A Hybrid Tabu Search-Based Artificial Immune Algorithm for Construction Site Layout Optimization

Vu Duc Quang[1], Nguyen Van Truong[1], Vu Thi Thuy[1], Hoang Xuan Huan[2]
[1] Thai Nguyen University of Education, Thai Nguyen, Vietnam
[2] Faculty of Information Technology, Vietnam National University, Hanoi, Vietnam
Correspondence: Hoang Xuan Huan, huanhx@vnu.edu.vn
Communication: received 20 May 2017, revised 18 May 2018, accepted 18 May 2018
Online early access: 8 November 2018, Digital Object Identifier: 10.32913/rd-ict.vol2.no15.470
The Area Editor coordinating the review of this article and deciding to accept it was Assoc. Prof. Le Nhat Thang

Abstract: Layout of temporary facilities on a construction site is essential to enhance productivity and safety. It is a complex issue due to the unique nature of construction. This problem is validated as an NP-hard and one of the challenging problems in the field of construction management. In this paper, we proposed a hybrid algorithm, named topt-aiNet, to solve the construction site layout problem by combining the aiNet algorithm with Tabu search. Experimental results showed that the proposed algorithm outperformed the state-of-the-art ones.

Keywords: *Artificial immune system (AIS), opt-aiNet, topt-aiNet, construction site layout, Tabu search.*

I. Introduction

Construction site layout (CSL) planning aims to arrange the locations and areas reserved for the temporary support facilities (Site Office, Storeroom, Warehouse, etc.), depending on the sizes and the locations of the predefined projects. This is an important task, should be considered early in construction planning, and is usually performed by construction managers. However, decisions are often made based on intuition, experiments, and experience. The impact of good layout planning on costs and time becomes more obvious during the implementation of the projects. Good site layout planning is important to promote safe and efficient operations, minimize travel time, decrease material handling, and avoid obstructing material and equipment movements, particularly for large-scale projects [1].

There are many methods proposed for the problem such as genetic algorithms (GA) [2, 3], ant colony optimization algorithms (ACO) [4], and the Clonalg algorithm [5]. However, these algorithms have to cope with high running time complexities when dealing with large datasets.

The Artificial Immune Network algorithm (aiNet) is an immune network algorithm derived from Artificial Immune Systems (AIS) [6]. The aiNet algorithm was developed for data compression and clustering. It was also extended slightly for applying to optimization problems, and hence called the Optimization Artificial Immune Network algorithm (opt-aiNet). The opt-aiNet algorithm has subsequently been developed further for bioinformatics and even for modeling of simple immune responses [7].

In this paper, we propose a hybrid algorithm, called topt-aiNet, by combining aiNet and the Tabu search to solve the CSL problem. Experimental results will show that the proposed algorithm outperforms some state-of-the-art ones.

The rest of the paper is organized as follows. In the next section, we present the background of the CSL problem and briefly review related works. Section III presents the combined algorithm called topt-aiNet in details. Section IV describes experiments on 8 case studies. Section V concludes the paper and discusses some possible future works.

II. Site Layout Problem and Related Works

1. Site Layout Problem

The problem is formally expressed as follows. Given n facilities and m available locations ($n \leq m$), our task is to arrange n facilities in m locations such that the objective function is optimized ($m - n$ remaining positions will be left empty). In case the number of locations is greater than the number of facilities, some "dummy" facilities with zero distance and frequency may be added to ensure that both numbers are equal. Therefore, we assume that the numbers of both predetermined facilities and places equal

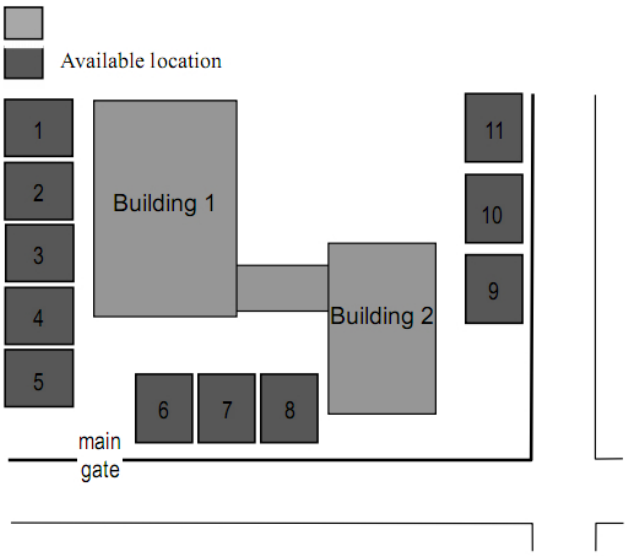

Figure 1. Location representation of the construction site.

n. There are many objective functions proposed for the problem [8, 9]. In this article, we consider two common objective functions. The functions are adapted from Yeh [1] and Li *et al.* [3]. For convenience, the objective functions are considered with two different sets of constraints, which represent two optimal (sub-) problems.

The first problem derives partly from the fact that the frequencies of trips made by construction personnel between facilities are not equal [4]. For example, a salesman usually moves between the Site Office and the Concrete Batch Workshop, but he rarely moves from the Site Office to the Storeroom. Therefore, the locations for facilities will be carefully chosen in order to minimize the total distance.

Problem 1: The problem aims to minimize the total distance between facilities, and is formulated as follows:

$$\text{minimize} \quad F_1 = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} \delta_{ik} f_{ij} d_{ij}, \qquad \text{(P1)}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \delta_{ik} = 1, \quad k = 1,2,...,n,$$

where n is the number of facilities, δ_{ik} is the permutation matrix variable (= 1 if facility i is assigned to location k), f_{ij} is the frequency of trips made by construction personnel between facilities i and j. Note that $f_{ij} = f_{ji}$ for all $i, j \leq n$. The frequency is expressed as the number of trips per time period, and is defined in this study as the number of trips per day, d_{ij} is the distance between locations i and j. Therefore, the objective function F_1 reflects the total traveling distance made by construction personnel. An example of locations with $n = 11$ is illustrated in Figure 1 [10].

TABLE I
THE FACILITIES TO BE LOCATED

Site facilities	Abbreviations
Site Office	SO
Falsework Shop	FS
Labor Residence	LR
Storeroom 1	S1
Storeroom 2	S2
Carpentry Workshop	CW
Reinforcement Steel Workshop	RW
Side Gate	SG
Electrical, water and other utilities control room	UR
Concrete Batch Workshop	BW
Main Gate	MG

Below, we select six test cases from the literature for Problem (P1). Test cases 1, 2 and 3 are widely used in literature and are selected for consideration. Moreover, three larger datasets used in test cases 4, 5 and 6 are created randomly as a further benchmark in this study.

The facilities to be located within the site boundaries are shown in Table I.

The frequencies of trips (in one day) and the distances between available locations are listed in [4, 8, 10, 11]. It should be noted that the site does not offer alternative roads from one location to another. The distances are measured in meters.

Test case 1:

Input: $n = 11$, f_{ij}, d_{ij}.

Output: $\min F_1$.

Constraints: Each of the predetermined location is available for accommodating any facility.

Test case 2:

Input: $n = 11$, f_{ij}, d_{ij}.

Output: $\min F_1$.

Constraints: Side Gate and Main Gate are assigned to locations 1 and 10, respectively [2]. This case represents a realistic approach in a construction site, which is usually determining the side and main gates before the construction to be started as the locations of gates are important for access and, thus, transportation. Therefore, these gates have to be positioned at predetermined locations.

Test case 3:

Input: $n = 11$, f_{ij}, d_{ij}.

Output: $\min F_1$.

Constraints: Site Office, Labor Residence and Concrete Batch Shop cannot be allocated to the relatively smaller

locations of 7 and 8 [3]. Test case 3 was used to illustrate the constraint under which facilities that are relatively larger than other facilities cannot be accommodated by every possible location. Unequal area constraint has to be stated to ensure that no larger facilities are positioned to smaller locations.

Test cases 4, 5, and 6 are similar to test case 1, except for $n = 20$, 40 and 60, respectively. The frequencies and distances in these cases are randomly generated[1].

Test case 4:

Input: $n = 20$, f_{ij}, d_{ij}.

Output: $\min F_1$.

Constraints: Similar to test case 1.

Test case 5:

Input: $n = 40$, f_{ij}, d_{ij}.

Output: $\min F_1$.

Constraints: Similar to test case 1.

Test case 6:

Input: $n = 60$, f_{ij}, d_{ij}.

Output: $\min F_1$.

Constraints: Similar to test case 1.

The second problem adapted from Yeh [1] is more complex than the first one.

Problem 2: The layout objectives include the cost that is calculated from the adjacency and distance between objects, the space availability for object location, the positions and views of objects in relation to the others. The optimization problem is formulated as follows:

minimize

$$F_2 = \sum_{x=1}^{n}\sum_{i=1}^{n}\delta_{xi}C_{xi} + \sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{y=1}^{n}\sum_{j=1}^{n}\delta_{xi}\delta_{yj}A_{ij}D_{xy}, \tag{P2}$$

subject to

$$\delta_{yi} = 0, \text{ if } \delta_{xi} = 1 \text{ and } y \neq x,$$
$$\delta_{xj} = 0, \text{ if } \delta_{xi} = 1 \text{ and } i \neq j,$$

where, the objective function, F_2, is an integration of total distance and cost, δ_{xi} is the permutation matrix variable, C_{xi} is the construction cost of assigning facility x to location i, $A_{ij} = 1$ if location i is neighboring location j, and D_{xy} is the interactive cost of assigning facility x on the location neighboring facility y. An example of locations with $n = 12$ is illustrated in Figure 2 adapted from [1].

We consider two test cases for Problem (P2). There are two permanent buildings on a campus to be constructed.

[1]Tthe datasets could be downloaded from https://goo.gl/v1kcTU.

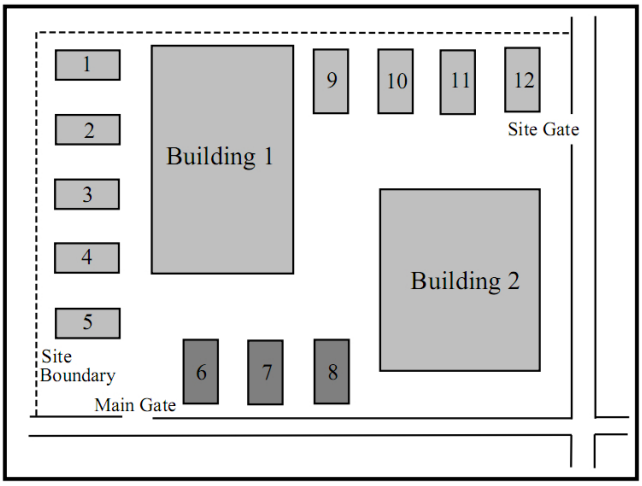

Figure 2. Example of site layout.

There are 12 available locations where the following facilities may be placed: Reinforcing Steel Shop 1 (R1), Reinforcing Steel Shop 2 (R2), Carpentry Shop 1 (C1), Carpentry Shop 2 (C2), Falsework Shop 1 (F1), Falsework Shop 2 (F2), Concrete Batch Plant 1 (B1), Concrete Batch Plant 2 (B2), Job Office (JO), Labor Residence (LR), Electricity Equipment and Water-supply Shop (E), and Warehouse (W). The construction cost matrix (C), the site neighboring index matrix (A), and the interactive cost matrix (D) (the unit of all costs in the test cases are 1,000) are shown in [12].

Test case 7:

Input: $n = 12$, f_{ij}, d_{ij}.

Output: $\min F_2$.

Constraints: Each of the predetermined location is available for accommodating any facility.

Test case 8:

Input: $n = 12$, f_{ij}, d_{ij}.

Output: $\min F_2$.

Constraints: Both site neighboring index matrix and interactive cost matrix are used, with different values as shown in [12].

2. Related Works

The CSL problem is classified as a quadratic assignment problem, which is an NP-hard problem [13]. Due to the complexity of the CSL problem, numerous techniques have been proposed to uncover its solutions, but it is nonperforming to obtain the optimal solutions by hand calculations. Therefore, optimization techniques are usually used, instead. The problem has been solved by researchers using two distinct techniques: exact algorithms

and approximation ones. Exact algorithms, such as the branch and bound algorithm, have been designed to find the optimal solutions. But they cannot be adopted for large-scale projects because of the need for huge calculations and efforts. So, heuristic and meta-heuristic algorithms have been proposed to reduce calculation costs of the algorithms but still producing acceptable solutions.

Evolutionary algorithms are mainly represented by the genetic algorithms, Clonalg, and opt-aiNet. Evolutionary algorithms mimic the process of natural evolution and are used to generate useful solutions to optimization and search problems. For the first time, Li and Love [2, 3] used the GA algorithm to solve these problems. Lately, Ioanna *et al.* [11] proposed a new GA algorithm to solve test case 2 for optimal results. Wang *et al.* [5] used Clonalg to solve test case 2. Quang *et al.* [14] proposed the lopt-aiNet algorithm, which is a hybrid algorithm employing the opt-aiNet and local search to solve test cases 1, 2, and 3.

The swarm intelligence algorithms mainly include ant colony optimization (ACO) and particle swarm optimization (PSO). Ning and Liu [13] used the Max-Min Ant System (MMAS), which is one variant of the ACO algorithm to solve the CSL problem, and used continuous dynamic searching to guide MMAS developed to solve the dynamic CSL problem under the objectives of minimizing safety concerns and reducing construction cost. In [4], Gulben *et al.* proposed an ACO algorithm with local analysis (ACO-LA) to improve the quality of the solution which led to better results than those in previous literature. They have also presented an improved ACO algorithm with parametric analysis (ACO-PA), potential to assess appropriate parameter values within a predefined parameter range. Adrian [12] presented a method of selecting optimal arguments for three algorithms GA, PSO, and ACO. ACO is considered the fastest among the three algorithms to find the optima.

There are other meta-heuristic algorithms. Kaveh *et al.* [8] proposed two algorithms, called the Colliding bodies optimization (CBO) and the Enhanced colliding bodies optimization (ECBO). CBO is an efficient meta-heuristic optimization algorithm that is based on one-dimensional collisions between bodies. ECBO improved CBO to make it faster and to obtain more reliable solutions.

Most recently, Quang *et al.* [14] have proposed an algorithm that combines aiNet with local search. Their approach reduces running times in many experiments. However, the algorithm has no capacity to prevent developing locally past candidate solutions. This could lead to limitations of the algorithm for large problems.

Algorithm 1: Topt-aiNet algorithm

1: Randomly initialize P including n_0 cells;
2: $M = \varnothing$; Tabu list = $\varnothing$;
3: **repeat**
  3.1: Determine the fitness of all cells in P;
  3.2: Sort P by value of the objective function;
  3.3: C = Select n_1 cells from P;
  3.4: Expand C;
  3.5: Apply hypermutation for all cells in C;
  3.6: Determine the fitness of mutated clones;
  3.7: $M = M + n_2$ best cells in C;
  3.8: All cells in M interact with each other;
  3.9: Apply Tabu search on M (as in Procedure TabuSearch(M))
  3.10: $P = M + (n_0 - |M|)$ cells randomly created;
until The stopping criteria have been met;
4: Choose a cell with the highest affinity as a solution from set M.

III. New Algorithm: Topt-aiNet

The aiNet algorithm was first proposed by de Castro *et al.* to perform data analysis and clustering tasks [6]. Opt-aiNet, an algorithm of the aiNet family, was used to optimize continuous functions. It evolves a population, which consists of a network of antibodies (considered as candidate solutions to the function being optimized, each antibody is called a cell). These antibodies undergo a process of evaluation against the objective function, clonal expansion, mutation, selection, and interaction between themselves. Opt-aiNet creates a memory set of antibodies that represent the best candidate solutions as used in [7].

The evolved population is similar to the population of GA in terms of both concept and representing models. It has selection and mutation methods like that of GA.

All opt-aiNet, PSO, and ACO have a memory set. However, the following features are specific to opt-aiNet:

- The population size is dynamically adjustable;
- Having the capability of maintaining many optimal solutions;
- Having interaction;
- Demonstrating exploitation and exploration of the search space.

To improve the performance of aiNet, Tabu search is used for fast locating of the local optima. The proposed topt-aiNet algorithm is described in Algorithm 1. In comparison with the original opt-aiNet, this algorithm is extended by adding Tabu search and resizing population by removing some worst cells for faster locating of the optima.

Parameters n_0, n_1, and n_2 in the topt-aiNet algorithm are such that $n_0 > n_1 > n_2$. In step 1, population P includes n_0

cells created randomly; each cell is a vector corresponding to a candidate solution for the problem and set M is initially empty. Table II shows an example of a cell in the network of antibodies. It means that facility 11 is located in location 1, facility 9 is located in location 2, etc. Iterations of the algorithm implemented in step 3. Meanwhile, the process of replication in step 3.4 simply is doubling the network cells in set C.

In step 3.5, hypermutation is applied to each network cell with an inverse ratio of fitness. In other words, the higher the network cell result is, the lower the hypermutation ratio is, and vice versa. The hypermutation ratio of cell i is defined as

$$r_i = \frac{f_i - f_{\text{best}}}{f_{\text{best}}} \times 100\%, \quad (1)$$

where f_i is the fitness of cell i and f_{best} is the fitness of the best cell found. In case $r_i > 100\%$, we assign 100% to r_i.

The topt-aiNet algorithm keeps a memory set called M to save the network cells having the best results. Through each iteration, set M is always updated. To avoid memory overflow in set M, which may be caused by adding similar cells, topt-aiNet performs the interactive process in step 3.8, in which each pair of cells in M will interact with each other, a "close together" pair will expel a lower antibody from M. There are a number of methods for evaluating the "close together" based on such as the results of the objective function, the ecliptic distance, etc. In this paper, we use the following objective function as a method for evaluating interaction: if the fitness of two cells differs less than 6%, the worse cell will be removed from M.

An important factor making the topt-aiNet algorithm different from the original opt-aiNet algorithm is that every iteration applies a Tabu search for all the cells in set M (step 3.9). This combination avoids the generation of repeated cells and thus speeds up the algorithm convergence. The Tabu search algorithm can be implemented as below.

```
Procedure TabuSearch(M);
begin
  repeat
      Create a set of neighborhood solutions of M;
      Determine the objective function
      Choose best neighborhood;
      Update Tabu list;
   until The stopping criteria have been met;
end
```

The fundamental difference between Tabu search and other local search algorithms is that, in each iteration, to avoid revisiting the previously reviewed solutions, Tabu search uses a list, called Tabu list, to store some previous moves. The list stores some of the transitions that have just been made in former iterations. The transitions in the Tabu list are called the Tabu transfers. These transfers will be banned to use again as long as they are in the Tabu list. Each Tabu transfer will be in the Tabu list for a t-iteration period, after that it will be removed from the Tabu list and can be used again. The value of t is called the Tabu tenure value of the transfer step. This value can be fixed for all transfers or it can also be a random number chosen for each transfer.

Before moving to the next iteration, the set P of cells was rebuilt by adding cells in M and randomly generating new cells added to P for enough n_0 cells (step 3.10).

TABLE II
AN EXAMPLE OF A CELL IN THE NETWORK

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11
11	9	4	5	3	8	6	10	7	2	1

TABLE III
AVERAGE RESULTS FOR CASES 1, 2 AND 3

Algorithms	**Case 1**	**Case 2**	**Case 3**
CBO 2016 [8]		12,558	
ECBO 2016 [8]		12,555	
ACO-PA 2015 [4]	12,150	12,582	12,616
GA 2016 [11]	12,150	12,546	12,606
CLONALG 2016 [5]		12,546	
Opt-aiNet 2016 [14]	12,436	12,582	12,616
Lopt-aiNet 2016 [14]	12,150	12,546	12,606
Topt-aiNet	12,150	12,546	12,606

IV. EXPERIMENTS

In this section, we conduct experimental comparisons of the topt-aiNet with GA, PSO, ACO, CBO, and CLONALG algorithms as proposed in [4, 5, 8, 11, 12] and with opt-aiNet, lopt-aiNet recently proposed in [14].

The topt-aiNet algorithm is tested on six datasets with different conditions as shown in Section II. We used a computer with a Pentium P6200 with 2.13 GHz of CPU and 2 GB of RAM.

Regarding the parameters used in the experiments, population $n_0 = 200$, $n_1 = 10$, and $n_2 = 1$, and the stopping criteria of topt-aiNet include a maximum of ten consecutive loops without improving the best solution. The stopping criteria of Tabu search include five iterations and the value of t in the Tabu search is 5 for all transfers. As seen from Table III, for test case 1 and test case 3, the average results, calculated as the arithmetic mean, of ACO-PA, GA, lopt-aiNet, and topt-aiNet algorithms are the same (12, 150 in

TABLE IV
AVERAGE RESULTS FOR CASES 4, 5 AND 6

Algorithms	Case 4	Case 5	Case 6
Lopt-aiNet [14]	28,490.2	142,185.2	320,765.6
Topt-aiNet	28,240.0	141,735.6	319,800.8

TABLE V
AVERAGE RUNNING TIMES FOR CASES 4-6

Algorithms	Case 4	Case 5	Case 6
Lopt-aiNet [14]	4.05	44.54	317.98
Topt-aiNet	3.29	21.69	99.21

test case 1 and 12,606 in test case 3). In test case 2, all GA, Clonalg, lopt-aiNet, and topt-aiNet algorithms produced the best results. Note that some cells in Table III are blank to indicate that the corresponding test cases are not presented in the references.

In term of running time, ACO-PA in [4] found the optimal solution in 1.15 seconds on an Intel Core 2 Duo processor at 2.66 GHz and 4 GB of RAM. Meanwhile, our topt-aiNet produced the optimal result in only 0.15 seconds on a less powerful computer.

Both CBO and ECBO algorithms run on a computer with Intel Core i7 processor (1.73 GHz) and 4 GB of RAM and they used 200 iterations to find the results. The CLONALG algorithm has been run on a computer with 2.30 GHz of CPU and 2 GB of RAM and after 50 iterations it found the solution for test case 2. Meanwhile, topt-aiNet only used 14 iterations on the average to find the same or better results.

For test cases 4, 5 and 6 on randomly created datasets, we implemented another combination of the ACO algorithm with the Tabu search, called Tabu-ACO. The TabuSearch(M) procedure used in Tabu-ACO is similar to that used in topt-aiNet. Table IV provides the average results of thirty five runs of the Tabu-ACO, lopt-aiNet and topt-aiNet algorithms. It shows that the topt-aiNet algorithm outperformed the ACO and lopt-aiNet in all three test cases 4, 5, and 6.

In term of the running time, it can be seen in Table V that topt-aiNet runs faster than lopt-aiNet.

Table VI shows the average performance with 35 runs of all algorithms GA, PSO, ACO, opt-aiNet, lopt-aiNet, and topt-aiNet. Experimental results show that topt-aiNet and lopt-aiNet could find the same optimal value of 90 in all runs while the others produce worse results (greater values). This implies that topt-aiNet was consistent in finding the minimum distance for both the test cases. About the average running time (in seconds), the lopt-aiNet algorithm

TABLE VI
AVERAGE RESULTS AND RUNNING TIMES FOR CASES 7 AND 8

Algorithms	Case 7		Case 8	
	Result	Time	Result	Time
GA [12]	91.0	0.54	91.6	0.54
PSO [12]	90.6	1.96	90.8	1.87
ACO [12]	90.8	0.33	91.0	0.35
Opt-aiNet [14]	101.2	0.19	103.2	0.19
Lopt-aiNet [14]	90.0	0.20	90.0	0.21
Topt-aiNet	90.0	0.20	90.0	0.20

performed fairly well and can be competitive with other approaches.

V. CONCLUSIONS

We have introduced a novel AIS-based approach for solving the CSL problem. The new algorithm, topt-aiNet, differs from the conventional aiNet algorithm in its combination with the Tabu search for effective localization of the optima.

Experiments have showed that topt-aiNet outperformed five other algorithms (lopt-aiNet, opt-aiNet, GA, ACO, and PSO) in terms of obtained objective function value and running time. Moreover, the optimal results found by all iterative runs (in test cases 1, 2, 3, 4, and 5) strongly supported the consistency of the proposed method.

The lopt-aiNet algorithm performed fairly well in terms of both running time and objective function values. This is, perhaps, caused by the support of the Tabu search in the combined algorithm for better convergence.

In the near future, we are planning to apply our algorithm to other construction tasks such as equipment routing planning and material storage layout planning for real projects. Besides, we are investigating how to dynamically calculate arguments like n_1 and n_2 for early ending of the algorithms. This will make our approach more suitable for larger test cases.

ACKNOWLEDGMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2016.05. We would like to thank Thai Nguyen University for providing research facilities while doing this work.

REFERENCES

[1] Y. I-Cheng, "Construction-site layout using annealed neural network," *Computing in Civil Engineering*, vol. 9, pp. 201–208, 1995.

[2] L. Heng and P. E. Love, "Comparing genetic algorithms and non-linear optimization for labor and equipment assignment," *Computing in Civil Engineering*, vol. 12, pp. 227–331, 1998.
[3] ——, "Genetic search for solving construction site level unequal area facility layout problems," *Automation in Construction*, vol. 9, pp. 217–226, 2000.
[4] G. Calis and O. Yuksel, "An improved ant colony optimization algorithm for construction site layout problems," *Building Construction and Planning Research*, vol. 3, pp. 221–232, 2015.
[5] X. Wang, A. S. Deshpande, G. B. Dadi, and B. Salman, "Application of clonal selection algorithm in construction site utilization planning optimization," in *International Conference on Sustainable Design, Engineering and Construction, Procedia Engineering 145*, 2016, pp. 267–273.
[6] D. Castro, L. Nunes, and J. Timmis, "An artificial immune network for multimodal function optimization," *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 1, pp. 699–704, 2002.
[7] J. Timmis and C. Edmonds, "A comment on opt-ainet: An immune network algorithm for optimization," *Genetic and Evolutionary Computation - GECCO*, pp. 308–317, 2004.
[8] A. Kaveh, M. Khanzadi, M. Alipour, and M. R. Moghaddam, "Construction site layout planning problem using two new meta-heuristic algorithms," *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, vol. 40, no. 4, pp. 263–275, 2016.
[9] H. M. Osman, M. E. Georgy, and M. E. Ibrahim, "A hybrid cad-based construction site layout planning system using genetic algorithms," *Automat Construct*, vol. 6, pp. 749–764, 2003.
[10] G. Ehsan, A. Afshar, and M. R. Jalali, "Site layout optimization with aco algorithm," in *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, 2006, pp. 90–94.
[11] I. N. Papadaki and A. P. Chassiakos, "Multi-objective construction site layout planning using genetic algorithms," *Creative Construction Conference 2016*, pp. 95–03, 2016.
[12] A. M. Adrian, A. Utamima, and K.-J. Wang, "A comparative study of GA, PSO and ACO for solving construction site layout optimization," *KSCE Journal of Civil Engineering*, vol. 19, pp. 520–527, 2014.
[13] N. Xin and W. H. Liu, "Max-min ant system approach for solving construction site layout," *Advanced Materials Research*, vol. 328-330, pp. 217–226, 2011.
[14] V. D. Quang, N. V. Truong, and H. X. Huan, "An improved artificial immune network for solving construction site layout optimization," in *Proceeding of the 12th IEEE-RIVF International Conference on Computing and Communication Technologies*, 2016, pp. 37–42.

Vu Duc Quang is a lecturer in the Department of Information Systems, Faculty of Mathematics at Thai Nguyen University of Education, since 2013. He received the M.Sc. degree in Information Systems at the University of Engineering and Technology, Vietnam National University Hanoi in 2016. He is currently pursuing his Ph.D. research in Taiwan. His research interests include machine learning, soft computing and optimization.

Nguyen Van Truong is a lecturer in the Faculty of Mathematics at Thai Nguyen University of Education, from where he received a Bachelor of Mathematics and Informatics in 2000. He completed his master degree in Computer Science at Vietnam National University Hanoi in 2003. He is currently pursuing his Ph.D. research at the Institute of Information Technology (IOIT), Vietnamese Academy of Science and Technology (VAST). His research interests include embedded systems and artificial immune systems.

Vu Thi Thuy is a second-year student in mathematics as well as a third-year student in informatics at Thai Nguyen University of Education.

Hoang Xuan Huan became a faculty member in the Faculty of Mathematics and Mechanics at the University of Natural Sciences, Vietnam National University Hanoi since 1980, where he obtained his Ph.D. degree in 1994. Since 1995, he became a faculty member in the Faculty of Information Technology, Vietnam National University Hanoi, where he is now an Associate Professor.