Journal of Soft Computing in Civil Engineering 3-2 (2019) 16-29

Contents lists available at SCCE

Journal of Soft Computing in Civil Engineering

Journal homepage: http://www.jsoftcivil.com/

Evolutionary Algorithm Performance Evaluation in Project Time-Cost Optimization

A.P. Chassiakos[1*] and G. Rempis[2]

1. Associate Professor, Department of Civil Engineering, University of Patras, Patras, Greece
2. Civil Engineer, Graduate, Department of Civil Engineering, University of Patras, Patras, Greece

Corresponding author: *a.chassiakos@upatras.gr*

http://dx.doi.org/10.22115/SCCE.2019.155434.1091

ARTICLE INFO

Article history:
Received: 08 November 2018
Revised: 24 June 2019
Accepted: 24 June 2019

Keywords:
Evolutionary algorithm;
Optimization;
Project scheduling;
Time-cost trade-off;
Construction management.

ABSTRACT

The time-cost trade-off problem pertains to the assessment of the best method of activity construction so that a project is completed within a given deadline and at least cost. Although several evolutionary-type of algorithms have been reported over the last two decades to solve this NP-hard combinatorial problem, there are not many comparative studies independently evaluating several methods. Such studies can provide support to project managers regarding the selection of the appropriate method. The objective of this work is to comparatively evaluate the performance potential of a number of evolutionary algorithms, each one with its own variations, for the time-cost trade-off problem. The evaluation is based on two measures of effectiveness, the solution quality (accuracy) and the processing time to obtain the solution. The solution is sought via a general purpose commercial optimization software without much interference in algorithm parameter setting and fine-tuning in an attempt to follow the anticipated project manager approach. The investigation has been based on case studies from the literature with varying project size and characteristics. Results indicate that certain structures of genetic algorithms, particle swarm optimization, and differential evolution method present the best performance.

1. Introduction

The time-cost trade-off (TCT) problem aims to determine the best method of activity execution so that the project finishes with the least cost and within any given completion deadline. The problem has been extensively studied for more than five decades and has been recognized as a particularly difficult combinatorial problem. Due to the problem size, which exponentially increases with project size, evolutionary algorithms have been implemented and tested in recent years for its solution. Several studies have been reported in the literature suggesting different methods, each one depending on specific assumptions and parameter settings. In real practice, however, the project manager does not have the ability to select the "best" method or perform fine-tuning of algorithm parameters. Instead, it is more likely to use some commercial software that implements such kind of algorithms.

The time-cost trade-off problem appears in two complementary forms. The first aims at developing the optimal project time-cost curve which results from assessing the minimum cost schedule at any feasible project duration. This curve provides the best schedule solution if a specific project completion deadline is set. While the above curve results solely from activity direct costs, if other indirect project costs, such as general project expenses, external consultancy fees, opportunity costs from late project completion, etc., are considered in addition, the total cost curve is developed by summing up direct and indirect costs along different project durations. This curve exhibits a minimum value at a particular project duration that is known as the optimal project duration.

The time-cost trade-off problem is considered as a NP-hard combinatorial problem and has been studied since 1960s. Proposed solution schemes have employed both exact and approximate methods. In the first category, linear programming (LP), integer programming (IP), or dynamic programming (DP) methods have been used. Because of certain limitations of these methods (especially in problems with large size), approximate methods have been developed which may converge faster to a solution although they do not guarantee that this solution is optimal. In recent years, the research on approximate methods has mainly focused on evolutionary-based algorithms (within the framework of metaheuristics), such as genetic algorithms (GA), ant-colony optimization (ACO) or particle swarm optimization (SPO) algorithms.

Among several mathematical programming methods, one can observe the work by Liu et al. [1] who proposed an LP/IP hybrid method and found that the required computational effort is much less than that of IP-alone based methods. In order to better simulate construction project characteristics, Sakellaropoulos and Chassiakos [2] presented a mixed integer-linear programming (MILP) method which aims to model generalized precedence relations among activities, activity planning and time constraints, as well as bonuses/penalties for early/late project completion respectively. An extension of this work was presented by Klansek and Psunder [3] for the nonlinear discrete time-cost trade-off problem. Other methods have used dynamic programming and network decomposition in which a project network is decomposed into several sub-networks, which are separately scheduled and finally put together (De et al. [4],

Akkan et al. [5], Hazir et al. [6]). The decomposition methods reduce the computational effort but they do not guarantee finding the optimal solution.

While exact methods can work satisfactorily in small networks, their effectiveness and efficiency drops as the project size and complexity rises. In this case, approximate algorithms can perform more efficiently even if they cannot guarantee the assessment of the optimal solution in any case. Among evolutionary-based algorithms, genetic algorithms were tested first. Feng et al. [7] presented such a method to develop the time-cost curve (Pareto front) via a multi-objective model which included both time and cost as distinct objectives. The method was evaluated on an 18-activity project network and the results were compared to those of a full enumeration analysis indicating that the algorithm could find almost all optimal values along the curve searching only a small part of the solution space. Hegazy [8] developed a practical problem for TCT optimization using the principle of genetic algorithms (GAs) and implementing it as a Visual Basic Application (VBA) macro program. Li and Love [9] presented certain improvements on existing GA methods in order to reduce computation times and increase algorithm effectiveness. Later, Li et al [10] proposed the integration of a machine learning method with GA with the former being used for the generation of quadratic time-cost curves from historical data in order to generalize the linear time-cost relationships. Zeng et al. ([11], [12]) presented a multi-objective model which integrates adaptive weights derived from previous generations and induces a search pressure toward an ideal point.

Besides GA methods, other evolutionary-type of algorithms have also been used. Yang [13] employed particle swarm optimization and adopted an elite archiving scheme to store non-dominated solutions while using members of the archive to direct further search. Zhang and Li [14] developed a multi-objective particle swarm optimization that adopts a combined scheme for determining the global best of each particle where the candidate solutions are represented through the multidimensional particles. Ng and Zhang [15] employed ant-colony optimization (ACO) to solve the multi-objective time-cost optimization problem while Xiong and Kuang [16] combined ACO with the modified adaptive weight approach to find the optimal solutions and define the Pareto front. Afshar et al [17] presented a multi-colony ant algorithm as an attractive alternative to the single ant colony optimization algorithm. Hybrid strategies have also been proposed and, among them, Sonmez and Bettemir [18] developed a hybrid model using GA and simulated annealing (SA) which is used to improve the hill-climbing ability of the GA.

While typical research efforts focus on examining certain methods and attempting to optimize their use through structure and parameter tuning, there is a scarcity of comparative studies which include several algorithms and formulations. One such study by Elbeltagi et al. [19] compared five evolutionary-based algorithms (genetic algorithms, memetic algorithms, particle swarm, ant-colony systems, and shuffled frog leaping) and, based on results from one test problem [7], concluded that the PSO method performs better than other algorithms in terms of success rate and solution quality, while being second best in terms of processing time. The lack of comparative studies may be a serious shortcoming in real practice where the project manager may not have the ability or willingness to search for the “best” method or perform fine-tuning of

algorithm structure and parameters. Instead, it is more likely that he/she will utilize some common use commercial software that implements such kind of optimization algorithms.

The objective of this work is to investigate the performance potential of several evolutionary algorithms for the time-cost trade-off problem. In particular, a number of existing algorithms, each one with its own variations, are comparatively evaluated. The evaluation is based on two measures of effectiveness, the solution quality (success rate and average of best solutions found under multiple algorithm applications) and processing time (average run time under multiple attempts). The solution is sought via a general purpose commercial optimization software with little interference in algorithm parameter setting and fine-tuning following more or less the expected project manager approach. The investigation has been done on case study projects from the literature with varying size and characteristics and with the exact solutions to be known. Numerical results and discussion is presented in the following sections.

2. Problem and Algorithm Description

The mathematical formultaion that describes the time-cost trade-off and minimum cost project duration problems has an objective function of the following form:

$$minTC = \sum_{i \in A} c_{ij} + c_0 f_0 \tag{1}$$

where *TC* is the total cost; *i* is the activity indicator; *A* is the group of project activities; *j* is the indicator of the activity method of construction; c_{ij} is the direct cost of activity *i* corresponding to method *j* and it directly related to the corresponding activity duration d_{ij}; f_0 is the project finish time; and c_0 is the project indirect cost per time unit. The first term of the objective function represents the project direct cost and the second one reflects the indirect cost. If indirect cost is not considered (i.e., c_0=0), the problem reduces to the time-cost trade-off problem. The problem variables are the activity durations d_{ij} while the constraints refer to: (a) project network structure and activity precedence relations, (b) activity duration-cost pairs for alternative construction methods (c) potential project deadline, (g) other time constraints for particular activities or sub-projects (e.g., “start no earlier than”).

Five evolutionary-based algorithms are considered in the comparative evaluation: hill climbing, genetic algorithms, particle swarm, differential evolution, and artificial bee colony. Alternative formulations are examined for some of these methods. A main concern when selecting methods for evaluation was to remain within a single developer of the corresponding software as this may guarantee a similar level of development sophistication and robustness of individual methods, reducing thus the possibility for bias existence in the evaluation results that is due to software capability discrepancies. In this regard, the XLOptimizer software [20], a generic optimization tool which implements customizable metaheuristic algorithms and is compatible with Microsoft Excel, was used. A short description of the examined algorithms and formulations are given below.

2.1. Hill Climber

Hill Climber is a local optimizer that can be applied to a binary chromosome [21]. It starts from a random individual chromosome and performs repetitive mutations of the chromosome bits in the form of flipping between 0 and 1 along the chromosome length. At each step, the objective value of the mutant is compared to that of the currently best chromosome and, if better, it becomes the currently best chromosome. The process ends when a full mutation loop along the chromosome length is completed without any improvement in the objective value.

2.2. Genetic Algorithms

Genetic Algorithms represent perhaps the first attempt to apply evolutionary algorithms for problem optimization. They progressively develop better and better solutions based on previous solutions that have been examined in the evolution process. Starting from an initial random population of candidate solutions with individuals representing distinct problem solutions and genes describing the problem parameters, new solutions evolve, generation by generation, using techniques such as selection, mutation and crossover. In each generation, the fitness of every individual (the objective function value associated with the individual) is evaluated. Next, multiple individuals are stochastically selected from the current population (considering fitness level) which are recombined (through crossover operation) or altered (through mutation) to produce offspring and form a new population by keeping the old and new individuals with the best fitness values within the population (retaining a constant size of it). Algorithm termination criteria may be the consideration of a given number of generations, the realization of a specific number of generations without an observable improvement of the fitness value or the achievement of a satisfactory fitness level (if this is known).

There are four alternative formulations of the genetic algorithms in XLOptimizer:

- The standard version works with predefined parameter settings regarding the population size, number of generation, crossover and mutation rate. A main disadvantage is that often terminates prematurely (long before reaching the best value) and, for this reason, its results are not presented in this study.
- In the custom version, the user can interfere and modify the genetic algorithm settings to fine-tune with the problem examined.
- The saw tooth GA (STGA) adopts a varying population size approach resembling a saw-tooth scheme [22]. In particular, the population size, starting from an initial value, is linearly decreased along generations. When it reaches a minimum value, the size is restored to its original value by the addition of randomly generated individuals.
- The micro GA (μGA) is a variant which utilizes a small population size of candidate solutions (typically 5-10 individuals) to reduce the computational effort for fitness value evaluations and memory storage requirement while the evolution is performed with crossover only [23]. At some point that the population tends to become homogeneous, a restart of the population is performed by keeping the best individuals of the existing population and replacing the rest by new randomly generated individuals. Population restarting several times during the process

allows effective exploration of the search space and prevents premature convergence to a local minimum.

2.3. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively improving a candidate solution with regard to a given measure of quality [24]. A initial population ("swarm") with random solutions is developed and PSO members of the swarm ("particles") move around in the search-space according to their current position and velocity. The velocity represents a progressive adjustment of particle position towards its own personal best position and the swarm's best position. This iterative process is expected to ultimately direct the swarm toward the best problem solution.

Besides the above standard PSO algorithm (SPSO), an enhanced formulation (EPSO) incorporates particle velocity restriction at the latest stages of the process to avoid overshooting (i.e., going back and forth around the optimal solution without the capacity to local convergence) which is a typical problem of PSO algorithms.

2.4. Differential Evolution

Differential Evolution (DE) is a relatively new stochastic method that optimizes a problem by maintaining a population of candidate solutions (called “agents”) subjected to iterations of recombination, evaluation, and selection while trying to improve a candidate solution with regard to a given measure of quality [25]. The agents move around in the search-space developing new agents by combining the positions of existing agents from the population. In particular, new candidate solutions are based on the weighted difference between two randomly selected population members added to a third population member. This perturbs population members relative to the spread of the broader population while the perturbation effect self-organizes the sampling of the problem space, bounding it to known areas of interest. In the iterative process, improved agent positions (based on fitness or score values) are accepted and become part of the population (replacing worse solutions), otherwise they are discarded. The process is repeated until a satisfactory solution is eventually revealed.

2.5. Artificial Bee Colony

Artificial Bee Colony (ABC) is a stochastic swarm intelligence algorithm inspired by the foraging behavior of honey bees [26]. In ABC algorithm, the food source position represents a possible solution of the optimization problem and the nectar amount of the food source is associated with the quality (fitness value) of the corresponding solution. Three groups of (artificial) bees are used in the process: employed bees, onlookers, and scouts. Employed bees search for food around the food source in their memory and, when they find a new food source position with higher nectar amount than that of the previous source, they memorize the new source position and abandon the old one. Further, they communicate this information to the onlookers which evaluate the nectar information taken from all employed bees and then choose a food source depending on the nectar amount of sources. The sources abandoned are replaced by

new sources, randomly produced by scouts, i.e., a few employed bees which have abandoned their food sources and search for new ones.

3. Algorithm Performance Evaluation

Three benchmark problems from the time-cost optimization literature have been used for the comparative algorithm evaluation, in particular, a 7-activity project described in [1], an 18-activity project presented in [7] and a 29-activity project analyzed in [2]. The first two cases solve the time-cost trade-off problem considering only the project direct cost with best cost solution to be 107,500 at a 78-day duration deadline and 106,270 at a 110-day duration deadline respectively; the third problem examines the optimal project duration problem considering, in addition, indirect cost as well as penalty/bonus parameters leading to a best cost solution of 45,500 at an 75-day project completion. Further, composite networks, including two single 18-activity projects set in series or in parallel, have been structured and tested within the evaluation effort. To obtain an indication of the problem size, the full enumeration of possible solutions adds up to 4,860 alternative schedules for the 7-activity network, 5.9×10^{9} alternatives for the 18-activity project, 8.3×10^{9} alternatives for the 29-activity project, and 3.5×10^{19} alternatives for each composite network consisting of two single 18-activity projects.

To obtain a robust indication of the algorithm merits, ten trial runs were performed for each problem and solution method (five for the composite projects). The performance of the compared algorithms was evaluated based on two criteria, the solution quality and the required processing time to reach the solution. The solution quality is determined by the success rate (i.e., the number of trials that led to the known target value of the objective function) and the average value of the objective function obtained in all trials. The processing time is recorded with its mean value and standard deviation among several trials.

The run time values are considered in a relative scale rather than in absolute terms. There are two reasons for such consideration. First, the computation time depends on the computer characteristics and lower time is expected with high-performance computing. In this study, all experiments took place on a laptop machine with Intel Core i5-460M processor, 2.5 GHz and 4 MB RAM (a rather typical configuration for the needs of a project manager). Second, from the project manager perspective, the main analysis goal may be to obtain a near-optimal solution at a reasonable computation time rather than waiting too long for a slightly better solution. In this regard, two types of output were developed. The algorithms were allowed to run as long as they were getting better values of the objective function (with a user-specified run time limit of half an hour for the three simple case studies and one hour for the two composite problems, ranges that may be barely acceptable for such project sizes). The run times that are reported in the result tables below correspond to the last best value found within the above run time limits. In addition, all intermediate solution upgrades were recorded within the calculation process and computational time scale. To get the whole picture, the algorithm performance is then illustrated with graphs showing the objective value evolution in time for representative cases.

The case study results are analytically presented in Tables 1 to 5 and graphically in Figures 1 and 2. Several indicative outcomes can be reported from the comparative evaluation. In particular:

Table 1.
Evaluation results for the 7-activity project (best solution 107,500).

Method	Success rate	Average solution	Deviation from best (%)	Average time (sec)	STD time (sec)
Hill Climber	4/10	110,190	2.5	1.1	0.3
GA	10/10	107,500	0	13	3
Saw Tooth GA	10/10	107,500	0	17	6
Micro GA	10/10	107,500	0	9	6
SPSO	10/10	107,500	0	7	4
EPSO	10/10	107,500	0	5	2
DE	10/10	107,500	0	8	3
ABC	10/10	107,500	0	22	11

Table 2.
Evaluation results for the 18-activity project (best solution 106,270).

Method	Success rate	Average solution	Deviation from best (%)	Average time (sec)	STD time (sec)
Hill Climber	10/10	106,270	0	6	1
GA	10/10	106,270	0	157	59
Saw Tooth GA	1/10	106,534	0.25	507	164
Micro GA	10/10	106,270	0	106	42
SPSO	6/10	106,505	0.22	74	21
EPSO	10/10	106,270	0	42	17
DE	10/10	106,270	0	64	9
ABC	10/10	106,270	0	180	30

Table 3.
Evaluation results for the 29-activity project (best solution 45,500).

Method	Success rate	Average solution	Deviation from best (%)	Average time (sec)	STD time (sec)
Hill Climber	2/10	45,573	0.16	5.5	0.5
GA	10/10	45,500	0	224	91
Saw Tooth GA	6/10	45,504	0.01	793	176
Micro GA	10/10	45,500	0	182	96
SPSO	1/10	45,608	0.24	86	32
EPSO	9/10	45,502	0.004	161	42
DE	10/10	45,500	0	85	10
ABC	10/10	45,500	0	405	71

The Hill Climber method is by far the fastest method in reaching its own best solution but it is generally unreliable in terms of optimization effectiveness (solution quality) since it presents a rather random performance with regard to problem size and other characteristics. In particular,

although in many cases the method reaches the absolute best, it cannot guarantee the best solution finding even in small-size problems where one expects higher performance than in larger projects. In addition, the method presents the highest deviations from the best solution among all examined methods. Nevertheless, its response time makes the method preferable for a fast solution approximation (at least in problems of these sizes).

Table 4.
Evaluation results for two 18-activity projects in series (best solution 212,540).

Method	Success rate	Average solution	Deviation from best (%)	Average time (sec)	STD time (sec)
Hill Climber	5/5	212,540	0	12	2
GA	2/5	212,579	0.02	2394	647
Saw Tooth GA	0/5	213,982	0.68	3166	329
Micro GA	5/5	212,540	0	1066	269
SPSO	0/5	214,010	0.69	422	45
EPSO	5/5	212,540	0	242	80
DE	5/5	212,540	0	214	30
ABC	5/5	212,540	0	864	53

Table 5.
Evaluation results for two 18-activity projects in parallel (best solution 212,540).

Method	Success rate	Average solution	Deviation from best (%)	Average time (sec)	STD time (sec)
Hill Climber	1/5	229,303	7.89	16	3
GA	3/5	212,798	0.12	2227	647
Saw Tooth GA	0/5	215,466	1.38	3416	173
Micro GA	2/5	213,525	0.46	2396	274
SPSO	0/5	221,998	4.45	50	5
EPSO	5/5	212,540	0	221	15
DE	5/5	212,540	0	226	24
ABC	0/5	220,454	3.72	1761	916

Among Genetic Algorithms, the custom GA has reached the best solutions in most cases examined and at competitive computational speeds. The Saw Tooth GA presents deteriorating performance in terms of solution accuracy and run time with the increase of the problem size (the method has consistently found the optimal solution only in the 7-activity problem). The micro GA has converged to the best solution in the majority of the cases with run time slightly lower in general than those of the custom GA.

The standard PSO algorithm (SPSO) performs rather poorly compared to other methods both in terms of solution accuracy and computational effectiveness with both indices to considerably deteriorate with the problem size. The enhanced PSO algorithm (EPSO), on the other hand, presents one of the best performances among the examined methods. In particular, the algorithm

has consistently found the best solutions in almost all case studies and attempts while its run time is comparable to the best ones (excluding the fast-running Hill Climber algorithm). The superior performance of the algorithm may be attributed to its capability to better manipulate the particle velocities at the final stages prior to convergence.

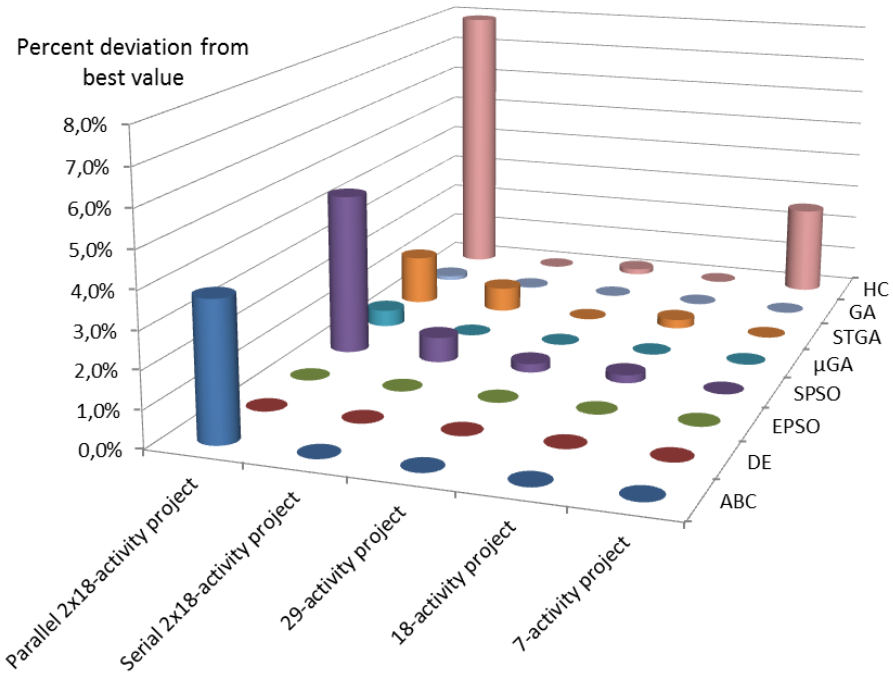

Fig 1. Evaluation results in terms of solution quality.

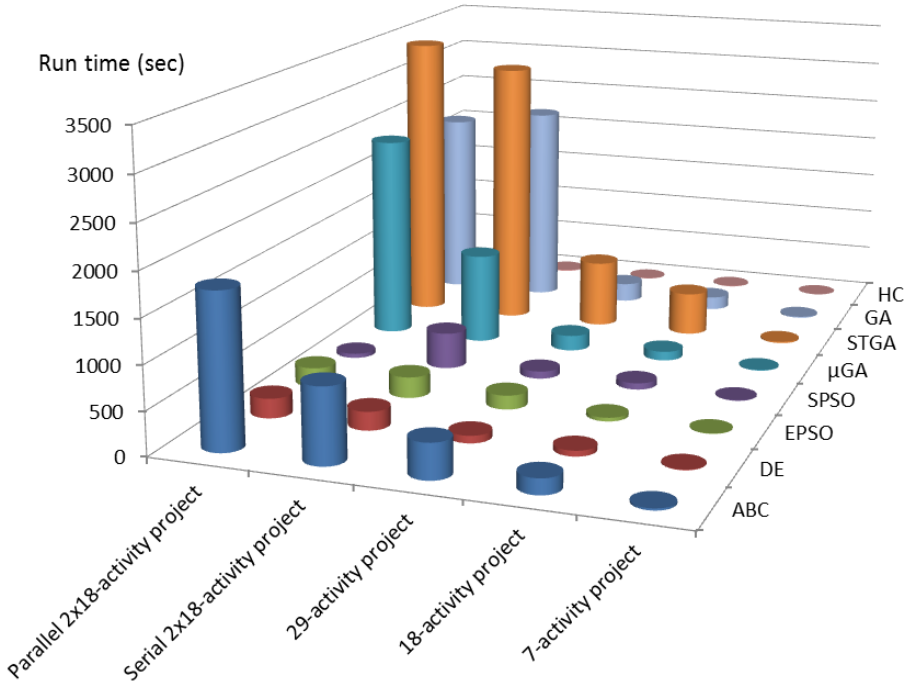

Fig 2. Evaluation results in terms of time requirement.

The Differential Evolution (DE) method exhibits one of the most competitive performances among all algorithms in terms of both solution quality and computational requirements. The algorithm has successfully reached the best solution in all case studies and repetitive attempts while its run times are among the lowest ones compared to other methods (with the exception of the Hill Climber method).

Finally, the Artificial Bee Colony (ABC) algorithm performs generally well in finding the optimal solution in almost all cases (with some degradation at larger problems), however, its computational efficiency is rather moderate compared to more effective methods, such as EPSO or DE.

In order to obtain a global perspective of method performance, Figures 3 and 4 illustrate the best performing algorithms with regard to the objective function (cost) value and run time value for the cases of the 18-activity and 29-activity networks. The point clouds for each method have been developed from selected (representative) result points of the multiple trials performed.

The schematic representation of the comparative evaluation indicates a slight outperformance of the enhanced PSO (EPSO) followed by the Differential Evolution (DE) and the micro GA methods. The outperformance primarily refers to the processing time that allows these methods to converge to their best solution faster than other methods. The custom GA presents a slower response in comparison to previous methods, while the ABC method, although ultimately reach the best solution (in most cases), generally takes more time than other methods to converge.

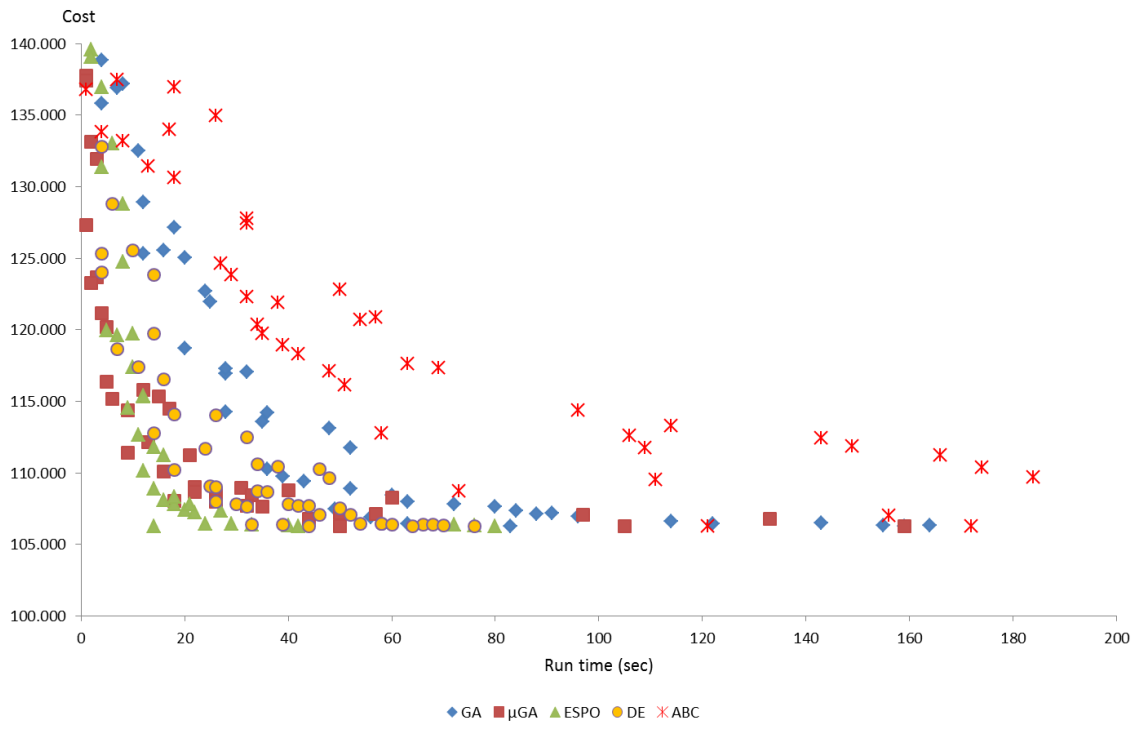

Fig 3. Evaluation results for best performing algorithms (18-activity network).

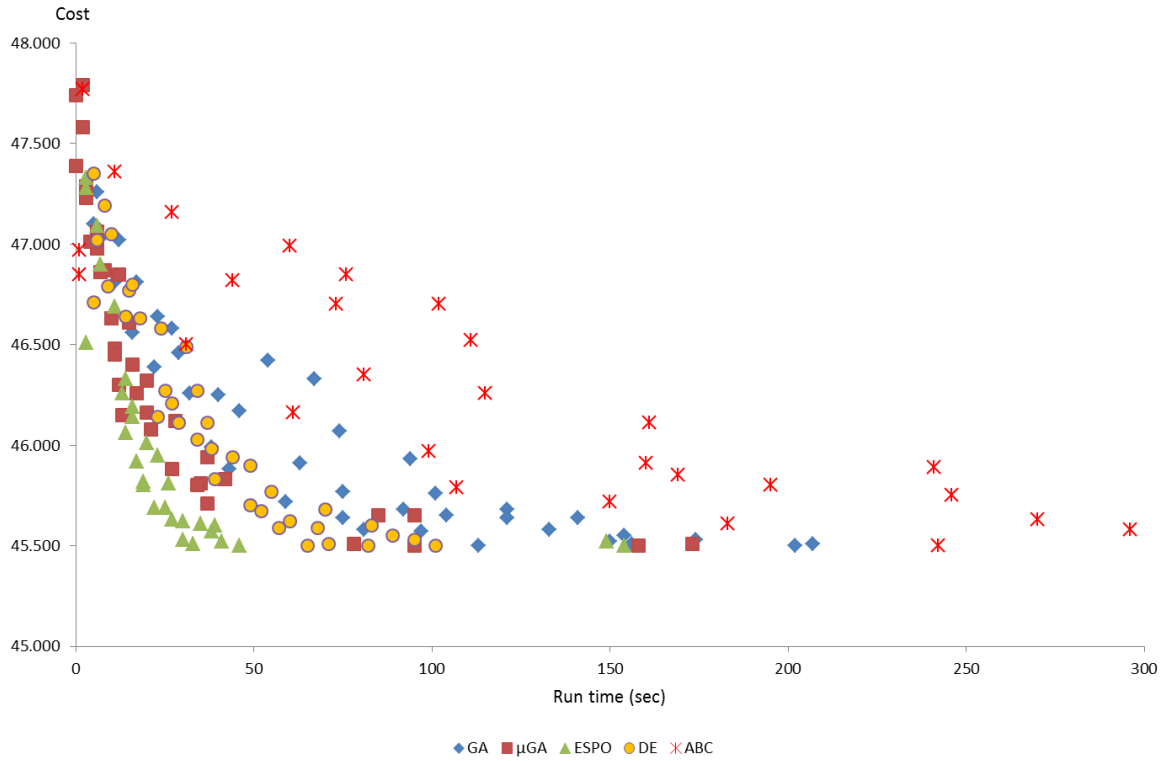

Fig 4. Evaluation results for best performing algorithms (29-activity network).

4. Conclusions

The time-cost trade-off problem is a typical problem of the project scheduling analysis and aims to determine the best method of activity construction so that the project is completed with the least cost and within any completion deadline. Due to the problem size, which exponentially increases with project size, evolutionary algorithms have been implemented and tested in recent years for the solution of the problem. Although several methods have been developed and tested over the years, there are not many comparative studies independently evaluating several methods. Such studies can provide support to project managers regarding the selection of the appropriate method. The objective of this work is to investigate the performance potential of several evolutionary algorithms for the time-cost trade-off problem. In particular, a number of existing algorithms, each one with its own variations, are comparatively examined. The solution is sought via a general purpose commercial optimization software without much interference in algorithm parameter setting and fine-tuning. The investigation was based on three benchmark case studies from the literature with varying size and characteristics. On the basis of the results presented, the following indicative conclusions can be drawn:

- There is no unique method that clearly supersedes others in all aspects, effectiveness to find the optimal solution and efficiency of computation (processing time). In fact, it appears that it may be preferable to utilize different methods to increase the probability of obtaining optimal results.

- In the case studies that were analyzed, all methods found the optimal or a near optimal solution with μGA, EPSO, DE to present a rather ideal behavior in converging to the best solution and Hill Climber (on the other hand) to exhibit the lowest performance in this regard. Regarding computational efficiency, the enhanced PSO and DE methods require the least time while certain versions of GA and the ABC method show the worst performance.
- Comparing different versions of an algorithm (e.g., CGA, STGA, μGA) indicates that experimenting with algorithm structure and parameter tuning leads to result improvement in terms of solution effectiveness and/or efficiency.
- The employment of general purpose commercial software highly reduces the effort to set up the problem, however, the results may be inferior to that of a tailor-made software development for the particular problem.

References

[1] Liu L, Burns SA, Feng C-W. Construction Time-Cost Trade-Off Analysis Using LP/IP Hybrid Method. J Constr Eng Manag 1995;121:446–54. doi:10.1061/(ASCE)0733-9364(1995)121:4(446).

[2] Sakellaropoulos S, Chassiakos AP. Project time–cost analysis under generalised precedence relations. Adv Eng Softw 2004;35:715–24. doi:10.1016/j.advengsoft.2004.03.017.

[3] Klanšek U, Pšunder M. MINLP optimization model for the nonlinear discrete time–cost trade-off problem. Adv Eng Softw 2012;48:6–16. doi:10.1016/j.advengsoft.2012.01.006.

[4] De P, James Dunne E, Ghosh JB, Wells CE. The discrete time-cost tradeoff problem revisited. Eur J Oper Res 1995;81:225–38. doi:10.1016/0377-2217(94)00187-H.

[5] Akkan C, Drexl A, Kimms A. Network decomposition-based benchmark results for the discrete time–cost tradeoff problem. Eur J Oper Res 2005;165:339–58. doi:10.1016/j.ejor.2004.04.006.

[6] Hazır Ö, Haouari M, Erel E. Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version. Comput Oper Res 2010;37:649–55. doi:10.1016/j.cor.2009.06.009.

[7] Feng C-W, Liu L, Burns SA. Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems. J Comput Civ Eng 1997;11:184–9. doi:10.1061/(ASCE)0887-3801(1997)11:3(184).

[8] Hegazy T. Optimization of construction time-cost trade-off analysis using genetic algorithms. Can J Civ Eng 1999;26:685–97. doi:10.1139/l99-031.

[9] Li H, Love P. Using Improved Genetic Algorithms to Facilitate Time-Cost Optimization. J Constr Eng Manag 1997;123:233–7. doi:10.1061/(ASCE)0733-9364(1997)123:3(233).

[10] Li H, Cao J-N, Love PED. Using Machine Learning and GA to Solve Time-Cost Trade-Off Problems. J Constr Eng Manag 1999;125:347–53. doi:10.1061/(ASCE)0733-9364(1999)125:5(347).

[11] Zheng DXM, Ng ST, Kumaraswamy MM. Applying a Genetic Algorithm-Based Multiobjective Approach for Time-Cost Optimization. J Constr Eng Manag 2004;130:168–76. doi:10.1061/(ASCE)0733-9364(2004)130:2(168).

[12] Zheng DXM, Ng ST, Kumaraswamy MM. Applying Pareto Ranking and Niche Formation to Genetic Algorithm-Based Multiobjective Time–Cost Optimization. J Constr Eng Manag 2005;131:81–91. doi:10.1061/(ASCE)0733-9364(2005)131:1(81).

[13] Yang I-T. Using Elitist Particle Swarm Optimization to Facilitate Bicriterion Time-Cost Trade-Off Analysis. J Constr Eng Manag 2007;133:498–505. doi:10.1061/(ASCE)0733-9364(2007)133:7(498).

[14] Zhang H, Li H. Multi-objective particle swarm optimization for construction time-cost tradeoff problems. Constr Manag Econ 2010;28:75–88. doi:10.1080/01446190903406170.

[15] Ng ST, Zhang Y. Optimizing Construction Time and Cost Using Ant Colony Optimization Approach. J Constr Eng Manag 2008;134:721–8. doi:10.1061/(ASCE)0733-9364(2008)134:9(721).

[16] Xiong Y, Kuang Y. Applying an Ant Colony Optimization Algorithm-Based Multiobjective Approach for Time–Cost Trade-Off. J Constr Eng Manag 2008;134:153–6. doi:10.1061/(ASCE)0733-9364(2008)134:2(153).

[17] Afshar A, Ziaraty AK, Kaveh A, Sharifi F. Nondominated Archiving Multicolony Ant Algorithm in Time–Cost Trade-Off Optimization. J Constr Eng Manag 2009;135:668–74. doi:10.1061/(ASCE)0733-9364(2009)135:7(668).

[18] Sonmez R, Bettemir ÖH. A hybrid genetic algorithm for the discrete time–cost trade-off problem. Expert Syst Appl 2012;39:11428–34. doi:10.1016/j.eswa.2012.04.019.

[19] Elbeltagi E, Hegazy T, Grierson D. Comparison among five evolutionary-based optimization algorithms. Adv Eng Informatics 2005;19:43–53. doi:10.1016/j.aei.2005.01.004.

[20] xlOptimizer software, www.xloptimizer.com, 2015.

[21] Eiben AE, Smith JE. Introduction to evolutionary computing. vol. 53. Springer; 2003.

[22] Koumousis VK, Katsaras CP. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 2006;10:19–28. doi:10.1109/TEVC.2005.860765.

[23] Krishnakumar K. Micro-genetic algorithms for stationary and non-stationary function optimization. Intell. Control Adapt. Syst., vol. 1196, International Society for Optics and Photonics; 1990, p. 289–96.

[24] Kennedy J, Eberhart R. Particle swarm optimization. Proc. IEEE Int. Conf. neural networks (Perth, Aust., 1995, p. 1942–8.

[25] Price K, Storn RM, Lampinen JA. Differential evolution: a practical approach to global optimization. Springer Science & Business Media; 2006.

[26] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 2008;8:687–97. doi:10.1016/j.asoc.2007.05.007.