Hindawi Publishing Corporation
Journal of Construction Engineering
Volume 2014, Article ID 136397, 7 pages
http://dx.doi.org/10.1155/2014/136397

Research Article

NIDE: A Novel Improved Differential Evolution for Construction Project Crashing Optimization

Nhat-Duc Hoang

Institute of Research and Development, Faculty of Civil Engineering, Duy Tan University, P809-K7/25 Quang Trung, DaNang 59000, Vietnam

Correspondence should be addressed to Nhat-Duc Hoang; hoangnhatduc@dtu.edu.vn

Received 6 June 2014; Revised 2 October 2014; Accepted 5 October 2014; Published 19 October 2014

Academic Editor: Yingfeng (Eric) Li

In the field of construction management, project crashing is an approach to shortening the project duration by reducing the duration of several critical project activities to less than their normal activity duration. The goal of crashing is to shorten the project duration while minimizing the crashing cost. In this research, a novel method for construction project crashing is proposed. The method is named as novel improved differential evolution (NIDE). The proposed NIDE is developed by an integration of the differential evolution (DE) and a new probabilistic similarity-based selection operator (PSSO) that aims at improving the DE's selection process. The PSSO has the role as a scheme for preserving the population diversity and fending off the premature convergence. The experimental result has demonstrated that the newly established NIDE can successfully escape from local optima and achieve a significantly better optimization performance.

1. Introduction

In the field of construction management, a construction project can be typically defined as a set of individual activities with their technical/managerial constraints. The nature of construction projects, which is characterized by constant changes in the environment, pressures to maintain schedules/costs with increasingly complex construction techniques, makes the task of project management a significant challenge [1]. Due to the complexity of construction projects, schedule management is proved to be very challenging [2]. Therefore, schedule overrun is not unusual in the construction field.

Moreover, there can be a great motivation for the construction contractor and the project owner to reduce the project time [3]. The reason is that as a project progresses, it consumes indirect costs, consisting of the cost of facilities, equipment, and machinery, interest on investment, utilities, labor, and the loss of skills and labor of the project team who are not working at their regular jobs [4]. There also may be severe financial penalties for not completing a project on time; many construction and government contracts have penalty clauses for exceeding the project completion date. In addition, the project owner may desire to reduce the completion time in order to put the facility into operation sooner.

In practice, to shorten the project schedule, the manager may accelerate some of the activities at an additional cost, that is, by allocating more or better resources, such as labor and equipment. Minimizing the sum of activity direct costs while meeting a specified deadline is of practical need and this has been known as the project crashing problem [5].

In the project crashing problem, the cost of an activity is generally expressed in forms of nonincreasing and bounded function of the duration. To simplify the problem, some previous works assumed that the activity time-cost function is linear [4]. Recent researches relaxed the linearity assumption by allowing the activity time-cost function to be of different forms: concave [6], convex [7], hybrid of concave and convex [5], and quadratic [8].

To better suit the real-world circumstance, the research interest has also been directed to solve the problem in which the activity cost function is not only continuous but also discrete [9]. Moreover, the time-cost functions may consist of discontinuous pieces, that is, defined at several separated domains [5]. This situation happens when an activity can

be performed by several methods, each of which leads to a different range of possible time and cost. For instance, an excavation task can be done by hand or by machine; while the former takes 10–15 days, the later may need only 3–5 days. Thus, the activity time-cost function is expressed as two distinct regions, each of which defines the possible time and cost for a particular construction method. Since the problem becomes less restrictive, the solution complexity grows and conventional optimization techniques are unable to deliver satisfactory solutions. Needless to say, with the increasing complexity of real world project crashing problem, the demand for an advanced tool is on the rise among scholars from both the academic area and the industry.

Due to the problem complexity, applying evolutionary algorithms to tackle the project crashing problem can provide feasible alternatives. Evolutionary algorithms inspired by the natural evolution of species have been successfully applied to solve numerous challenging optimization problems in diverse fields [10–12]. Evolutionary algorithms are characterized by iterative progress used to guide the randomly initiated population to the final optimal solution. The advantage of evolutionary algorithms is its ability to find good solutions to complex problems in a relatively short time.

The differential evolution (DE) [13] is a population-based stochastic search engine which is efficient and effective for global optimization in the continuous domain. It uses mutation, crossover, and selection operators at each generation to move its population toward the global optimum [14]. Superior performance of the DE over other algorithms has been verified in optimization problems that span many fields [15–21].

Notably, the DE's ability to deal with complex optimization problem can be hindered because it employs a greedy criterion to make decision whether or not to accept a newly created individual [13]. Under this criterion, a new individual is allowed to survive if and only if it reduces the value of the cost function [22]. In other words, this scheme only takes into account the fitness of individual to determine its survival.

Although the greedy criterion facilitates the optimization process to converge fast, it harbors the risk of premature convergence. The reason is that the population diversity is deteriorated in parallel with the fast convergence. As a consequence, the population tends to be trapped in some local optimum. Obviously, this disadvantage hampers the algorithm to deal with complex objective functions.

Hence, this paper replaces the original selection operator of the DE by proposing a new selection scheme, namely, the probabilistic similarity-based selection operator (PSSO). In the PSSO, besides the fitness values, the similarity between individuals in the population is taken into account to determine their survivals. This scheme can help maintain population diversity and prevent premature convergence. In addition, to strike balance between the optimization performance and computational cost, the PSSO is activated with a certain probability. This probability is fairly large when the optimization commences and gradually reduces as the searching progress gets matured.

In this research, the PSSO is integrated into the DE algorithm to establish the novel improved differential evolution (NIDE) for solving the project crashing problem. The remaining part of this paper is organized as follows. In Section 2, the project crashing problem and the DE algorithm are reviewed. The overall picture of the NIDE, including the PSSO, is presented in Section 3. Section 4 describes applications to demonstrate the capacity of new approach. Finally, some conclusions are mentioned in the last section of this paper.

2. Literature Review

2.1. The Project Crashing Problem Formulation. The project crashing problem can be defined as minimizing the sum of activity direct costs while meeting a specified project deadline [4, 5, 7]. In the project crashing analysis, the objective function is set to minimize the project direct cost [23]. The project direct cost is the summation of all activities' costs. Herein, the decision variables are the duration of all activities. The parameters of the problem are the activities' relationships, the time-cost functions, and the prespecified project duration.

The project crashing problem can be mathematically formulated as follows:

$$\text{Minimize } \sum_{\forall i} c_i, \quad \text{where } \sum_{\forall i} c_i \text{ is the sum of the activity direct cost} \tag{1}$$

subject to

$$\text{ES}_i + t_i - \text{ES}_j \le 0, \quad \forall j \in A_i \tag{2}$$

$$D = \max_{\forall i} \{\text{ES}_i + t_i\} \le D_o \tag{3}$$

$$\text{ES}_i,\ t_i \ge 0, \quad \forall i \tag{4}$$

$$c_i = f(t_i), \quad \forall i. \tag{5}$$

Equation (2) is used to specify the precedence constraints between activity i and all the activities in its successor set A_i; t_i represents the duration for activity i; ES_i denotes the early start time for activity i. Equation (3) computes the total project duration, which must not exceed the project deadline D_o. Equation (4) guarantees that all the early start times and activity duration are nonnegative. Equation (5) expresses that the cost of an activity (c_i) is a function of its duration (t_i).

2.2. Differential Evolution. The differential evolution (DE) is an evolutionary algorithm proposed by Price et al. and Storn and Price [13, 22], which is designed for real parameter optimization. The DE is based on the utilization of a novel crossover-mutation operator, based on the linear combination of three different individuals and one subject-to-replacement parent (or target vector) [24]. The crossover-mutation operator yields a trial vector (or child vector) which will compete with its parent in the greedy selection operator. These two terms, trial vector and child vector, can be used interchangeably. This selection process is performed via selection between the parent and the corresponding offspring [25, 26].

Initialize population of NP individuals
Do
  For each individual j in the population
    Generate three random integers r_1, r_2, and $r_3 \in (1, NP)$
    with $r_1 \neq r_2 \neq r_3 \neq j$
    Generate random integer $i_{\text{rand}} \in (1, D)$
    For each parameter i

$$U_{j,i,g} = \begin{cases} X_{j,r_3,g} + F \times \left(X_{j,r_1,g} - X_{j,r_2,g}\right) & \text{if } \text{rand}_j(0,1) < \text{Cr or } j = \text{rnb}(i) \\ X_{j,i,g} & \text{otherwise} \end{cases}$$

    End For
    Replace X_i with the offspring U_i if U_i is better
  End For
Until the stopping condition is met

Algorithm 1: Differential evolution algorithm.

The algorithm of the DE is depicted in Algorithm 1. In this figure, it is noted that NP represents the size of the population; $X_{j,i}$ is the ith decision variable of the jth individual in the population; g is the current generation; D denotes the number of decision variables. $\text{rand}_j(0, 1)$ is a uniform random number lying between 0 and 1 and r_i is a randomly chosen index ranging from 1 to NP. Moreover, Cr denotes the crossover probability.

In the selection process, the trial vector is compared to the target vector (or the parent) using the greedy criterion [22]. Thus, if the trial vector can yield a lower objective function value than its parent, then the trial vector supersedes the target vector (see Figure 1). The selection operator is expressed as follows:

$$X_{i,g+1} = \begin{cases} U_{i,g} & \text{if } f\left(U_{i,g}\right) \leq f\left(X_{i,g}\right), \\ X_{i,g} & \text{if } f\left(U_{i,g}\right) > f\left(X_{i,g}\right), \end{cases} \tag{6}$$

where $X_{i,g}$ represents the parent vector at generation g, $U_{i,g}$ denotes the trial vector at generation g, and $X_{i,g+1}$ is the chosen individual which survives to the next generation ($g + 1$).

The optimization process terminates when the stopping criterion is met. The user can set the type of this stopping condition. Commonly, maximum generation ($G_{\max}$) or maximum number of function evaluations (NFE) can be used as the stopping condition. When the optimization process terminates, the final optimal solution is readily presented to the user.

3. Novel Improved Differential Evolution (NIDE) for Project Crashing Optimization

3.1. Probabilistic Similarity-Based Selection Operator (PSSO). As mentioned earlier, the DE employs a selection operator based on greedy criterion. This means that, to decide the survivability of an individual, only its fitness value is taken into account. According to Storn and Price [22], this approach, besides its simplicity, has the advantage of fast convergence. This explains why the DE oftentimes overcomes other benchmarked algorithms in terms of convergence rate for a wide range of optimization problems [22, 27, 28].

Nevertheless, the capability of fast convergence is not the only desirable characteristic of an optimizer. Accuracy or, in other words, the solution quality is also of major concern. When dealing with difficult problems, which landscape may contain many local minima, the performance of the DE algorithm may deteriorate. It is because the greedy criterion selection operator reduces the population diversity significantly in order to achieve a fast rate of convergence. Thus, the algorithm is frequently subjected to premature convergence or becoming trapped in local minima [22].

In case of the DE algorithm, to avoid being trapped in a suboptimal solution, it is beneficial to compromise the convergence speed and put more stress on the exploration of the search space. In this case, this research proposes to replace the original selection operator in the DE by the PSSO. Being beneficial from the ideal of the crowding technique [29], the replacement policy in the PSSO involves a candidate solution (trial vector) competing for a place in the population with the most similar parent. Herein, the similarity can be measured by the Euclidean distance.

If a trial vector is better than its most similar parent, which is not necessarily the target vector, the parent is replaced; otherwise, the candidate solution is discarded (see Figure 2). Hence, besides fitness value, the PSSO considers the similarity of individuals quantified by distances among them. The algorithm basically prefers the competition among similar individuals and maintains the diversity of the population.

Moreover, to strike balance between optimization performance and computational cost, the PSSO is activated with a certain probability. This probability is very high at the beginning stage of the optimization process to facilitate the algorithm to explore the search space thoroughly. At the latter stage, it is beneficial to utilize the traditional greedy selection operator to exploit the currently found solutions.

When the algorithm needs to decide whether to accept a newly created trial vector, a random number P_{cs} is generated within the range of (0, 1). If $P_{cs} \leq P_o$, the PSSO is activated; otherwise, the conventional greedy selection is performed. The threshold probability P_o is set to be 1 at the

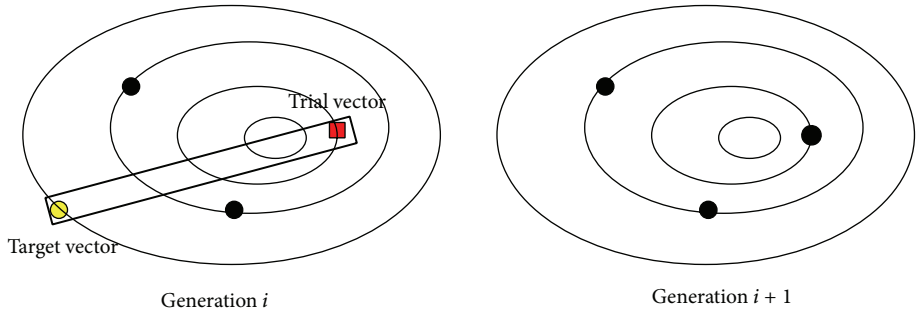

Figure 1: Greedy criterion based selection approach.

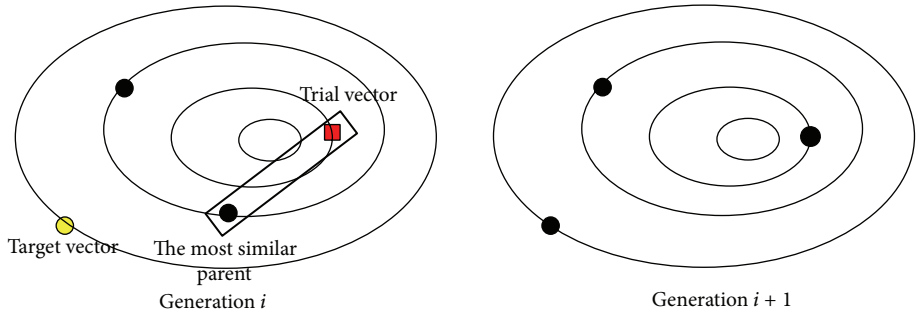

Figure 2: Probabilistic similarity-based selection operator (PSSO).

first generation and it gradually decreases when the searching process reaches latter stages. The formula for calculating P_o is given in the following decaying function:

$$P_o = a \times (1 - r)^{(g-1)}, \tag{7}$$

where a is the initial amount before decaying ($a = 1$); r represents the decaying rate ($r = 0.02$); g denotes the current generation ($g = 1, 2, \ldots, G_{\max}$).

When the PSSO is performed, Euclidean distances from the trial vector to each of the parent vectors are calculated and sorted to identify the most similar parent vector ($X^s_{i,g}$), which is corresponding to the closest distance. In the modified selection operator, the most similar parent vector ($X^s_{i,g}$) is not necessarily identical to the target vector. The trial vector and its corresponding $X^s_{i,g}$ compete with each other based on fitness value. The one that can reduce the cost function survives to the next generation. The selection operator is described as follows:

$$X^s_{i,g+1} = \begin{cases} U_{i,g} & \text{if } f\left(U_{i,g}\right) \le f\left(X^s_{i,g}\right), \\ X^s_{i,g} & \text{if } f\left(U_{i,g}\right) > f\left(X^s_{i,g}\right), \end{cases} \tag{8}$$

where $X^s_{i,g}$ denotes the most similar parent and $U_{i,g}$ represents the trial vector.

3.2. The Proposed NIDE Algorithm. This section is dedicated to describing the newly developed NIDE optimization model in detail. The overall picture of the proposed model is depicted in Figure 3. The NIDE operates through 5 consecutive steps: (1) initialization of population, (2) mutation process, (3) crossover process, (4) probabilistic similarity-based selection operator, and (5) stopping condition verification.

(1) Initialization of Population. Before the searching process can commence, an initial population of *NP* feasible solutions is created by a uniform random generator; herein *NP* denotes the population size. A solution is represented as a vector with D elements as follows:

$$X_i = \left[X_{i,1}, X_{i,2}, \ldots, X_{i,D}\right], \tag{9}$$

where D is the number of decision variables of the problem at hand. The index i denotes the ith individual in the population. The following equation is used to generate a solution within the feasible domain:

$$X_{i,j} = \mathrm{LB}\left(j\right) + \mathrm{rand}\left[0, 1\right] \times \left(\mathrm{UB}\left(j\right) - \mathrm{LB}\left(j\right)\right), \tag{10}$$

where $X_{i,j}$ is the decision variable jth of the solution ith, rand[0, 1] denotes a uniformly distributed random number between 0 and 1, and LB(j) and UB(j) are the lower and upper bound of the decision variable jth, respectively.

(2) Mutation Process. For each target vector, a mutant vector is created by three randomly chosen vectors in the current population. It is worth noticing that the magnitude of the mutation scale factor (F) influences the search step of mutation operator. This process is mathematically shown as follows:

$$V_{i,g+1} = X_{r1,g} + F\left(X_{r2,g} - X_{r3,g}\right), \tag{11}$$

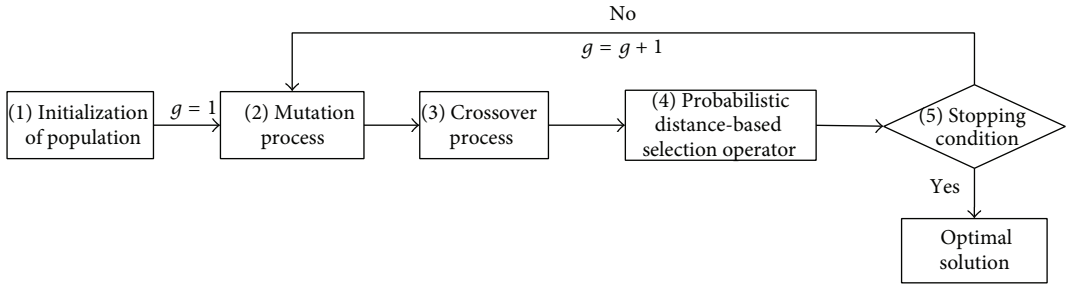

Figure 3: Novel improved differential evolution (NIDE).

Table 1: Project information.

ID	Activity description	Pre.	Time-cost function	Type
1	Site preparation	—	$(t, c) = [(14, 23), (20, 18), (24, 12), (28, 8), (32, 4)]$	Discrete
2	Forms and rebars	1	$(t, c) = [(15, 3), (18, 2.4), (20, 1.8), (23, 1.5), (25, 1)]$	Discrete
3	Excavation	1	$(t, c) = [(15, 4.5), (22, 4), (33, 3), (42, 2.2), (48, 1.9)]$	Discrete
4	Precast concrete girders	1	$(t, c) = [(12, 45), (16, 35), (20, 30), (25, 27), (33, 25)]$	Discrete
5	Pour foundation and piers	2, 3	$(t, c) = [(20, 20), (24, 17.5), (28, 15), (30, 10), (32, 8)]$	Discrete
6	Deliver precast concrete girders	4	$c = -0.3333t^2 + 7.6667t - 6,\ 14 \le t \le 18$ $c = 0.25t^2 - 12.5t + 172,\ 19 \le t \le 25$	Nonlinear
7	Erect girders	5, 6	$c = -4t + 68,\ 7 \le t \le 9$ $c = -2t + 50,\ 10 \le t \le 12$ $c = -t + 39,\ 14 \le t \le 18$	Piecewise linear
8	Finish up	7	$c = -0.5t + 8,\ 2 \le t \le 4$	Linear

where $r1$, $r2$, and $r3$ are three random indexes lying between 1 and NP. These three randomly chosen integers are also selected to be different from the index i of the target vector. F denotes the mutation scale factor, which controls the amplification of the differential variation between $X_{r2,g}$ and $X_{r3,g}$. $V_{i,g+1}$ represents the newly created mutant vector.

(3) Crossover Process. The crossover operation is to diversify the current population by exchanging components of target vector and mutant vector. In this stage, a new vector, named as trial vector, is created. The trial vector is also called the child vector. Herein, the terms trial vector and child vector can be used interchangeably. The trial vector can be formed as follows:

$$U_{i,j,g+1} = \begin{cases} V_{i,j,g+1}, & \text{if } \text{rand}_j \le \text{Cr or } j = \text{rnb}(i), \\ X_{i,j,g}, & \text{if } \text{rand}_j > \text{Cr and } j \ne \text{rnb}(i), \end{cases} \quad (12)$$

where $U_{i,j,g+1}$ is the trial vector, j denotes the index of element for any vector, rand_j is a uniform random number lying between 0 and 1, Cr is the crossover probability, which is needed to be determined by the user, and rnb(i) is a randomly chosen index ranging from 1 to NP which guarantees that at least one parameter from the mutant vector ($V_{i,j,g+1}$) is copied to the trial vector ($U_{i,j,g+1}$). In this step, the crossover probability (Cr) determines the contribution of the target vector and the mutant to the component of the trial vector.

(4) Probabilistic Similarity-Based Selection Operator. At this stage, the newly created trial vector is evaluated by the PSSO which is described in the previous section of the paper. If the trial vector is accepted, it will survive to the next generation; otherwise, the trial vector is discarded.

(5) Stopping Condition. The optimization process continues until the maximum generation $G_{\max}$ is achieved. Once the optimization process terminates, the final optimal solution is ready for the user's assessment.

4. Model Application

In this section, the capability of the NIDE is demonstrated through a project crashing optimization problem adopted from a previous work [5]. This case study is fairly complex because the time-cost relation exists in a variety of forms: nonlinear, discrete, and piecewise discontinuous functions. The detail of project information, which includes activity's ID, activity description, predecessor, time-cost function, and type of function, is provided in Table 1. The required project crash time is 110 days.

To verify the performance of the proposed algorithm, three different algorithms are used for performance comparison: the Genetic Algorithm (GA) [30], the Particle Swarm Optimization (PSO) [5], the DE [13], and the Adaptive Differential Evolution with Optional External Archive (JADE) [31]. The PSO has been applied by Yang (2007) to solve the project crashing optimization [5]. The GA and DE methods are popular and commonly used evolutionary algorithms in the field of civil engineering. The JADE is a recently developed metaheuristic. Thus, comparing the performance

Table 2: Result comparison for model applications.

Methods	Generations				
	1000	3000	5000	8000	10000
GA	85.50	85.50	85.50	85.50	85.50
PSO	85.50	85.50	85.50	85.50	85.50
DE	85.50	78.75	73.00	70.50	70.50
JADE	85.50	73.00	65.40	65.40	65.40
NIDE	85.50	64.50	63.75	63.75	63.75

of the NIDE and these benchmarking methods can point out the superior of the proposed approach.

The population size (NP) and the maximum number of generations ($G_{\max}$) for each algorithm are set to be $6 \times D$ and 10000, respectively. Herein, the parameter D denotes the number of decision variables of the optimization problem at hand. Moreover, as recommended by Price et al. [13], the tuning parameters F and Cr of the DE and NIDE are fixed to be 0.5 and 0.8, respectively.

The experimental results are shown in Table 2 which shows the performance of each optimization algorithm (in terms of the project cost) at each threshold of $G_{\max}$. Observably, the proposed NIDE has achieved the most desirable project cost. The minimum total direct cost found by the NIDE is 63.75. Accordingly, the optimal duration of eight activities in the project is given as follows: [32, 25, 22, 33, 32, 25, 13, 4]. The JADE is the second best method with the project cost of 65.40. The DE is the third best method with the project cost of 70.50. The GA and PSO algorithms both attain the project cost of 85.5. As observed from the experiment, the GA and PSO got trapped in an inferior solution at early states of their evolutionary progress. Compared with the GA and the PSO, the JADE and DE algorithms have discovered better solutions, but they still get trapped in some local minima.

Moreover, it is noted that the NIDE can find a project crashing solution which is significantly improved compared to the outcome found by other benchmark algorithms. As seen from Table 2, the proposed method has also converged faster than other algorithms. The NIDE has located the best found solution within 5000 generations. Based on the experiment, the advantage of the PSSO is clearly exhibited. And the replacement of the PSSO over the conventional selection operator is shown to be helpful in boosting the optimization performance.

5. Conclusion

This study has proposed a novel optimization model, namely, NIDE, to tackle complex nature of the project crashing problem. To preserve the population diversity during the evolutionary process, the NIDE replaces the original selection in the DE by the PSSO—a new selection operator. The PSSO helps the optimization algorithm preserve the diversity during the evolutionary process and better compromise between exploration in the initial stage and exploitation in the latter stage of the searching process. By doing so, the possibility of being trapped in a suboptimal solution is diminished considerably.

The optimization result obtained from the experiment section has verified the capability of the newly established approach. This fact proves that the newly established NIDE can be a promising tool to help the project manager better deal with the project crashing problem. The future direction of this research may include applying the NIDE to solve other optimization problems and integrating a mechanism for tuning the model's parameters adaptively.

Conflict of Interests

The author declares that he has no conflict of interests regarding to the publication of this paper.

References

[1] M.-Y. Cheng, N.-D. Hoang, A. F. V. Roy, and Y.-W. Wu, "A novel time-depended evolutionary fuzzy SVM inference model for estimating construction project at completion," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 4, pp. 744–752, 2012.

[2] S. Mubarak, *Construction Project Scheduling and Control*, John Wiley & Sons, 2010.

[3] R. J. Dzeng, "Identifying a design management package to support concurrent design in building wafer fabrication facilities," *Journal of Construction Engineering and Management*, vol. 132, no. 6, pp. 606–614, 2006.

[4] K. Sears, G. Sears, and R. Clough, *Construction Project Management: A Practical Guide to Field Construction Management*, John Wiley & Sons, Hoboken, NJ, USA, 5th edition, 2008.

[5] I.-T. Yang, "Performing complex project crashing analysis with aid of particle swarm optimization algorithm," *International Journal of Project Management*, vol. 25, no. 6, pp. 637–646, 2007.

[6] J. E. Falk and J. L. Horowitz, "Critical path problems with concave cost-time curves," *Management Science*, vol. 19, no. 4, pp. 446–455, 1972.

[7] S. Foldes and F. Soumis, "PERT and crashing revisited: mathematical generalizations," *European Journal of Operational Research*, vol. 64, no. 2, pp. 286–294, 1993.

[8] R. F. Deckro, J. E. Hebert, W. A. Verdini, P. H. Grimsrud, and S. Venkateshwar, "Nonlinear time/cost tradeoff models in project management," *Computers and Industrial Engineering*, vol. 28, no. 2, pp. 219–229, 1995.

[9] P. De, E. James Dunne, J. B. Ghosh, and C. E. Wells, "The discrete time-cost tradeoff problem revisited," *European Journal of Operational Research*, vol. 81, no. 2, pp. 225–238, 1995.

[10] M.-Y. Cheng, N.-D. Hoang, and Y.-W. Wu, "Hybrid intelligence approach based on LS-SVM and differential evolution for construction cost index estimation: a Taiwan case study," *Automation in Construction*, vol. 35, pp. 306–313, 2013.

[11] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[12] L.-C. Lien and M.-Y. Cheng, "A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9642–9650, 2012.

[13] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, Springer, 2005.

[14] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[15] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2008.

[16] A. Ponsich and C. A. C. Coello, "Differential Evolution performances for the solution of mixed-integer constrained process engineering problems," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 399–409, 2011.

[17] R. Mallipeddi, "Harmony search based parameter ensemble adaptation for differential evolution," *Journal of Applied Mathematics*, vol. 2013, Article ID 750819, 12 pages, 2013.

[18] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[19] M.-Y. Cheng and N.-D. Hoang, "Risk score inference for bridge maintenance project using evolutionary fuzzy least squares support vector machine," *Journal of Computing in Civil Engineering*, vol. 28, no. 3, Article ID 04014003, 2014.

[20] M. Cheng and N. Hoang, "Groutability estimation of grouting processes with microfine cements using an evolutionary instance-based learning approach," *Journal of Computing in Civil Engineering*, vol. 28, no. 4, Article ID 04014014, 2014.

[21] H.-H. Tran and N.-D. Hoang, "An artificial intelligence approach for groutability estimation based on autotuning support vector machine," *Journal of Construction Engineering*, vol. 2014, Article ID 109184, 9 pages, 2014.

[22] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[23] H. Ahuja, *Project Management Techniques in Planning and Controlling Construction Projects*, Wiley, New York, NY, USA, 1984.

[24] R. Landa Becerra and C. A. Coello, "Cultured differential evolution for constrained optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33–36, pp. 4303–4322, 2006.

[25] E. Mezura-Montes, C. A. C. Coello, and E. I. Tun-Morales, "Simple feasibility rules and differential evolution for constrained optimization," in *Proceedings of the 3rd Mexican International Conferenceon Artificial Intelligence (MICAI '04)*, pp. 707–716, April 2004.

[26] H.-H. Tran and N.-D. Hoang, "A novel resource-leveling approach for construction project based on differential evolution," *Journal of Construction Engineering*, vol. 2014, Article ID 648938, 7 pages, 2014.

[27] J. Zhang and A. C. Sanderson, "JADE: self-adaptive differential evolution with fast and reliable convergence performance," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 2251–2258, September 2007.

[28] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043–3074, 2008.

[29] K. A. de Jong, *Analysis of the behavior of a class of genetic adaptive systems [Ph.D. dissertation]*, University of Michigan, Ann Arbor, Mich, USA, 1975.

[30] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, John Wiley & Sons, 2004.

[31] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, 2009.