A Portfolio Approach to Algorithm Selection for Discrete Time-Cost Trade-off Problem

Santosh Mungle
Department of Systems and Industrial Engineering, University of Arizona, USA - 85721-0020

Abstract

It has been widely known that performance of algorithms for NP-Hard problems varies from instance to instance. This phenomenon has been observed, when we comprehensively studied multi-objective evolutionary algorithms (MOEAs) on a six benchmark instances of discrete time-cost trade-off problem (DTCTP). Instead of using single algorithm to solve DTCTP, we use a portfolio approach that takes multiple algorithms as its constituent. In this paper, we proposed portfolio comprising of four MOEAs, Non-dominated sorting genetic algorithm 2 (NSGA 2), the strength Pareto EA 2 (SPEA 2), Pareto archive evolutionary strategy (PAES) and Niched Pareto Genetic Algorithm 2 (NPGA 2) to solve DTCTP. The result shows that the portfolio approach is computationally fast and qualitatively superior than its constituent algorithms for all benchmark instances. Moreover, portfolio approach provides an insight in selecting the best algorithm for all instances of DTCTP.

Keywords: Time-cost trade-off, Algorithm portfolio, Multi-objective optimization.

1. Introduction

The discrete time-cost trade-off problem (DTCTP) is an important aspect in the scheduling of construction project. In project planning and scheduling, selection of appropriate resources including crew sizes, equipment, methods and technologies to carry out a project are challenging decisions. These decisions ultimately affect the duration and cost of the project. In practice, if project is running behind the scheduled plan, acceleration of activity becomes essential by allowing additional resources which ultimately incurs additional cost (De et al., 1997). DTCTP is a process to identify a suitable activity for speeding up, and deciding “by how much” so as to attain the best possible savings in both time and cost (Zheng et al., 2005). Such optimization process includes hidden trade-off relationship between time and cost. This will create complex
situation for decision maker’s to predict whether the total cost (i.e., the direct and indirect costs) would increase or decrease as a result of the schedule compression.

Over the last few decades, various approaches have been proposed to optimize time and cost. They can be broadly classified based on mathematical models or heuristics. Many researchers developed new mathematical models by using linear programming, integer programming or dynamic programming to solve the time-cost optimization problem (Robinson, 1975; Siemens, 1971; Goyal, 1975). Heuristic or mathematical models showed both strengths and weaknesses in solving DTCTP. The heuristic approaches select the activities to be shortened or expanded based on certain selection criteria, which do not guarantee optimal solutions. On the other hand, mathematical models require great computational effort and some approaches do not provide the optimal solution either. In addition, for large-scale networks (several hundred activities with discrete time-cost relationships, which are common for most construction projects), neither heuristic methods nor mathematical models can offer optimal solutions efficiently (Ozgen, 2009). DTCTP is known as strong NP-hard problem that failed to maintain pseudo-polynomial time guarantee for complex project network using exact solution algorithms (De et al., 1997).

Recently, many researchers proposed various metaheuristics such as genetic algorithm (GA), ant colony optimization (ACO), simulated annealing (SA) etc. to solve the DTCTP for large-scale networks to obtain near-optimal Pareto solution (Ozgen, 2009; Chassiakos and Sakellaropoulos, 2005; Feng and Liu, 1997; Pour et al., 2010; Zheng et al., 2004; Feng et al., 2000; Afshar et al., 2009; Mungle et al., 2013). However, it is difficult to inform in advance for decision maker which metaheuristic is the best to solve an instance of NP-hard optimization problem. Since a behavior of most metaheuristics for NP-hard problems are usually very difficult to characterize analytically and experimentally that further lead to algorithm selection problem (De et al., 1995). Previously, few attempts were made by researchers to solve algorithm selection problem for NP-hard problem by using experimental methods (Rice 1976; Guo and Hsu, 2007).

DTCTP is a multi-objective optimization problem that attempts to strike a delicate balance between project schedule time and cost. A multi-objective problem, in its real sense, requires the determination of a representative set of non-dominated or Pareto optimal solutions in the stipulated time frame to be presented to the decision maker. Last decade has witnessed a significant growth in the use of evolutionary meta-heuristics for complex multi-objective
optimization problems (Deb, 2001). Continuous improvements in the past few years have spectacularly reduced the time of response of these meta-heuristics without much depreciation in terms of solution quality. In particular, state of the art MOEAs like Non-dominated Sorting Genetic Algorithm II (NSGA-II) ( Deb et al., 2002), Strength Pareto Evolutionary Algorithm II (SPEA-II) (Zitzler et al., 2001) and Niched Pareto Genetic Algorithm II (NPGA-II) (Erickson et al., 2001) etc. have been extensively used and their robustness and capability to solve complex Multi-objective Combinatorial Optimization (MOCO) problems are well established.

The objective of this paper is to design a portfolio of four MOEAs namely NSGA-II ( Deb et al., 2002), SPEA-II (Zitzler et al., 2001), NPGA-II (Erickson et al., 2001) and PAES (Knowles and Corne, 1999), and its implementation on DTCTP. The four MOEAs are parallelly processed on two and four processors system. In addition, average quality metric is used to evaluate the quality of the approximated Pareto-optimal front. This aids in the decision making process to investigate the relative performance of various strategies embedded in the portfolio to get an insight into the solution quality of the Pareto-optimal front with the increasing problem size and complexity. A portfolio of algorithms is formally expressed as “A collection of different algorithms and or different copies of the same algorithm running on different processors” (Gomes and Selman, 2001; Huberman et al., 1997, Brown et al., 2003). Algorithm portfolio seeks to identify the best suited strategy and thereby amalgamate the performance of various algorithms thus enhancing the algorithm performance in dynamic environment. The effect can broadly be termed as performance maximizing superposition of algorithms where an algorithm performance is tailored by the performance of counterparts embedded in the portfolio. Finally, AHP method is adapted to determine the best portfolio case for DTCTP.

The remainder of the paper is organized as follows: Section 2 presents the mathematical formulation of DTCTP. Section 3 reviews some of the terminologies and techniques employed in the field of multi-objective optimization. Section 4 presents the computational study. Section 5 presents design of algorithm portfolio and its implementation. Section 6 includes the numerical results. Finally, section 7 concludes the paper with suggested future research works.

2. Mathematical Formulation

To model the DTCTP, we start with the standard assumption for modeling the project network: the project network has no cycles, the start activity (activity 0, a dummy activity) is the
only activity that is not an immediate successor of any activity, and the finish activity (activity \(N+1\), also a dummy activity) is the only activity that has no successors.

In time-cost optimization study, project completion time and implementation cost are considered as the main objectives in determining the best possible option for every activity in the project network.

2.1. Project completion time

The project completion time is determined by scheduling the activities using the assigned activity durations. It can be expressed as:

\[
\text{min} \left\{ \max_{L_k \in L} \sum_{i \in L_k} \sum_{j=1}^{n_i} t_{ij} x_{ij} \right\}
\]

Where,

\( N = \) Total number of activities in the project network
\( t_{ij} = \) Duration of option \( j \) of activity \( i \), for \( i = 1, ..., N, j = 1, ..., n_i \)
\( x_{ij} = \) If \( x_{ij} = 1 \) then activity \( i \) perform \( j^{th} \) option, while \( x_{ij} = 0 \) means not
\( L_k = \) Activity sequence of \( k^{th} \) path (\( ij_k = i1_k, i2_k, ..., in_k \))
\( L = \) Sets of all path of a network (\( 1, 2, ..., m \))
\( m = \) Number of paths in a project network
\( n_i = \) Number of time-cost option for each activity \( i \), for \( i = 1, ..., N \)

2.2. Project cost

The total project cost consists of direct and indirect cost. The direct cost is taken as the sum of direct cost of all activities within a project network. On the other hand, indirect cost is comprised of the expenditure on management during project implementation, which depends heavily upon the project duration, i.e., longer the duration, the higher the indirect cost. It can be expressed as:

\[
\text{min} \sum_{i=1}^{N} \sum_{j=1}^{n_i} C_{ij} x_{ij} + \text{min} \left\{ \max_{L_k \in L} \sum_{i \in L_k} \sum_{j=1}^{n_i} t_{ij} x_{ij} \right\} \times IC_{ij}
\]

Where,

\( C_{ij} = m_{ij} + d_{ij} R_{ij} \)
\[ C_{ij} = \text{Direct cost of option } j \text{ for activity } i, \text{ for } i = 1, \ldots, N, j = 1, \ldots, n_i \]
\[ IC_{ij} = \text{Indirect cost of option } j \text{ for activity } i, \text{ for } i = 1, \ldots, N, j = 1, \ldots, n_i \]
\[ m_{ij} = \text{Material cost of option } j \text{ for activity } i, \text{ for } i = 1, \ldots, N, j = 1, \ldots, n_i \]
\[ d_{ij} = \text{Duration of activity } i \text{ for option } j, \text{ for } i = 1, \ldots, N, j = 1, \ldots, n_i \]
\[ R_{ij} = \text{Daily cost rate in } \frac{\$}{\text{day}} \text{ for using option } j \text{ in activity } i, \text{ for } i = 1, \ldots, N, j = 1, \ldots, n_i \]

2.3. Constraints

2.3.1. Network logic constraints

The completion time of a project can be constrained by one of two methods (Crowston, 1970). The first method allows precedence constraints for each immediate preceding relationship in the project network. The second method allows one constraint for each path from the first activity to the last one in the project network. In the present formulation, the first method is adopted. The logical relationship between any two consecutive activities \( i \) and its immediate successor \( k \) is expressed as:

\[ SS_k \geq SF_i \quad \forall i = 1, \ldots, N, \forall k \in S_i \] (3)

Where,
\( SS_k \) = Scheduled start of immediate successor of activity \( i \)
\( SF_i \) = Scheduled finishing time of activity \( i \)

The \( SF_i \) equal scheduled start time (\( SS_i \)) plus its duration. The logical relationship constraint can be expressed using following equation, in which \( S_i \) is the set of successor activities to activity \( i \).

\[ SF_k \geq SF_i + \sum_{j=1}^{n_i} t_{ij} x_{ij} \quad \forall i = 1, \ldots, N, \forall k \in S_i \] (4)

2.3.2. Project completion constraints

In real life scenario, we imply a constraint on project completion time and cost to ensure that the both the objective will complete under restriction. These constraints are expressed by the following equations in which \( T_{\text{max}} \) is denoted as maximum allowable project duration and \( C_{\text{max}} \) is denoted as maximum allowable project cost.
\begin{align}
\min \left\{ \max_{L_k \in L} \sum_{i \in L_k} \sum_{j=1}^{n_i} t_{ij} x_{ij} \right\} < T_{\text{max}} \\
\min \sum_{i=1}^{N} \sum_{j=1}^{n_i} C_{ij} x_{ij} + \min \left\{ \max_{L_k \in L} \sum_{i \in L_k} \sum_{j=1}^{n_i} t_{ij} x_{ij} \right\} \times I_{C_{ij}} < C_{\text{max}}
\end{align}

3. Basic Concepts of Multi-objective Optimization

In this section, we briefly present the basic concepts of multi-objective optimization. Generally, multi-objective optimization problem can be formulated as follows:

\begin{align}
\text{maximize } \left\{ f_1(x) = z_1, ..., f_J(x) = z_J \right\} \\
\text{s.t. } x \in X
\end{align}

Where, solution \( x = [x_1, ..., x_I] \) is a vector of decision variables in a problem at hand and \( X \) is the set of feasible solutions available in a search space. If the variables in a problem at hand are discrete, the multi-objective optimization problem is called multiple-objective combinatorial optimization problems.

The image of a solution \( x \) in the objective space is a point

\[ z^x = [z_1^x, ..., z_J^x] = f(x), \text{ such that } z_j^x = f_j(x) \quad j = 1, ..., J \]

Point \( z^1 \) dominates \( z^2 \), \( z^1 > z^2 \), if \( \forall j \quad z_j^1 \geq z_j^2 \) and \( z_j^1 > z_j^2 \) for at least one \( j \). Solution \( x^1 \) dominates \( x^2 \) if the image of \( x^1 \) dominates the image of \( x^2 \). A solution \( x \in X \) is considered as Pareto-optimal if there is no \( x' \in X \) that dominates solution \( x \). A point being an image of a Pareto-optimal solution is known as non-dominated point. The set of all Pareto-optimal solution is called the Pareto-optimal set. The image of the Pareto-optimal set in the objective space is called the non-dominated set or Pareto front. An approximation of the non-dominated set is a set \( A \) of feasible points (and corresponding solutions) such that \( z^1, z^2 \in A \) such that \( z^1 > z^2 \).

The point \( z^* \) composed of the best attainable objective function value is called the ideal point.

\[ z_j^* = \max\{f_j(x) | x \in X\}, \quad j = 1, ..., J \]

An approximation of the ideal point based on set \( A \) is denoted by \( Z^{**}(A) \)

\[ z_j^{**}(A) = \max\{z_j | z \in A\}, \quad j = 1, ..., J \]
Finding the complete Pareto front is a very difficult task computationally which is attributed to the presence of a large numbers of sub-optimal Pareto front. Considering an existing memory limitation, determining the complete Pareto front becomes infeasible, and thus requires the Pareto front to be diverse covering maximum possible regions of it. Therefore, a multi-objective algorithm aims to obtain an approximation of the parent front (Jaszkiewicz, 2003).

In this paper, performance of algorithm portfolio is measured based on the quality of the obtained Pareto front, which is quantitatively evaluated using standard criteria or metrics. One such metric ‘Average Quality’ (AQ) (Jaszkiewicz, 2003) is utilized here. In past, the quality of the Pareto-optimal set was measured in terms of Tchebycheff function $S_\infty$ (Jaszkiewicz, 2003) and is defined in the following way:

$$S_\infty(z, z^0, \land) = \max_j \left\{ \lambda_j (z^0_j - z_j) \right\}
= \max_j \left\{ \lambda_j (z^0_j - f_j(x)) \right\}$$  \hspace{1cm} (11)

Where $z^0$ is a reference point, $\land = [\lambda_1, \ldots, \lambda_j]$ is a weight vector such that $\lambda_j \geq 0 \ \forall j$ and $\exists j|\lambda_j > 0$. Each weighted Tchebycheff scalarizing function has at least one global optimum belonging to the set of Pareto-optimal solutions. Note, however, that if the optimum is not unique, then some of the optima may be dominated, but must have at least one objective component equal to a Pareto-optimal solution. For each Pareto optimal solution $x$ there exist a weighted Tchebycheff scalarizing function $S$ such that $x$ is a global optimum of $S$.

However, this function tends to hide certain aspects regarding the solution set’s quality because poor performance with respect to proximity may get compensated by good performance in distribution of solutions. In order to overcome this limitation, diversity indicators of spread and space are added to the formulation of AQ metric.

Achievement scalarizing functions are defined in the following ways:

$$S_a(z, z^0, \land, \rho) = \max_j \left\{ \lambda_j (z^0_j - z_j) \right\} + \rho \sum_{j=1}^{j} \lambda_j (z^0_j - z_j)$$ \hspace{1cm} (12)

Where $\rho = \text{small positive number, } 0 < \rho << 1$. All global optima of each achievement scalarizing function belong to the set of Pareto-optimal solutions.
Weight vector that meets the following condition are called normalized weight vectors.

\[ \forall j \lambda_j \geq 0, \quad \sum_{j=1}^{J} \lambda_j = 1 \]  \hfill (13)

The AQ metric uses a representative sample \( \Psi_s \) of normalized weight vectors which is represented as:

\[ \Psi_s = \{ \Lambda = [\lambda_1, ..., \lambda_J] \in \Psi | \lambda_j \in \{0, \frac{1}{l}, \frac{2}{l}, ..., \frac{l-1}{l}, 1\} \} \]  \hfill (14)

Where,

- \( \Psi_s \) = Set of normalized vector
- \( l \) = Sampling parameter

We observe that \( \Psi_s \) contain \( \binom{l + J - 1}{J - 1} \) weight vectors. For each number of objectives \( J \), the size of \( \Psi_s \) should at least 50 i.e., sampling parameter \( l \) is set as the lowest value that assures \( |\Psi| \geq 50 \). Therefore \( l \) set to be 49 for \( J = 2 \). Each weight vector \( \Lambda \in \Psi_s \) defines an achievement scalarizing function \( S_a(z, z^0, \Lambda, \rho) \). All scalarizing functions defined by vectors from set \( \Psi_s \) constitute set \( S_a \) of functions.

In order to evaluate the quality of solution generated by algorithm portfolio, we first run the algorithm portfolio. In result, an approximation \( A \) of the whole non-dominated set is obtained. Then for each function \( S_a(z, z^0, \Lambda, \rho) \in S_a \) the best solution \( x \) on this function is selected from set \( A \). i.e., \( \forall z \in A \quad S_a(f^x, z^0, \Lambda, \rho) \leq S_a(z, z^0, \Lambda, \rho) \). Thus the average quality of solutions generated by the portfolio is

\[ AQ = \frac{\sum_{A \in \Psi_s} S_a(f^x, z^0, \Lambda, \rho)}{|\Psi_s|} \]  \hfill (15)

Where,

- \( z^0 \) = reference point which is an approximation of the ideal point
- \( X \) = best solution on function \( S_a(z, z^0, \Lambda, \rho) | x \in A \)
- \( \rho \) = parameter set as 0.01

For each test instance, an approximation of the ideal point was found by local optimization of individual objectives started from an initial solution. This approximation of the
ideal point is used as reference point in achievement scalarizing function \( S_a(z, z^0, \lambda, \rho) \). The objective ranges observed during local optimization of every individual objective is used to determine the range equalization factor which further helps to normalize the objective values before calculating scalarizing function.

The range equalization factors (Steur, 1986) are defined in the following ways:

\[
\Pi_j = \frac{1}{R_j}, \quad j = 1, ..., J
\]  

(16)

Where, \( R_j \) is the approximate range of objective function \( Z_j \) in the non-dominated set \( A \). Objective values multiplied by the range equalization factors are called normalized objective function values.

4. Computational Study

4.1. Benchmark instances of DTCTP

In this paper, data sets of DTCTP are characterized based on the number of activities, possible schedules and number of total paths in the project network. Table 1 shows the benchmark instances of DTCTP with their aforementioned characteristics. In this study, total six data sets of DTCTP from very large to very small size have been considered. Among them, problems 1, 2, 3, 5 and 6 have been adapted respectively from (Ozgen, 2009), (Chassiakos and Sakellaropoulos, 2005), (Feng and Liu, 1997), (Pour et al., 2010) and (Zheng et al., 2004), and problem 4 is hypothetically generated with maximum 4 and minimum 2 execution modes for project activities. The solution to these problems has been attempted by integrating them in an algorithm portfolio that works on the strategy of minimizing the risk in terms of computational cost and the solution quality obtained.

| Sr. No | Number of Activities | Possible Schedules | Number of Network Paths |
|--------|----------------------|--------------------|-------------------------|
| 1      | 63                   | \(1373\times10^{92}\) | 28                      |
| 2      | 29                   | \(8264\times10^6\)  | 46                      |
| 3      | 18                   | \(1494\times10^7\)  | 11                      |
| 4      | 14                   | \(2831\times10^4\)  | 11                      |
| 5      | 9                    | \(1500\times10^3\)  | 5                       |
| 6      | 7                    | 5569               | 3                       |
4.2. Multi-objective Evolutionary Algorithms (MOEAs)

In this paper, four establish MOEAs namely, NSGA-II (Deb et al., 2002), SPEA-II (Zitzler et al., 2001), NPGA-II (Erickson et al., 2001) and PAES (Knowles and Corne, 1999) is taken into consideration to design and implement the algorithm portfolio. The following discussion presents a brief overview of these four MOEAs.

4.2.1. Non-dominated Sorting Genetic Algorithm-II (NSGA-II)

It was proposed by Deb et al. (2002). This algorithm is a revised version of Non-dominated Sorting Genetic algorithm (NSGA) proposed by Srinivas and Deb (1995) which has been criticized mainly for its high computational complexity, non-elitism approach and a need for specifying a sharing parameter. NSGA was based on Genetic algorithm (GA), which utilizes several layers of classification of the individuals. In the slow sorting approach, in order to find the first non-dominated front each solution in the population is compared with every other to find all the members which are non-dominated compared to others. These individuals comprises first non-dominated front. To find the members of the next front, all the members of the previous fronts are left temporarily and the aforementioned procedure is repeated. This leads to a high $O(MN^3)$ computational complexity where $M$ is the number of objectives and $N$ is the population size.

However, NSGA-II followed a fast non-dominated sorting approach which requires $O(MN^2)$ comparisons. It also replaced the use of sharing function with the new crowded comparison approach that eliminates the need of any user defined parameter for maintaining diversity among population members. The other basic steps of NSGA-II are same as those of GA. Initially, a population double in size ($2N$) is randomly generated and is then sorted on the basis of non-dominance. Thereafter, the selection of best $N$ members from the population is carried out. Crossover and Mutation is performed in order to utilize these genetic operations to produce better offspring. The parents and offspring are then combined to form the initial population for the next generation and the aforementioned procedure is repeated. The algorithm finally provides a well distributed Pareto set of solutions. In the present paper, two point crossover and bit-flip mutation have been used.
4.2.2. **Strength Pareto Evolutionary Algorithm II (SPEA-II)**

It is a revised version of SPEA proposed by Zitzler and Thiele (2001). SPEA was developed on the basis of a previously carried comparative study and establish the multi-objective techniques commonly used in Evolutionary Algorithms (EAs) into a single meta-heuristic skeleton (Zitzler and Thiele, 1999). SPEA-II, an improved version of SPEA, utilizes an external memory for storing the non-dominated individuals. Each individual in the extended memory as well as in the current population is assigned a strength value based on the domination and density information. The rank value of an individual is determined by the summation of strength values of the individuals that dominate it. Density of each individual is estimated based on the $k^{th}$ nearest neighbor density estimation method. The final fitness value is then calculated as the sum of rank and density values. In addition, a hierarchical clustering based method (Morse, 1980) is adopted for the pruning of the external archive thereby maintaining diversity in the obtained Pareto front.

4.2.3. **Niched Pareto Genetic Algorithm-II (NPGA-II)**

Erickson et al. (2001) proposed a revised version of the NPGA (Horn et al., 1994) called the NPGA-II. This algorithm uses Pareto ranking but keeps tournament selection (solving ties through fitness sharing as in the original NPGA). In this case, no external memory is used and the elitist mechanism is similar to the one adopted by the NSGA-II. Niche counts in the NPGA-II are calculated using individuals in the partially filled next generation, rather than using the current generation. This is called continuously updated fitness sharing, and was proposed by Oei et al. (1991).

4.2.4. **Pareto Archived Evolution Strategy (PAES)**

This algorithm was introduced by Knowles and Corne (1999). PAES consists of a $(1 + 1)$ evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records the non-dominated solutions previously found. This archive is used as a reference set against which each mutated individual is being compared. Such a historical archive is the elitist mechanism adopted in PAES. However, an interesting aspect of this algorithm is the procedure used to maintain diversity which consists of a crowding procedure that divides objective space in a recursive manner. Each solution is placed in a certain grid location based on the values of its objectives (which are used as its “coordinates” or “geographical location”). A map of such grid is maintained, indicating the number of solutions
that reside in each grid location. Since the procedure is adaptive, no extra parameters are required (except for the number of divisions of the objective space). Next subsection develops the mathematical insights supporting the use of portfolios.

4.3. Computational Analysis

As we know that MOEAs do not always reach the optimal solution, even for long computation times. In addition, it is often impossible to obtain an analytical prediction of either the solution achievable within a given computation time or the time taken to find a solution of a given quality. The analysis of these measures is critical for the comparison between MOEAs. In literature, statistical analysis of these measures is known as “Univariate Model” (Chiarandini et al., 2007). More specifically, Univariate model is the one in which either the solution cost or run-time is taken into consideration while analyzing the various metaheuristics.

In this paper, our concern is run-time that means computation time is measured when a solution with a desired property is found. We assume that run-time corresponds to the number of iteration needed to achieve the desired solution quality.

The performance measure $X$ (run-time) of a metaheuristics on the instances can be described by a probability distribution function $f(x) = P[X = x]$ or equivalently by its cumulative (discrete) distribution function (CDF) $F(x) = P[X \leq x] = \sum_{x_i \leq x} p(x_i)$

In the computational experiments, we observe data $x_1, x_2, \ldots, x_n$ and then calculate the CDF. Once the CDF is known, parameter such as mean and variance of the iterations is calculated to compare the performance of each MOEA.

Initiation of the experiment takes place by testing the four MOEAs on all the benchmark instances for 50 trials and the results have been reported in Figures 1-6. In Figure 1-6, $X$-axis represents the iterations needed to satisfy the termination criteria and $Y$-axis represents the CDF value. The termination criteria for each MOEA set as the performance level within 10% of the best obtained AQ (average quality) metric value. The mean and variance of the number of iterations required by each algorithm for the six problems are provided in Table 2.
Figure 1: CDF of 4 MOEAs for Problem 1

Figure 2: CDF of 4 MOEAs for Problem 2
Figure 3: CDF of 4 MOEAs for Problem 3

Figure 4: CDF of 4 MOEAs for Problem 4
Figure 5: CDF of 4 MOEAs for Problem 5

Figure 6: CDF of 4 MOEAs for Problem 6
Table 2: Mean and Variance of the iterations for the six problems

| Problem  | NSGA-II Mean | NSGA-II Variance | SPEA-II Mean | SPEA-II Variance | PAES Mean | PAES Variance | NPGA-II Mean | NPGA-II Variance |
|----------|--------------|------------------|-------------|-----------------|-----------|---------------|-------------|-----------------|
| Problem 1| 1800         | 1679159          | 1583        | 1481943         | 1575      | 1452351       | 1999        | 1747775         |
| Problem 2| 1091         | 1028848          | 1252        | 1365382         | 1479      | 1179406       | 1392        | 1126270         |
| Problem 3| 1123         | 572831           | 992         | 581708          | 808       | 577827        | 1012        | 809507          |
| Problem 4| 642          | 321288           | 385         | 154736          | 340       | 109901        | 361         | 147748          |
| Problem 5| 155          | 34635            | 295         | 86506           | 231       | 51919         | 402         | 68844           |
| Problem 6| 13           | 147              | 21          | 400             | 15        | 237           | 26          | 1163            |

A statistical analysis in Table 2 shows that for problem 1, 3 and 4 PAES takes less number of iterations to reach the termination criteria compared to other three MOEAs. For problem 2 and 5, NSGA-II takes less number of iterations compared to PAES, NPGA-II and SPEA-II. In addition, it shows expected variations in the outcome with the changing problem size. Moreover, it is important to compare the number of success amongst the 50 trials for each MOEA to reach the termination criteria. Figures 1-6 reveals that the performance of the all four MOEAs remained competitive for the six benchmark instances considered. More specifically, for problem 1, 3 and 4 PAES perform well compared to other three MOEA. For problem 2 and 5, NSGA-II performs well compared to other three MOEAs.

5. Algorithm Portfolio

In this section, we present the design of algorithm portfolio and its implementation on DTCTP.

5.1. Algorithm Portfolio Design

The deviation in the performance of the four MOEAs for DTCTP is the key to portfolio realization. Making use of the obtained variations, different portfolios have been conceived and analysis has been carried out. These experiments were performed with portfolios containing 2 and 4 algorithms on 2 and 4 processor systems. The detailed list containing the information pertaining to the algorithms embedded in various portfolios is provided in Table 3.

Following criteria has been considered while designing these portfolios:

- Those algorithms were combined and placed in same portfolios which showed varied performance.
- Strategies utilizing external memory for storing non dominated particles are tied with the strategies that store such particles in the current population.
- Portfolios were designed keeping in mind the requirement to analyze the relative performance of various strategies utilized for maintaining diversity in the population.

### Table 3: Portfolio Design

| No. of Algorithms | Case | 2 Processors | 4 Processors |
|-------------------|------|--------------|--------------|
| 2                 | 1    | NSGA II      | PAES         |
|                   | 2    | NPGA-II      | SPEA II      |
|                   | 3    | SPEA II      | NSGA II      |
|                   | 4    | SPEA II      | PAES         |
| 4                 | 1    |              | NSGA II      |

Similarly, the relative advantage that could be obtained with various Pareto diversity maintaining strategies is also considered. Aforementioned criteria are clearly evident in the portfolio design utilized. The inferior performance of NPGA-II compared to other meta-heuristics lowers the expectancy of competitive performance by the portfolios containing it. In addition, the similarity of its operators with NSGA-II, and much better performance by NSGA-II, limits its utilization (as is evident from the lowest number of its portfolios) in further portfolio analysis.

### 5.2. Implementation of Portfolio Approach

In practice, one may implement the portfolio following a few steps. First, a set of portfolio cases for different processors and its constituent algorithm should be identified. In this paper, we developed three portfolio cases, namely, 2 Algorithm-2 Processor (2A-2P), 2 Algorithm-4 Processor (2A-4P) and 4 Algorithm-4 Processor (4A-4P) (See Table 3). We then identified the constituent algorithm for each of these portfolio cases, as given in Table 3. When choosing the constituent algorithm, an intuition is that they should be more or less complementary. The constituent algorithms are combined and placed in portfolio based on their performances on all benchmark problems. As will be shown in our numerical results interpretation, choosing complementary constituent algorithms leads to good performances with same algorithms exploit on the different parallel processors on the same benchmark problems. More specifically, the
constituent algorithms should not only employ different operators but also exhibit different behaviors on problem sets. For second steps, algorithm portfolio framework should be highly capable to accommodate existing algorithms. However, due to the fact that some existing algorithms might have their own configuration, they merit little bit attention when incorporated into portfolio framework. In the present portfolio, some algorithms such as NSGA-II are highly capable to provide good solution quality beside its more run time complexity. This helps respective portfolio terminates quickly if the status of solution has reached predefined termination criteria.

In this paper, the termination criteria for the each portfolio run was set to be the performance level within 10% of the best obtained AQ (average quality) metric value. Average quality (AQ) is used as standard criteria here to measure the quality of the obtained Pareto front of algorithm portfolio. More details of this issue are given in section 3. In final step of algorithm portfolio, number of iterations will be recorded after the termination criterion is satisfied. Number of iterations is taken as criteria to evaluate the computational efficiency of the each portfolio case.

Before running portfolio, part of our work is to allocate population size, generation size and other parameters values for each constituent algorithms of portfolio. Their values should be always same to evaluate the performances of each portfolio case.

6. Results and Discussion
In order to perform the analysis, six benchmark instances are considered with varying problem size and complexity. Simulations were performed with various combinations of algorithms and processor systems. The algorithm combinations embedded in different portfolio cases is given in Table 3. Various combinations of algorithms are implemented in parallel on 2 and 4 processor systems which are detailed in coming subsection. These combinations or portfolios are represented by a ‘/’ notation. For example, the symbol 1/1 represents 2 algorithm-2 processor portfolio where first algorithm is run on first processor whiles the other one on the second processor. Similarly, 4/0/0/0 represents 4 algorithms-4 processor portfolios in which four copies of the first algorithm (Table 3) were run on all the four processors. Each combination was evaluated on the basis of its performance averaged over 50 independent runs.
6.1. Experimental Runs

This subsection contains the experimental results obtained by considering various combinations utilized. Average number of iterations required by the portfolio to reach the desired quality level (10% of the best) of Pareto-optimal front is again taken as the performance criteria. Different combinations of algorithms to be run on the processors and utilized in the experiment are given as follows:

- **2 Processor System**: Only three possible combination of algorithms exist in this case *i.e.* [2/0], [1/1] and [0/2] which are thoroughly analyzed. All the four cases were simulated with the aforementioned algorithm combinations and the obtained results are detailed in Figure 7.

- **4 Processor System** The analysis with 4-Processor system is carried out in two parts viz. 2 algorithm cases and 4 algorithm cases. In case of two algorithms, the possible cases are [4/0], [3/1], [2/2], [1/3] and [0/4]. However, in 4 algorithm case, the numbers of possible combinations are extremely large. In order to alleviate the computational burden due to the large number of possible portfolio combinations, authors have restricted the number of combinations utilized in this study to 16. These selected portfolio cases are provided in Table 4 and the results obtained with two and four processors systems are shown in Figures 8 and 9 respectively.

| Case | 4A- 4P Case | Case | 4A- 4P |
|------|-------------|------|--------|
| 1    | 4/0/0/0    | 9    | 0/3/1/0|
| 2    | 3/1/0/0    | 10   | 0/4/0/0|
| 3    | 3/0/1/0    | 11   | 1/0/3/0|
| 4    | 3/0/0/1    | 12   | 1/2/0/1|
| 5    | 2/1/1/0    | 13   | 0/0/4/0|
| 6    | 2/1/0/1    | 14   | 1/1/0/2|
| 7    | 1/0/1/2    | 15   | 0/1/0/3|
| 8    | 0/2/1/1    | 16   | 0/0/0/4|
Figure 7: Results for different portfolio cases pertaining to two algorithm-two processor (2A-2P) system. The six rows present results for the six problems considered and the columns are corresponding to different algorithms selected. Y axis is scaled to number of iterations; X axis denotes the portfolio cases explored.
Figure 8: Result for different portfolio cases pertaining 2 algorithms– 4 processors (2A-4P) system. The six rows present results for six problems considered and the column are corresponding to different algorithm selected. Y axis is scaled to number of iterations; X axis denotes the portfolio cases explored.
6.2. Interpreting the Results

Having had the experimental results, the task to be accomplished is concerned to portfolio assessment. Generally processor availability for the parallel runs is limited in an organization; hence the selection strategy has to be executed separately for each of the two types of systems (2 Processor, 4 Processor) under consideration. The varying performance of different algorithms, and eventually the portfolios, poses enough challenge to select the best strategy among the instances explored in order to get the quality solution with minimum risk. Analytical Hierarchy Process (AHP) has long been utilized as a tool to decision makers for selecting the best alternative from the given alternatives in such situations. AHP employs hierarchical pairwise comparison to induce the weights of alternatives thorough their pairwise comparison.

This paper also addresses the selection of best portfolio problem from an AHP perspective. Each type of portfolio is recognized as an alternative for the particular processor system to which the portfolio belongs to and the results over six different problems are considered as of different attributes. First, a matrix $A$ is constructed which is defined as follows,
\[ A = \begin{pmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \]  

(17)

Where, the column represent attribute \( \mathcal{A} \), \( \mathcal{A} = 6 \) in the underlying case (corresponding to problem 1-6); row represent the alternatives (portfolio) explored; and \( a_{ij} \) represent the normalized the objective value for the experiment characterizing by set \((i, j)\), i.e.

\[
 a_{ij} = \frac{OBJ_{ij}}{\sum_j \{OBJ_{ij}\}} 
\]

(18)

where, \( OBJ_{ij} \) represent the objective value based on which the solution quality is to be measured. In this case, \( OBJ_{ij} \) symbolizes the average number of generation required by the particular test alternative.

Thereafter a priority matrix \( \Omega_{i,k}^{j} \) is calculated for each attribute \( i \), where

\[
 \Omega_{jk}^{i} = \frac{a_{ij}}{a_{ik}} 
\]

(19)

The associated weight vector \( W^{i} \) is then calculated for each attribute \( i \) by taking geometric mean for the row corresponding to matrix \( \Omega^{i} \)

\[
 W_{1x,\mathcal{A}}^{i} = \{w_{j}^{i}\} 
\]

\[
 = \sqrt[\mathcal{A}]{\prod_{k}^{\mathcal{A}} \Omega_{jk}^{i}} 
\]

(20)

The calculation of weight vector is followed by their normalization, thus, \( \mathcal{A} \) normalized priority vector \( PV_{j}^{i} \) are obtained as

\[
 PV_{j}^{i} = \frac{w_{j}^{i}}{\sum_{j=1}^{\mathcal{A}} w_{j}^{i}} 
\]

(21)
The weighted sum of six priority vectors is now obtained that gives the relative rank vector $RV$ for the alternatives. Here, priority weights are taken as $1/6$ to ensure equal weightage to all attributes. For the two processor system, 8 portfolio combinations (due to repeat of several portfolios) have been evaluated. Similarly for four processor systems 16 combinations corresponding to both 2 algorithm and 4 algorithm cases are evaluated.

The final rank vectors and ranks for different alternatives are provided in Tables 5, 6 and 7. As is evident, the aforementioned strategy provides a relative ranking between the various portfolio alternatives available. For the two algorithms with two processors system the portfolios comprising of algorithms NSGA-II and PAES characterized by (1/1) system is found to be the best. One of the possible reasons for this might be the complementary performance of both the algorithms on the problem scenario considered. As is evident from the individual performance profiles (Figures 1-6), in general, the performance of PAES was good for on all instances considered. On the contrary, the performance of NSGA-II was good for large and medium sized instances while considerably poor for the small sized problem. The results obtained with portfolios embedding NSGA-II are generally better thereby supporting our early assumptions for its increased number in the portfolios considered. It is also evident that NPGA-II, which performed poorly when considered individually, continued the show with the portfolios embedding it resulting in worst case performance.

**Table 5:** Priority vectors and corresponding rank for the different alternatives related with 2 Algorithms and 2 Processors case

| Algorithm selected | Portfolio | Priority | Rank |
|--------------------|-----------|----------|------|
| NSGA II - PAES     | 2/0       | 0.1666   | 3    |
|                    | 1/1       | 0.1623   | 1    |
|                    | 0/2       | 0.1723   | 5    |
| NPGA-II-SPEA II    | 2/0       | 0.1785   | 7    |
|                    | 1/1       | 0.1974   | 12   |
|                    | 0/2       | 0.1886   | 9    |
| SPEA II – NSGA II  | 2/0       | 0.1913   | 10   |
|                    | 1/1       | 0.1634   | 2    |
|                    | 0/2       | 0.1766   | 6    |
| SPEA II - PAES     | 2/0       | 0.1820   | 8    |
|                    | 1/1       | 0.1924   | 11   |
|                    | 0/2       | 0.1683   | 4    |
Table 6: Priority vectors and corresponding rank for the different alternatives related with 2 Algorithms and 4 Processors case

| Algorithm Portfolio | Priority | Rank |
|---------------------|----------|------|
| NSGAII - PAES       | 4/0 0.1712 | 4    |
|                     | 3/1 0.1667 | 3    |
|                     | 2/2 0.1631 | 1    |
|                     | 1/3 0.1784 | 7    |
|                     | 0/4 0.2464 | 6    |
| NPGAII - SPEA II    | 4/0 0.2185 | 14   |
|                     | 3/1 0.2213 | 15   |
|                     | 2/2 0.2313 | 20   |
|                     | 1/3 0.2286 | 19   |
|                     | 0/4 0.2256 | 17   |
| SPEA II - NSGA II   | 4/0 0.2234 | 16   |
|                     | 3/1 0.2133 | 13   |
|                     | 2/2 0.1934 | 10   |
|                     | 1/3 0.1833 | 8    |
|                     | 0/4 0.1634 | 2    |
| SPEA II - PAES      | 4/0 0.2266 | 18   |
|                     | 3/1 0.2076 | 12   |
|                     | 2/2 0.1986 | 11   |
|                     | 1/3 0.1867 | 9    |
|                     | 0/4 0.1767 | 5    |

For the two algorithms with four processor systems, the portfolio comprising of algorithms NSGA-II and PAES characterized by (2/2) system is found to be the best. From result, it is clear that running algorithm on parallel processor is also found to be the best strategy when harmonizing portfolio. Similarly for the four algorithms case, the portfolio comprising of algorithms NSGA-II and PAES characterized by (3/1/0/0) system is found to be the best suited strategy.

In fact, the finding is attributed to the element of randomness inherited in the search procedure. Randomness here does not mean uncertainty rather it refers to the ability of the heuristic to vary its performance based on the initial random seed. However, the reason can be deduced on the basis of performance profiles obtained with the single processor system (Figures 1-6).
### Table 7: Priority vectors and rank for the different alternatives related with 4 Algorithms and 4 Processors case

| Algorithm Portfolio | Priority | Rank |
|---------------------|----------|------|
| NSGA II – PAES - NPGA-II - SPEA II | 0/2/1/1 | 0.1789 | 10 |
|                      | 0/3/1/0 | 0.1766 | 9   |
|                      | 0/4/0/0 | **0.1616** | 3   |
|                      | 1/0/3/0 | 0.1833 | 13  |
|                      | 1/2/0/1 | 0.1746 | 8    |
|                      | 0/0/4/0 | 0.1889 | 14   |
|                      | 1/1/0/2 | 0.1813 | 11   |
|                      | 0/1/0/3 | 0.1867 | 13   |
|                      | 0/0/0/4 | 0.1937 | 16   |

As is evident, NSGA-II shows higher CDF in regions of less number of iterations thereby empirically suggestive of the aforementioned fact. Similar other interesting and supporting conclusions (for portfolio implementation) can be drawn from the analysis. However, before the implementation of this concept to the practical scenario a much detailed experimentation is required. This comprises of a much exhaustive list of algorithms and an augmented use of processors.

### 7. Conclusion and Future Research

This paper proposes a portfolio approach to algorithm selection for discrete time-cost trade-off problem in multi-objective environment. The proposed algorithm portfolio containing NSGA-II, PAES, NPGA-II and SPEA-II is used to minimize the associated risk related to the selection of algorithm. Six benchmark instances of DTCTP of varying dimension and complexities are included to analyze the performances of portfolio approach. The four algorithms are parallelly processed on two and four processors system. The results showed an insight, suggesting the application of portfolios to select the best algorithm for computationally expensive multi-
objective optimization problems. Furthermore, the result shows a considerable decrease in the computational time by the parallel processing of algorithms.

As perspective for future work, much detailed analysis is needed prior to any suggestion for algorithm portfolio usage. However, our preliminary results are encouraging new directions for implementation of algorithm portfolio on different multi-objective decision making problems.

References

Afshar, A., Ziaraty, A. K., Kaveh, A. and Sharifi, F. (2009). Non-dominated Archiving Multic colony Ant Algorithm in Time-Cost Trade-off Optimization, *Journal of Construction Engineering and Management*, 135 (7), 668-674.

Brown, K. L., Nudelman, E., Andrew G., McFadden J. and Shoham, Y. (2003). A Portfolio Approach to Algorithm Selection, *International Joint Conference in Artificial Intelligence*, 1542-1543.

Chassiakos, A. P. and Sakellaropoulos, S. P. (2005). Time Cost Optimization of Construction Projects with Generalized Activity Constraints, *Journal of Construction Engineering and Management*, 131(10), 1115-1124.

Chiarandini, M., Paquete, L., Preuss, M., and Ridge, E., (2007). Experiments on metaheuristics: Methodological overview and open issues. Technical report, Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark. URL http://ls11-www.cs.tu-dortmund.de/people/preuss/pages/publications/DMF-2007-03-003.pdf.

Crowston, W. B. (1970). Decision CPM: Network Reduction and Solution, *Operational Research Quarterly*, 21(4), 435-452.

De, P., Dunne, E. J., Ghosh, J. B., and Wells, C. E. (1995). The Discrete Time-Cost Trade-off Problem Revisited, *European Journal of Operation Research*, 81(2), 225-238.

De, P., Dunne, E. J., Ghosh, J. B. and Wells, C. E. (1997). Complexity of the Discrete Time-Cost Trade-off Problem for Project Networks, *Operations Research*, 45(2), 302-306.
Deb, K. (2001) Multi-objective Optimization using Evolutionary Algorithms, Chichester, U.K.: Wiley, New York.

Deb, K., Pratap, A., Agrawal S. and Meyarivan, T. (2002). A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6(2), 182-197.

Erickson, M., Mayer, A. and Horn, J. (2001). The Niched Pareto Genetic Algorithm 2 Applied to the design of Groundwater Remediation Systems, *Lecture Notes in Computer Science*, 1993(1), 681-695.

Feng, C. and Liu, L. (1997). Using Genetic Algorithm to Solve Construction Time-Cost Trade-off Problem, *Journal of Computing in Civil Engineering*, 11(3), 184-189.

Feng, C., Liang, L. and Burns, S. (2000). Stochastic Construction Time Cost Trade-off Analysis”, *Journal of Computing in Civil Engineering*, 14 (2), 117-126.

Gomes, C. and Selman, B. (2001). Algorithm Portfolios, *Artificial Intelligence*, 126(1), 43–62.

Goyal, S. (1975). A Simple CPM Time-Cost Trade-off Algorithm & quot, *Management Science*, 21(6), 718-722.

Guo, H. and Hsu, W. H. (2007). A Machine Learning Approach to Algorithm Selection for NP-Hard Optimization Problems: A Case Study on the MPE Problem, *Annals of Operations Research*, 156(1), 61-82.

Horn, J., Nafpliotis, N. and Goldberg, D. E. (1994). A Niched Pareto Genetic Algorithm for Multi-objective Optimization in *Proc. of the First IEEE Conference on Evolutionary Computation*, Z. Michalewicz, Ed. Piscataway, NJ: IEEE Press, pp. 82–87.

Huberman, B. A., Lukose, R. M. and Hogg, T. (1997). An Economics Approach to Hard Computational Problems, *Science*, 275(5296), 51–54.

Jaszkiewicz, A. (2003). Do Multiple-Objective Meta-heuristics Deliver on their Promises? A Computational Experiment on the Set-Covering Problem, *IEEE Transaction on Evolutionary Computation*, 7(2), 133-143.
Knowles, J. and Corne, D. (1999). The Pareto archived evolution strategy: A New Baseline Algorithm for Multi-objective Optimization, Proceedings of the 1999 Congress on Evolutionary Computation, Piscataway: New Jersey: IEEE Service Center, 98–105.

Morse, J. (1980). Reducing the Size of the Non-dominated Set: Pruning by Clustering, Computers & Operations Research, 7(1), 55-66.

Mungle, S., Benyoucef, L., Son, Y. J. and Tiwari, M. K. (2013). A fuzzy clustering-based genetic algorithm approach for time-cost-quality trade-off problems: A case study of highway construction project, Engineering Applications of Artificial Intelligence, 26(8), 1953–1966.

Oei, C. K., Goldberg, D. E. and Chang, S. J. (1991). Tournament Selection, Niching, and the Preservation of Diversity, IlliGAL Report No. 91011. Illinois Genetic Algorithms Laboratory. University of Illinois at Urbana-Champaign.

Ozgen, C. (2009). Optimization of Time-Cost Resource Trade-off Problem in Project Scheduling using Metaheuristic Algorithms, Doctoral Thesis Report, Middle East Technical University.

Pour, N., Modarres, N., Aryanejad, M. B. and Moghadam, R. D. (2010). The Discrete Time-Cost-Quality Trade-off using a Novel Hybrid Genetic Algorithm, Applied Mathematical Sciences, 4(42), 2081-2094.

Rice, J. (1976). The Algorithm Selection Problem, Advances in Computers, 15(1), 65-118.

Robinson, D. (1975). A Dynamic Programming Solution to Time-Cost Trade-off for CPM, Management Science, 22(2), 158-166.

Siemens, N. (1971). A Simple CPM Time-Cost Trade-off Algorithm, Management Science, 18(3), 354-363.

Srinivas, N. and Deb, K. (1995). Multi-objective Optimization using Non-dominated Sorting Genetic Algorithms, IEEE Transactions on Evolutionary Computation, 2(3), 221-248.

Steur, R. (1986). Multiple Criteria Optimization: Theory, Computation and Application, Wiley: New York.
Zitzler, E. and Thiele, L. (1999). Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation*, 3(4), 257-271.

Zitzler, E., Laumanns, M. and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Swiss Federal Institute of Technology, Lausanne, Switzerland, Tech. Rep. IK-Rep.103.

Zheng, D., Thomas, S. and Kumaraswamy, M. (2004). Applying Genetic Algorithm-Based Multi-objective Approach for Time-Cost Optimization, *Journal of Construction Engineering and Management*, 130(2), 168-176.

Zheng, D., Thomas S. and Kumaraswamy M. (2005). Applying Pareto Ranking and Niche Formation to Genetic Algorithm-Base Multi-objective Time-Cost Optimization, *Journal of Construction Engineering and Management*, 131(1), 81-91.