Fifth International Conference on Construction in the 21st Century (CITC-V)
"Collaboration and Integration in Engineering, Management and Technology"
May 20-22, 2009, Istanbul, Turkey

A Comparison of Tabu Search, Simulated Annealing and Genetic Algorithms for Discrete Time-Cost-Quality Trade-Off Analysis

Serdar Kale, M. Emre İlal
Balıkesir University, Balıkesir, Turkey
*skale@balikesir.edu.tr, ilal@balikesir.edu.*tr

Abstract
The discrete time-cost-quality trade-off analysis is a classical Multi-Objective-Combinatorial Optimization (MOCO) problem. Construction management researchers have been using combinatorial optimization algorithms to solve this multi-objective optimization problem. These algorithms are both approximate and blind. These features of combinatorial optimization algorithms have raised a number of questions on their relative performance and triggered a number of research studies. The research presented in this paper compares the relative performance of three popular combinatorial optimization techniques, namely Simulated Annealing (SA), Genetic Algorithm (GA), and Tabu Search (TS), on a test problem with respect to three performance criteria: (1) quality of optimal solution, (2) computational time and (3) number of candidate solutions evaluated before reaching the optimal solution. The test results suggest that TS, GA and SA perform well in finding the optimal solution. TS outperforms both SA and GA in terms of computational time and number of candidate solutions evaluated before reaching the optimal solution for the test problem.

Keywords
Project management, Time-cost-quality trade-off, Combinatorial optimization, Algorithms

1. Introduction

The time-cost-quality trade-off analysis has been receiving growing interest in recent years (e.g., El-Rayes and Kandil, 2005; Pollack-Johnson and Liberatore, 2006; Tareghian and Taheri, 2006; Afsar *et al*., 2007; Rahimi and Iranmesh, 2008; Huang *et al*., 2008). The time-cost-quality trade-off analysis is an extension of traditional time-cost trade-off analysis (e.g., Siemens, 1971; Panagiotakopoulos, 1977; Moleshi, 1993; Liu *et al*., 1995; Feng *et al*., 1997). The main impetus behind this extension can be attributed to strong dissatisfaction from traditional cost-time trade-off analysis models that assume an equal level of quality is maintained throughout the analysis (Pollack-Johnson and Liberatore, 2006) and changing nature of the construction business environment (El Rayes and Kandil, 2005). Several time-cost-quality trade-off analysis models have been developed in order to assist construction project managers to meet this challenge and in turn complete construction project within specified time, cost and quality. The earliest time-cost-quality trade-off analysis models (e.g., Babu and Suresh, 1996; Khang and Myint, 1999) are based on mathematical programming methods. These models transform the time-cost-quality trade-off problem to mathematical models and solve it by using linear programming technique. Yet linear programming technique requires a great amount of computational effort (Panagiotakopoulos, 1977). Furthermore it is suitable for solving problems with linear time, cost and quality relationships and small sized construction projects (Feng *et al*., 2000). More recently a number of time-cost-quality trade-off analysis models (e.g., El-Rayes and Kandil, 2005; Iranmanesh *et al*., 2008; Rahimi and Iranmanesh 2008; Afsar *et al*., 2007; Huang and Zhang, 2008) have been proposed to overcome limitations of the

earlier models. These set forth models differ from the earlier models on two counts: (1) assumption that tradeoffs among time, cost and quality are discrete rather than linear and (2) use of combinatorial optimization techniques rather than mathematical programming techniques. These differences can be attributed the efforts to meet arising practical needs (Vanhoucke, 2005) and to better capture the realties of the construction industry.

A construction project consists of hundreds of construction activities. Each construction activity can be performed by using different crew sizes, equipment, methods, and technologies. The primary challenge facing construction project managers is selecting appropriate option for each activity that can minimize project duration and cost and maximize project quality. This challenge can be more appropriately addressed by using combinatorial optimization algorithms because selecting appropriate option for each activity among several alternatives is a classical combinatorial optimization problem (Feng *et al*., 2000; Xiong and Kuang, 2008). Combinatorial Optimization (CO) is the process of finding optimum selection, arrangement, grouping, and ordering of discrete objects (Osman and Kelly, 1996). Different combinatorial optimization algorithms (e.g., Genetic Algorithms, Tabu Search, Simulated Annealing Ant Colony Optimization, Swarm Particle Optimization) have been set forth to solve combinatorial problems. Combinatorial optimization algorithms are approximate algorithms (i.e., they do not guarantee finding optimal solution) and blind (i.e., they do not know if they reached an optimal solution) (Youssef *et al*., 2001). Mainly due to these two characteristics of combinatorial optimization algorithms, it is very common in the literature to come across implicit or explicit statements that suggest that one of the combinatorial optimization algorithms outperforms its counterparts (e.g., Arostegui *et al*., 2006; Xiong and Kuang, 2008). Implicit or explicit statements on performance of combinatorial optimization algorithms have triggered a number of research studies to compare their performances. Youssef *et al*. (2001) used floor planning of very large scale integrated (VSLI) problem to compare performances of three combinatorial optimization algorithms namely Tabu Search (TS), Genetic Algorithms (GS), and Simulated Annealing (SA) algorithms. They report that TS outperforms GA and SA and GA outperforms SA. Hasan *et al*. (2001) explored performance of TS, GA and SA in solving quadratic programming (QP) problem. They report that GA is superior to SA and TS in solving the QP and overall performance of TS compared with GA and SA is unsatisfactory. Arostegui *et al*. (2006) used three facility layout problems (FLP) to compare the performances of TS, GA, and SA algorithms. They report that TS has superior performance for all of the three problems studies. These research findings jointly suggest that performance of a combinatorial optimization algorithm is sensitive to the problem category. Therefore, exploring performance of combinatorial optimization algorithms in different problem categories is needed in order to identify the most preferable combinatorial optimization algorithm(s) for that category of problems. Yet a research study on performance of combinatorial optimization algorithms in the category of discrete time-cost-quality trade-off analysis is missing in literature. The research presented in the paper presents a comparison of combinatorial optimization algorithms in solving discrete time-cost-quality trade-off problem. The objectives of the paper are twofold: (1) to compare performance of combinatorial optimization algorithms in solving discrete time-cost-quality trade-off problem and (2) to identify whether a combinatorial optimization algorithm is more preferable than others in solving discrete time-cost-quality trade-off problem. A comparison study on combinatorial optimization algorithms, like any other comparison studies, requires selecting combinatorial optimization algorithms to be explored and defining a set of criteria for comparing the selected combinatorial optimization algorithms. The selection of combinatorial optimization algorithms for this study is guided by their popularity. The most popular combinatorial optimization techniques include Simulated Annealing (SA), Genetic Algorithms (GA), and Tabu Search (TS). A set of criteria for comparing performances of the selected combinatorial algorithms was developed in the light of the previous research studies (Hasan *et al*., 2001; Arostegui *et al*., 2006). The performances of the combinatorial algorithms in this study were evaluated along three dimensions: (1) quality of solution identified by each combinatorial optimization technique, (2) number of solutions evaluated by each combinatorial optimization technique to find optimum solution and (3) computational time required for each combinatorial optimization technique to find optimum solution.

2. Problem Description

The discrete time cost and quality trade-off analysis model presented in this paper follows the notations used in previous research studies (e.g., Vanhoucke, 2005; Tareghian and Taheri, 2006). It assumes a construction project is represented by activity-on-arc (AoA) network G = (N, A) where the set of nodes (N) represents precedence relationships for activities and the set of arcs (V) represents activities. A set of M_i of modes is available for performing each construction project activity $i \in A$. Each mode, $r \in M_{i,}$, represents time(t_{ir}), cost (c_{ir})and quality(q_{ir}) of activity *i* performed in mode *r*.

The discrete time cost and quality tradeoff is stated as follows:

$$T = s_n + \sum_{k=1}^{r(n)} x_{nk} t_{nk} \quad (1)$$

$$C = \sum_{i=1}^{n} \sum_{k=1}^{r(i)} x_{ik} c_{ik} + icT \quad (2)$$

$$Q = \sum_{i=1}^{n} w_i \sum_{k=1}^{r(i)} x_{ik} q_{ik} w_{ik} \quad (3)$$

Subject to

$$\sum_{k=1}^{r(i)} x_{ik} = 1 \qquad \forall i \in V \quad (4)$$

$$s_j - s_i \geq \sum_{k=1}^{r(i)} t_{ik} x_{ik} \qquad \forall (i,j) \in E \quad (5)$$

$$s_i \geq 0 \qquad \forall i \in V \quad (6)$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V, k \in [1,2,....,r(i)] \quad (7)$$

The model parameters are described below:

n	number of activities
r_i	number of modes for activity i
t_{ik}	duration of activity i when performing the kth option
q_{ik}	quality of performing activity i under kth option
c_{ik}	direct cost of performing activity i under kth option
ic	indirect cost rate of a project
x_{ik}	index variable of activity i when performing the kth option
w_i	quality weight of activity i
w_{ik}	quality indicator weight for performing activity i under kth option

Eqs. 1-3 represent objective functions of the model. Eq.1 minimizes the project duration (*T*). Eq. 2 minimizes the project's cost (*C*). Eq. 3 maximizes project's quality (*Q*). Eqs. 4-7 represent constraints of the model. Eq. 4 ensures that one and only execution mode is assigned to each activity. Eq. 5 preserves the precedence relations between project activities. Eq. 6 forces project not to start earlier than time zero. Eq. 7 is a binary mode indicator which is 1 when mode k assigned to activity i and otherwise 0. The above defined problem is not only a Combinatorial Optimization (CO) problem but also a Multi-Objective Optimization (MOO) problem. Multi-Objective Optimization is the process of

optimizing systematically and simultaneously a collection of objective functions. The objective functions in multi-objective optimization problem can be entirely maximization, minimization or combination of both as in the case of discrete time-cost-quality trade-off analysis which involves minimizing project duration and cost and maximizing project quality (i.e. Eqs. 1-3). Multi-objective optimization problems are commonly solved by transforming the original objective functions into a non-dimensional objective function with upper limit of one and lower limit of zero and then minimizing it. The following single-objective function can be used to combine three objectives functions (i.e. Eqs. 1-3) of the discrete time-cost-quality trade-off problem:

$$Min\ d = \sum_{j}^{3} \left| \frac{F_j(s^*) - F_j(s)}{F_j(s^*)} \right| \qquad 8$$

where *d* is relative distance, $F_j(s)$ is value of a candidate solution for j th objective function and $F_j(s^*)$ is value of best solution found so far for j th objective function.

3. Combinatorial Optimization Algorithms

3.1 Simulated Annealing

For this combinatorial optimization algorithm, the analogy is to the annealing process in metallurgy. Annealing is performed to increase crystal size and reduce defect in metals. It involves heating up a solid until it melts into liquid form allowing particles to freely rearrange themselves. During the slow cooling cycle, particles settle down in their lowest energy states. Success of the annealing process depends on the highest temperature reached and the rate of cooling. The simulated annealing algorithm was introduced for solving complicated combinatorial optimization problems. The analogy between the algorithm and the physical system is that the objective function and the solution (configuration) in the optimization process correspond to the energy function and the state of particles, respectively. Each iteration of the algorithm picks a random neighboring solution and evaluates it. The search moves to better solutions automatically but also has a probability to move to worse solutions depending on how worse the solution is as well as the global temperature. At high temperatures, the algorithm moves freely exploring the landscape. As temperature decreases moves to worse solutions gradually become unlikely and the algorithm adopts a greedier descent strategy. An appropriate neighborhood creation mechanism, movement strategy and cooling schedule are essential in finding optimal solutions with simulated annealing. SA's effectiveness becomes more significant as the system size increases.

3.2 Genetic Algorithms

A genetic algorithm is a combinatorial optimization algorithm inspired by biological evolution. The features of genetic algorithms are different from other search techniques in several aspects. First, the algorithm does not search in one locale. The search takes place at a number of locations represented by the population. Therefore, the possibility of being trapped at a local minimum is reduced. Secondly, GA works with a coding of parameters (the chromosome) instead of the parameters themselves. Movement is random and depends on the stochastic selection of phenotypes, a crossover and a probability based mutation that evolves the next generation. The chromosome allows the genetic operator to evolve the next generation with minimum computations. Thirdly, GA simply evaluates the fitness of each string to "guide" its search instead of an "intelligent" optimization function that requires auxiliary knowledge.

3.3 Tabu Search

Tabu search is a local neighborhood search procedure with short-term memory. The memory preserves a number of previously visited solutions along with a number of solutions that might be considered

unwanted. This information is stored in a tabu list for a number of turns in order to avoid backtracking. At each iteration the algorithm evaluates the neighboring solutions and moves to the best. If no better solution exists, moves to worse solutions not on the tabu list are allowed in order to escape local minima. The definition of a solution, the area around it and the length of the tabu list are critical design parameters. In addition to these tabu parameters, two extra parameters are often used: Aspiration and diversification. Aspiration is used when a solution that is otherwise tabu should still be selected. In that case, the tabu obstacle is overridden by selecting a new solution. Diversification adds randomness to this otherwise deterministic search. If the Tabu search is not converging, a random restart can be carried out.

4. Implementation

In order to compare the performance of these techniques a new tool has been developed in Java. The tool reads a network and provides a common platform for the three algorithms to carry out their optimization tasks. A user interface allows fine tuning of algorithm parameters. For all algorithms a *solution* is defined as the list of selected options for every activity in the network. The implementations utilize an array (or string) of integers where the array index represents an activity and the integer value represents the option selected for that activity.

4.1 Simulated Annealing Algorithm

For simulated annealing, the *neighborhood* to be used during the search process is defined for any given solution as the set of solutions that are different only in one activity. The pseudo-code for the SA algorithm is given in Figure 1.

```
initialize temperature
for i := 1...coolingIterationLimit do
  temperature := coolingRate* temperature
  for j := 1...moveIterationLimit do
    randomly pick a move to a neighbor
    evaluate the move
    delta := current_value – move_value
    if delta > 0 then  // new best
       make the move permanent
       increment good_moves
    else
       p := random number in range [0...1]
       m := exp( delta / temperature )
       if p < m then  //The Metropolis criterion
            make the move permanent
            increment good_moves
       end if
    end if
    exit when good_moves > moveLimit
  end for
end for
```

Figure 1: The Simulated Annealing Pseudo-code

The Metropolis criterion (Metropolis *et al*., 1953) which is used during the search is defined as:

$$m = e^{delta/temperature} \qquad 9$$

This criterion is what allows the search to move uphill and avoid being trapped at a local minimum. The value of m will be between 0 and 1 and approach 0 for large negative values of delta representing a solution that is a big step back and/or lower values of temperature. Towards the latter stages of the search, uphill moves become harder and forces a detailed exploration of the neighborhood around the best

solution found up to that point. The five SA parameters users are able to explore are: Starting temperature, cooling rate (as a percentage), maximum number of cooling cycles, maximum number of iterations at each temperature and the maximum number of moves allowed at each temperature.

4.2 Genetic Algorithm

A solution is referred to as a *chromosome* in GA and represents a phenotype (an individual). The GA search does not make use of a neighborhood but instead relies on the genetic operator that produces the next generation from the current one. The genetic operator utilizes a combination of crossover and mutation mechanisms. Crossover between two chromosomes is the swapping of option selections from one array to another beyond a random point on the two arrays. Mutation alters the option selection for each activity with a probability defined by the user. Elitism is also implemented as a user option. When elitism is turned on the best two phenotypes in each current generation are automatically passed on to the next providing a slot for recording the best solutions found during the search. The pseudo-code for GA is given in Figure 2.

```
initialize random population(size)
for i := 1...iterationLimit do
  if elitism is on then
    place best 2 into nextGeneration
    while currentGeneration is not empty do
      // CROSSOVER
      seperate the best 2 from currentGeneration
      pick a random point on both chromosomes
      swap strings after the point
      place as offspring into nextGeneration
    end while
    foreach chromosome in nextGeneration do
      // MUTATION
      foreach activity on chromosome do
        p := random number in range [0...1]
        if (p < mutationProbability) then
          pick a new random option
        end if
      end foreach
    end foreach
  end if
  sort nextGeneration
end for
```

Figure 2: The Genetic Algorithm Pseudo-code

The four user controlled parameters of GA are: The size of the population, the number of iterations, the mutation probability as a percentage and whether or not elitism will be employed.

4.3 Tabu Search Algorithm

Tabu search (TS) implementation utilizes the same solution objects as simulated annealing as well as using the same neighborhood definition. During a TS iteration, first, the complete neighborhood of the current solution is generated and evaluated. The algorithm proceeds by making a list of possible moves. This stage is where the tabu list is utilized and prevents backtracking on recent moves unless the result would meet an aspiration criterion such as creating the best ever solution encountered. Then, the best move among the possible moves is selected even if it results in a worse solution than the current solution. This constraining mechanism is what guides TS away from local minima. The length of the tabu list needs to be long enough to constrain the algorithm out of wider local minima areas. A reactive tabu search dynamically resizes the tabu list. If better solutions are not encountered, tabu list grows in order to force

the search into unexplored areas. The reactive tabu list shrinks in order to speed up the search when going downhill reaching better solutions. The tabu search pseudo-code is given in Figure 3.

```
select random start
create tabuList(listSize)
for i := 1...iterationLimit do
  generate neighboorhod from current position
  evaluate all moves
  foreach move do
    if move is tabu AND
         aspiration critera not met then
      eliminate possibility of move
    end if
  end foreach
  pick bestMove among possible moves
  if bestMove is better than bestEver then
    bestEver := bestMove
    if reactive then
      listSize := listSize/2
    end if
  else if bestMove not better than current then
    if reactive and listSize < maxList then
      listSize := listSize + 2
    end if
  end if else
  move
  add move to tabuList
end for
```

Figure 3: The Tabu Search Pseudo-code

The four parameters available to users for tuning tabu search are: Initial tabu list size, number of iterations to be conducted, whether or not the search will employ a reactive tabu list and the maximum tabu list size to be used when a reactive tabu list is employed.

5. Test Project and Results

Three combinatorial optimization techniques discussed in preceding sections were tested and compared on an 18-activity CPM network project introduced by Feng *et al*. (1997) and subsequently modified by El Rayes *et al*. (2005). The original project was developed to implement GA to solve discrete time-cost trade-off analysis problems. Therefore, it has a set of time and quality pair for the options of each project activity. Yet the modified project has a set of time, cost and quality triplet for the options of each project activity. The test project does not have indirect costs. The tests for comparing performance of the selected combinatorial algorithms were performed in two stages. The first stage of tests involves running combinatorial algorithms in order to fine tune their parameters for the test problem. The parameters for each algorithm are incrementally adjusted to the point where the algorithm is allowed to run just as long as is necessary for it to find the optimal solution in all instances of a set of 100 runs. The second stage of the tests involves running combinatorial algorithms 10 times with settings determined in the first stage and tracking the progress of each individual run.

The first stage of the tests reveals that the test problem was optimally solved by TS, GA and SA. The results suggest that with regard to the quality of the optimal solution all three algorithms have equal performances. They all converged on the identical solution. For the test project, this optimum solution as represented by an array of option selections for each activity is: [1,3,1,3,4,3,3,1,1,1,1,1,1,1,1,5,1,1]. Selecting these options (i.e., execution modes) for performing the project activities would result in a total project duration of 114 days, a cost of $110620, and a quality level of 81.40%. The optimum solution is comparable to that reported by El-Rayes and Kandil (2005). Also in the first stage, based on the 100 runs, averages are determined for computational time and number of solutions evaluated by the algorithms.

Table 1 presents average computational time, standard deviation, minimum and maximum values for TS, SA and GA over 100 runs. The average computation time for TS to find the optimum solution is 4.06 milliseconds. It is shorter than the minimum computation times of both SA and GA. SA has the second shortest average computational time while GA has the longest. TS outperforms both its counterparts on this criterion. Table 1 also presents average, minimum and maximum number of candidate solutions evaluated in finding the optimum solution. The average number of candidate solutions evaluated in finding optimum solution for TS is 573. It is fewer than the minimum number of candidate solutions evaluated by SA and GA in finding the optimum solution. SA evaluates fewer number of candidate solutions than GA in finding the optimum solution. It is clear that TS performs better than SA and GA on this performance criterion as well.

Table 1: Performance of TS, SA and GA for the Test Project (over 100 runs)

Algorithm	*Computational Time (millisecond)*				*Number of Candidate Solutions Evaluated*			
	Mean	Std. Dev.	Min	Max	Mean	Std. Dev.	Min	Max
TS	4.06	1.01	2	8	573	86.55	370	783
SA	7.52	0.95	6	11	1027	110.22	1186	1787
GA	21.61	8.56	9	45	2424	980.59	953	5021

The second stage of tests involves running each algorithm 10 times while recording each discovery of a solution that is better than all solutions evaluated up to that point or in other words the discovery of a new best solution. Figures 4a, 4b, and 4c present the incremental progress over time of TS, SA and GA algorithms respectively. In these figures the time of discovery of a new best solution is plotted against the objective value of the solution for 10 sample runs. It is clear from Figures 4a-c that for the test problem TS converges on the optimum solution quicker than SA and GA. GA is the slowest of the three. Figures 5a, 5b, and 5c present the incremental progress over number of solutions evaluated for TS, SA and GA algorithms respectively. The order in which new best solutions are discovered is plotted against the objective value of these new best solutions during 10 sample runs. Figures 5a-c suggest that TS requires the fewest number of candidate solution evaluations before reaching the optimum solution for the test problem while GA requires the most number of evaluations.

The better performance TS and SA on these two criteria can be attributed to their search strategies that are well suited for the solution space involved in discrete time-cost-quality trade-off problems. The pareto-front for the test project is rather smooth and GA that relies on randomness is not able to perform as well as TS and SA that are able to guide their search downhill. SA falls behind TS while it waits for its Metropolis criterion to initiate the downhill search. In light of the results of the first and second stage tests, it can be argued that for discrete time-cost-quality trade-off analysis TS is superior in performance compared to SA and GA and is preferable.

The research presented herein has some limitations. First the results are based on a single test project. Therefore comparing performance of TS, SA and GA on large number of test problems would provide better insights on performance of combinatorial optimization algorithms. Second, the combinatorial optimization algorithms have different set of parameters and tuning of those parameters might have influenced the results. Therefore the results should be interpreted in the light of these limitations.

6. Conclusions

The research presented in this paper compares relative performance of TS, GA and SA in solving a test project with an 18 activity network. The relative performances of three combinatorial optimization algorithms were evaluated based on three criteria: (1) quality of the optimum solution computed by each combinatorial optimization algorithm, (2) number of candidate solutions required to find optimum solution by each combinatorial optimization algorithm and (3) computational time required to find

optimum solution by each combinatorial optimization algorithm. The results reveal that TS outperforms GA and SA on computational time and number of candidate solutions evaluated in finding optimal solution.

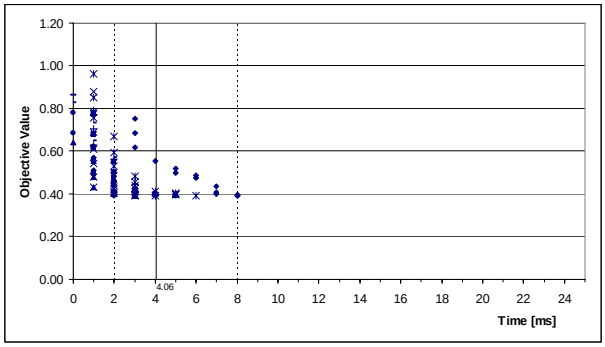

Figure 4a: Discovery of new best solutions over time for 10 TS runs

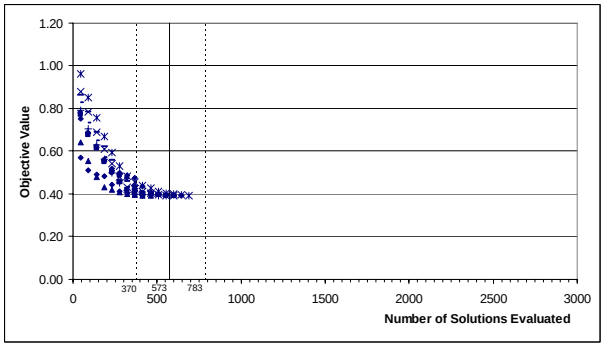

Figure 5a: Discovery of new best solutions among all candidate solution evaluations for 10 TS runs

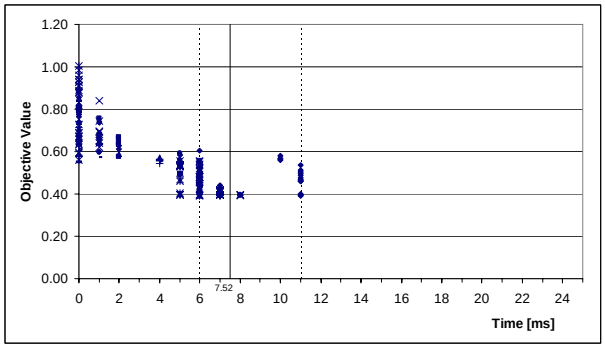

Figure 4b: Discovery of new best solutions over time for 10 SA runs

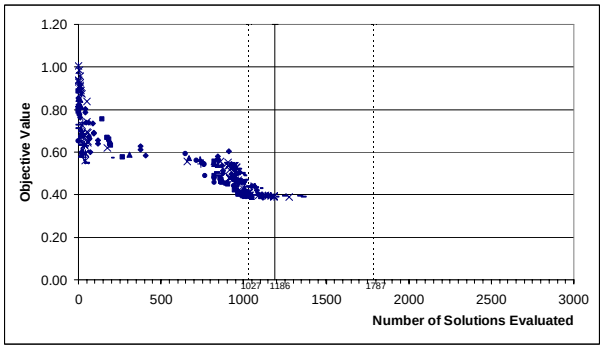

Figure 5b: Discovery of new best solutions among all candidate solution evaluations for 10 SA runs

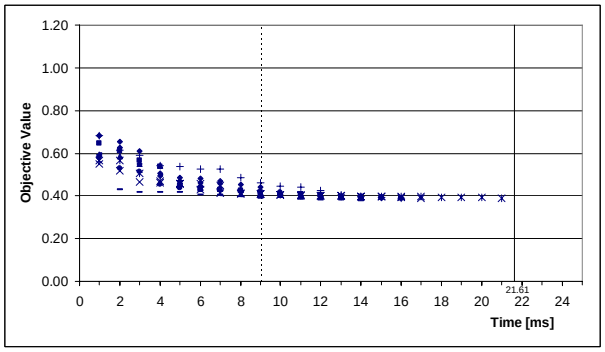

Figure 4c: Discovery of new best solutions over time for 10 GA runs

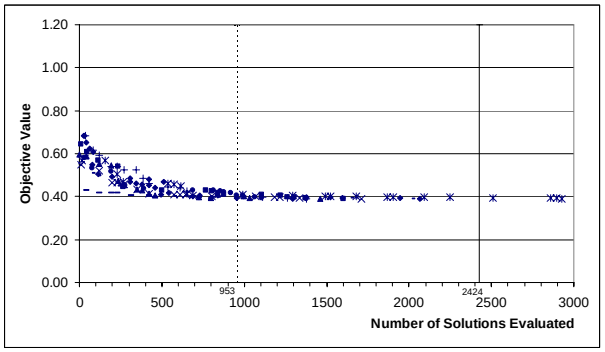

Figure 5c: Discovery of new best solutions among all candidate solution evaluations for 10 GA runs

7. References

Afshar, A., Kaveh A., and Shoghl, O. R. (2007). "Multi-objective optimization of time-cost-quality using multi-colony ant algorithm", *Asian Journal of Civil Engineering*, 8, 2, 113-124.

Arostegui, M. A., Kadipasaoglu, S. N., and Khumawala, B. M. (2006). "An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems", *International Journal of Production Economics*, 103, 2, 742-754.

Babu, A. J. G., and Suresh, N. (1996). "Project management with time, cost, and quality considerations", *European Journal of Operational Research*, 88,2, 320-327.

El-Rayes, K., and Kandil, A. (2005). "Time-cost-quality trade-off analysis for highway construction", *Journal of Construction Engineering and Management*, 131(4), 477-486.

Feng, C-W., Liu L., and Burns, S. A. (2000). "Stochastic construction time-cost trade-off analysis", *Journal of Computing in Civil Engineering*, 14, 2, 117-126.

Feng, C-W., Liu, L., and Burns, A. S. (1997). "Using genetic algorithms to solve construction time-cost trade-off problems", *Journal of Computing in Civil Engineering*, 11, 3, 184-189.

Hasan, M., Alkhamis, T., and Ali J. (2000). "A comparison between simulated annealing, genetic algorithm and tabu search methods for the unconstrained quadratic pseudo-boolean function", *Computers & Industrial Engineering*, 38, 3, 323-340.

Huang, Y-S., Deng, J-J., and Zhang, Y-Y. (2008). "Time-cost-quality tradeoff optimization in construction project based on modified ant colony algorithm", *International Conference on Machine Learning and Cybernetics*, 1031-1035.

Iranmanesh, H., Skandari, M. R., and Allahverdiloo, M. (2008). "Finding pareto optimal front for the multi-mode time, cost quality trade-off in project scheduling", *International Journal of Computer, Information, and Systems Science, and Engineering*, 2, 2 118-122.

Khang, D. B., and Myint, Y. M. (1999). "Time, cost and quality trade-off in project management: A case study", *International Journal of Project Management*, 17, 4, 249-256.

Kuang Y-P., and Ying X. (2006). "Construction time-cost trade-off analysis using ant colony optimization algorithm", *Management Science and Engineering, 2006. ICMSE '06. International Conference*, 2039-2044

Liu, L., Burns, S. and Feng. C. (1995). "Construction time-cost trade-off analysis using LP/IP", *Journal of Construction Engineering and Management*, 121, 4, 446-454.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953). "Equations of state calculations by fast computing machines", *Journal of Chemical Physics*, 21, 1087-1092.

Moselhi, O. (1993). "Schedule compression using the direct stiffness method", *Canadian Journal of Civil Engineering*, 20, 65-72.

Panagiotakopoulos, D. (1977). "Cost-time model for large CPM project networks", *Journal of the Construction Division*, 103, 201-211.

Pollack-Johnson, B., and Liberatore, M.J. (2006). "Incorporating quality considerations into project time/cost tradeoff analysis and decision making", *IEEE Transactions on Engineering Management*, 53, 4, 534-542.

Rahimi, M., and Iranmanesh, H. (2008). "Multi objective particle swarm optimization for a discrete time, cost and quality trade-off problem", *World Applied Sciences Journal*, 4 2, 270-276.

Sadegheih, A., and Drake P. R. 2008. "System network planning expansion using mathematical programming, genetic algorithms and tabu search", *Energy Conversion and Management*, 49, 6, 1557-1566.

Siemens, N. (1971). "A simple CPM time-cost trade-off algorithm", *Management Science*, 17, 6, 354-363.

Tareghian, H. R., and Taheri, S. H. (2006). "On the discrete time, cost and quality trade-off problem", *Applied Mathematics and Computation*, 181, 2, 1305-1312.

Vanhoucke, M. (2005). "New computational results for the discrete time-cost trade-off problem with time-switch constraints", *European Journal of Operational Research*, 165, 2, 359-374.

Xiong, Y., and Kuang, Y. (2008). "Applying an ant colony optimization algorithm-based multiobjective approach for time – cost trade-off", *Journal of Construction Engineering and Management*, 134, 2, 153-156.

Youssef, H., Sait M. S., and Adiche H. (2001). "Evolutionary algorithms, simulated annealing and tabu search: A comparative study", *Engineering Applications of Artificial Intelligence*, 14, 2, 2001, 167-181.