Computers & Operations Research 38 (2011) 44–50

Contents lists available at ScienceDirect

Computers & Operations Research

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

A Neurogenetic approach for the resource-constrained project scheduling problem

Anurag Agarwal [a], Selcuk Colak [b], Selcuk Erenguc [c,*]

[a] Department of Information Systems and Decision Sciences, College of Business, University of South Florida, Sarasota, FL 34243, USA
[b] Department of Business, College of Economics and Administrative Sciences, Cukurova University, Adana, Turkey
[c] Department of Information Systems and Operations Management, Warrington College of Business Administration, University of Florida, Gainesville, FL 32611-7164, USA

ARTICLE INFO

Available online 25 January 2010

Keywords:
Project management
Resource constrained project scheduling
Neural networks
Genetic algorithms
Neurogenetic

ABSTRACT

A variety of metaheuristic approaches have emerged in recent years for solving the resource-constrained project scheduling problem (RCPSP), a well-known NP-hard problem in scheduling. In this paper, we propose a Neurogenetic approach which is a hybrid of genetic algorithms (GA) and neural-network (NN) approaches. In this hybrid approach the search process relies on GA iterations for global search and on NN iterations for local search. The GA and NN search iterations are interleaved in a manner that allows NN to pick the best solution thus far from the GA pool and perform an intensification search in the solution's local neighborhood. Similarly, good solutions obtained by NN search are included in the GA population for further search using the GA iterations. Although both GA and NN approaches, independently give good solutions, we found that the hybrid approach gives better solutions than either approach independently for the same number of shared iterations. We demonstrate the effectiveness of this approach empirically on the standard benchmark problems of size J30, J60, J90 and J120 from PSPLIB.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is a classical problem in scheduling. The problem is widely applicable in project management, construction engineering, software development and production scheduling and is known to be strongly NP-hard [1]. The objective is to schedule the activities of a project so as to minimize the project makespan, subject to precedence and resource constraints. The quantities of available resources are assumed to be known and fixed for the entire duration of the project. The resources are considered renewable, i.e., they do not get consumed. Resource requirements and processing times for each activity are also known and fixed a priori. Preemption of activities is not allowed. This problem has been well researched for over four decades. Since exact approaches are not applicable to larger problems, due to the NP-hard nature of the problem, research efforts in recent years have focused on developing a variety of heuristic and metaheuristic approaches. Single-pass heuristics are generally based on priority rules such as minimum latest finish time next or maximum remaining work next, etc. Multi-pass heuristics or metaheuristics are also very popular because they tend to improve the solution quality over single-pass heuristics significantly by using some extra computation time. A variety of metaheuristic approaches such as genetic algorithms, tabu search, simulated annealing, ant-colony optimization and neural network-based approaches have been developed in the last 15 years.

In this paper we propose a hybrid approach called the Neurogenetic approach for solving the RCPSP. The proposed approach is a hybrid of genetic algorithms (GA) and neural networks-based (NN) approaches. The GA approach has shown remarkable success in solving the RCPSP and is one of the most preferred approaches for this problem. Colak et al. [2] proposed a NN-based approach which also gave very competitive results for this problem. Although both GA and NN-based approaches give some of the best known results in the literature, the two approaches are very different from each other in terms of search strategies. While the GA approach is very effective for global search, the NN-based approach is basically a nondeterministic local-search technique. We propose hybridizing these approaches in order to benefit from the complementary advantages of the two approaches—i.e. GAs providing the diversification in search while NNs providing intensification.

We propose an interleaving approach in which GA and NN iterations are interleaved, feeding their best solutions to each other alternately. Interleaving requires that switching back and forth between the two techniques be technically feasible.

* Corresponding author.
E-mail addresses: agarwala@sar.usf.edu (A. Agarwal), scolak@cu.edu.tr (S. Colak), selcuk.erenguc@cba.ufl.edu (S. Erenguc).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.01.007

Switching between NN and GA approaches is not straightforward given that the two approaches work quite differently. While GA is a solution-space based approach, NN-based approach is a problem-space based approach. In a solution-space based approach, the solution is perturbed from one iteration to the next, using some mechanism (such as crossover and mutation in Genetic Algorithms). Tabu search, simulated annealing, genetic algorithms and ant-colony optimization all belong to the class of solution-space based approaches. In a problem-space approach, the problem parameters are perturbed from one iteration to another, while using the same heuristic. For example, in an RCPSP, the problem parameter such as processing time may be altered from one iteration to another, while applying the same heuristic to obtain a different schedule. The makespan for the new schedule is still calculated using the original problem parameters. The NN-based approach provides a framework for applying a problem-space based approach. With the help of a weight vector which is modified after each iteration, weighted problem parameters are used instead of original parameters. Using the same heuristic, new solutions are generated in each iteration. A suitable weight modification strategy guides the search.

In this paper, we will describe how to hybridize GA and NN-based approaches. Our empirical testing demonstrates the effectiveness of the Neurogenetic approach. We find that the hybrid approach works better than either NN or GA approach alone, for the same number of iterations. The rest of the paper is organized as follows. In Section 2 we describe the problem formulation. In Section 3 we discuss some solution techniques such as serial and parallel schedule generation schemes and double justification schemes used for solving the RCPSP. In Section 4 we review the current literature for this problem. Sections 5–7 describe the neural network approach, the genetic algorithms approach and the Neurogenetic approach. Empirical results are shown in Section 8. Finally, Section 9 provides a summary of the paper and discusses future research ideas.

2. The problem formulation

Let N represent the set of activities of a project and let A represent the set of arcs (or precedence relationships). The project is then represented by a directed graph $G=\{N, A\}$. The activities are numbered 0 to $n+1$, where the 0th and the $(n+1)$th activities are dummy activities representing start and end activities, respectively. The processing time of the ith activity is d_i $(1 \le i \le n)$. There are K types of resources. The resources are renewable in nature. The ith activity requires r_{ik} units of resource k in each period of its execution and there are R_k units of the kth resource. The start and finish time of each activity is represented by S_i and F_i, respectively.

The RCPSP with the objective of minimizing the makespan is formulated as Christofides et al. [18]

$$\text{Minimize} \quad S_{n+1}$$

$$\text{Subject to}: \quad S_j - S_i \ge d_i \quad (j,i) \in A \qquad \sum_{i \in P(t)} r_{ik} \le R_k,$$

$$t = 1, \ldots, T, \quad k = 1, \ldots, K \quad S_i \ge 0, \quad i \in N$$

where $P(t)$ represents the set of activities in process at time t and T is an upper bound on the project's makespan.

3. Some solution methods

Since our paper focuses on heuristics and metaheuristics, we will describe some solution methods that pertain to the heuristic approach. We will not focus on exact solution approaches like complete enumeration, branch and bound and dynamic programming. In a heuristic approach, a schedule is generated activity by activity. We will briefly describe the serial and parallel schedule generation schemes and describe how these schemes are used in conjunction with a priority heuristic. We also describe forward and backward scheduling and backward–forward improvement (BFI) or double justification scheme.

3.1. Serial and parallel schedule generation scheme

The heuristic approach for generating a solution to the RCPSP is based on building a schedule from an activity list. An activity list is a precedence feasible list of all activities of the given project. Given an activity list L of all activities, a schedule can be built using either a serial schedule generation scheme (S-SGS) or parallel schedule generation scheme (P-SGS). In S-SGS, activities in L are scheduled in the order in which they appear in L; they are scheduled at the earliest clock time at which the required resources become available.

In P-SGS, a clock is maintained. At each point in time, activities that are precedence and resource feasible are scheduled. If more than one activity can be assigned at a certain clock time, priority is given to activity based on L. If no activities can be assigned at that point in time, the clock is forwarded to the finish time of the shortest in-process activity.

In a priority heuristic approach, L is determined using some heuristic parameter such as latest finish time (LFT) or total remaining work (RWK) etc.

3.2. Forward and backward scheduling

An RCPSP graph $G=\{N, A\}$ can be viewed either as a forward problem or a backward problem. In the backward problem, activity $(n+1)$ is regarded as activity 0 and the entire graph viewed in reverse. Using the same SGS and the same heuristic such as LFT, RWK to generate the activity list L, the backward approach often gives a different solution than the forward approach. The better of the two solutions can be considered as our solution. This method was first proposed by Li and Willis [3]. For any heuristic or metaheuristic technique, the problem can be solved twice—once with the forward approach and once with the backward approach and the better of the two solutions considered.

3.3. Backward–forward improvement

Once a schedule has been generated using some SGS and some L, it is sometimes possible to reduce the empty gaps in the Gantt chart by scanning the Gantt chart in the reverse direction and shifting the activities to the right. This shifting packs the Gantt chart more densely thus reducing the makespan. Having thus obtained a new packed schedule, a forward scan shifts the activities again to the left, further packing the Gantt chart. This approach of reducing the makespan is called backward–forward improvement (BFI) or double justification. It was first introduced by Valls et al. [4]. The counter is updated by adding two schedules each time double justification is used. This approach can be used to refine any solution obtained through any heuristic or metaheuristic.

4. Literature review

The RCPSP literature dates back to the 1960s—Carruthers and Battersby [5]. Several good review papers appeared in the

1990s—Icmeli et al. [6], Ozdamar and Ulusoy [7], Herroelen et al. [8], Brucker et al. [9]. More recent reviews can be found in Hartmann and Kolisch [10], Kolisch and Padman [11] and Kolisch and Hartmann [12]. Due to the NP-hard nature of the problem, exact approaches work only for relatively smaller problems, although many such approaches have been developed such as zero-one programming [13–15], dynamic programming [5] and implicit enumeration with branch and bound [16–23].

Many heuristic or priority-rule based approaches have been proposed for this problem [24–29]. In multi-priority rule methods a different priority rule is used at each iteration [27,30,31]. Sampling methods are used by Cooper [25], Alvares-Valdes and Tamarit [26], Drexl [32], Kolisch [29], Kolisch [33], Kolisch and Drexl [34], Schirmer and Riesenberg [35], Schirmer [36].

Genetic algorithms have been applied in Leon and Ramamoorthy [37], Lee and Kim [38], Hartmann [39], Hartmann [40], Alcaraz and Maroto [41], Coelho and Tavares [42], Hindi et al [43], Toklu [44], Valls et al. [45]. Boctor [46], Cho and Kim [47] and Bouleimen and Lecocq [48] have applied simulated annealing. Tabu search based metaheuristics are proposed by Pinson et al. [49], Baar et al. [50], Nonobe and Ibaraki [51] and Thomas and Salhi [31]. Merkle et al. [52] proposed an ant-colony approach to the RCPSP. Colak et al. [2] proposed a neural-network based technique.

5. Neural-network based approach

The neural-network based approach was first proposed by Agarwal et al. [53]. Colak et al. [2] applied that approach for the RCPSP. In this approach a chosen priority rule (such as LFT or RWK) and a chosen SGS (i.e. serial or parallel) is applied multiple times. The best solution, after a certain number of trials, is saved as the final solution. In each iteration, the activity list L is different. How is the new activity list generated? Using a weight vector $W=(w_0, w_1,\ldots, w_{n+1})$. Suppose Q represents the vector of parameter used in the chosen priority rule. For example, if the chosen priority rule is "latest finish time", then let q_i represent the LFT for activity i and $Q=(q_0, q_1,\ldots,q_{n+1})$ represents the vector of latest finish times of all activities. If the chosen heuristic is say remaining work, then q_i in Q represents the remaining work for activity i. Let Q_w represent a vector of weighted parameters $(w_0*q_0,\ldots, w_1*q_1,\ldots, w_{n+1}*q_{n+1})$. If we assume a unit vector W, then $Q_w=Q$. For the first iteration we obtain L using Q. For subsequent iterations, we use Q_w to obtain a different L. After each iteration W is updated using a weight update strategy to give a new Q_w, which in turn generates a new L, which produces a new solution.

This NN-based approach is basically a local search approach because the perturbed vector Q_w produces a perturbed activity list in the local neighborhood of the original activity list. The approach is called NN-based because of its similarity with the traditional neural networks in which a weight vector is used as the perturbation mechanism. If a good priority rule and a good SGS are used to produce the initial solution, the local search around this original solution produces very competitive results as shown in Colak et al. [2].

6. Genetic algorithms

Genetic Algorithms is a well established approach. For details, see Goldberg [54]. When applying genetic algorithms to the RCPSP, the activity list L is treated as a chromosome of the population. Each activity's ordered position number acts as a gene of the chromosome. An initial population of chromosomes is generated either randomly or by using some priority rules. New chromosomes, for subsequent populations are generated using crossover and mutation mechanisms. In this study two point crossover and random mutation are used. Each new L can be used to decode a new schedule using either serial or parallel SGS. Double justification and forward and backward scheduling can be applied to further refine the solution quality for the same L.

Since a crossover between two different chromosomes can result in a child chromosome which is very different from either parent, the GA approach results in a more global search. This is in contrast with the NN-based approach in which the search is limited to a local neighborhood.

7. The Neurogenetic approach

In the Neurogenetic approach, we interleave GA search iterations with NN search iterations. The idea is that after a few GA iterations, we can develop a set of "good" activity lists (chromosomes), i.e. activity lists that produce good solutions, and that are distributed globally in the solution-space, i.e. belong to different search neighborhoods. If we feed these good activity lists as initial activity lists for NN search to perform some intensive local search around each of these good activity lists, we can get improved solutions not possible by GA alone or NN alone. To further improve the solution quality, the better solutions obtained using NNs can be introduced to the GA population thus improving the genetic material for future GA iterations. The interleaving steps of the Neurogenetic approach are outlined in Fig. 1.

We now describe the algorithm of Fig. 1 in some detail. Here g represents the number of GA solutions that we want to find at each interleaving. For finding g, we are multiplying p (the proportion for GA iterations) by u (the total number of iterations that we need to find) and then we are dividing the result by t which is the number of interleaving that we require. For example say that we want to find 1000 total solutions and 80% of which we want to devote to GA. Also say the number of interleaving is 4. So, here u is 1000, p is 0.8 and t is 4. We can find g using the formula: $g=(1000*0.8)/4=200$ which is number of solutions that we will find at each interleaving. The next formula $a=(1-p)*u/(t*m)$ is similar to the first one and used for finding the number of neural-network iterations at each interleaving. Assume that m is 5. In this example, a will be $(1–0.8)*1000/(4*5)$. So a, which represents the number of solutions found by NN per interleaving, per GA solution is 200/20=10.

7.1. The challenge: switching form GA to NN

Once m good solutions are selected from the GA stage (Step 4 of Fig. 1), we have m activity lists corresponding to these m solutions. Let us call these activity lists $L_{GA,i}$, where i goes from 1 to m. Each of these activity lists is now fed to the neural network approach (Step 5), which will try to find an improved solution in the local neighborhood of that activity list. For the NN approach to perform local search around $L_{GA,i}$, we need to generate a vector W such that $W*Q$ generates $L_{GA,i}$. Once such a W has been determined, then NN-search iterations can perform a local search around $L_{GA,i}$. If we apply the NN-based approach from scratch we do not have an issue, because we always start with a unit vector W and W_{n+1} is derived from W_n, where n is the iteration number. To start from an arbitrary L as the initial solution, we need to determine a W that corresponds to L. In the next subsection, we describe an algorithm to obtain the W vector.

Step 1: Decide on the number of iterations to search, say u.
Decide on the number of interleavings to use, say t.
Decide the proportion for GA/NN iterations, say p and $(1-p)$
Decide on the number of GA solutions to feed to NN at each interleaving, say m
Decide population size, say s
Determine number of GA iterations per interleaving $g=(p*u)/t$
Determine number of NN iterations per GA solution per interleaving, $a = (1-p)*u/(t*m)$
Step 2: Generate Initial Population for GA using the NN approach
Step 3: Run GA search for g number of iterations
Step 4: Select m solutions from the GA population
Step 5: Determine NN weights for each of m selected solutions
Step 6: Run NN search for a iterations for each of m solutions
Step 7: If better solutions found by AugNN than those in the GA population, translate solution to GA chromosome and include in population, replacing the worst solution in the population.
Step 8: If u solutions are not found then go to step 3, else stop.

Fig. 1. Steps of Neurogenetic approach.

Loop until for each gene, the cardinal position of gene on source is the same as the target cardinal position
If a gene is out of place, i.e. its position is different from the target position
Let w_a and w_b represent the weights corresponding to the out of place gene and the target position gene
If ($w_a * q_a > w_b * q_b$ and $position_a > position_b$) and heuristic is based on non-increasing order of q then
Set $w_a = 0.1 + w_b * (q_b/q_a)$
Elseif ($w_a * q_a > w_b * q_b$ and $position_a > position_b$) and heuristic is based on non-decreasing order of q then
Set $w_a = w_b * (q_b/q_a) - 0.1$
End If
End If
End Loop

Fig. 2. Algorithm to generate W.

7.1.1. Algorithm to generate W:

Suppose one of the good activity lists generated through GA iterations is L_t (Target L). The problem is to determine a W such that $W*Q$ produces L_t. We start with L_s (Source L) which is obtained by using a unit vector W and Q. The weight generation algorithm given in Fig. 2 is applied to modify L_s until it equals L_t.

Let us look at a numerical example. Suppose there are five activities and the vector Q based on the minimum latest finish time heuristic is (4, 7, 9, 14, 15). Assume a vector of weights W=(1, 1, 1, 1, 1). So, the ranking based on $W*Q$=(4, 7, 9, 14, 15) is (1, 2, 3, 4, 5). Assume that GA produces a string of (1, 4, 3, 2, 5). If we look at the source and target, we notice that the gene at position 2 (activity 2) is out of place. Here, instead of activity 2, we need activity 4. So we set $w_4=w_2*(q_2/q_4)-0.1$, or $w_4=1*(7/14)-0.1=0.4$. Now the new weight vector becomes W=(1, 1, 1, 0.4, 1) and the new $W*Q$=(4, 7, 9, 5.6, 15). The new ordering based on $W*Q$ is (1, 4, 2, 3, 5). If we look at source and target again, we see that position 3 (activity 2) of source is out of place. Therefore, we set $w_3=w_2*(q_2/q_3)-0.1$, or $w_3=1*(7/9)-0.1=0.67$. The new W is (1, 1, 0.67, 0.4, 1) and the new $W*Q$ is (4, 7, 6.03, 5.6, 15). The new ordering based on $W*Q$ is (1, 4, 3, 2, 5). At this point we check the source and target strings again. The orderings are equal to each other and we stop.

An alternative approach for finding the weight vector is to let the weight vector be the vector of elements ta_i/q_i where q_i is the ith element of Q vector and ta_i is the target activity at the ith element or position. For the example above, W=(1/4, 4/7, 3/9, 2/14, 5/15) because the ta vector is (1, 4, 3, 2, 5) and the Q vector is (4, 7, 9, 14, 15). The $W*Q$ vector becomes (1, 4, 3, 2, 5) which gives the ranking of (1, 4, 3, 2, 5).

7.2. Switching form NN to GA

Switching from NN encoding to GA encoding is quite straightforward. Here, we first find the ordering of activities using weighted parameters ($W*Q$). This ordering represents the order of activities in the GA chromosome. Suppose that the weight vector W is (0.3, 0.7, 0.9, 0.75, 0.6) and Q is (9, 7, 4, 6, 11). Here $W*Q$ becomes (2.7, 4.9, 3.6, 4.5, 6.6) which gives a ranking of (1, 3, 4, 2, 5) which is also the GA encoding.

8. Computational experiments and results

We implemented three approaches, the NN approach, the GA approach and the Neurogenetic approach in Visual Basic 6.0 and executed the experiments on a Pentium IV, 2.8 GB personal computer. Well-known benchmark problem instance sets from PSPLIB (http://www.bwl.uni-kiel.de/Prod/psplib/index.html) were used to evaluate the algorithm. The sets J30, J60 and J90 consist of 480 problem instances with four resource types and 30, 60 and 90 activities, respectively. The set J120 consists of 600 problem instances with four resource type and 120 activities.

For NN-based approach, we use the priority rule LFT. The stopping criterion is to stop if the solution is equal to the lower bound which is the critical path calculated without resource constraints or if a predetermined number of maximum schedules are reached—in our case either 1000 or 5000. The learning rate is set to 0.05 and the weights are initialized at 1 and modified after each iteration. For the GA part, we use two point crossover and mutation probability is set to 0.5. Population size is used as 30. As the neurogenetic parameters, the number of interleavings is set to

Table 1
Average percent deviations for NN, GA and Neurogenetic approaches.

Approach	Dataset	Number of schedules evaluated	
		1000	5000
NN approach alone	J30	0.25	0.11
GA	J30	0.19	0.15
Neurogenetic	J30	0.13	0.10
NN approach alone	J60	11.72	11.39
GA	J60	11.66	11.52
Neurogenetic	J60	11.51	11.29
NN approach alone	J90	11.21	11.10
GA	J90	11.31	11.11
Neurogenetic	J90	11.17	11.06
NN approach alone	J120	34.94	34.57
GA	J120	35.11	34.95
Neurogenetic	J120	34.65	34.15

Table 2
Average deviations from the optimal solution for J30.

Algorithm	SGS	Reference	Number of schedules	
			1000	5000
Scatter search, path relinking	Both	[55]	0.05	0.02
Filter and fan	Serial	[56]	0.09	0.00
Hybr scatter search	Both	[57]	0.10	0.03
GA, TS, path relinking	Both	[58]	0.10	0.04
Decomposition based GA	Both	[59]	0.12	0.04
Neurogenetic (FBI)	*Both*	*this paper*	*0.13*	*0.10*
Sampling—LFT–FBI	Both	[60]	0.23	0.14
GA—forw.–backw.	Both	[61]	0.25	0.06
HNA—FBI	Both	[2]	0.25	0.11
Sampling—LFT–FBI	Both	[62]	0.25	0.15
GA—hybrid, FBI	Serial	[45]	0.27	0.06
Scatter search—FBI	Serial	[63]	0. 27	0.11
GA—forw.–backw.	Serial	[41]	0.33	0.12
GA—FBI	Serial	[4]	0.34	0.20

Table 3
Average deviations from the critical path based lower bound for J60.

Algorithm	SGS	Reference	Number of schedules	
			1000	5000
PSO		[64]	9.52	9.01
Filter and fan	Serial	[56]	10.66	10.56
Scatter search, path relinking	Both	[55]	11.12	10.74
Decomposition Based GA	Both	[59]	11.31	10.95
Neurogenetic (FBI)	*Both*	*this paper*	*11.51*	*11.29*
GA—hybrid, FBI	Serial	[45]	11.56	11.10
Hybr scatter search	Both	[57]	11.59	11.07
HNA—FBI	Both	[2]	11.72	11.39
Scatter search—FBI	Serial	[63]	11.73	11.10
GA—forw.–backw	Both	[61]	11.89	11.19
Sampling—LFT–FBI	Both	[60]	12.04	11.72
GA—FBI	Serial	[4]	12.21	11.27

5 and the proportion of GA is taken as 90%. Also the number of GA solutions to feed NN is used as four. We attempted both the weight generation schemes suggested in Section 7.1, but did not find any significant difference between the two and report the result of the first approach.

Table 1 presents the results of NN, GA and NG approaches for 1000 and 5000 solutions for each of the four datasets. For each dataset, NG performed better than NN or GA alone. Tables 2–5 display the results obtained by our algorithm and other tested

Table 4
Average deviations from the critical path based lower bound for J90.

Algorithm	SGS	Reference	Number of schedules	
			1000	5000
GA—hybrid, FBI	Both	[45]	NA	10.46
Filter and fan	Serial	[56]	10.52	10.11
Decomposition based GA	Both	[59]	10.80	10.35
Neurogenetic (FBI)	*Both*	*this paper*	*11.51*	*11.29*

Table 5
Average deviations from the critical path based lower bound for J120.

Algorithm	SGS	Reference	Number of schedules	
			1000	5000
Filter and fan	Serial	[56]	32.96	31.42
Decomposition based GA	Both	[59]	33.55	32.18
GA—hybrid, FBI	Serial	[45]	34.07	32.54
Scatter search, path relinking	Both	[55]	34.49	32.61
Neurogenetic (FBI)	*Both*	*this paper*	*34.65*	*34.15*
HNA—FBI	Both	[2]	34.94	34.57
Scatter Search–FBI	Serial	[63]	35.22	33.10
GA—FBI	Serial	[4]	35.39	33.24
Sampling—LFT–FBI	Both	[60]	35.98	35.30
Sampling—LFT–FBI	Both	[62]	36.32	35.62
GA—forw.–backw.	Both	[61]	36.53	33.91

heuristics for 1000 and 5000 schedules, respectively. In these tables we present the type of heuristics, the type of schedule generation scheme used, the authors of each heuristic and the average deviation from the critical path based lower bound (from the optimal solution for J30 instances) for 1000 and 5000 schedules, respectively. In each table, the heuristics are sorted according to descending performance with respect to 1000 schedules.

Table 2 presents the percentage deviations from the optimal makespan for the instance set J30 in which all problem instances have been solved to optimality by Demeulemeester and Herroelen's [20] branch and bound procedure. Our algorithm solved 456 out of 480 problems to optimality and the average deviation from the optimal solution is 0.13% and 0.10% for 1000 and 5000 schedules, respectively.

For J60, J90 and J120 problem sets, the optimal solutions are not known, so the results are reported in terms of average percentage deviation from the critical-path based lower bound. Table 3 summarizes the results for J60 test instances. 295 out of 480 instances are solved to critical path based lower bound. The average deviation from the critical path based lower bound is 11.51 and 11.29 percent for 1000 and 5000 schedules, respectively.

Table 4 summarizes the results for J90 test instances. 331 out of 480 instances are solved to critical path based lower bound. The average deviation from the critical path based lower bound is 11.17 and 11.06 percent for 1000 and 5000 schedules, respectively.

Table 5 summarizes the results for J120 set. One hundred and fifty seven out of 600 problems matched the critical path based lower bound. The average deviations are 34.65% and 34.15%, respectively, for 1000 and 5000 schedules. These results are very competitive with other techniques used in the literature.

9. Conclusions

In this paper we addressed a well-known scheduling problem, the resource constrained project scheduling problem (RCPSP).

Given the NP-hard nature of the problem, heuristics and metaheuristics are needed to solve larger instances of this problem. We proposed, developed and tested a new hybrid metaheuristic approach called the Neurogenetic approach, which is a hybrid of a neural network approach and the genetic algorithms approach. In this hybrid approach, we interleave the NN and the GA approaches so each technique feeds its best solution to the other technique. GAs are known to perform well as a global search technique, whereas the NN approach is good at intensive local search. Because of the complementary advantages of these approaches, the two approaches complement each other well when interleaved. Empirical testing demonstrated that the NG approach gave better results than either NN or GA approach alone for the same number of schedules generated. We also implemented several solution enhancement techniques such as double justification, better of forward and backward schedules and better of serial and parallel schedule. The results from the NG approach were quite competitive with other techniques used in the literature.

References

[1] Blazewicz J, Lenstra JK, Rinnooy Kan AHG. Scheduling projects to resource constraints: classification and complexity. Discrete Applied Mathematics 1983;5:11–24.

[2] Colak S, Agarwal A, Erenguc SS. Resource-constrained project scheduling problem: a hybrid neural approach. In: Weglarz J, Jozefowska J, editors. Perspectives in modern project scheduling. 2006. p. 297–318.

[3] Li K, Willis R. An iterative scheduling technique for resource-constrained project scheduling. European Journal of Operational Research 1992;56: 370–9.

[4] Valls V, Ballestin F, Quintanilla MS. Justification and RCPSP: a technique that pays. European Journal of Operational Research 2005;165(2):375–86.

[5] Carruthers JA, Battersby A. Advances in critical path methods. Operational Research Quarterly 1966;17:359–80.

[6] Icmeli O, Erenguc SS, Zappe CJ. Project scheduling problems: a survey. International Journal of Operations & Production Management 1993;13(11): 80–91.

[7] Ozdamar L, Ulusoy G. A survey on the resource-constrained project scheduling problem. IIE Transactions 1995;27:574–86.

[8] Herroelen W, Demeulemeester E, De Reyck B. Resource-constrained project scheduling: a survey of recent developments. Computers & Operations Research 1998;25(4):279–302.

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

[10] Hartmann S, Kolisch R. Experimental evaluation of stateof-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 2000;127:394–407.

[11] Kolisch R, Padman R. An integrated survey of deterministic project scheduling. OMEGA 2001;29:249–72.

[12] Kolisch R, Hartmann S. Experimental investigation of heuristics for resource-constrained project scheduling: an update. European Journal of Operational Research 2006;174(1):23–37.

[13] Pritsker AAB, Watters LJ, Wolfe PM. Multiproject scheduling with limited resources: a zero-one programming approach. Management Science 1969;16:93–107.

[14] Patterson JH, Huber WD. A horizon-varying, zero-one approach to project scheduling. Management Science 1974;20:990–8.

[15] Patterson JH, Roth GW. Scheduling a project under multiple resource constraints: a zero-one programming approach. AIIE Transactions 1976;8: 449–55.

[16] Davis EW, Heidorn GE. An algorithm for optimal project scheduling under multiple resource constraints. Management Science 1971;17:803–16.

[17] Talbot FB, Patterson JH. An efficient integer programming algorithm with network cuts for solving resource constrained scheduling problems. Management Science 1978;24(11):1163–74.

[18] Christofides N, Alvarez-Valdes R, Tamarit JM. Project scheduling with resource constraints: a branch and bound approach. European Journal of Operational Research 1987;29(3):262–73.

[19] Demeulemeester E, Herroelen W. A branch-and-bound procedure for the multiple resource-constrained project scheduling problems. Management Science 1992;38(12):1803–18.

[20] Demeulemeester E, Herroelen W. New benchmark results for the resource-constrained project scheduling problem. Management Science 1997;43(11): 1485–92.

[21] Brucker P, Knust S, Schoo A, Thiele O. A branch & bound algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 1998;107(2):272–88.

[22] Mingozzi A, Maniezzo V, Ricciardelli S, Bianco L. An exact algorithm for project scheduling with resource constraints based on new mathematical formulation. Management Science 1998;44(5):714–29.

[23] Dorndorf U, Pesch E, Phan-Huy T. A branch-and-bound algorithm for the resource-constrained project scheduling problem. Mathematical Methods of Operations Research 2000;52:413–39.

[24] Davis EW, Patterson JH. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. Management Science 1975;21(81): 944–55.

[25] Cooper DF. Heuristics for scheduling resource-constrained projects: an experimental investigation. Management Science 1976;22:1186–94.

[26] Alvares-Valdes R, Tamarit JM. Heuristic algorithms for resource-constrained project scheduling: a review and an empirical analysis. In: Slowinski R, Weglarz J, editors. Advances in project scheduling. Amsterdam: Elsevier; 1989. p. 113–34.

[27] Boctor FF. Some efficient multi-heuristic procedures for resource-constrained project scheduling. European Journal of Operational Research 1990;49:3–13.

[28] Ozdamar L, Ulusoy G. A local constraint based analysis approach to project scheduling under general resource constraints. European Journal of Operational Research 1994;79:287–98.

[29] Kolisch R. Efficient priority rule for the resource-constrained project scheduling problem. Journal of Operations Management 1996;14(3):179–92.

[30] Ulusoy G, Ozdamar L. Heuristic performance and network/resource characteristics in resource-constrained project scheduling. Journal of the Operational Research Society 1989;40:1145–52.

[31] Thomas PR, Salhi S. A tabu search approach for the resource constrained project scheduling problem. Journal of Heuristics 1998;4:123–39.

[32] Drexl A. Scheduling of project networks by job assignment. Management Science 1991;37:1590–602.

[33] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. European Journal of Operational Research 1996;90:320–33.

[34] Kolisch R, Drexl A. Adaptive search for solving hard project scheduling problems. Naval Research Logistics 1996;43:23–40.

[35] Schirmer A, Riesenberg S. Class-based control schemes for parameterized project scheduling heuristics, Manuskripte aus den Instituten für etriebswirtschaftslehre. Germany: Universität Kiel; 1998. p. 471.

[36] Schirmer A. Case-based reasoning and improved adaptive search for project scheduling. Naval Research Logistics 2000;47:201–22.

[37] Leon VJ, Ramamoorthy B. Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. OR Spektrum 1995;17: 173–82.

[38] Lee J-K, Kim Y-D. Search heuristics for resource-constrained project scheduling. Journal of the Operational Research Society 1996;47:678–89.

[39] Hartmann S. A competitive genetic algorithm for the resource-constrained project scheduling. Naval Research Logistics 1998;45:733–50.

[40] Hartmann S. A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics 2002;49:433–48.

[41] Alcaraz J, Maroto C. A robust genetic algorithm for resource allocation in project scheduling. Annals of Operations Research 2001;102:83–109.

[42] Coelho J, Tavares L. Comparative analysis of meta-heuristics for the resource constrained project scheduling problem. Technical report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal; 2003.

[43] Hindi KS, Yang H, Fleszar K. An evolutionary algorithm for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation 2002;6:512–8.

[44] Toklu YC. Application of genetic algorithms to construction scheduling with or without resource constraints. Canadian Journal of Civil Engineering 2002;29:421–9.

[45] Valls V, Ballestin F, Quintanilla MS. A hybrid genetic algorithm for the resource constrained project scheduling problem. European Journal of Operational Research 2008;185:495–508.

[46] Boctor FF. An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problems. International Journal of Production Research 1996;34:2335–51.

[47] Cho JH, Kim YD. A simulated annealing algorithm for resource-constrained project scheduling problems. Journal of the Operational Research Society 1997;48:736–44.

[48] Bouleimen K, Lecocq H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 2003;149:268–81.

[49] Pinson E, Prins C, Rullier F. Using tabu search for solving the resource-constrained project scheduling problem. In: Proceedings of the fourth international workshop on project management and scheduling, Leuven, Belgium; 1994. p. 102–6.

[50] Baar T, Brucker P, Knust S. Tabu search algorithms for resource-constrained project scheduling problems. In: Voss S, Martello S, Osman I, Roucairol C, editors. Metaheuristics: advances and trends in local search paradigms for optimisation. Dordrecht: Kluwer Academic Publishers; 1997. p. 1–18.

[51] Nonobe K, Ibaraki T. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Ribeiro CC, Hansen P, editors. Essays and surveys in metaheuristics. Dordrecht: Kluwer Academic Publishers; 2002. p. 557–88.

[52] Merkle D, Middendorf M, Schmeck H. Ant colony optimization for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation 2002;6:333–46.

[53] Agarwal A, Jacob VS, Pirkul H. Augmented neural networks for task scheduling. European Journal of Operational Research 2003;151(3):481–502.
[54] Goldberg DE. Genetic algorithms in search optimization and machine learning. Reading, MA: Addison Wesley; 1989.
[55] Mahdi Mobini MD, Rabbani M, Amalnik MS, Razmi J, Rahimi-Vahed AR. Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. Soft Computing 2009;13:597–610.
[56] Ranjbar M. Solving the resource constrained project scheduling problem using filter-and-fan approach. Applied Mathematics and Computation 2008;201:313–8.
[57] Ranjbar M, Reyck BD, Kianfar F. A hybrid scatter search for the discreet time/resource trade-off problem in project scheduling. European Journal of Operational Research 2009;193:35–48.
[58] Kochetov Y, Stolyar A. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In: Proceedings of the third international workshop of computer science and information technologies, Russia; 2003.
[59] Debels D, Vanhoucke M. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. Operations Research 2007;55(3):457–69.
[60] Tormos P, Lova A. An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research 2003;41(5):1071–86.
[61] Alcaraz J, Maroto C, Ruiz R. Improving the performance of genetic algorithms for the RCPS problem. In: Proceedings of the ninth international workshop on project management and scheduling, 2004. p. 40–3.
[62] Tormos P, Lova A. A competitive heuristic solution technique for resource constrained project scheduling. Annals of Operations Research 2001;102:65–81.
[63] Debels D, De Reyck B, Leus R, Vanhoucke M. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. European Journal of Operational Research 2006;169(2):638–53.
[64] Tchomte SK, Gourgand M, Quilliot A. Solving resource-constrained project scheduling problem with particle swarm optimization. In: Proceedings of fourth multidisciplinary international scheduling conference. 2007. p. 251–8.