Solving the Resource Constrained Project Scheduling Problem to Minimize the Financial Failure Risk

Zhi-Jie Chen, Chiuh-Cheng Chyu
Department of Industrial Engineering and Management
Yuan-Ze University
Taiwan

***Abstract*—In practice, a project usually involves cash in- and out-flows associated with each activity. This paper aims to minimize the payment failure risk during the project execution for the resource-constrained project scheduling problem (RCPSP). In such models, the money-time value, which is the product of the net cash in-flow and the time length from the completion time of each activity to the project deadline, provides a financial evaluation of project cash availability. The cash availability of a project schedule is defined as the sum of these money-time values associated with all activities, which is mathematically equivalent to the minimization objective of total weighted completion time. This paper presents four memetic algorithms (MAs) which differ in the construction of initial population and restart strategy, and a double variable neighborhood search algorithm for solving the RCPSP problem. An experiment is conducted to evaluate the performance of these algorithms based on the same number of solutions calculated using ProGen generated benchmark instances. The results indicate that the MAs with regret biased sampling rule to generate initial and restart populations outperforms the other algorithms in terms of solution quality.**

Keywords-RCPSP; cash availability; memetic algorithms; variable neighborhood search.

I. INTRODUCTION

Cash flow is critical to the success of executing a project. The term cash flow is used to describe the net difference, at any point in time, between income (revenue) and project expenditures; negative cash flow is outgoing (cash out-flow), while positive cash flow is income (cash in-flow). In practice, cash in-flows often arise from payments due to the completion of specified parts of the project. On the other hand, cash out-flows are caused by the execution of activities, such as resource usage and necessity expenditure. Both cash in- and out- flows may occur at several points in time during execution of an activity. Usually, discount rates are taken into consideration. Some commonly used NPV maximization RCPSP models include progress payments, lump-sum payment at the pre-specified project deadline, payments at activity completion times, etc. [1-4] carried out comparative studies on the above payment models. For an overview of RCPSP with objectives based on NPV, we refer to [5, 6].

This research presents a different financial model, which aims to minimize the risk of payment failure or maximize the cash availability during project execution. The goal is to provide a cautious and less complex model to minimize the payment failure risk during the project execution. To achieve this goal, the money-time value, which is the product of the cash in-flow and the length from the time the cash received to the project makespan, can provide a financial evaluation of project cash availability. The cash availability of a project schedule is defined as the total money-time values associated with all activities. This financial metric does not consider discount rate, and it will provide a conservative estimate of cash in-flows during the project execution, since cash on hand will grow in value over time. In the proposed model, the cash in-flows are assumed to occur at the completion time of each activity, and the cash amounts can be used during the rest of project execution time. Hereafter, we shall refer to this model as the project cash availability maximization problem (PCAMP) for the resource constrained project scheduling problem (RCPSP).

The PCAMP is mathematically equivalent to the RCPSP with the objective of minimizing total weighted completion time (also known as total weighted flow time). This problem is strongly NP-hard since its sub-problem, single machine scheduling with total flow time minimization objective subject to precedence constraints, is strongly NP-hard [7]. Thus, the heuristic approach will be appropriate for solving the PCAMP for RCPSP of large size. As the objective function of the PCAMP for RCPSP is regular, the optimal solution must be an active schedule [8]. The serial SGS procedure will generate the active schedules, where each activity is scheduled as early as possible [9]. A regular objective function is a non-decreasing function of the activity start times or finish times.

To solve the PCAMP for RCPSP, we propose several memetic algorithms (MAs) that differ in initial population and restart strategy. The performance of these MAs will be compared to each other, as well as a double variable neighborhood search (DVNS) [10]. MAs are a population-based meta-heuristic that incorporates evolutionary algorithms with local search [11]. In MAs, the term "memes" refers to the strategies that are employed to improve individuals or solutions. The strategies involve local refinement, perturbation, constructive methods, restart policy, etc. Ong and Keane [12] discussed the importance of selecting local search methods in MAs. An appropriate local search method will significantly improve the efficiency of the solution search. Merz and Freisleben [13] presented an MA framework with a restart strategy, in which mutation operator is used to generate a new

and diverse population when the current population is judged to be convergent.

Variable neighborhood search (VNS) was introduced by Mladenović and Hansen [14], and its principles and applications were further detailed in [15]. The basic idea of VNS is to perform a systematic change of neighborhood and to find a local optimal solution with respect to each neighborhood using a local search algorithm. This algorithm explores increasingly distant neighborhoods of the incumbent solution and jumps from there to a new one when an improvement has been made.

The remainder of this paper is organized as follows: Section 2 defines the problem; Section 3 illustrates the solution methods; Section 4 presents the numerical results; Section 5 concludes this research.

II. Problem Description

The RSPSP can be described as follows. A project consists of a set $N = \{0, 1,\ldots, J+1\}$ of activities (nodes), where activities 0 and $J+1$ are dummies, and respectively represent the start time and the completion time of the project. A set of precedence relationships between activity pairs must be specified for the project. An activity list (AL) is a sequence of activities that follows the precedence constraints. Activity preemption is not allowed. The duration of an activity j is denoted by d_j, and its requirement for renewable resource type k is r_{jk}, $k = 1,\ldots, K$. The availability of a resource type k for each time period is R_k units.

This model aims to minimize the risk of payment failure during project execution, which in turn to maximize the project cash availability. The following are notations and formulation of the PCAMP model.

Notation

- j : Index of activity, $j = 0, 1, 2,\ldots, J + 1$
- t : Index of time, $t = 1,\ldots, D$
- D : Project deadline
- k : Index of renewable resource type, $k = 1,\ldots, K$
- d_j : Duration of activity j
- ω_j : Cash in-flow of executing activity j
- r_{jk} : Per period usage of resource k by activity j
- R_k : Availability level of resource k per period
- DP_j : Direct predecessors of activity j
- f_j : completion time of activity j
- S(t) : Set of activities in progress at time t

Mathematical model

$$\text{Maximize} \quad \sum_{j=1}^{J} \omega_j \cdot (D - f_j) \tag{1}$$

Subject to

$$f_j \geq f_i + d_j \quad \text{for all } i \in DP_j \tag{2}$$

$$\sum_{j \in S(t)} r_{jk} \leq R_k \quad k = 1,\ldots, K;\ t=1,\ldots, D \tag{3}$$

$$f_j \geq 0 \text{ and integers} \quad j = 0, 1,\ldots, J+1 \tag{4}$$

Equation (1) is the model objective that computes the cash availability during the project execution. Constraint set (2) describes the precedence relationships among activities. Constraint set (3) specifies the usage limit per period for each renewable resource type at any time during project execution.

III. Solution Methods

All algorithms presented in this research will employ local search to refine a newly produced solution. Three local search methods are used: (1) left move at random, (2) 2-swap, and (3) forward/backward improvement (FBI; also known as justification procedure [16]). This section first introduces the three local search methods, and then the proposed MAs and DVNS. These algorithms use activity list (AL) as the encoding scheme and forward serial list scheduling (F-SLS) as the decoding scheme. An AL offers the order to schedule the activities in the project. Fig. 1 displays a simple example with a single resource availability of 5 units, where activities 0 and 9 are dummy and they represent the start and finish events of the project, respectively. Fig. 2 shows the encoding and decoding schemes through an AL = {1, 3, 4, 2, 6, 5, 8, 7}.

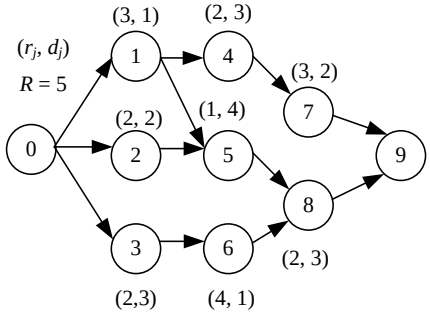

Figure 1. Example of project network

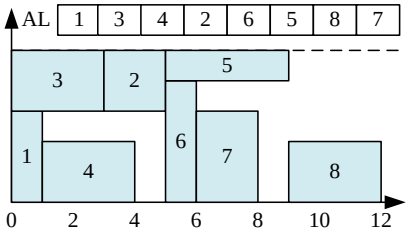

Figure 2. Example of decoding AL

A. Local Search methods

In the left move method, a position is randomly selected from the AL, and the activity on that position is then randomly placed leftward into a position allowed by precedence relations.

In Fig. 2 example, if position 5 is selected, the corresponding activity 6 can be placed on either position 3 or 4, but not position 2 (i.e. activity 3).

In the 2-swap method, an activity is randomly selected from the AL, and then the corresponding left- and right-move limits are determined. Then, within these two limits, a second activity is randomly selected for possible swapping. The procedure performs swapping when these two activities have different start times in the current schedule and the swap does not violate the precedence constraints. In Fig. 2 example, if activity 6 is selected, then it can swap with activities 4, 2, and 5.

The FBI method consists of two steps: in the forward step, the activities are scheduled as late as possible according to the order of their finish times in the current schedule; then in the backward step, the activities are scheduled as early as possible based on the order of their start times in the schedule generated in the forward step. The FBI method has been proved very effective in improving solutions for RCPSP with the objective of minimizing makespan [17]. Fig. 3 shows the forward step (i.e. right justification) and Fig. 4 shows the backward step (left justification) using the schedule in Fig. 2.

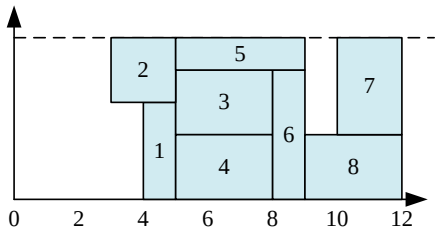

Figure 3. Forward step schedule

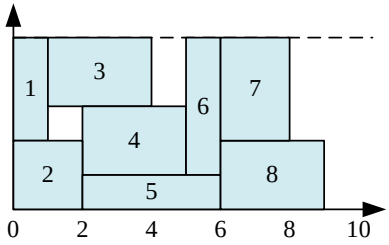

Figure 4. Backward step schedule

B. Memetic Algorithms

Memetic algorithms (MAs) are sometimes referred to as genetic local search algorithms. In this research, we present a multi-start MA framework which employs the FBI to improve each newly produced individual. In each generation, a modified order based recombination operator is applied to produce offspring. The MA will restart with a new population when the current population contains at least 80% identical individuals. This situation is regarded as the population having entered into a convergence state, and continuing evolution will have little chance of making further improvement under the recombination operation.

Our MA considers the following policies to produce individuals of the initial population and restart population: (1) randomly generated individuals followed by FBI for initial population, and a series of 2-swap followed by FBI for restart population; (2) regret biased sampling [18] followed by FBI for both initial and restart populations (also denoted as RBS + RBS); (3) regret biased sampling followed by FBI for initial population, and a series of 2-swap followed by FBI for restart population (also denoted as RBS + 2-swap); (4) randomly generated individuals followed by FBI for initial population and no restart. For any method, the best individual will be retained when generating a new population. We shall refer to the MA with policy k as MA^k for k = 1,…, 4. Fig. 5 describes the pseudo code of the MAs.

When generating a population using the RBS rule, at each step an activity i is selected from current candidate list CL according to the following probability:

$$P(i) = \frac{[\mathrm{Max}_{m\in \mathrm{CL}}(\omega_m \cdot f_m) - \omega_i \cdot f_i + \varepsilon]}{\sum_{j\in \mathrm{CL}}(\mathrm{Max}_{m\in \mathrm{CL}}(\omega_m \cdot f_m) - \omega_j \cdot f_j + \varepsilon]} \quad (5)$$

Initialize Population P (Policy k);
While (termination conditions not met) do
{ for i = 1 to *psize* do
{Select a, $b \in$ **P** by tournament;
Offspring c = recombine (a, b);
c = FBI(c);
}end for;
Elitist selection to produce next population **P**;
If P converges (80% of individuals are identical)
Construct **P** by Policy k;
} end while;
Output the best solution in P.

Figure 5. Framework of memetic algorithms

The recombination is performed by a modified order-based procedure described as follows: Randomly select k positions and determine the set A(k) that contains the elements that correspond to those positions in parent b. Place the elements of A(k) one by one to the k positions according to their order in parent a. Fill out the remaining positions by the activities in parent b at the corresponding positions. Repair the offspring if it violates precedence relations of activities. Fig. 6 illustrates this operation via Fig. 1 network. Suppose k = 3 and the positions are 2, 4, and 7. The corresponding activities in parent b, A(k) = {1, 5, 6}. Since DP_3 = 6, the child {2, 1, 4, 6, 7, 3, 5, 8} violates the precedence order and the violation will be fixed by moving activity 3 to the left position of activity 6.

C. Double variable neighborhood algorithm

Similarly, the proposed VNS uses AL and serial F-SLS for the coding schemes. Hansen and Mladenović [15] mentioned that three problem-specific questions must be considered when designing a VNS: (a) What neighborhood structures should be used and how many of them? (b) What should be their local search? (c) What strategy should be used in changing neighborhoods? The proposed VNS consists of the following features: (1) The initial solution is constructed by a greedy heuristic using the objective function values. (2) Two fundamental generators are used to construct the variable neighborhoods: 2-swap and left move. The distance of the generated neighborhood to the current solution is evaluated by

the number of the operations performed. (3) An enhanced local search procedure named short-term or inner VNS is used. The inner VNS uses the efficient local search method FBI, and aims to seek the best solution in the nearby area.

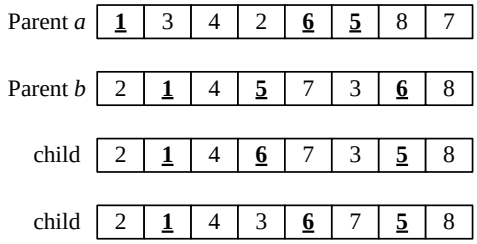

Figure 6. Modified order-based recombination

Fig. 7 describes the framework of the DVNS. The relevant parameters are explained as follows: N_k is the k-th neighborhood, k = 1,…, $Kmax$ and $s < K1 <$ Kmax, where $\{N_1,\ldots,N_s\}$ are used for the inner VNS and the remaining $\{N_{s+1},\ldots, N_{Kmax}\}$ are used for the outer VNS; each $N_k \in \{N_1,\ldots, N_{K1}\}$ is generated by performing k times of two-swap operations, and each $N_k \in \{N_{K1+1},\ldots, N_{Kmax}\}$ by performing $k - K1$ times of left-move operations. In general, a 2-swap operation takes a longer computational time than a left-move operation, but the former will make a bigger change on the current solution structure. Each N_k will generate m neighboring solutions, each of which will produce two additional solutions by FBI. The DVNS will continue to move ahead to next neighborhood regardless of whether or not the current neighborhood search has found a new best solution. In the DVNS, parameters are set to m = 20, s = 5, $K1$ = 20, $Kmax$ = 30. In the performance comparison study, each algorithm will calculate MaxSol solutions. The DVNS will return to N_{s+1} if the number of solutions calculated after N_{Kmax} is smaller than $MaxSol$.

Initialization:
Select N_k, $k = 1,\ldots, N_{Kmax}$; determine s, $K1$, $MaxSol$.
Generate an initial solution x by greedy heuristic;
Set $k = s+1$; current best solution $x^* = x$;
While (# of solutions calculated < $MaxSol$) do
{ $y = N_k(x)$; y = FBI(y); set x = best of $\{x, y\}$;
 for i = 1 to s do
 { for j = 1 to m do
 { $y = N_i(x)$; y = FBI(y); set x = best of $\{x, y\}$;
 Set $x^* = x$ if objective (x) > objective (x^*);
 } end for j;
 } end for i and inner VNS;
 $k = k+1$; if ($k > Kmax$) set $k = s+1$;
} end While;
Output the best solution x^*.

Figure 7. Framework of algorithm DVNS

IV. NUMERICAL RESULTS

This section presents the experimental results for the four MAs and DVNS. The test sets come from J.60 and J.120 of the PSPLIB [19]. Test set J.60 contains 480 instances, while the set J.120 has 600 instances. Each instance set is characterized by three factors: (1) Network complexity (NC) with three levels, 1.5, 1.8, 2.1; (2) Resource factor (RF) with four levels, 0.25, 0.50, 0.75, 1.0; (3) Resource strength (RS). The RS for J.60 has four levels: 0.2, 0.5, 0.7, 1.0, whereas the RS for J.120 has five levels: 0.1, 0.2, 0.3, 0.4, and 0.5. The factor NC is the average number of successors of each job, RF is the average number of resource types used by a job, and RS signifies the strength of resource availability. Each combination of factors has 10 instances. The J.120 test instances are much more difficult than J.60, mainly due to larger problem size and lower resource strength. The test environment is as follows: CPU: Intel Core i5 3G, RAM: 2G DDRIII, HD: 500G 7200 rpm, OS: Win7, Language: Visual C#.Net.

A. Generation of activity profit

The profit of each activity is assumed to be a function of the resources consumed. The following describes the method to generate the profit for activity j using r_{jk} units of resource type k, k = 1,…, K. For each resource type k, we generate a value b_k at random from (1,000, 1,300), and set the unit cost for resource type k to $c_k = b_k/R_k$. The larger the resource limit, the cheaper the unit cost. The cost of activity j is defined as $C_j = \sum_{k=1}^{K} c_k \cdot r_{jk}$, and the profit $\omega_j = \lambda_j \cdot C_j$, where the multiplier λ_j is randomly generated from the interval of [0.3, 0.5]. The project deadline of each instance is set to 1.5 multiples of the minimum makespan, which can be found in PSPLIB (URL: http://129.187.106.231/psplib/). The critical path method (CPM) based upper bound is used to compare the results of these algorithms.

B. Computation results

An experiment was conducted to compare the performance of the proposed algorithms based on 1000 and 5000 schedules for J.60 and J.120 test sets. Each test set has 10 repetitions on each instance. The performance of an algorithm on a test set is evaluated based on the average deviation from the CPM upper bound. The deviation (in percentage) of an algorithm A for solving an instance is defined as follows:

{(CPM upper bound – best objective value found by A) / CPM upper bound} · 100%.

The 1000-schedule experiment evaluates short running time performance of an algorithm, whereas the 5000-schedule evaluates moderate running time performance of an algorithm. A small average deviation indicates that the algorithm is able to achieve a high quality performance.

TABLE I displays the performance of the algorithms for J.60 based on 1000 and 5000 schedules. The "min" column shows the best performance among 10 runs, the "avg" column presents the average performance, and the "max" column gives the worst performance. The results indicate that MA^2 (i.e. RBS + RBS) and MA^3 (RBS + 2-swap) produce the best results in short and moderate running times. The DVNS, which uses enhanced local search on each new neighboring solution, outperforms the MA with randomly generated initial population and 2-swap restart strategy (MA^1), as well as the MA without restart strategy (MA^4). It is also observed that DVNS improves its performance the most when increasing schedules from 1000 to 5000.. MA^2 and MA^3 are next in this respect, and MA^1 and MA^4 are least effective in calculating additional solutions. Similar results can be observed for J.120 in TABLE II. The deviations grow significantly for all algorithms since the problem size of J.120 is much larger.

TABLE I. ALGORITHM PERFORMANCE ON J.60

	1000 schedules			
	min	avg	max	CPU
DVNS	5.63%	5.98%	6.43%	0.11s
MA^1	5.85%	6.11%	6.39%	0.12s
MA^2	5.54%	5.76%	6.00%	0.18s
MA^3	5.52%	5.75%	6.01%	0.15s
MA^4	5.89%	6.13%	6.37%	0.21s
	5000 schedules			
	min	avg	max	CPU
DVNS	5.46%	5.69%	5.96%	0.55s
MA^1	5.75%	6.01%	6.29%	0.36s
MA^2	5.39%	5.59%	5.84%	0.86s
MA^3	5.37%	5.58%	5.85%	0.76s
MA^4	5.78%	6.03%	6.31%	0.47s

TABLE II. ALGORITHM PERFORMANCE ON J.120

	1000 schedules			
	min	avg	max	CPU
DVNS	14.68%	15.29%	15.94%	0.43s
MA^1	14.97%	15.41%	15.82%	0.50s
MA^2	14.38%	14.77%	15.16%	0.46s
MA^3	13.76%	14.76%	15.15%	0.45s
MA^4	15.08%	15.50%	15.91%	0.49s
	5000 schedules			
	min	avg	max	CPU
DVNS	14.19%	14.65%	15.17%	2.02s
MA^1	14.51%	14.95%	15.40%	2.43s
MA^2	13.76%	14.16%	14.59%	2.11s
MA^3	13.73%	14.14%	14.60%	2.01s
MA^4	14.48%	15.04%	15.49%	2.07s

TABLE III shows the performance of MA^3 for J.120 based on the factor RF. Each instance contains four resource types. Thus an RF of 0.25 implies that each activity in the project consumes one of the four resource types, whereas an RF of 1.0 indicates that each activity consumes all four resource types. The problem becomes more difficult when RF increases, and the computation time increases as well. TABLE IV shows the results of MA^3 based on RS for J.120. The smaller the RS value, the more scarce the resource. When RS decreases, the problem becomes harder and the project completion time will be longer. TABLE V presents the results of MA^3 on four hardest problems. The most difficult problem is (RF, RS) = (1.0, 0.1). In addition, the RS influences the problem difficulty more significantly than the RF. The average deviation for (1.0, 0.1) is 30.48% and for (0.75, 0.1) is 29.46%, but for (1.0, 0.2) is 24.18% and for (0.75, 0.2) is 22.58%.

TABLE III. RESULTS OF MA^3 BASED ON RF FOR J.120

RF	min	avg	max
0.25	6.40%	6.65%	6.95%
0.5	13.35%	13.82%	14.34%
0.75	16.74%	17.24%	17.77%
1	18.42%	18.85%	19.32%

TABLE IV. RESULTS OF MA^3 BASED ON RS FOR J.120

RS	min	avg	max
0.1	24.12%	24.76%	25.45%
0.2	18.39%	18.91%	19.46%
0.3	12.62%	13.02%	13.49%
0.4	8.60%	8.91%	9.24%
0.5	4.91%	5.11%	5.34%

TABLE V. RESULTS OF MA^3 FOR HARD INSTANCES OF J.120

(RF,RS)	min	avg	max
(0.75,0.1)	28.74%	29.46%	30.22%
(0.75,0.2)	21.99%	22.58%	23.25%
(1,0.1)	29.87%	30.48%	31.10%
(1,0.2)	23.69%	24.18%	24.72%

V. CONCLUSIONS

This research presents a new model to minimize the financial failure risk during the project execution. The model assumes that cash out-flows may occur at any times during activity execution, and all returns (in-flows) will be received upon activity completion times. In practice, the project owner would like to schedule activities to best prepare for cash out-flows until project completion time from a simple and conservative financial viewpoint.

The proposed project cash availability maximization model can be shown to be mathematically equivalent to the RCPSP with the objective of minimizing total weighted flow time, which is strongly NP-hard. Thus, the meta-heuristic approach is appropriate for solving this problem. Two solution approaches are presented: memetic algorithms (MAs) and a double variable neighborhood search termed DVNS. Our experimental results indicate that MA with good constructive heuristic for initial population and with restart strategy will produce high quality results.

ACKNOWLEDGMENT

This work was supported by the National Science Council in Taiwan under Grant NSC 99-2221-E-155-029.

REFERENCES

[1] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "Maximizing the net present value of a project with linear time-dependent cash flows," Int. J. Prod. Res., vol. 39, pp. 3159-3181, 2001.

[2] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "Progress payments in project scheduling problems," Eur. J. Oper. Res., vol. 148, pp. 604-620, 2003.

[3] G. Ulusoy, F. Sivrikaya-Serifoglu, and S. Sahin, "Four payment models for the mult-mode resource constrained project scheduling problem with discounted cash flows," Ann. Oper. Res., vol. 102 , pp. 237-261, 2001.

[4] M. Mika, G. Waligóra, and J. Węglarz, "Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discount cash flows and different payment models," Eur. J. Oper. Res., vol. 164, pp. 639-668, 2005.

[5] E. Demeulemmester, and W. S. Herroelen, Project Scheduling: A research handbook. Kluwer's International Series, 2002.

[6] S. Hartmann, and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," Eur. J. Oper. Res., vol. 207, pp. 1-14, 2010.

[7] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, 3rd edition, Springer, 2008, pp. 605.

[8] K. Neumann, C. Schwindt, and J Zimmermann, Project scheduling with time windows and scarce resources, Springer, 2002.

[9] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited – Theory and computation", Eur. J. Oper. Res., vol. 90, pp. 320-333, 1996.

[10] C. C. Chyu, and Z. J. Chen, "Scheduling jobs under constant period-by-period resource availability to maximize project profit at a due date", Int. J. Adv. Manuf. Technol., vol. 42, pp. 569-580, 2009.

[11] P. Moscato, On evolutions, search, optimization, genetic algorithms and martial arts: toward memetic algorithms, Technical Report, Caltech Concurrent Computer Program Report, California Institute Technology, Pasadena, CA, 1989.

[12] Y. S. Ong, and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," IEEE Trans. Evol. Comput., vol. 8, pp. 99-110, 2004.

[13] P. Merz, and B. Freisleben, "A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem," In 1999 Congress on Evolutionary Computation (CEC'99), IEEE Press, Piscataway, NJ, 1999, pp. 2063–2070.

[14] N. Mladenović, and P. Hansen, "Variable neighborhood search," Comput. Oper. Res., vol. 24, pp. 1097–1100, 1997.

[15] P. Hansen, and N. Mladenović, "Variable neighborhood search: principles and applications," Eur. J. Oper. Res., vol. 130, pp. 449–467, 2001.

[16] V. Valls, F. Ballestin, and S. Quintanilla, "Justification and RCPSP: A technique that pays," Eur. J. Oper. Res., vol. 165, pp. 375-386, 2005.

[17] R. Kolisch, S. Hartmann, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling: An update," Eur. J. Oper. Res., vol. 174, pp. 23-37, 2006.

[18] R. Kolisch, "Project scheduling under resource constraints – efficient heuristics for several problem classes," Physica, Heidelberg, 1995.

[19] R. Kolisch, and A. Sprecher, "PSPLIB – a project scheduling problem library," Eur. J. Oper. Res., vol. 96, pp. 205-216, 1996.

AUTHORS PROFILE

Zhi-Jie Chen is currently a PhD student of the Industrial Engineering and Management department at Yuan-Ze University, Chung-Li, Taiwan. His research interests include applied operations research, project scheduling, and meta-heuristics for combinatorial optimization problems.

Chiuh-Cheng Chyu is currently an associate professor of the department of Industrial Engineering and Management at Yuan-Ze University. His current research interests are in the areas of applied operations research, multiple criteria decision-making, scheduling, and meta-heuristics for combinatorial optimization problems.