IEMS Vol. 9, No. 2, pp. 70-79, June 2010.

A Dual-Population Memetic Algorithm for Minimizing Total Cost of Multi-Mode Resource-Constrained Project Scheduling

Zhi-Jie Chen
Department of Industrial Engineering and Management
Yuan-Ze University, Jongli 320, TAIWAN
E-mail: s948907@mail.yzu.edu.tw

Chiuh-Cheng Chyu†
Department of Industrial Engineering and Management
Yuan-Ze University, Jongli 320, TAIWAN
E-mail: iehshsu@saturn.yzu.edu.tw

Received, February 18, 2010; Revised, April 19, 2010; Accepted, May 17, 2010

Abstract. Makespan and cost minimization are two important factors in project investment. This paper considers a multi-mode resource-constrained project scheduling problem with the objective of minimizing costs, subject to a deadline constraint. A number of studies have focused on minimizing makespan or resource availability cost with a specified deadline. This problem assumes a fixed cost for the availability of each renewable resource per period, and the project cost to be minimized is the sum of the variable cost associated with the execution mode of each activity. The presented memetic algorithm (MA) consists of three features: (1) a truncated branch and bound heuristic that serves as effective preprocessing in forming the initial population; (2) a strategy that maintains two populations, which respectively store deadline-feasible and infeasible solutions, enabling the MA to explore quality solutions in a broader resource-feasible space; (3) a repair-and-improvement local search scheme that refines each offspring and updates the two populations. The MA is tested via ProGen generated instances with problem sizes of 18, 20, and 30. The experimental results indicate that the MA performs exceptionally well in both effectiveness and efficiency using the optimal solutions or the current best solutions for the comparison standard.

Keywords: Memetic Algorithms, Project Scheduling, Multiple Modes, Truncated Branch and Bound.

1. INTRODUCTION

In the past three decades, the resource constrained project scheduling problem (RCPSP) has attracted the attention of the research community and has been widely used in software development, architectural engineering, production manufacturing, and project management. Its application includes shortening product cycle times, giving a greater variety of products, and reducing total project cost. In practice, a project involves a number of activities with precedence relations. The multi-mode resource constrained project scheduling problem (MRCPSP) refers to the situation where each activity is executed in one of several alternative modes. The set of execution modes for each activity is usually created by different resource/resource and resource/ duration trade-offs. In construction and software development projects (see e.g. De Reyck, 1998), it is frequently the case that only one renewable bottleneck resource (e.g. labor or machine) is available in a constant amount throughout the project. Also, it often occurs that one or more non-renewable resources (fuel, raw materials, or money), each with a limited amount, are available for the project. In knowledge- intensive industries or high-technology enterprises, project costs usually consist of expenditure on manpower, such as salaries periodically paid by the enterprise (renewable resource cost), and rewards paid by the project leader for the employees' performance in the project (non-renewable resource cost) (Wuliang and Chengen, 2009).

In MRCPSP, discrete time-cost trade-offs problems (DTCTP) and discrete time-resource trade-offs problems

† : Corresponding Author

(DTRTP) have been extensively studied in the literature. Prabuddha *et al.* (1997) has shown that both 1, *T*| *cpm*, *disc*, *mu*|C_{max} (DTRTP) and 1, T|*cpm*, δ_J, *disc*, *mu*|*av* (DTCTP) are strongly NP-hard for general networks, where the notation used to represent the problems follows Demeulemeester and Herrolen (2002). Notation (1, *T*) stands for one non-renewable resource, *cpm* for Strict finish-start precedence constraints with zero time-lag, *disc* for discrete, *mu* for multiple activity-modes, C_{max} for project makespan, δ_J for project deadline, and *av* for resource availability cost. Demeulemeester and Herrolen (2002) also show that for the renewable resource case, the problem 1, 1|*cpm*, *disc*, *mu*|C_{max} is strongly *NP-hard*. For the MRCPSP, a project-mode (PM) or mode assignment is a *J*-tuple, $\omega = \{\omega(j): j = 1, \cdots, J\}$, where activity j will be executed with mode $\omega(j)$. Kolisch and Drexl (1997) show that existence of a resource-feasible project-mode for an MRCPSP with at least two non-renewable resource constraints is *NP-complete*.

The literature on the standard discrete time-resource trade-off problem, *m*, 1*T*|*cpm*, *disc*, *mu*|C_{max}, is relatively sparse, where notation (*m*, 1*T*) represents *m* types of resources that are both renewable and non-renewable (Talbot, 1982; Bouleimen and Lecocq, 2003). Hsu and Kim (2005) and Yamashita *et al.* (2006) studied a discrete time-cost trade-off problem, *m*, 1|*cpm*, δ_n, *disc*, *mu*|$\Sigma C(R_k)$, with the objective of minimizing the total renewable resource availability cost while meeting a prespecified deadline. For further information on the classification and investigation of RCPSP, we refer to (Herroelen *et al.*, 1998; Weglarz, 1999; Brucker *et al.*, 1999; Demeulemeester and Herroelen, 2002; Kolisch and Hartmann, 2006).

In this research, we focus on a different discrete time-cost trade-off problem, *m*, 1*T*|*cpm*, δ_n, *disc*, *mu*|$\Sigma c_j(m_j)$. Each activity must not be interrupted and is limited to renewable resource and non-renewable resource constraints, and the model objective is to minimize the project-mode cost while meeting a given deadline. In this model, the renewable resource availability of each type is predetermined, and this cost will depend on the deadline and thus is fixed. Such can occur when labors and equipments are owned by the company, and salaries and equipment depreciation costs are incurred whether or not the project is under execution. Additionally, each activity has a cost which is a function of the selected mode. The activity-mode cost consists of three kinds: consumptions of renewable and non-renewable resources, and an overhead cost depending on the mode. The cost of a non-renewable resource is linear in terms of the quantity consumed, although a quota is imposed and may not be exhausted; the cost of a renewable resource can either be zero or linear in the quantity consumed.

Memetic algorithm (MA) is a population-based algorithm with a meme defined as a unit of cultural evolution that is capable of performing local refinements (Moscato, 1989; Moscato, 1999). Unlike genetic algorithms (GAs), MAs can employ one or more local search methods following a recombination or a mutation. According to Buriol *et al.* (2004), a good MA implementation should have suitable recombination and mutation operators, efficient and effective local searches, and a well-structured population. In addition, a preprocessing is allowed to collect sufficient information, and it often leads to better solutions. However, this preprocessing may sometimes take much computational efforts (Ljubić and Raidl, 2003).

In this paper, we propose an MA and an exact solution method for solving the DTCTP model. The MA is characterized by three features. First, a truncated branch and bound heuristic is applied to form the initial population. Second, an adaptive evolution strategy is implemented using two parallel populations, one of which maintains individuals that are both (non-renewable) resource-feasible and deadline-feasible, and the other contains individuals that are resource-feasible but not deadline-feasible. Meanwhile, the number of parents drawn from each population will conversely depend on its present population size. Third, a three-stage local search method termed RDC is used for selecting a new project-mode: repairing into resource-feasible, deadline-feasible, and finally reducing mode cost. A well-known local search, backward-forward (BF) method, is applied to verify deadline-feasibility (Li and Willis, 1992; Tormos and Lova, 2001). The BF method has been proven to be the most powerful local refinement method for minimizing the makespan of RCPSP (Tormos and Lova, 2001; 2003; Vall *et al.*, 2005).

The remainder of this paper is organized as follows: Section 2 defines the problem; Sections 3 illustrates the problem-solving methods; Section 4 summarizes the numerical results; Section 5 presents the concluding remarks.

2. PROBLEM DESCRIPTION

This paper studies a DTCTP aiming to minimize project-mode cost subject to a deadline. The problem can be stated as follows: A project consists of a set of activities labeled 0 to *J*+1, where activities 0 and *J*+1 are dummy activities and represent the events of the project start time and finish time, respectively. There is a set of finish-start precedence relations with zero-time lags between activities, which cannot be violated and can be represented by an acyclic directed network $G = (N, A)$, where $N = \{0, 1, \cdots, J+1\}$ and A is the set of arcs. An activity j cannot start unless all of its direct predecessors indexed DP_j are completed. Each activity has to be processed in order to complete the project, and cannot be interrupted during execution.

Given a project-mode ω, MRCPSP is reduced to a single mode RCPSP, with activity j executed in mode $\omega(j)$ and with consumption cost $C(\omega(j))$. A project-mode is feasible if it contains at least one project schedule satisfying the following conditions: (1) precedence constraints; (2) renewable and non-renewable resource con-

straints; (3) deadline feasibility. Let Ω denote the set of all project-modes in the problem. The aim of the problem is to identify a feasible project mode (FPM) which has the minimum cost among Ω.

The availability of renewable resource $r \in R$ per period is limited to an integer quantity Q_r^R, where R is a set of renewable resource types. For each non-renewable resource type $r \in NR$, there is a restriction on the total amount of the resource for the entire project. This restricted amount is denoted as TQ_r^{NR}, where NR is a set of non-renewable resource types. Each activity j can be executed by one of the modes, $\omega(j) \in M_j$. Each activity-mode $\omega(j)$ specifies the corresponding duration $d_{j\omega(j)}$, the requirement for each renewable resource type $r \in R$ per period $q^R_{j\omega(j)r}$, and the requirement for each non-renewable resource type $r \in NR$, $q^{NR}_{j\omega(j)r}$, during activity execution. Finally, a project deadline T is defined. A project schedule S can be represented by either the start time of each activity (S_0, S_1, $\cdots$, S_{J+1}), or the finish time (f_0, f_1, $\cdots$, f_{J+1}), where $S_0 = f_0 = 0$ and project makespan $S_{J+1} = f_{J+1}$. Note that $f_j = S_j + d_{jw(j)}$ for all j.

It is assumed that the resource availability setup of each renewable type (machine, labor, or equipment) per period will not incur any additional costs, since they are a constant expenditure for the company. The total cost of a project-mode is the sum of the cost of the selected activity-modes. Each activity-mode cost $C(\omega(j))$ is a function of the amounts of renewable and non-renewable resources consumed, plus an overhead cost $h_{j\omega(j)}$ which is depending on the mode. The mathematical formulation of this problem is presented as follows:

$$\underset{\omega \in \Omega}{Minimize}\ C(\omega) = \sum_{j=1}^{J} \left(\sum_{r \in R} c_r \cdot q^R_{j\omega(j)r} \cdot d_{j\omega(j)} + \sum_{r \in NR} c_r \cdot q^{NR}_{j\omega(j)r} + h_{j\omega(j)} \right) \quad (1)$$

Subject to

$$S_i + d_{i\omega(i)} \le S_j \text{ for all arcs } \langle i, j \rangle \in A,\ i \in DP_j \quad (2)$$

$$\sum_{i \in S(t)} q_{i\omega(i)r} \le Q_r^R \quad \text{for } r \in R,\ t = 1, \cdots, T \quad (3)$$

$$\sum_{i \in N} q_{i\omega(i)r} \le TQ_r^{NR} \quad \text{for } r \in NR \quad (4)$$

$$\omega(j) \in M_j \quad j \in N \quad (5)$$

$$S_j \ge 0 \quad j \in N \quad (6)$$

Equation (1) is the model objective, where c_r is the additional cost of resource type r per unit during activity execution, and $h_{j\omega(j)}$ is the associated overhead cost. Constraint set (2) describes the precedence relationships. Constraint set (3) confines the resource usage per period for each renewable resource type, where $S(t)$ is the set of activities in progress during time period [t-1, t]. Constraint (4) restricts the usage of non-renewable resource r during the entire project.

3. MEMETIC ALGORITHM

The MA consists of three features: (1) two parallel populations for evolution, one of which stores deadline- and resource-feasible individuals (DRPop) and the other of which stores resource-feasible only individuals (RPop); (2) preprocessing of initial populations for DRPop and RPop; (3) a local search DRC (deadline-resource-cost) as a repairing function. First, we apply a truncated branch and bound (TB&B) heuristic to create a quality initial population for each of DRPop and RPop. Our experimental results have proven the effectiveness of the preprocess phase, especially for instances with tight non-renewable resource constraints or for those of moderate size. Each individual is near local optimal in terms of project makespan on the assigned project-mode, as the individual has been refined by the BF method. When either population is full, the worst solution is replaced with a newly generated and better solution. The preprocessing is limited to a preset CPU time.

At each generation of the evolution phase, two parents are selected from DRPop and RPop respectively, and produce two offspring using recombination and mutation operations. Note that the union of DRPop and RPop belong to resource-feasible solution space. Thus, the evolution is performed in a broader solution space, rather than in the sparsely scattered deadline- and resource-feasible solution space. This evolution strategy will enable the MA algorithm to explore quality solutions. The DRC local search is applied to each offspring to improve the quality of both populations and the current best solution. The algorithm terminates when a predetermined number of project-modes have been found.

The main framework of the MA is shown in Figure 1.

```
Begin
Preprocessing of initial population by branch and bound heuristic
While (termination condition not met) do
   Select parents from RPop and DRPop;
Apply recombination operator to generate an offspring followed by a mutation operation;
   Apply BF method to the offspring,
If the offspring is an FPM with cost lower than that of current best solution;
      If DRPop is full, replace the worst solution in DRPop;
      Else record into DRPop;
   Else apply RDC;
End (while)
End.
```

Figure 1. Pseudo code of MA.

3.1 Encoding and Decoding Schemes

The MA adopts a double list (ω, AL) to represent a solution, where ω is a project-mode (PM) and AL is an activity list. An AL represents the order of activities to be scheduled in the project. Each AL is decoded into a schedule by the forward serial-list-scheduling (F-SLS) method, and the schedule is further improved by BF method. Section 3.3 gives an example of the encoding scheme.

3.2 Preprocessing of Initial Populations

A truncated branch and bound method is applied to construct initial populations for DRPop and RPop. The TB&B uses the depth-first search as the branch selection rule in the search tree, where a node represents an activity, and an arc reflects the selected mode. The algorithm processes stage by stage according to the order of activity indices. Each node at the jth stage will have M_{j+1} branches. Let $\underline{\boldsymbol{m}}(j) = \{\omega(1), \cdots, \omega(j)\}$ be the partial project mode formed at stage j of the current iteration, $Q_r^{NR}(j)$ the consumed amount of non-renewable resource type r in $m(j)$, the consumed amount of non-renewable resource type r in $\underline{\boldsymbol{m}}(j)$ plus the minimum amount required to accomplish the remaining activities j+1, $\cdots$, J, $CT(j)$ the sum of the cost based on $\underline{\boldsymbol{m}}(j)$ plus the minimum cost to complete the remaining activities, and $CPM(j)$ the makespan computed by employing the critical path method on the project-mode $\{\underline{\boldsymbol{m}}(j) \cup$ minimum duration modes of the remaining activities$\}$. The TB&B algorithm fathoms a branch at stage $j < J$ if one of the following three conditions is met: (1) $CT(j) \geq C^*$; (2) $CPM(j) > T$; and (3) $Q_r^{NR}(j) > TQ_r^{NR}$ for $r \in NR$. The notation C^* is the current best project-mode cost. When the algorithm successfully reaches J once, the resource-feasible PM which currently has the lowest cost is found. The SPT rule, followed by the BF method, is then applied to investigate this PM's deadline feasibility. If the PM is deadline feasible, then it will be recorded into DRPop; otherwise, it will be recorded into RPop. In either case, a newly found PM will replace the worst solution when the population is full.

The preprocessing terminates when the running time reaches 0.1 seconds multiplied by the number of project activities. The sizes of the two populations are determined by an experiment on a set of instances with 30 activities, where each activity has three mode alternatives. The procedure of the TB&B is shown in Figure 2.

3.3 Recombination and Mutation Operator

For each recombination, two parents (a, b) are selected by *tournament*, and two offspring are generated.

```
Begin
While (termination condition not met) do
   Branch and bound search for next PM ω;
Check (1) Q_r^NR constraints, (2) CPM ≤ T, and (3)
  C(ω) < C*;
If all of the three conditions are met, investigate dead-
   line feasibility:
        Apply SPT rule with BF method to ω;
          If the makespan ≤ T, set C* = C(ω);
             If DRPop is not full, record ω into DRpop ;
             Else replace the worst solution by ω;
          Else
             If RPop is not full, record ω into Rpop;
             Else replace the worst solution by ω;
     End (if three conditions)
End (while)
```

Figure 2. Pseudo code of truncated B&B.

The probability of selecting a parent from DRPop is p_{DR} = (RPop size)/(DRPop size + RPop size), while the probability from RPop is $p_R = 1\text{-}p_{DR}$. If one of DRPop and RPop is empty and the other has only one individual, one parent is selected from the non-empty one, and the other will be generated randomly. If both DRPop and RPop are empty, two parents will be randomly generated. Both cases may occur at the end of preprocessing initial populations.

The recombination operator is performed in two lists: ω and AL. Each $\omega(j)$ records the information about cost and non-renewable consumption of activity j; thus, we apply a simple crossover to implement the mode recombination. About 30% to 70% of ω will be randomly selected from parent a, while the others are from parent b. Hence, each $\omega(j)$ of the offspring may maintain the cost and non-renewable consumption relationship between $\omega(j)$ and j. The recombination for the parent ALs is performed using a modified order-based procedure in which parent a randomly determines 50% of the positions and their respective activities. These selected activities are then placed into the offspring in the same positions as parent a, but follow the ordering of parent b. The offspring's unoccupied positions are filled by the unselected activities based on their sequences in parent b. The precedence feasibility of the offspring is then examined. Violated activities are shifted leftward one by one to the first feasible positions. Figure 3 depicts the recombination operation of producing one offspring for a project containing ten activities. Another offspring will be produced in the same manner by (b, a).

In the example, a random number 0.62 is generated. This number is then multiplied by 10, and rounded to an integer, so that 6 positions of parent a are randomly selected. Thus, offspring c will have 4 positions from parent b, which are 2, 5, 7, 8, and these mode assignments are replaced with the corresponding mode assignments in parent b. The order based recombination

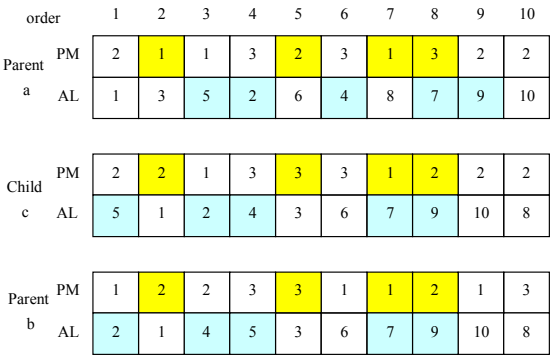

Figure 3. A recombination operation example.

of AL is performed by randomly selecting five positions, 3, 4, 6, 8, 9.

For each offspring, the mutation operator randomly selects one position in PM and changes it to another mode. If the resulting ω is a non-renewable feasible solution with a lower cost, then the F-SLS decoding scheme followed by the BF method will be applied to investigate the deadline feasibility; otherwise, the local search RDC will be applied. Note that mutation is only applied on PM. If the offspring is deadline-feasible, it will be recorded into DRPop; otherwise, the offspring is placed into RPop.

3.4 RDC Local Search

The RDC procedure is applied to each offspring with several aims-improve the current best solution and the quality of DRPop and RPop. RDC consists of three phases: resource-feasible, deadline-feasible, and cost reduction.

Phase 1: If the offspring is non-renewable resource-feasible, proceed to Phase 2 to check the deadline feasibility; otherwise, repetitively select an $\omega'(j) \neq \omega(j)$ with $q^{NR}_{j\omega'(j)r} < q^{NR}_{j\omega(j)r}$ at random to replace $\omega(j)$ until either a non-renewable resource-feasible solution is obtained, or a preset number of trials F has been reached. If a non-renewable resource-feasible PM is obtained, go to the Phase 2; otherwise, terminate RDC. For any phase, F is set to the integer rounded by 0.1 × problem size.

Phase 2: Apply the BF method to investigate the deadline feasibility. If not, update RPop; otherwise, a process that randomly selects and replaces an $\omega'(j) \neq \omega(j)$ with $d_{j\omega'(j)} \sum_{r\in R} \frac{q^R_{j\omega'(j)r}}{Q^R_r} < d_{j\omega(j)}$ $\sum_{r\in R} \frac{q^R_{j\omega(j)r}}{Q^R_r}$ is repeated until an FPM is found, or F trials has been completed. If an FPM is found, update DRPop and go to Phase 3; else if the PM is resource-feasible, update the RPop; else, terminate the RDC.

Phase 3: If the cost of the FPM is smaller than the current best, update C^*; otherwise, consecutively and randomly select an $\omega'(j) \neq \omega(j)$ with $C(\omega'(j)) < C(\omega(j))$ until a new best is found or F trials have been completed. An update on DRPop or RPop is performed as needed during the process.

3.5 Exact Solution Method

An exact solution method is applied to find optimal solutions for problems of moderate size. The branch and bound method introduced in Section 3.2 is used to find all potential PMs. For each potential PM, the deadline feasibility is verified by the following two-phase process:

Apply SPT rule plus BF method to find the PM's makespan f'_{J+1}. If $f'_{J+1} \leq T$, it is deadline feasible; otherwise, a precedence-tree branch and bound algorithm (Patterson, 1984; Sprecher, 1994) using T as the initial upper bound is employed to verify the deadline feasibility.

4. EXPERIMENTAL RESULTS

In this section, we present the performance of the algorithms described in Section 3. The algorithms are evaluated using the following criteria: project-mode feasibility (*PMF*), project-mode cost deviation (δ_C), optimality (*OPT/BEST*), and CPU time. *PMF* is presented as the percentage of instances in which at least one FPM is obtained. δ_C is shown as the percentage deviation of the project cost found from the best or optimal solutions, and is expressed in three ways: minimum (min), average (avg), and maximum (max) percentages. *OPT/BEST* indicates the percentage of instances reaching the optimal/best solutions. All algorithms were coded in Visual Studio C# NET and run on a computer with Intel core duo, 1.8GHz processor and 1 Giga bytes DDR566. The TB&B phase is limited to a preset CPU time, 0.1 multiplied by problem size. The termination conditions of the evolution phase is a maximum of 1000×J PMs found. Except for the one-hour experiments, others using MA are based on 10 runs.

In Section 4.1, we introduce the benchmark instances. Section 4.2 shows the performance of the MA on the instances.

4.1 Generation of Benchmark Instances

The instances include three sets, j18, j20, and j30 that contain 18, 20, and 30 activities, respectively. These instances were generated through ProGen (Kolisch *et al.*, 1995), and introduced in the PSPLIB (Kolisch and Spre-

cher, 1996). All data are available via *http://129.87.106.231/psplib/*. The PSPLIB provides the optimal makespans for j18 and j20, but only the current best makespans for j30. These instances are subject to two renewable and two non-renewable resource constraints. Each activity contains three modes and the duration of each activity-mode ranges from one to ten. The sets j18, j20, and j30 include 552, 554, and 552 resource-feasible instances, respectively.

There are three direct costs associated with each activity-mode. The unit cost of each renewable resource type per period is randomly generated from the integer interval [10, 20], while the unit costs of the first and second non-renewable resource types are from [20, 100] and [30, 150], respectively. Other than the resource aspect, each activity-mode includes an overhead cost randomly generated from [50, 200]. The deadlines are specified as the minimum or the current best makespan multiplied by a number randomly generated from [1.3, 1.5].

4.2 Numerical Results

In the experiment, the population size for (DRPop, RPop) is set to (20, 50), and the number F for RDC is an integer rounded from $0.1 \times J$. We use different stopping criteria for the TB&B to investigate the performance of the evaluation phase. One criterion is $0.1 \times J$ seconds CPU time and the other is the time doubled. The proposed MA algorithm consists of two phases: (1) TB&B phase finds quality solutions and construct initial populations for DRPop and RPop, and (2) MA phase (GA + RDC local search) executes subsequent evolutions. For simplicity, we shall denote the method by assigning 0.1 seconds $\times$ J termination criterion as "_a", 0.2 seconds $\times$ J criterion as "_b", and one hour as "_c." Tables 1 and 2 show the performance of the first two BB-MAs compared to the optimal solutions found by the exact solution method. Note that the exact solution method and TB&B heuristic differ in the method for determining the deadline feasibility (see Sections 3.2 and 3.5). The CPU time by the exact solution method grows significantly from j18 to j20. For j18, the TB&B finds the optimal solutions for approximately 95% of the total instances. The MA phase further improves the percentage of optimality by about 4%. For j20, the TB&B contributes roughly 94% of optimality, while the MA phase improves another 3%. For both instance sets, the TB&B also works exceptionally well in determining the resource- and deadline-feasibility. The MA phase concludes that all instances are feasible. The average computational times of the TB&B+MA for both instance sets are short. Furthermore, the TB&B+MA is robust as the cost deviation percentages δ_C in terms of average, minimum, and maximum are identical up to the second decimal point. The minimum (maximum) δ_C is defined as the average of the minimum (maximum) deviation in ten runs from each instance in the test set. Such results have shown that the TB&B+MA is very efficient and effective in solving problems of moderate size.

For the large size instance j30, we set a maximum of 3,600 seconds for the exact solution method. The exact method obtained 499 optimal solutions and one best solution, but cannot determine a deadline-feasible solution for 12 instances. The average CPU time is around 6 minutes, compared to a total of 43.65 seconds in average by implementing the TB&B+MA with BB phase set to a maximum of one hour. Apparently, this TB&B+MA with one-hour is superior to the exact solution method in terms of cost deviations, percentage of reaching feasibility and best solutions, and CPU time. There are only three instances where the modes cannot be completed by this heuristic. The performance of the MA with a short TB&B phase approaches the exact solution method, but the CPU time is much shorter. If the TB&B phase is set to more than nine minutes, the heuristic will find at least one FPM for all j30 instances. All three BB-MAs have roughly the same CPU running time for MA phase since their termination conditions are to investigate 30,000 PMs.

The performance of the exact solution method for j30 may be improved by replacing the simple SPT+BF scheme with another heuristic, such as simulated annealing with BF or genetic algorithm with BF; however, more running time will be required to implement these heuristics. The SPT+BF scheme is generally not effective in determining whether a PM is deadline-feasible for a project instance with 30 activities. In the exact solution method, the precedence tree branch and bound algorithm has to be executed to find the exact answer whenever the makespan computed by the SPT+BF scheme is greater than the specified deadline. It yields the worst performance when an instance contains an enormous number of resource-feasible project modes that have decreasing cost, and are deadline-infeasible in the search sequence. For such problems, the exact solution method will execute the precedence tree branch and bound method numerous times, which is time-consuming with 30-activity instances. As a result, the algorithm will exceed the time limit and generate no deadline-feasible solution or a non-optimal FPM.

The preprocessing BB phase of the MA is very effective and efficient for j18 and j20, and accounts for about 94% of optimal solutions. For j30, the BB phase yields approximately 80% of optimality for short CPU times. When the CPU time is set to one hour, the BB phase is able to investigate all resource-feasible modes. When the deadline is loose, the BB phase may outperform the exact solution method. However, if the deadline is tight, the exact solution method becomes a favorable method.

The second phase of the proposed MA uses GA+RDC local search for generational evolution. We shall call this evolutionary stage MA phase, which plays the role of further improving solution quality. The evolutions are performed by continuously maintaining dual populations containing individuals in resource-feasible

solution space. For j18 and j20 instances, this phase improves approximately 3 to 4 percent, which aggregates to nearly 100% optimality for j18 and 97% optimality for j20. For j30, the two-phase method performs almost perfectly when CPU time is set to one hour, while the exact solution method only reaches 90.5%.

The three bounding rules, along with SPT+BF used in the preprocessing phase, are effective in pruning inefficient project modes. However, when the deadline is tight, using a meta-heuristic rather than SPT+BF will be helpful in reducing the CPU time of the exact solution method. In such a situation, the exact solution method will be favorable in terms of finding the optimal solution within an acceptable computation time for j18 and j20.

4.3 Solving time/cost trade-off profile problems

The proposed algorithms can easily be applied to solve discrete time-cost trade-off profile problems using horizon varying approach (Demulemeester *et al.*, 1998). This approach is similar to ε-constraint method (Haimes *et al.*, 1971), which suggests reformulating the multi-objective optimization problem by just keeping one of the objectives and restricting the rest of the objectives within user-specified values.

While applying horizon varying approach, a set of deadlines are prespecified in advance. For each deadline, the proposed MA and exact solution method are employed to find the minimum total cost, which is the sum of total resource availability cost (fixed cost) and project mode execution cost (variable cost). Note that the total resource availability cost is defined as the resource availability cost per period multiplied by the deadline.

To test the performance of the proposed MA and exact solution method on the time-cost profile problem, an experiment was conducted on the problem instance j2057-10.mm from the PSPLIB. For this instance, the availability of resource types 1 and 2 are set to eight and six units per period, respectively. The unit cost of both types is ten per unit, which yields a total resource availability cost of 140 per period.

Figure 4 presents the computational results of applying the exact solution method to the instance. This figure displays total resource availability cost, total variable cost, and total cost with respect to each of the deadlines ranging from 35 to 60 periods, where 35 is the minimum makespan. After comparing their total costs, only five non-dominated solutions are concluded and shown in Figure 5. The five solutions have makespans coinciding with their deadlines, and have the following dual-objective values (35, 22452), (36, 22018), (37, 21798), (38, 21649), and (39, 21617). When MA is applied to this instance, the time-cost relation is displayed and compared to the exact solution method in Figure 6. The total cost increases after deadline reaches 40 because the marginal cost of the resource availability increases constantly while the optimal variable cost converges to a constant. For this test instance, MA_b produces two non-dominated solutions, (38, 21704) and (39, 21672), which are slightly inferior to their counterparts produced by the exact solution method. Although MA_b cannot find a feasible solution when the deadline is strictly specified-hat is, between 35 and 37-it uses much less computation time than the exact solution method. Finally, we can infer that the number of non-dominated solutions will increase when the resource availability cost per unit falls.

5. CONCLUSION

This paper studies the MRCPSP with the objective of minimizing the project-mode cost within a deadline. An MA algorithm and an exact solution method are proposed to solve this problem. The MA consists of three features: an evolution strategy of using two parallel populations, an efficient and effective B&B heuristic to form a quality initial solution for both populations, and a RDC local search. The MA performs very well in terms of cost deviations, percentage of optimality attained, and CPU time, when compared to the performance of the exact solution method. While the exact solution method is efficient and effective for problem sizes j18 or less, the MA is more suitable for large sizes j20 and j30. The research is generally useful in practice, since a major managerial objective is to complete the project in the most economic manner.

Many studies on the MRCPSP consider only one of the two objectives: makespan and cost. Vanhouche *et al.* (2002) mention the third objective of this problem-to construct a complete and efficient makespan/cost profile over the set of feasible project makespans. One prospective research direction of the MRCPSP may simultaneously consider the two minimization objectives. With a set of near Pareto-optimal schedules, management can select the best alternative based on his preference and the environment. For such bi-objective optimization problems, the proposed hybrid methods can be a very useful solution approach.

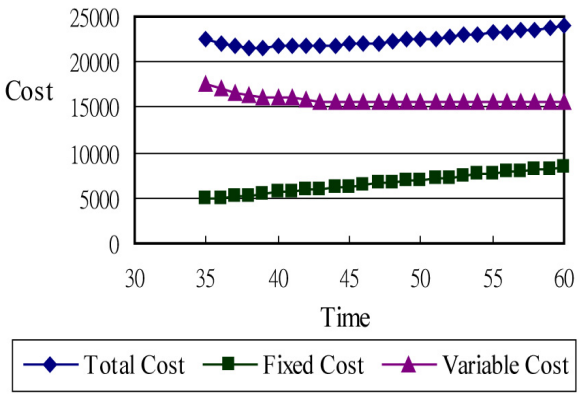

Figure 4. Three costs generated by exact solution method on the test instance.

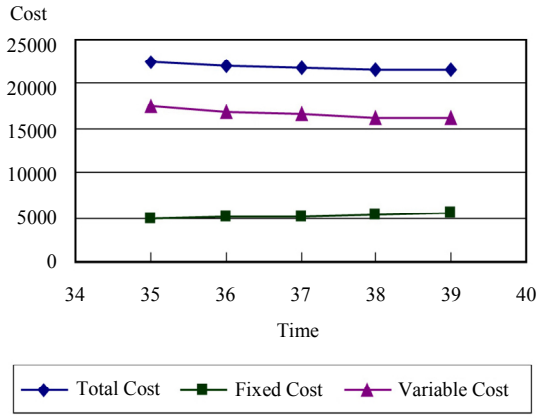

Figure 5. Five non-dominated solutions generated by exact solution method on the test instance.

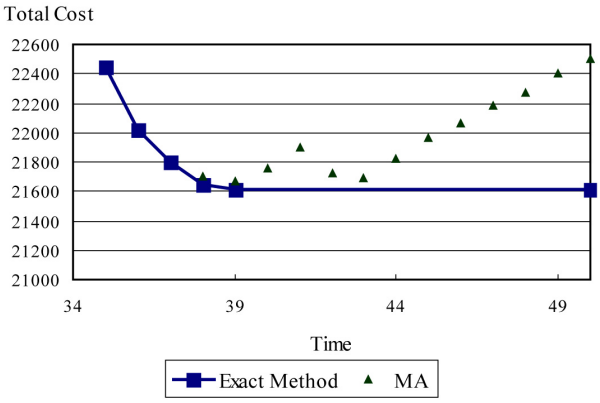

Figure 6. Comparison of time and cost objectives between exact solution method and MA_b.

Table 1. Experimental results of j18.

Algorithm	TB&B limitation (seconds)	avg δ_C	min δ_C	max δ_C	PMF	OPT	$\delta_C \le 1\%$	$\delta_C \le 3\%$	CPU time (second)
Exact solution method	-	0.00%	-	-	100.00%	100.00%	100.00%	0.00%	8.91
BB phase_a	1.8	0.03%	-	-	100.00%	95.47%	99.09%	100.00%	0.04
MA_a		0.00%	0.00%	0.00%	100.00%	99.87%	100.00%	100.00%	0.27
BB phase_b	3.6	0.02%	-	-	100.00%	95.83%	99.27%	100.00%	0.06
MA_b		0.00%	0.00%	0.00%	100.00%	99.82%	100.00%	100.00%	0.28

Table 2. Experimental results of j20.

Algorithm	TB&B limitation (seconds)	avg δ_C	min δ_C	max δ_C	PMF	*OPT*	$\delta_C \le 1\%$	$\delta_C \le 3\%$	CPU time (second)
Exact solution method	-	0.00%	-	-	100.00%	100.00%	100.00%	0.00%	170.14
BB phase_a	2	0.04%	-	-	99.64%	94.04%	98.91%	99.64%	0.12
MA_a		0.01%	0.01%	0.01%	100.00%	97.38%	99.78%	100.00%	0.38
BB phase_b	4	0.04%	-	-	99.82%	94.40%	98.91%	99.64%	0.17
MA_b		0.01%	0.00%	0.01%	100.00%	97.56%	99.96%	100.00%	0.40

Table 3. Experimental results of j30.

Algorithm	TB&B limitation (seconds)	avg δ_C	min δ_C	max δ_C	PMF	*BEST*	$\delta_C \le 1\%$	$\delta_C \le 3\%$	CPU time (second)
Exact soluteon method	3600	1.17%	-	-	97.83%	90.58%	92.96%	93.89%	360.60
BB phase_a	3	0.97%	-	-	99.09%	78.44%	84.64%	88.67%	0.81
MA_a		0.17%	0.07%	0.33%	99.51%	85.80%	94.59%	98.69%	1.49
BB phase_b	6	0.64%	-	-	99.09%	82.61%	87.39%	92.87%	1.26
MA_b		0.10%	0.04%	0.19%	99.44%	88.99%	96.37%	99.32%	1.97
BB phase_c	3600	0.02%	-	-	100.00%	95.11%	99.09%	100.00%	42.93
MA_c		0.00%	-	-	100.00%	99.09%	99.82%	100.00%	43.65

ACKNOWLEDGEMENT

This research was supported by the National Science Council in Taiwan under grant NSC 98-2221-E-155-039.

REFERENCES

Bouleimen, K. and Lecocq, H. (2003) A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, *European Journal of Operational Research*, **149**(2), 2684-281.

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

Buriol, L., Franca, P. M., and Moscato, P. (2004) A new memetic algorithm for the asymmetric traveling salesman problem, *Journal of Heuristics*, **10**(5), 483-506.

De Reyck, B. (1998) Scheduling Projects with Generalized Precedence Relations-Exact and Heuristic Procedures, *Ph.D. Dissertation*, *Department of Applied Economics, Katholieke University Leuven*.

Demeulemeester, E., De Reyck, B., Foubert, B., Herroeleon, W., and Vanhoucke, M. (1998) New computational results on the discrete time/cost trade-off problem in project networks, *Journal of the Operational Research Society*, **49**(11), 1153-1163.

Demeulemeester, E. L., and Herroelen, W. S. (2002) *Project scheduling*: *A Research Handbook*, Norwell, MA: Kluwer.

Halman, N., Li, C. L., and Simchi-Levi, D. (2009) Fully polynomial-time approximation schemes for time-cost tradeoff problems in series-parallel project networks, *Operations Research Letters*, **37**(4), 239-244.

Hamimes, Y. Y., Lasdon, L. S., and Wismer, D. A. (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization, *IEEE Transactions on Systems, Man and Cybernaties*, **1**(3), 296-297.

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

Hsu, C. C. and Kim, D. S. (2005) A new heuristic for the multi-mode resource investment problem, *Journal of the Operational Research Society*, **56**(4), 406-413.

Kolisch, R. and Drexl, A. (1997) Local search for non-preemptive multi-mode resource-constrained project scheduling, *IIE Transactions*, **29**(11), 987-999.

Kolisch, R. and Hartmann, S. (2006) Experimental investigation of heuristics for resource-constrained project scheduling: An update, *European Journal of Operational Research*, **174**(1), 23-37.

Kolisch, R. and Sprecher, A. (1997) PSPLIB–A project scheduling problem library, *European journal of Operational Research*, **96**(1), 205-216.

Kolisch, R., Sprecher, A., and Drexl, A. (1995) Characterization and generation of a general class of resource-constrained project scheduling problems, *Management Science*, **41**(10), 1693-1703.

Li, K.Y. and Willis, R. J. (1992) An iterative scheduling technique for resource-constrained project scheduling, *European Journal of Operational Research*, **56**(3), 370-379.

Ljubić, I. and Raidl, G. R. (2003) A memetic algorithm for minimum-cost vertex-biconnectivity augmentation of graphs, *Journal of Heuristics*, **9**(5), 401-427.

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

Moscato, P. (1999) Memetic algorithms: a short introduction, In D. Corne, F. Glover and M. Dorigo, eds. *New Ideas in Optimization*, *McGraw-Hill*, 219-234.

Patterson, J. H. (1984) A comparison of exact procedures for solving the multiple constrained resource project scheduling problem, *Management Science*, **30**(7), 854-867.

Prabuddha, D. E., Dunne, E. J., and Ghosh, J. B. (1997) Complexity of the discrete time-cost tradeoff problem for project networks, *Operations research*, **45**(2), 302-306.

Sprecher, A. (1994) Resource-constrained project sche-duling -exact methods for the multi-mode case, *Lecture Notes in Economics and Mathematics*, *No 409, Springer*, *Berlin*, *Germany*.

Talbot, F. B. (1982) Resource constrained project scheduling with time-resource tradeoffs: the nonpreemptive case, *Management Science*, **28**(10), 1197-1210.

Tormos, P. and Lova, A. (2001) A competitive heuristic solution technique for resource-constrained project scheduling, *Annals of Operations Research*, **102**(1-4), 65-81.

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

Valls, V., Ballestin, F., and Quintanilla, S. (2005) Justi-

fication and RCPSP: a technique that pays, *European Journal of Operational Research*, **165**(2), 375-386.

Vanhoucke, M., Demeulemeester, E., and Herroelen, W. (2002) Discrete time/cost trade-offs in project scheduling with time-switch constraints, *Journal of the Operations Research Society*, **53**(7), 741-751.

Weglarz, J. (1999) *Project Scheduling*: *Recent models, Algorithms and Applications, Norwell, MA*: *Kluwer*.

Wuliang, P. and Chengen, W. A. (2009) Multi-mode resource-constrained discrete time-cost tradeoff problem and its genetic algorithm based solution, *International Journal of Project Management*, **27** (6), 600-609.

Yamashita, D. S. Armentano, V. A., and Laguna, M. (2006) Scatter search for project scheduling with resource availability cost, *European Journal of Operational Research*, **169**(2), 623-637.