	0.07	11.40	11.40	35.57
	LFT	0.66	12.97	12.66	38.63	0.20	12.05	12.00	37.19	0.08	11.43	11.44	35.65
	MTS	0.73	13.01	12.62	38.56	0.18	12.06	11.93	37.07	0.06	11.44	11.40	35.68
	CPR	0.50	12.91	12.58	38.58	0.16	11.98	11.93	36.95	0.06	11.40	11.37	35.61
SJ	LST	0.57	12.92	12.89	39.40	0.13	12.13	12.26	38.07	0.04	11.53	11.70	36.71
	LFT	0.46	12.92	12.85	39.33	0.14	12.12	12.27	38.22	0.06	11.52	11.71	36.67
	MTS	0.55	12.92	12.87	39.19	0.13	12.15	12.26	38.07	0.06	11.62	11.66	36.63
	CPR	0.39	12.90	12.72	38.77	0.13	12.15	12.19	37.96	0.06	11.59	11.68	36.62
SDJ	LST	0.51	12.46	12.13	37.23	0.11	11.73	11.60	36.13	0.03	11.17	11.15	34.89
	LFT	0.49	12.52	12.16	37.30	0.09	11.64	11.57	36.04	0.02	11.19	11.14	34.93
	MTS	0.54	12.50	12.11	37.22	0.09	11.68	11.60	36.04	0.03	11.19	11.11	34.94
	CPR	0.38	12.39	12.08	37.04	0.08	11.63	11.59	36.02	0.03	11.21	11.13	34.90
ASDJ	LST	0.40	12.76	12.51	38.59	0.14	12.04	12.13	37.37	0.06	11.49	11.52	35.99
	LFT	0.46	12.71	12.59	38.55	0.14	12.07	12.07	37.38	0.05	11.48	11.53	36.00
	MTS	0.47	12.76	12.61	38.56	0.09	12.10	12.09	37.39	0.05	11.51	11.57	36.00
	CPR	0.36	12.73	12.52	38.58	0.13	12.05	12.06	37.29	0.03	11.50	11.50	35.92

Table 6
Test results for SDJ under different constriction factor values.

C.F. (χ)	Priority rule	1000 Schedules (10F/10B Particles)				5000 Schedules (10F/10B Particles)				50,000 Schedules (20F/20B Particles)			
		AD_O		AD_{CP}		AD_O		AD_{CP}		AD_O		AD_{CP}	
		j-30	j-60	j-90	j-120	j-30	j-60	j-90	j-120	j-30	j-60	j-90	j-120
0.6	LST	0.39	12.29	12.06	36.97	0.09	11.39	11.39	35.46	0.03	11.02	10.95	34.44
	LFT	0.44	12.44	12.18	37.49	0.10	11.42	11.42	35.53	0.02	10.98	10.94	34.46
	MTS	0.41	12.33	12.06	37.06	0.08	11.46	11.38	35.50	0.03	11.01	10.92	34.49
	CPR	0.31	12.40	12.16	37.25	0.08	11.43	11.33	35.53	0.02	10.99	10.95	34.45
0.5	LST	0.36	12.16	11.94	36.63	0.11	11.31	11.12	34.85	0.04	10.92	10.66	33.73
	LFT	0.32	12.17	12.01	37.06	0.15	11.29	11.07	34.84	0.04	10.89	10.61	33.78
	MTS	0.42	12.13	11.89	36.73	0.12	11.32	11.12	34.94	0.04	10.85	10.65	33.83
	CPR	0.28	12.16	11.95	36.79	0.09	11.24	11.05	34.80	0.03	10.89	10.65	33.71
0.4	LST	0.38	12.00	11.72	36.15	0.22	11.25	10.84	34.18	0.06	10.92	10.38	32.96
	LFT	0.35	12.03	11.81	36.39	0.20	11.31	10.88	34.18	0.07	10.97	10.39	32.94
	MTS	0.41	11.96	11.70	36.20	0.20	11.27	10.88	34.21	0.08	10.99	10.45	33.02
	CPR	0.25	12.01	11.73	36.22	0.12	11.19	10.84	34.09	0.07	10.93	10.42	32.85
0.3	LST	0.39	11.91	11.56	35.81	0.30	11.28	10.79	33.82	0.16	11.07	10.38	32.57
	LFT	0.39	11.97	11.65	35.96	0.26	11.31	10.79	33.83	0.12	11.07	10.37	32.59
	MTS	0.49	11.93	11.56	35.78	0.39	11.31	10.80	33.88	0.14	11.04	10.41	32.57
	CPR	0.26	11.88	11.54	35.83	0.17	11.29	10.86	33.91	0.09	11.04	10.35	32.54
0.2	LST	0.53	11.97	11.55	35.56	0.31	11.37	10.82	33.88	0.14	11.07	10.36	32.46
	LFT	0.51	12.06	11.59	35.80	0.35	11.37	10.85	33.82	0.13	11.07	10.36	32.43
	MTS	0.73	11.92	11.56	35.54	0.61	11.41	10.87	33.87	0.26	11.08	10.35	32.50
	CPR	0.28	11.94	11.54	35.65	0.26	11.29	10.81	33.84	0.11	11.03	10.37	32.44
0.1	LST	0.73	12.14	11.59	35.63	0.46	11.61	11.07	34.28	0.17	11.05	10.42	32.78
	LFT	0.78	12.22	11.68	35.91	0.39	11.55	11.04	34.31	0.16	11.05	10.42	32.78
	MTS	0.96	12.15	11.57	35.66	0.65	11.61	11.05	34.26	0.31	11.09	10.44	32.75
	CPR	0.38	12.15	11.61	35.76	0.28	11.55	11.02	34.19	0.14	11.02	10.40	32.75

Table 7
Test results (best values) for *MJPSO* using neighbouring topology (SDJ implemented).

Topology	GR	N_{Sch}	j-30 AD_O	Best results param.			j-60 AD_O	Best results param.			j-120 AD_O	Best results param.		
				Priority rule	χ	Particles Fo./ Ba.		Priority rule	χ	Particles Fo./ Ba.		Priority rule	χ	Particles Fo./ Ba.
Neighbouring	0.75	1000	0.22	CPR	0.3	10/10	11.88	CPR	0.3	10/10	35.69	CPR	0.2	10/10
		5000	0.06	LFT	0.6	10/10	11.22	CPR	0.3	20/20	33.78	CPR	0.3	10/10
		50,000	0.02	CPR	0.6	20/20	10.85	CPR	0.5	20/20	32.40	LFT	0.2	20/20
	0.5	1000	0.22	CPR	0.4	10/10	11.86	CPR	0.3	10/10	35.75	CPR	0.2	10/10
		5000	0.05	CPR	0.6	20/20	11.19	LFT	0.4	10/10	33.83	CPR	0.3	10/10
		50,000	0.02	CPR	0.6	20/20	10.83	CPR	0.5	20/20	32.40	LFT	0.3	20/20
	0.25	1000	0.24	CPR	0.4	10/10	11.97	CPR	0.3	10/10	35.81	CPR	0.2	10/10
		5000	0.06	CPR	0.5	20/20	11.20	LFT	0.4	10/10	33.93	CPR	0.3	10/10
		50,000	0.02	CPR/LFT	0.4	20/20	10.78	LFT	0.4	20/20	32.47	CPR	0.3	20/20
lBest	0.0	1000	0.25	CPR	0.3	10/10	11.98	LFT	0.3	10/10	35.87	CPR	0.2	10/10
		5000	0.06	CPR	0.6	10/10	11.22	CPR	0.4	10/10	33.99	CPR	0.3	10/10
		50,000	0.02	CPR	0.6	20/20	10.86	LFT	0.4	20/20	32.57	CPR	0.2	20/20

schedules to be generated); however, the quality of results using *CPR* was also efficient in most cases in comparison with the other priority rules.

8.4. Neighbouring topology test results

Since *Justification PSO* (or *JPSO*) presented by Chen (2011) is currently the best performing PSO algorithm for single mode RCPSP in literature, and since some of the *JPSO*'s best results were obtained with *neighbouring* PSO topology, then for the sake of comparison, *neighbouring* PSO topology was applied with different *global ratio* (*GR*) values. Table 8 contains the best results achieved with the *MJPSO* algorithm after applying *neighbouring* PSO topology. The implementation of *neighbouring* PSO topology showed some improvement to the *MJPSO* solutions quality (as shown in Table 7).

Finally, Table 8 shows a comparison between the performances of *JPSO* & *MJPSO*; the comparison was performed on the *gbest* & *neighbouring* PSO topologies, and on the same priority rule, while adding the *CPR* for *MJPSO* for best results listing. Results show clearly that the implementation of *SDJ* & *CPR* lead to a significant improvement than *JPSO*.

8.5. MJPSO algorithm rating

Kolisch and Hartmann (2006) performed a detailed survey on the best performing algorithms for solving RCPSPs. The top 20 performance results in this survey were amended with both experimental results of the *JPSO* (Chen, 2011) and the *MJPSO* (proposed in this paper). For detailed review of these algorithms, refer to the original references included in the original survey. Results of Tchomté and Gourgand (2009) were not considered in this

Table 8
Performance comparison between JPSO & MJPSO algorithms.

PSO Topology	Priority Rule	Algorithm	1000 Schedules			5000 Schedules			50,000 Schedules		
			AD_O	AD_{CP}		AD_O	AD_{CP}		AD_O	AD_{CP}	
			j-30	j-60	j-120	j-30	j-60	j-120	j-30	j-60	j-120
gBest	LFT	JPSO	0.29	12.03	35.71	0.14	11.43	33.88	0.07	11.41	33.72
		MJPSO	0.32	11.97	35.80	0.05	11.31	***33.82***	***0.02***	***10.85***	***32.43***
	CPR	MJPSO	***0.25***	***11.88***	***35.65***	***0.08***	***11.19***	33.84	***0.02***	10.89	32.44
Neighboring (incl. lBest)	LFT	JPSO	0.29	12.03	35.71	0.14	11.43	33.88	0.04	11.00	32.89
		MJPSO	0.31	11.96	35.80	0.06	11.19	33.86	***0.02***	***10.78***	***32.40***
	CPR	MJPSO	***0.22***	***11.86***	***35.69***	***0.05***	11.21	***33.78***	***0.02***	10.83	32.43

comparison as best PSO results because the value presented for critical path average deviation AD_{CP} for j-60 problem (9.01) is lower than that of the j-60's lower bounds (9.42), which is not feasible if their calculation method for the AD_{CP} is similar to the common method used in literature (Eq. (8)).

The performance comparison presented in Tables 9–11 corresponding to SRCPSP j-30, j-60 & j-120, shows that the *MJPSO* is highly ranked between state-of-the-art algorithms for solving single mode RCPSPs; and it also outperformed other PSO algorithms.

Table 9
Algorithms comparison for AD_O of ProGen SRCPSP j-30.

Algorithm	SGS	Reference	Max. # of schedules		
			1000	5000	50,000
GA, TS—path relinking	Both	Kochetov and Stolyar (2003)	0.10	0.04	0.00
Scatter Search—FBI	Serial	Debels, De Reyck, Leus, and Vanhoucke (2006)	0.27	0.11	0.01
PSO—SDJ, FBI (MJPSO)	***Both***	***This paper***	***0.22***	***0.05***	***0.02***
GA—hybrid, FBI	Serial	Valls et al. (2008)	0.27	0.06	0.02
GA—FBI	Serial	Valls et al. (2005)	0.34	0.20	0.02
GA—forw.–backw., FBI	Both	Alcaraz et al. (2004)	0.25	0.06	0.03
GA—forw.–backw.	Serial	Alcaraz et al. (2001)	0.33	0.12	–
PSO—DJ, FBI (JPSO)	Serial	Chen (2011)	0.29	0.14	0.04
Sampling—LFT, FBI	Both	Tormos and Lova (2003b)	0.25	0.13	0.05
TS—activity list	Serial	Nonobe and Ibaraki (2001)	0.46	0.16	0.05
Sampling—LFT, FBI	Both	Tormos and Lova (2001)	0.30	0.16	0.07
GA—self-adapting	Both	Hartmann (2002)	0.38	0.22	0.08
GA—activity list	Serial	Hartmann (1998)	0.54	0.25	0.08
Sampling—LFT, FBI	Both	Tormos and Lova (2003a)	0.30	0.17	0.09
TS—activity list	Serial	Klein (2000)	0.42	0.17	–
Sampling—random, FBI	Serial	Valls et al. (2005)	0.46	0.28	0.11
SA—activity list	Serial	Bouleimen and Lecocq (2003)	0.38	0.23	–
GA—late join	Serial	Coelho and Tavares (2003)	0.74	0.33	0.16
Sampling—adaptive	Both	Schirmer (2000)	0.65	0.44	–
TS—schedule scheme	Related	Baar et al. (1998)	0.86	0.44	–

Table 10
Algorithms comparison for AD_{CP} of ProGen SRCPSP j-60.

Algorithm	SGS	Reference	Max. # of schedules		
			1000	5000	50,000
Scatter search—FBI	Serial	Debels et al. (2006)	11.73	11.10	10.71
GA—hybrid, FBI	Serial	Valls et al. (2008)	11.56	11.10	10.73
GA, TS—path relinking	Both	Kochetov and Stolyar (2003)	11.71	11.17	10.74
GA—FBI	Serial	Valls et al. (2005)	12.21	11.27	10.74
GA—forw.–backw., FBI	Both	Alcaraz et al. (2004)	11.89	11.19	10.84
PSO—SDJ, FBI (MJPSO)	***Both***	***This paper***	***11.86***	***11.19***	***10.85***
PSO—DJ, FBI (JPSO)	Serial	Chen (2011)	12.03	11.43	11.00
GA—self-adapting	Both	Hartmann (2002)	12.21	11.70	11.21
GA—activity list	Serial	Hartmann (1998)	12.68	11.89	11.23
Sampling—LFT, FBI	Both	Tormos and Lova (2003b)	11.88	11.62	11.36
Sampling—LFT, FBI	Both	Tormos and Lova (2003a)	12.14	11.82	11.47
GA—forw.–backw.	Serial	Alcaraz et al. (2001)	12.57	11.86	–
Sampling—LFT, FBI	Both	Tormos and Lova (2001)	12.18	11.87	11.54
SA—activity list	Serial	Bouleimen and Lecocq (2003)	12.75	11.90	–
TS—activity list	Serial	Klein (2000)	12.77	12.03	–
TS—activity list	Serial	Nonobe and Ibaraki (2001)	12.97	12.18	11.58
Sampling—random, FBI	Serial	Valls et al. (2005)	12.73	12.35	11.94
Sampling—adaptive	Both	Schirmer (2000)	12.94	12.58	–
GA—late join	Serial	Coelho and Tavares (2003)	13.28	12.63	11.94
GA—random key	Serial	Hartmann (1998)	14.68	13.32	12.25
GA—priority rule	Serial	Hartmann (1998)	13.30	12.74	12.26
Sampling—adaptive	Both	Kolisch and Drexl (1996)	13.51	13.06	–

Table 11
Algorithms comparison for AD_{CP} of ProGen SRCPSP j-120.

Algorithm	SGS	Reference	Max. # of schedules		
			1000	5000	50,000
GA—hybrid, FBI	Serial	Valls et al. (2008)	34.07	32.54	31.24
GA—forw.–backw., FBI	Both	Alcaraz et al. (2004)	36.53	33.91	31.49
Scatter Search—FBI	Serial	Debels et al. (2006)	35.22	33.10	31.57
GA—FBI	Serial	Valls et al. (2005)	35.39	33.24	31.58
GA, TS—path relinking	Both	Kochetov and Stolyar (2003)	34.74	33.36	32.06
PSO—SDJ, FBI (MJPSO)	***Both***	***This paper***	***35.60***	***33.78***	***32.40***
Population-based—FBI	Serial	Valls et al. (2005)	35.18	34.02	32.81
PSO—DJ, FBI (JPSO)	Serial	Chen (2011)	35.71	33.88	32.89
GA—self-adapting	Both	Hartmann (2002)	37.19	35.39	33.21
Sampling—LFT, FBI	Both	Tormos and Lova (2003b)	35.01	34.41	33.71
Ant system	Serial	Merkle et al. (2002)	–	35.43	–
GA—activity list	Serial	Hartmann (1998)	39.37	36.74	34.03
Sampling—LFT, FBI	Both	Tormos and Lova (2003a)	36.24	35.56	34.77
Sampling—LFT, FBI	Both	Tormos and Lova (2001)	36.49	35.81	35.01
GA—forw.–backw.	Serial	Alcaraz et al. (2001)	39.36	36.57	–
TS—activity list	Serial	Nonobe and Ibaraki (2001)	40.86	37.88	35.85
GA—late join	Serial	Coelho and Tavares (2003)	39.97	38.41	36.44
Sampling—random, FBI	Serial	Valls et al. (2005)	38.21	37.47	36.46
SA—activity list	Serial	Bouleimen and Lecocq (2003)	42.81	37.68	–
GA—priority rule	Serial	Hartmann (1998)	39.93	38.49	36.51
Sampling—adaptive	Both	Schirmer (2000)	39.85	38.70	–
Sampling—LFT	Parallel	Kolisch (1996a)	39.60	38.75	37.74

9. Conclusions & future research

Efficient utilization of project resources with minimal computational burden is the aim of practical scheduling applications. This paper has proposed few techniques/approaches which can be used to support the improvement of RCPSP solving techniques in order to minimize the analysis time and improve solutions quality. The paper introduced a new justification technique, *Stacking Justification* (*SJ*), a new optimization algorithm *multiple justification PSO* (*MJPSO*), and a new approach for swarm particles initialization, *combined priority rules* (*CPR*).

The performance of the proposed techniques was tested using well recognized problem sets to enable proper comparability with other algorithms in literature. Experimental results illustrated that the combination of *SJ* with the original *Double Justification* (*DJ*) technique achieved a considerable improvement to solutions quality. The combination of both justification techniques was tested using particle swarm optimization; and the developed algorithm, MJPSO, outperformed many of the best performing algorithms in literature, and achieved best results with respect to applications of *PSO* technique for RCPSP.

Additionally, the use of the proposed *CPR* approach was proven to have significant improvement to results, especially for small number of generated schedules; while the improvement decreased (but still existed) with the amount of generated schedules. This behaviour is suitable for practical applications where achieving quick good-to-high quality solutions is necessary. *CPR* involves no additional computational burden, which makes its use also a free quality improvement approach.

For future research, several research directions can be suggested as follows: Firstly, this study, as well as few other studies in the scheduling context, adopted the justification of solutions of RCPSP type; so, it is suggested that future studies can develop improved schedule generation schemes and/or justification techniques for other problem types such as *Resource Investment Problem* (*RIP*) [*time-cost trade-off*] or *Resource Levelling Problem* (*RLP*) [*time-resource trade-off*].

Secondly, the examples in this paper showed that the combination of the two justification techniques caused considerable reduction to search space size, which accordingly impacted the solutions quality despite the large computational effort this combination exerted on the optimization process (about 40% reduction to the number of iterations to reach the schedules limit). Accordingly, future research can be performed for checking if adding more justification techniques can further reduce search space (i.e. improve optimization results).

Finally, according to an observation on the outcomes of SGS in the presented examples, the distribution resulting from eliminating priorities lists which does not respect precedence is not fairly distributed among other lists which do respect precedence. This will cause difficulties to any optimization algorithm in finding the optimum solution if it received less probability share from the eliminated options. This point requires another research to study the possibility of improving the modelling of SGS in order to have a resulting fairly distributed search space which will definitely support the generation of better quality solutions in shorter analysis time.

References

Alcaraz, J., Maroto, C., & Ruiz, R. (2004). Improving the performance of genetic algorithms for the RCPS problem. In: *Proceedings of the ninth international workshop on project management and scheduling*, Nancy, (pp. 40–43).

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

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. 1–8). Dordrecht: Kluwer Academic Publishers.

Bedworth, D. D., & Bailey, J. E. (1982). *Integrated production control systems – Management, analysis, design*. New York: Wiley.

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

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its simple mode versions. *European Journal of Operational Research, 140*(2), 268–281.

Bratton, D., & Kennedy, J. (2007). Defining a standard for particle swarm optimization. *Proceeding of IEEE swarm intelligence symposium, SIS, 2007*, 120–127.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research, 112*(1), 3–41.

Brucker, P., Knust, S., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 107*, 272–288.

Chen, R.-M. (2011). Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. *Expert Systems with Applications, 38*, 7102–7111.

Chen, X., & Li, Y. (2007). A modified PSO structure resulting in high exploration ability with convergence guaranteed. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 37*(5), 1271–1289.

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

Coelho, J., & Tavares, L. (2003). *Comparative analysis of metaheuricstics for the resource constrained project scheduling problem*. Technical report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal.

Coelho, J., & Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research, 213*(1), 73–82.

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

Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science, 38*, 1803–1818.

Demeulemeester, E., & Herroelen, W. (2002). *Project scheduling: A research handbook*. Kluwer Academic Publishers.

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

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

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research, 207*(1), 1–14.

Herroelen, W., Reyck, B. D., & Demeulemeester, E. (1998). Resource-constrained project scheduling: a survey of recent developments. *Computers and Operations Research, 25*(4), 279–302.

Higashi, H., & Iba, H., (2003). Particle swarm optimization with Gaussian mutation. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 72–79).

Icmeli, O., Erenguc, S. S., & Zappe, C. J. (1993). Project scheduling problems: A survey. *International Journal of Operations & Production Management, 13*(11), 80–91.

Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 35*(6), 1272–1282.

Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation, 195*, 299–308.

Kanaga, G. M., & Valarmathi, M. L. (2012). Multi-agent based Patient Scheduling Using Particle Swarm Optimization. *Procedia Engineering, 30*, 386–393.

Kelley, J. E. Jr., (1963). The critical-path method: Resources planning and scheduling. In J. F. Muth & G. L. Thompson (Eds.), *Industrial scheduling, prentice-hall* (pp. 347–365). NJ: Englewood Cliffs.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE international conference on neural networks*, Vol. 4, (pp. 1942–1948).

Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2010). Fractional particle swarm optimization in multi-dimensional search space. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 40*(2), 298–319.

Kiranyaz, S., Pulkkien, J., & Gabbouj, M. (2011). Multi-dimensional particle swarm optimization in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 40*(2), 298–319.

Klein, R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research, 38*, 3937–3952.

Klein, R., & Scholl, A. (1999). Progress: Optimally solving the generalized resource constrained project scheduling problem. *Mathematical Methods of Operations Research, 52*(3), 467–488.

Kochetov, Y., & Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In: *Proceedings of the 3rd international workshop of computer science and information technologies*, Russia.

Kolisch, R. (1996a). Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. *European Journal of Operational Research, 90*, 320–333.

Kolisch, R. (1996b). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management, 14*, 179–192.

Kolisch, R., & Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics, 43*, 23–40.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*, 23–37.

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega International Journal of Management Science, 29*, 249–272.

Kolisch, R., & Sprecher, A. (1997). PSPLIB – A project scheduling problem library: OR software – ORSEP operations research software exchange program. *European Journal of Operational Research, 96*, 205–216.

Krohling, R. A., & Coelho, L. S. (2006). Co-evolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 36*(6), 1407–1416.

Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource constrained project scheduling. *European Journal of Operational Research, 56*, 370–379.

Liang, J. J., & Qin, A. K. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*(3), 281–295.

Lo, S. T., Chen, R. M., Huang, Y. M., & Wu, C. L. (2008). Multiprocessor system scheduling with precedence and resource constraints using an enhance ant colony system. *Expert Systems with Applications, 34*(3), 2071–2081.

Lova, A., Tormos, P., Cervantes, M., & Barber, F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics, 117*(2), 302–316.

Lovberg, M., & Krink, T. (2002). Extending particle swarm optimisers with self-organized criticality. *Proceedings of the IEEE Congress on Evolutionary Computation, 2*, 1588–1593.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation, 8*(3), 204–210.

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

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation. *Management Science, 44*(5), 714–729.

Neumann, K., Schwindt, C., & Zimmermann, J. (2003). *Project Scheduling with Time Windows & Scarce Resources*. Springer.

Nonobe, K., & Ibaraki, T. (2001). "Formulation and Tabu search algorithm for the resource constrained project scheduling problem (RCPSP)". In C. C. Ribeiro & P. Hansen (Eds.), *Essays and surveys in meta-heuristics*. Kluwer Academic Publishers, pp. 557-588.

Paliwal, P., Patidar, N. P., & Nema, R. K. (2014). Determination of reliability constrained optimal resource mix for an autonomous hybrid power system using Particle Swarm Optimization. *Renewable Energy, 63*, 194–204.

Peteghem, V. V., & Vanhoucke, M. (2008). *A Genetic Algorithm for the Multi-Mode Resource-Constrained Project Scheduling Problem*. Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium 08/494, Ghent University, Faculty of Economics and Business Administration.

Peteghem, V. V., & Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research, 235*, 62–72.

Ranjbar, M., De Reyck, B., & Kianfar, F. (2009). A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research, 193*(1), 35–48.

Ratnaweera, A. C., Halgamuge, S. K., & Watson, H. C. (2002). Particle swarm optimiser with time varying acceleration coefficients. In *Proceedings of the international conference on soft computing and intelligent systems* (pp. 240–255).

Ratnaweera, A. C., Halgamuge, S. K., & Watson, H. C. (2003). Particle swarm optimization with self-adaptive acceleration coefficients. In *Proceedings of the first international conference on fuzzy systems and knowledge discovery* (pp. 264–268).

Rutenbar, R. A. (1989). Simulated annealing algorithms: An overview. *Circuits and Devices Magazine, IEEE, 5*, 19–26.

Schirmer, A. (2000). Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics, 47*, 201–222.

Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 1958–1962). IEEE Press.

Tchomté, K. S., & Gourgand, M. (2009). Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems. *International Journal of Production Economics, 121*(1), 57–67.

Thomas, P. R., & Salhi, S. (1998). A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics, 4*(2), 123–139.

Tormos, P., & Lova, A. (2003b). *Integrating heuristics for resource constrained project scheduling: One step forward*. Technical report, Department of Statistics and Operations Research, Universidad Politécnica de Valencia.

Tormos, P., & Lova, A. (2001). A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research, 102*, 65–81.

Tormos, P., & Lova, A. (2003a). An efficient multi-pass heuristic for project scheduling with constrained resources. *International Journal of Production Research, 41*(5), 1071–1086.

Valls, V., Ballestín, F., & Quintanilla, S. (2005). Justification and RCPSP: a technique that pays. *European Journal of Operational Research, 165*, 375–386.

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

Van den Bergh, F., & Engelbrecht, A. P. (2002). A new locally convergent particle swarm optimizer. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 96–101.

Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 3*, 225–239.

Wang, S., Zheng, J., Zheng, K., Guo, J., & Liu, X. (2012). Multi Resource Scheduling Problem Based on an Improved Discrete Particle Swarm Optimization. *Physics Procedia, 25*, 576–582.

Xie, X., Zhang, W., Yang, Z. (2002a). Adaptive particle swarm optimization on individual level. In *Proceedings of the sixth international conference on signal processing* (Vol. 2, pp. 1215–1218).

Xie, X., Zhang, W., & Yang, Z. (2002b). A dissipative particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation, 2*, 1456–1461.

Xie, X., Zhang, W., & Yang, Z. (2002c). Hybrid particle swarm optimizer with mass extinction. *Proceedings of the International Conference on Communication, Circuits and Systems, 2*, 1170–1173.

Zamani, R. (2013). A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. *European Journal of Operational Research, 229*(2), 552–559.

Zarghami, M., & Hajykazemian, H. (2013). Urban water resources planning by using a modified particle swarm optimization algorithm. *Resources, Conservation and Recycling, 70*, 1–8.

Zhang, H., Li, H., & Tam, C. M. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction, 14*, 393–404.

Zhang, C., Sun, J., Zhu, X., & Yang, Q. (2008). An improved particle swarm optimization algorithm for flow-shop scheduling problem. *Information Processing Letters, 108*(4), 204–209.

Zhang, H., Tam, C., & Li (2006). Multi-mode project scheduling based on particle swarm optimization. *Computer-Aided Civil and Infrastructure Engineering, 21*, 93–103.

Zhang, W.-J., & Xie, X.-F. (2003). DEPSO: Hybrid particle swarm with differential evolution operator. *Proceedings of the IEEE International Conference on System, Man, and Cybernetics, 4*, 3816–3821.