Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 5136327, 12 pages
http://dx.doi.org/10.1155/2016/5136327

Research Article

AntStar: Enhancing Optimization Problems by Integrating an Ant System and A* Algorithm

Mohammed Faisal, Hassan Mathkour, and Mansour Alsulaiman

College of Computer and Information Sciences (CCIS), King Saud University, P.O. Box 5117, Riyadh 11543, Saudi Arabia

Correspondence should be addressed to Mohammed Faisal; mfaisal@ksu.edu.sa

Received 13 September 2015; Revised 17 December 2015; Accepted 24 December 2015

Academic Editor: Stéphane Caro

Copyright © 2016 Mohammed Faisal et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, nature-inspired techniques have become valuable to many intelligent systems in different fields of technology and science. Among these techniques, Ant Systems (AS) have become a valuable technique for intelligent systems in different fields. AS is a computational system inspired by the foraging behavior of ants and intended to solve practical optimization problems. In this paper, we introduce the AntStar algorithm, which is swarm intelligence based. AntStar enhances the optimization and performance of an AS by integrating the AS and A* algorithm. Applying the AntStar algorithm to the single-source shortest-path problem has been done to ensure the efficiency of the proposed AntStar algorithm. The experimental result of the proposed algorithm illustrated the robustness and accuracy of the AntStar algorithm.

1. Introduction

Natural swarms have inspired swarm intelligence. Many algorithms and techniques based on swarm intelligence have been developed to solve optimization problems, such as the Ant System (AS) [1, 2], particle swarm optimization (PSO) [3], artificial bee colony (ABC) [4, 5], Firefly Algorithm (FA) [6], and Intelligent Water Drops (IWD) [7, 8]. Dorigo et al. introduced the AS in the 1990s [1, 2]. AS is an optimization algorithm inspired by the foraging behavior of natural ant colonies. In nature, during foraging, ants manage to establish the shortest paths from their nests to food sources by depositing pheromones on the ground as they move. AS uses this same idea. Kennedy introduced PSO, which is inspired by the swarm of bird and fish schools [3] and which simulates the behaviors of bird flocking. ABC was introduced by Karaboga in 2005 [4, 5]. ABC tries to simulate the natural food foraging behavior of real honeybees, as broken into three groups: employed bees, scouts, and onlookers. In ABC, employed bees go to food sources to determine the amount of nectar present there. Next, ABC calculates the probability value of the food sources. The onlooker selects the preferred sources. The scouts are sent to search areas to discover new food sources. The Firefly Algorithm, introduced by Yang in 2009, was inspired by the flashing (flight) behavior of insects [6]. Shah-Hosseini proposed the IWD algorithm in 2007 [7, 8], trying to simulate natural river systems and the interaction between water drops and their environment. During movement, IWD velocity is increased nonlinearly, proportional to the inverse of the soil between the two locations. Soil in IWD is increased by removing some soil from the path between the two locations. This increase is inversely proportional to the time that IWD needs to pass from its current location to the next location. This time, in turn, is proportional to the velocity of the IWD and inversely proportional to the distance between the two locations. Many swarm intelligence systems have been used for path-planning problems. The advantages of such systems include the possibility to add expert knowledge to the search operation and the ability to work with several candidate solutions simultaneously rather than only exploring a single alternative. Głąbowski et al. used Ant Colony Optimization (ACO) to solve the shortest-path problem [9]. Hsiao et al. used ACO to search for the best path of a map [10]. Tan et al. used ACO for real-time planning of the globally optimal path for mobile robots [11]. Montiel et al. solved the path-planning problem of mobile robots using the Bacterial Potential Field (BPF) approach [12]. Contreras-Cruz et al. solved the mobile robot path-planning problem by

combining an artificial bee colony algorithm as a local search procedure with an evolutionary programming algorithm that refines the feasible path found by the set of local procedures [13]. Mo and Xu combined biogeography-based optimization (BBO), PSO, and an Approximate Voronoi Boundary Network (AVBN) to plan the global path of mobile robots in a static environment [14]. Many swarm intelligence algorithms, such as AS, ACO, ABC, and FA, have been used to solve path-planning problems in the field of robotics [15]. In this paper, we propose AntStar, a technique based on swarm intelligence. AntStar enhances the performance of AS by inserting an evaluation function, the A* [16] algorithm, into the transition probability function of AS. This guides the random movements of ants in order that they perform an admissible search from the first iteration. Therefore, AntStar reaches a suboptimal solution early.

This paper is organized as follows. In Section 2, background is presented. The AntStar algorithm is explained in Section 3. Section 4 explains the application of the AntStar algorithm. The application of AntStar to a single-source, shortest-path problem is presented in Section 5; Section 6 explains the experiments with and performance of AntStar. Applying the AS to a single-source, shortest-path problem is presented in Section 7 in order to compare AntStar with AS in Section 8. Section 9 presents the discussion, and the conclusion is given in Section 10.

2. Background

The proposed AntStar algorithm integrates AS and the A* algorithm. Therefore, this section discusses, as follows, the Natural Ant System, the Artificial Ant System, and the A* algorithm.

2.1. Natural Ant System. In nature, ants work in colonies that differ in size. The main work of ants is to forage for food. Many researchers have studied ant behavior [1, 17–19]. Understanding the foraging behavior of natural ant colonies was one of the main problems studied by ethologists. Ants have trail-laying and trail-following behavior when foraging [20]. During foraging, ants manage to establish the shortest paths from their nests to food sources and then return. This is done by laying communications media (chemical substances called pheromones) in varying quantities on the ground, used for communication between individuals. Each ant lays information (pheromones) regarding their paths and decides where to go according to these pheromone trails. The foraging behavior of the natural ant can be described as follows:

(1) At first, ants move randomly and lay down pheromone on the ground.

(2) If a food source is discovered, the ants return to the nest and lay down a pheromone trail.

(3) If a pheromone is discovered during movement, the probability of following the pheromone trail will increase.

(4) If an ant reaches the nest, it goes again to search for a food source.

The collective behavior of ants is *autocatalytic*: the more the ants follow a trail, the more the trail becomes attractive to follow [21]. This operation can be characterized as positive feedback, where the probability of choosing a path increases with the number of ants that have before chosen the same path [21]. Consider Figure 1(a) [21]. The ants walk from nest A to food source E. The path the ants followed when free of obstacles is in Figure 1(a). Unexpectedly, an obstacle appears in the path, breaking it off, as in Figure 1(b). Thus, the ants at place B (ants walking from nest A to food source E) or at place D (ants walking from food source E to nest A in the opposite direction) must decide whether to go left or right. This decision is influenced by the degree of pheromones left by the preceding ants.

As we can see, more pheromone on a route provides an ant with stronger motivation and higher probability to choose that route. At the beginning, the first ant at place B (or D) in Figure 1(b) moves randomly, laying down pheromones without caring about the lack of previous pheromones. An ant that uses the route BCD will reach D before the first ant on the route BHD (Figure 1(c)), since BCD is shorter than BHD. As a result, more ants returning from the food source E to nest A via D will select DCB, because it has stronger, less-decayed pheromones than DHB. This in turn causes the amount of pheromones on BCD (the shorter path) to grow faster than on the DHB (longer one).

2.2. Artificial Ant System. The artificial AS simulates the foraging behavior of real ant colonies as follows.

Consider an ant colony looking for food like solving a problem. Consider N ants as N solutions. In each tour, each ant looks for food and returns to the nest, representing one solution. An artificial AS is derived from the Natural Ant System in this way, with these major differences [21]:

(1) Artificial ants are designed with memory capacity.

(2) Artificial ants are not completely blind like real ants.

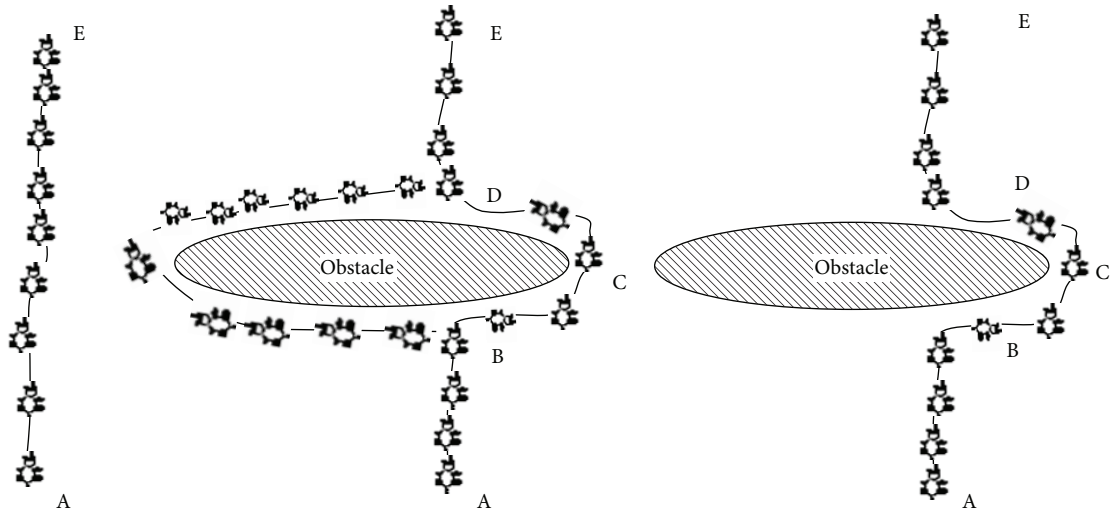
(3) Working time is discrete.

AS uses a swarm of computer agents (a swarm of ants), each agent simulating a real ant. During the movement (search) of each agent, the ants manage to establish the shortest paths from their nest to the food source (source to destination) using the transition probability and trail intensity. The transition probability of an artificial ant (agent) from node i to node j for the kth ant is given by (1) below [1] and in Figure 2:

$$P_{ij}^{k}(t) = \frac{\tau_{ij}^{\alpha}(t) * \eta_{ij}^{\beta}}{\sum_{k \in \text{allowed}_k} \tau_{ik}^{\alpha}(t) * \eta_{ik}^{\beta}}, \quad (1)$$

where η is a heuristic value on edge (i, j), $\tau_{ij}(t)$ is the intensity of trail on edge (i, j) at time t, and α and β are parameters that control the importance of the pheromone versus a heuristic. The trail intensity equation of edge (i, j) is [1]

$$\tau_{ij}(t + n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij} \quad \text{if ANT-cycle is used} \quad (2a)$$



(a) Ants walk on a route between nest A and food source E

(b) An obstacle appears and the ants try to avoid it

(c) Ants select the shorter route

Figure 1: An example of Natural Ant System [6].

or

$$\tau_{ij}(t+1) = \rho * \tau_{ij}(t) + \Delta\tau_{ij} \quad \text{otherwise} \tag{2b}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2c}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if the } k\text{th ant used the edge } (i,j) \text{ in the current tour} \\ 0 & \text{Otherwise,} \end{cases} \tag{2d}$$

where ρ is an evaporation coefficient of the trail between time t and $(t + n)$ and $\Delta\tau_{ij}$ is the laying quantity of pheromone on edge (i, j) by the kth ant over the period of time $(t, t + n)$. Moreover, Q is a constant, and L_k is the tour length of the kth ant. This updating evaporation of the pheromone allows an indirect form of communication called *stigmergy*. Dorigo et al. [1, 21] tested the artificial AS with the well-known travelling salesman problem (TSP) [22]. Given a graph (N, E), a set of cities N, and a set of edges E, the TSP is defined as the problem of finding a path with minimal length that visits each city once. The distance between cities (i, j) is given by the Euclidean distance

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}. \tag{3}$$

Let $M_i(t)$ be the number of ants in town i at time t, with $i = 1$ to N, where N is number of towns. M is the total number of ants.

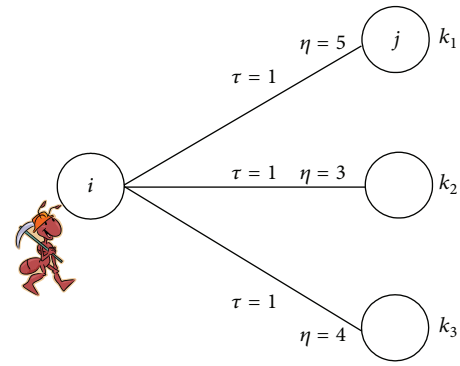


Figure 2: Transition probability of an artificial ant.

As we said before, each ant is considered an agent with these properties as follows:

(A) An agent selects a town to move to using a probability function $P_{ij}^{k}(t)$ (see (1)), which depends on the amount of trail on the edge τ_{ij} and the town distance η_{ij}.

(B) Going to visited towns is disallowed using a tabu list.

(C) After each tour, an agent lays a substance called a *trail* on each edge of its trip $\tau_{ij}(t + n)$ using (2a), (2b), (2c), and (2d).

2.3. A Algorithm.* A* algorithm is an extension of the Dijkstra algorithm [23] described in 1968 by Hart et al. [16]. It is a searching algorithm incorporating heuristic information

about the problem domain into a formal mathematical theory of graph searching to find a good enough (perhaps not the best) path from point to point. The admissible searching property is used by A* to expand the fewest nodes in the searching operation of the optimal path. A* evaluates all accessible nodes using the evaluation function $f(n)$ to decide which one should be expanded next [16]. The evaluation function $f(n)$ is defined as

$$f(n) = g(n) + h(n), \tag{4}$$

where $g(n)$ is the past path-cost function (actual cost) of an optimal path from source s to current node n and where $h(n)$ is the future path-cost function (heuristic cost) of an optimal path from current node n to the destination [16]. To maintain the admissibility of A*, the estimates of $h(n)$ must not be higher than the lowest possible cost [16].

The A* algorithm uses two lists (OPEN and CLOSED) to maintain the operation. The OPEN list saves all available nodes that were not yet accessed. The CLOSED list saves all visited nodes. The OPEN list is sorted according to $f(n)$, and A* uses the OPEN list to select the next node to visit with the minimum cost to access. Gradually, A* finds the relatively shortest path from the source to the destination. Let us consider a connected subgraph G with start node s and $T \sqsubseteq G$ as goal nodes. The A* algorithm is summarized below.

Search algorithm A* [16] is as follows:

(1) Insert start node s to the open list

$$\text{Open\_List}(1) = s. \tag{5}$$

(2) Calculate the evaluation function:

$$f(s) = g(s) + h(s). \tag{6}$$

(3) Select the node n with the minimum ($f(n)$) from the Open_List().

(4) If ($n \in T$),

insert n to the closed list Closed _List(),
terminate the algorithm.

(5) Else

insert n to the Closed _List();
calculate evaluation function for each successor and insert them to open list;
go to Step (2).

3. Proposed AntStar Algorithm

AntStar is a proposed algorithm that enhances the optimization and the performance of AS by integrating the AS with the A* algorithm, gaining advantages from both. AntStar inserts the evaluation function of A* into the transition probability function of AS. Therefore, AntStar is able to improve on the performance of the AS by combining the collective behavior and self-organization of AS with the evaluation function of the A* algorithm. So, at the beginning (the first iteration) of the AntStar algorithm, no pheromones are available in the search area. However, the evaluation function of AntStar guides the random movements of ants to perform an admissible search from the first iteration, while in AS each ant moves randomly with their only guide of the heuristic function of AS. Actually, this integration gives AntStar the following properties:

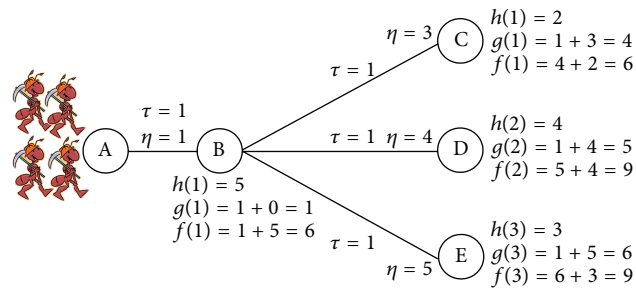


Figure 3: Transition probability of AntStar.

(1) *Multiagents*. They take advantage of multiagents from the AS.

(2) *Stochastic Best-First-Search*. It takes advantage of the best-first-search from A* using the A* evaluation function $f(n)$. In addition, AntStar inherits the probabilistic advantage of solving computational problems from the AS.

(3) *Collective Behavior and Self-Organization*. They take advantage of swarm intelligence from the AS.

The AntStar algorithm works as follows.

During the movement (search) of each artificial ant (agent) in AntStar, the agents manage to establish the shortest paths from their nest to the food source (source to destination) using the transition probability and trail intensity equation. AntStar builds its solution by repeatedly applying a stochastic, best-first-search rule using the transition probability rule. Agents prefer to move to nodes that are connected by a short edge and that have a larger amount of pheromone. The transition probability of an agent at node i to node j for the kth ant is obtained by integrating the evaluation function of A* and the transition probability, as in (7) below and in Figure 3:

$$P_{ij}^{k}(t) = \frac{\tau_{ij}^{\alpha}(t) * f(j)_{ij}^{\beta}}{\sum_{k \in \text{allowed}_k} \tau_{ik}^{\alpha}(t) * f(j)_{ik}^{\beta}}, \tag{7}$$

where

$$f(j) = g(j) + h(j), \tag{8}$$

where $g(j)$ is the past path-cost function (actual cost) of an optimal path from source s to current node j at time t, $h(j)$ is the future path-cost function (heuristic cost) of an optimal path from current node j to the destination, and

$\tau_{ij}(t)$ is the intensity of the trail on edge (i, j) at time t. $\tau_{ij}(t)$ provides evidence of how many ants have chosen that same edge in the past. α and β are parameters that control the importance of the pheromone versus the heuristic. Once all agents have completed their tours and reached their targets, the pheromones are updated on all edges according to the trail intensity given by (2a), (2b), (2c), and (2d) above [1].

4. Applications of the AntStar Algorithm

AntStar is a swarm intelligence algorithm applicable to any problems that can be addressed by such algorithms (e.g., AS, ACO, ABC, PSO, and IWD) or by the A* algorithm, including the single-source, shortest-path problem [24], travelling salesman problem (TSP) [22], N-Queen Puzzle [25], or Multiple Knapsack Problem (MKP) [26]. Any problem the state spaces of which can be represented as graphs may benefit from the AntStar algorithm.

In fact, AntStar can be adapted to other kinds of problems. In the single-source, shortest path and TSP, AntStar uses the distances between vertices (cities) in the evaluation function $f(n)$ of the transition probability (7). On the other hand, in the N-Queen Puzzle, AntStar's evaluation function $f(n)$ of the transition probability represents not the distance between two vertices, but rather the positions on the chessboard where queens will kill each other if placed. In the MKP, AntStar uses the benefit and the capacity values in the evaluation function $f(n)$ of the transition probability (7); the allowed set K will be the remaining feasible items. The AntStar algorithm is adapted to a wide range of problems through the proper usage of the evaluation function.

As detailed in Section 5, we experimented with AntStar in the single-source, shortest-path problem. In the TSP problem, which is represented by a graph $G(V, E)$ with $|V|$ nodes and $|E|$ edges [22], the AntStar algorithm is applied by considering the cities as the nodes of the graph, with the links between cities as edges. Each link has a distance value, and each city has a specific location. Using this information, the AntStar algorithm can be directly applied to the TSP problem.

Another example application is the N-Queen Puzzle [25]. Khan et al. [27] and Shah-Hosseini [7] introduced ACO and IWD as methods to solve the N-Queen Puzzle with a decreased search space. The AntStar algorithm, IWD, and ACO all have the same working principle, making it possible for AntStar to solve the N-Queen Puzzle in a similar manner.

In the knapsack problem, there are several items, each with a weight, a profit, and a knapsack [26]. The aim is to put some of the items in the knapsack for sale in order to obtain maximum profit such that the total weight is less than or equal to a given limit. In the multiple knapsacks (MKP) problem, there are many items (i) and multiple knapsacks (m). The aim is to put some of the items in the knapsacks for sale in order to obtain maximum profit $b_i \operatorname{Max}(\sum y_i * b_i)$ with the condition that none of the knapsacks is overflowing. Many algorithms have been developed to solve the MKP, such as [28], where the MKP is represented as a graph G with V and T edges: (V, T). The vertices V denote the items, while T represents the arcs (paths) between the items. Since AntStar and IWD have the same working principle, it may be deduced that AntStar is applicable to the MKP problem.

5. Applying AntStar to the Single-Source Shortest-Path Problem

As mentioned above, AntStar is a search algorithm that integrates the A* algorithm with AS. In this section, we apply the AntStar algorithm to the single-source shortest-path problem. In graph theory, a connected graph $G = (V, E)$ is a directed graph with V vertices, E edges, and weight function w: $E \text{-}\gg R$. Each path has a weight that is the sum of the weights of each edge in the path $p = (v_0, v_1, \ldots, v_k)$ [24]:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i). \tag{9}$$

The shortest-path weight from u to v is defined by

$$\partial(u, v) = \begin{cases} \min\left\{w(p) : u \xrightarrow{p} v\right\} & \text{if there is a path between } u \text{ and } v \\ \infty & \text{otherwise.} \end{cases} \tag{10}$$

The shortest path between vertex u and vertex v is defined as any path p with the weight

$$w(p) = \partial(u, v). \tag{11}$$

The application of the AntStar algorithm to a single-source shortest-path problem is described by the flowchart in Figure 4.

Algorithm 1 details the steps of the AntStar algorithm with the shortest-path problem.

6. Experimentation Processes

Various scenarios were devised to test the performance of the proposed AntStar algorithm on the shortest-path problem. These experimental results determined the robustness, accuracy, adaptability, and efficiency of the proposed algorithm. The proposed algorithm was tested successfully many times in the search space with different configurations that were as big as 100 square meters (10 m × 10 m). We divided the search space into 100 nodes, each one square meter in size, so the length of each edge within each node is one meter, as illustrated in Figure 5.

(1) Initialize:
  Set $t \coloneqq 0$
  Set NC $\coloneqq 0$ {NC is the cycles counter}
  For every edge (i, j) set an initial value $\tau_{ij}(t) = c$ $(c = 0.1)$ for trail intensity and $\Delta\tau_{ij} = 0$
(2) Place the m ants on the source node
(3) Set $s \coloneqq 1$ {s is the tabu list index}
  For $k \coloneqq 1$ to m do
    Place the starting town of each ant in $\text{tabu}_k(s)$
(4) Repeat until each ant reaches the target or in local minima
  Set $s \coloneqq s + 1$
  For $k \coloneqq 1$ to m do
    // at time t the kth ant is in town i
    Calculate the evaluation function of each neighbor j
      $f(j) = g(j) + h(j)$
    Choose the town j to move to, with probability

$$P_{ij}^{k}(t) = \frac{\tau_{ij}^{\alpha}(t) * f(j)_{ij}^{\beta}}{\sum_{k \in \text{allowed}_k} \tau_{ik}^{\alpha}(t) * f(j)_{ik}^{\beta}}$$

    Move the kth ant to the selected town
    Insert town j in $\text{tabu}_k(s)$
(5) For $k \coloneqq 1$ to m do
    Compute the length L_k
    Update the shortest tour found
    For every edge (i, j)
      For $k \coloneqq 1$ to m do

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if the } k\text{th ant used the edge } (i, j) \text{ in the curent tour} \\ 0 & \text{Otherwise} \end{cases}$$

(6) For every edge (i, j)
    Compute $\tau_{ij}(t + n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij}$
(7) *Set* $t \coloneqq t + n$
  Set NC $\coloneqq$ NC + 1
  For every edge (i, j)
    Set $\Delta\tau_{ij} = 0$
(8) If (NC < NCMAX) and (not stagnation behavior) then
    Empty all tabu lists
    Go to step (2)
Else
    Print shortest tour
    Stop

Algorithm 1: AntStar algorithm with shortest-path problem.

Table 1: Beta setting.

β	0	1	2	**5**	10	20
α	1	1	1	1	1	1
Shortest path	17.071	13.656	13.071	12.485	12.485	12.485

Table 2: Alpha setting.

β	5	5	5	**5**	5	5
α	0	0.5	1	2	5	10
Shortest path	12.485	12.485	12.485	13.071	13.071	13.071

6.1. Selection of Parameters. Parameter values indirectly affect the performance of AS, as well as the proposed AntStar. In this section, we will discuss the effectiveness and sensitivity of tuning the parameters α and β of the probability equation (7). We tested several values for each parameter. First, we fixed α of AntStar based on many studies [1, 2, 21] and then tested AntStar with several values of β (Table 1).

As shown in Table 1, the length of the path decreases until $\beta = 5$, after which the length becomes fixed. Similarly, then, we fixed $\beta = 5$ and tested AntStar with several values for α (Table 2).

As shown in Table 2, the length of the path is fixed (optimal) up to $\alpha = 1$, after which the length becomes fixed at a higher value (13.071). Therefore, in our investigation in Sections 6.2 and 7, we set $\beta = 5$ and $\alpha = 1$.

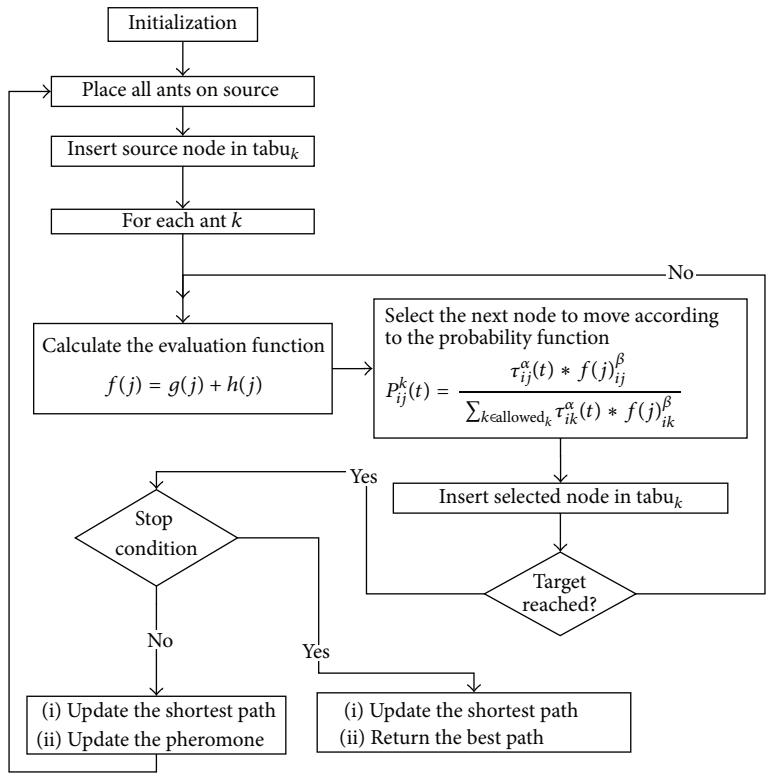


Figure 4: AntStar algorithm with shortest-path problem.

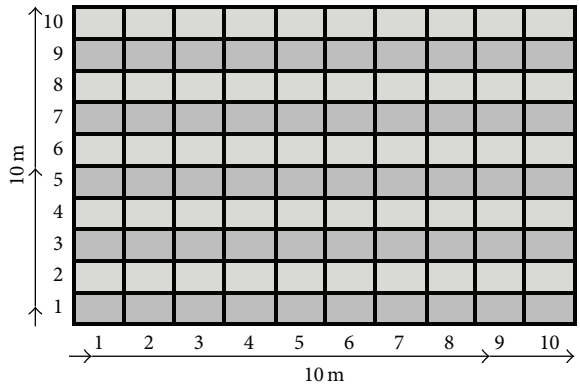
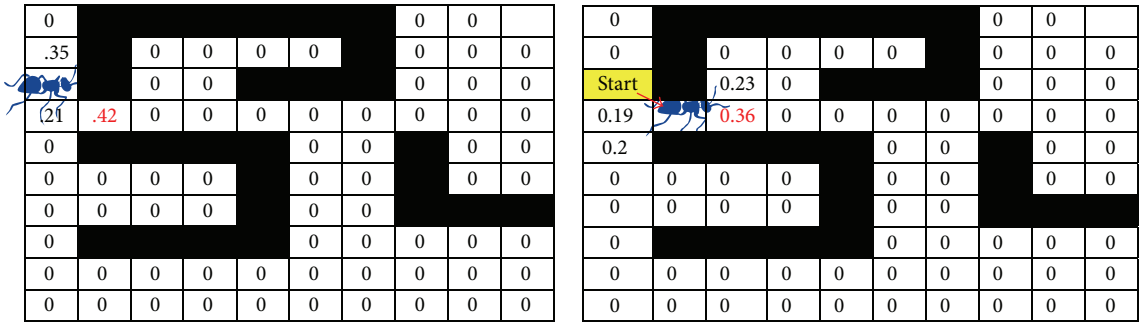


Figure 5: Search space of experiments.

6.2. Experiments Results. The parameters of AntStar were set based on our experiments and many studies [1, 2, 21] as $\alpha = 1$, $\beta = 5$, $p = 0.5$, and $c = 0.1$. We used 10 ants and 10 iterations to test the AntStar algorithm on the shortest-path problem. As we mentioned, the movement of each ant from node i to node j depends on the transition probability function (see (7)), which is influenced by actual cost, heuristic cost, and pheromone trail values. Two scenarios using AntStar are discussed in this section. The source node of the first scenario is at 1 m, 8 m, with target node at 10 m, 10 m. The source node of the second scenario is at 5 m, 5 m, with target node at 7 m, 5 m. The probability values of movement for one ant in the first scenario are illustrated in Figure 6. Figure 6(a) illustrates the probability values of first movement. As can be seen, there are three different probability values for movement; the ant decides to move toward the node with the highest value (0.424). As we will see, it is not necessary to move toward the node with the highest value. Figure 6(b) illustrates the probability values for the second movement. As can be seen, there are four different probability values for movement; again, the ant decides to move toward the node with the highest value (0.3689). As can be seen in Figures 6(c) and 6(d), the ant decided not to move toward the node with the highest value probability due to stochastic movement, which helps AntStar to explore the search space. Work continues in this manner until the ant reaches the target.

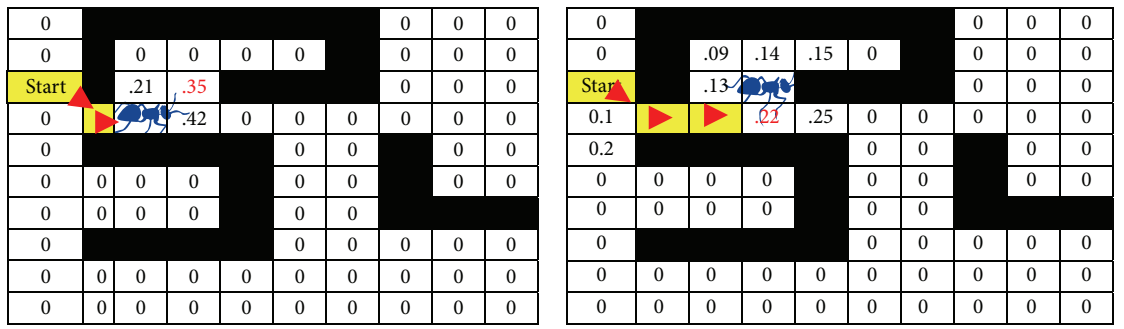
A snapshot of the results of different paths in the experiment of the first scenario with AntStar is displayed in Figure 7. Figure 7 illustrates the output paths of the first scenario using the AntStar algorithm on the shortest-path problem. As we can see, it displays four different paths, each of different length: (1) 16.49 m; (2) 13.07 m; (3) 13.07 m; and (4) 12.485 m, which is the best. Those paths are the best paths of the first four iterations of the algorithm. The shortest path found using the AntStar algorithm in this scenario is plotted in Figure 8.

As AntStar executed with 10 ants and 10 iterations, at each iteration 10 different paths were generated, one path per ant. The best path of each iteration is a candidate for the best path. Therefore, Figure 7 showed only the best path of each iteration. Figure 9 illustrates the length of the shortest path of each iteration in this scenario.



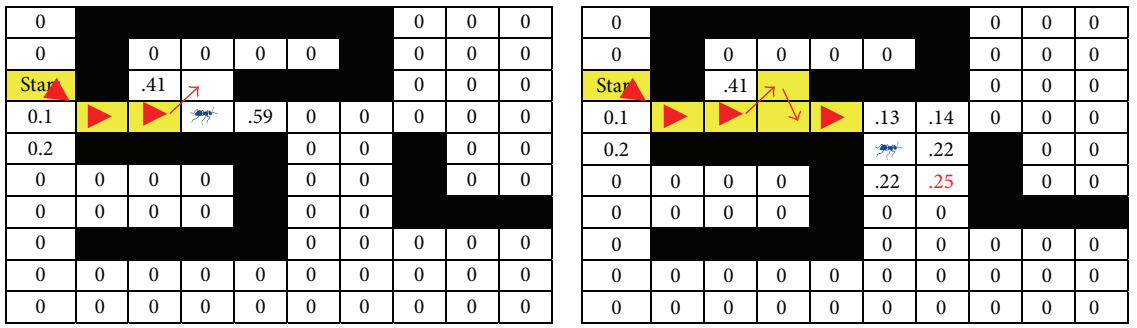
(a) Probability values of first movement

(b) Probability values of second movement



(c) Probability values of third movement

(d) Probability values of fourth movement



(e) Probability values of fifth movement

(f) Probability values of another movement

Figure 6: Movement of one ant with the probability values at each movement (first scenario).

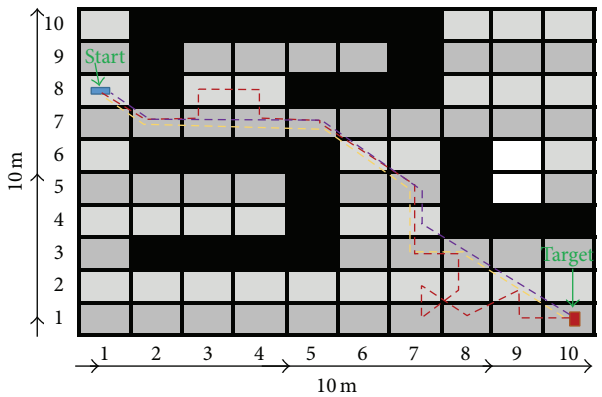


Figure 7: A snapshot of multiple paths of the AntStar algorithm applied to the shortest-path problem (first scenario).

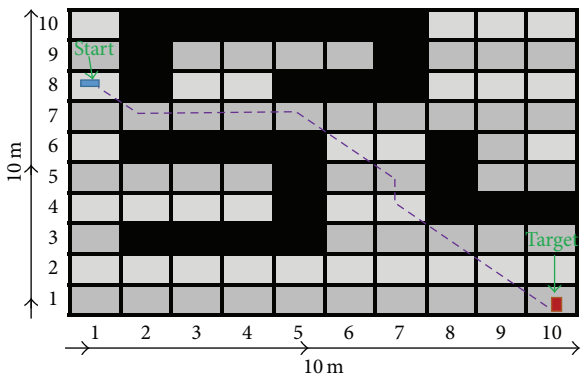


Figure 8: The shortest path of the AntStar algorithm applied to the shortest-path problem.

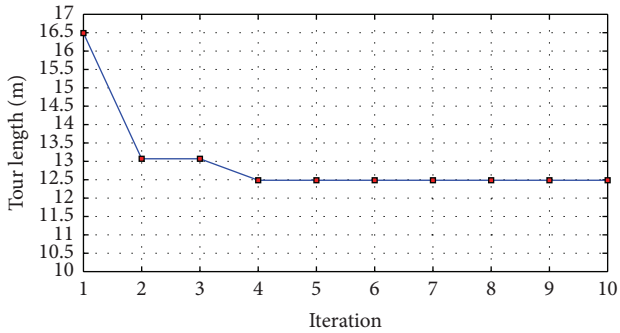


Figure 9: The shortest path of each iteration (first scenario).

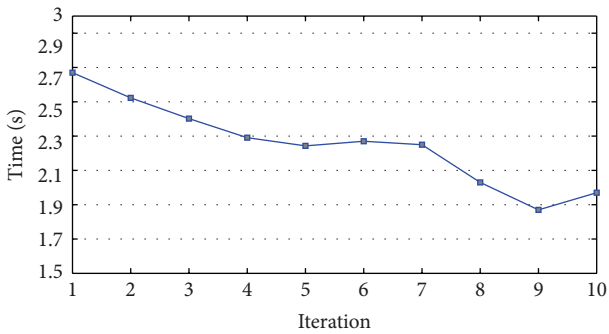


Figure 10: Time of each iteration (first scenario).

At the beginning, no pheromones are available in the search area. Therefore, upon the first iteration of AntStar, each ant moves randomly, constrained by the evaluation function $f(j)$ (7). As we can see in Figures 7 and 9, on the first iteration the best path was of length 16.49 m. The best path of the second and third iterations was enhanced to length 13.07 m. We executed the A* algorithm using the same search space, configuration, start point, and target point to determine the shortest path of the searching space. Therefore, we know the shortest path of this configuration. On the fourth iteration, AntStar found the best path in the search space, of length 12.485 m. This quick enhancement was due to the collective behavior, self-organization, and evaluation function of the AntStar algorithm. Over the rest of the iterations, there was no change in the length of the found path, as we can see in Figure 9. The time of finding the best path of each iteration is illustrated in Figure 10. The output paths of the second scenario using the AntStar algorithm on the shortest-path problem are illustrated in Figure 11. As we can see, three different paths are displayed, each of different length: (1) 16 m; (2) 13.2 m; and (3) 10.24 m, the shortest path. Those paths are the best paths of all the iterations of the algorithm. Let us recall that AntStar executes with 10 ants and 10 iterations, so, at each iteration, 10 different paths are generated, one path per ant. Figure 12 illustrates the length of the shortest path of each iteration in this scenario. The time of finding the best path of each iteration is illustrated in Figure 13.

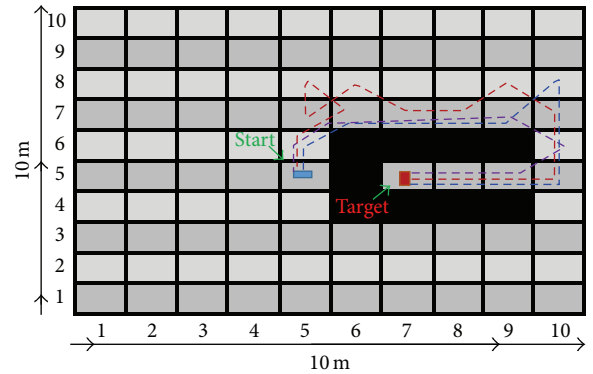


Figure 11: A snapshot of multiple paths of the AntStar algorithm applied to the shortest-path problem (second scenario).

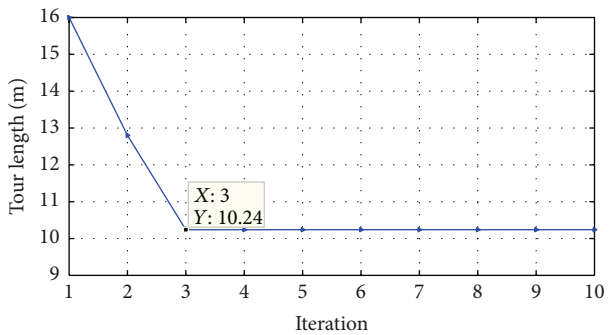


Figure 12: The shortest path of each iteration (second scenario).

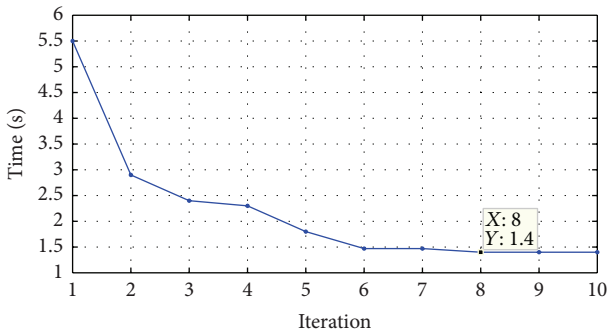


Figure 13: Time of each iteration (second scenario).

7. AS Applied to the Shortest-Path Problem

In this section, we apply the AS algorithm to the shortest-path problem for the same 100-square meter (10 m × 10 m) search space and configuration as we used with AntStar and setting the parameters of the AS the same as AntStar, $\alpha = 1$, $\beta = 5$, $p = 0.5$, and $c = 0.1$. We used 10 ants and 10 iterations to test the AS system on the shortest-path problem. A snapshot of the results captured from some of these experiments is displayed in Figure 14, illustrating the output paths of the AS system applied to the shortest-path problem over 10 iterations. As we can see, it displays many

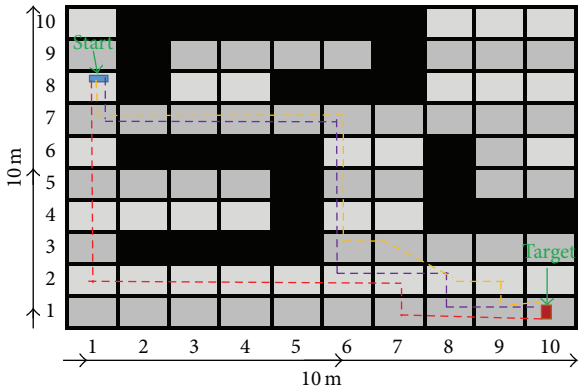


Figure 14: A snapshot of multiple paths of the AS system applied to the shortest-path problem.

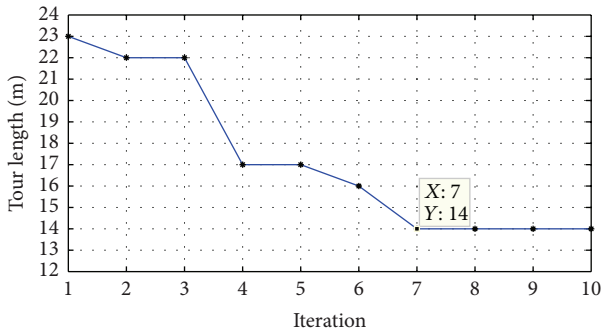


Figure 15: The shortest path of each iteration using AS.

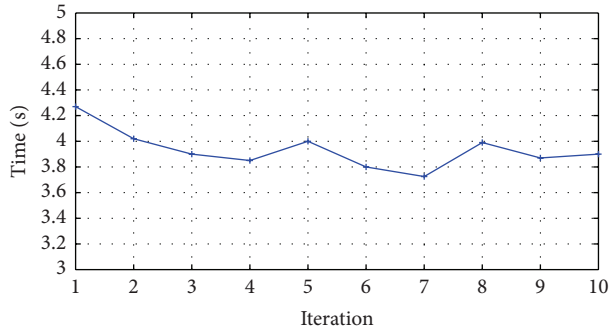


Figure 16: Time of each iteration using AS.

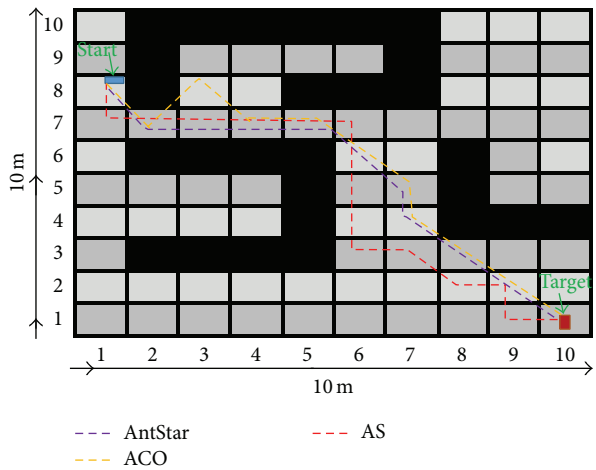


Figure 17: Shortest paths using the AntStar, ACO, and the AS algorithms.

different paths, each of different length: (1) 17 m; (2) 16 m; and (3) 14 m, the best in this experiment. Those paths are the best from the 10 iterations of the AS. Figure 15 illustrates the lengths of the best 10 paths of each iteration. The time cost of each iteration is illustrated in Figure 16.

8. Comparison

In this section, we compare the proposed AntStar system with AS and ACO as applied to the shortest-path problem using the same search space and configuration. The comparison is from an experimental point of view, so it includes traveled distance in meters, used path, and execution time. Plots of the results of the used path and traveled distance using the AntStar algorithm and AS are shown in Figure 17. We executed the AntStar algorithm and AS with the same start point (1 m, 8 m) and target point (10 m, 10 m). As can be seen, the travel distance of AntStar is 12.485 m for the given search space and configuration, while AS and ACO produced another path and travel distance (14 m and 13.31 m, resp.). We executed the A* algorithm using the same search space, configuration, start point, and target point to determine the shortest path. Therefore, we know the shortest path of this configuration.

AntStar, AS, and ACO are competitive in finding the path operation, but in terms of execution time and traveled distance the AntStar prevailed in the competition. Let us recall that AntStar, AS, and ACO have been executed for ten iterations each with ten ants. The best (minimum) time was that of AntStar (1.85 sec), while the best time of AS is 3.7 and that of ACO is 1.9. The best average time was that of AntStar (2.241 sec) while the average time of AS was 3.932 and that of ACO was 3.03. The best-traveled distance was that of AntStar (12.485 m), while the best-traveled distance of AS was 14 m and that of ACO was 13.31 m. The best average traveled distance was that of AntStar (13.05 m), while the best average traveled distance of AS was 17.3 and that of ACO was 14.36. Actually, AntStar, AS, and ACO had a different execution time, different traveled distance, and path for each iteration as illustrated in Figures 18 and 19. Detailed performance indexes are listed in Table 3.

9. Discussion

From the obtained experimental results of the AntStar and AS applied to the shortest-path problem, we focus on three performance indexes: the traveled distance, the used path, and, finally, the time to reach the target.

Used Path and Traveled Distance. As illustrated in Figure 17 and in Table 3, AntStar, AS, and ACO all succeeded in reaching the target from the initial point. In terms of execution

Table 3: Detailed performance indexes of AntStar and AS.

Measurement		AntStar	AS	ACO $q_0 = 0.5$
Traveled distance of 10 iterations (meter)	Min	**12.485**	14	13.31
	Max	16.49	23	18.14
	Mean	13.05	17.3	14.36
	Std.	1.61	3.683	1.77
Execution time of 10 iterations (seconds)	Min	**1.85**	3.7	1.9
	Max	2.7	4.3	4.69
	Mean	2.241	3.932	3.03
	Std.	0.262	0.150	1.02
Used path in 10 iterations	Figure 17	Shortest	Not good	Not good

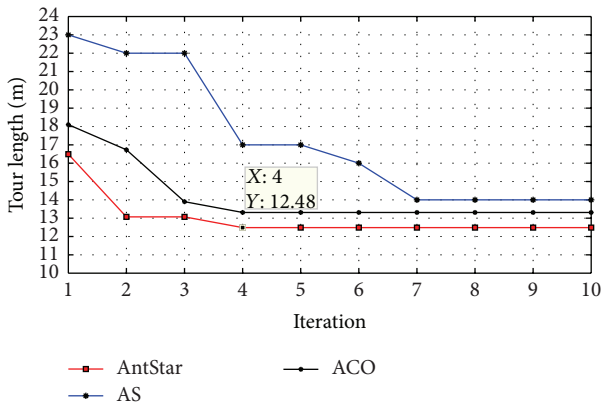


Figure 18: Comparison between the shortest paths for each iteration using AntStar, AS, and ACO.

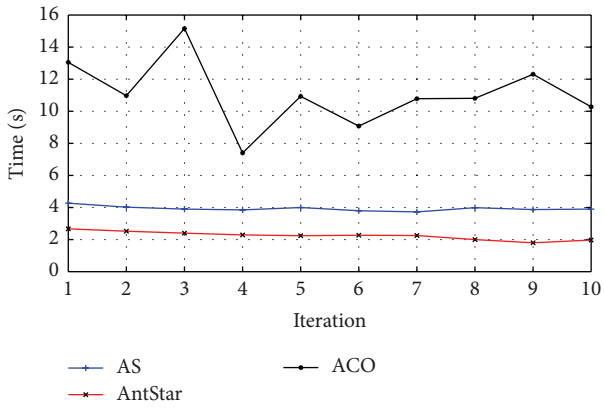


Figure 19: Comparison of elapsed times for each iteration using AntStar, AS, and ACO.

time and traveled distance, however, AntStar prevailed in the competition. Let us recall that AntStar, AS, and ACO were all executed for ten iterations, each with ten ants. The minimum traveled distance was AntStar's (12.485 m), compared to 14 m for AS and 13.31 m for ACO. As can be seen in Figure 18, upon the first iteration, the best path of all ants using AntStar was 16.49 m long, while the best path of AS was 23 m long and that of ACO was 18.14 m long. The best path on the second and third iterations using AntStar was enhanced to a length of 13.07 m, while AS on the second and third iterations had best-path lengths of 22 m and that of ACO was 16.14 m. On the fourth iteration, AntStar found the best path in the search space, which is of length 12.485 m, while AS found its best path (which is not the best for the given search space) on the seventh iteration and ACO found its best path on the fourth iteration. This quick enhancement of AntStar was due to the collective behavior, self-organization, and evaluation function of the AntStar algorithm. So, at the beginning (on the first iteration) of the AntStar, AS, and ACO, no pheromones were available in the search area, but the evaluation function of AntStar guides the random movements of ants to perform an admissible search from the first iteration, while, in AS and ACO, each ant moves randomly with their only guide of the heuristic function.

Time to Arrive at Target. Travel distance is linearly related to time for any fixed speed. In all experiments, the ants of AntStar realized the best time (1.85 sec) to attain the target, as the probability function, which includes the evaluation function, was better in terms of guiding the ants and had an optimal control strategy.

10. Conclusion

In this paper, we proposed the AntStar algorithm, which enhances the optimization and the performance of AS by integrating AS with the A* algorithm. As we have seen, applying the AntStar algorithm to the single-source, shortest-path problem demonstrated its efficiency. The experimental results showed the quick enhancement of AntStar, which is due to the evaluation function, the collective behavior, and the self-organization of the AntStar algorithm. Further, these results illustrate the robustness and accuracy of the AntStar algorithm on the single-source, shortest-path problem, which encourages us to apply AntStar to other problems, such as the TSP, N-Queen Puzzle, and MKP.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This project was funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia (Award no. 12-MED2474-02).

References

[1] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[2] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*, pp. 36–39, Springer, 2010.

[3] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, New York, NY, USA, 2010.

[4] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[5] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. tr06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[6] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.

[7] H. Shah-Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm," *International Journal of Bio-Inspired Computation*, vol. 1, no. 1-2, pp. 71–79, 2009.

[8] H. Shah-Hosseini, "Problem solving by intelligent water drops," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 3226–3231, IEEE, Singapore, September 2007.

[9] M. Głąbowski, B. Musznicki, P. Nowak, and P. Zwierzykowski, "Shortest path problem solving based on ant colony optimization metaheuristic," *Image Processing & Communications*, vol. 17, pp. 7–17, 2012.

[10] Y.-T. Hsiao, C.-L. Chuang, and C.-C. Chien, "Ant colony optimization for best path planning," in *Proceedings of the IEEE International Symposium on Communications and Information Technologies (ISCIT '04)*, vol. 1, pp. 109–113, IEEE, Sapporo, Japan, October 2004.

[11] G.-Z. Tan, H. He, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279–285, 2007.

[12] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177–5191, 2015.

[13] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Applied Soft Computing Journal*, vol. 30, pp. 319–328, 2015.

[14] H. Mo and L. Xu, "Research of biogeography particle swarm optimization for robot path planning," *Neurocomputing*, vol. 148, pp. 91–99, 2015.

[15] N. S. Pal and S. Sharma, "Robot path planning using swarm intelligence: a survey," *International Journal of Computer Applications*, vol. 83, no. 12, pp. 5–12, 2013.

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[17] J.-L. Deneubourg, J. M. Pasteels, and J.-C. Verhaeghe, "Probabilistic behaviour in ants: a strategy of errors?" *Journal of Theoretical Biology*, vol. 105, no. 2, pp. 259–271, 1983.

[18] J. L. Deneubourg and S. Goss, "Collective patterns and decision-making," *Ethology Ecology & Evolution*, vol. 1, no. 4, pp. 295–311, 1989.

[19] S. Goss, R. Beckers, J.-L. Deneubourg, S. Aron, and J. M. Pasteels, "How trail laying and trail following can solve foraging problems for ant colonies," in *Behavioural Mechanisms of Food Selection*, vol. 20 of *NATO ASI Series*, pp. 661–678, Springer, Berlin, Germany, 1990.

[20] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Oxford, UK, 1999.

[21] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the 1st European Conference on Artificial Life*, pp. 134–142, Paris, France, December 1991.

[22] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, pp. 61–75, 1956.

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

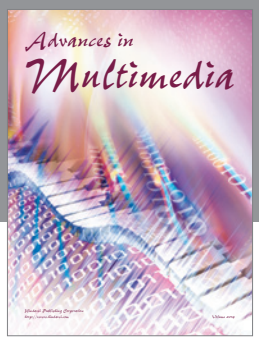
[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Mass, USA, 2001.

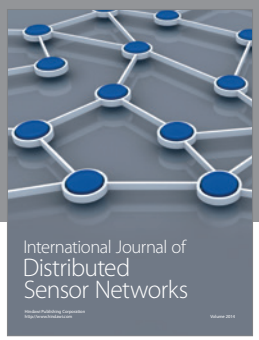
[25] J. J. Watkins, *Across the Board: The Mathematics of Chessboard Problems*, Princeton University Press, Princeton, NJ, USA, 2007.

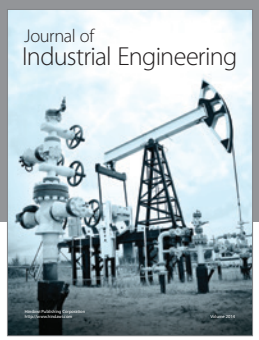
[26] H. Kellerer, U. Pferschy, and D. Pisinger, *Introduction to NP-Completeness of Knapsack Problems*, Springer, 2004.

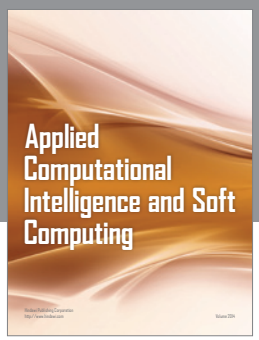
[27] S. Khan, M. Bilal, M. Sharif, M. Sajid, and R. Baig, "Solution of n-Queen problem using ACO," in *Proceedings of the IEEE 13th International Multitopic Conference (INMIC '09)*, pp. 1–5, IEEE, Islamabad, Pakistan, December 2009.

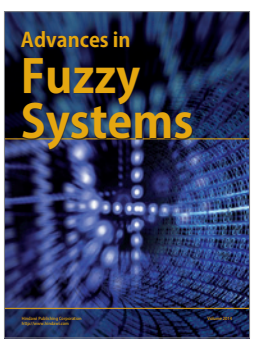
[28] H. Shah-Hosseini, "Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem," *International Journal of Intelligent Computing and Cybernetics*, vol. 1, no. 2, pp. 193–212, 2008.

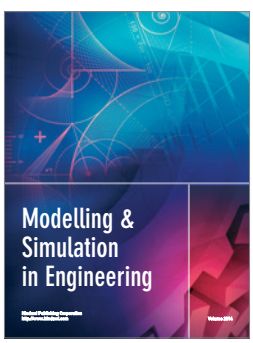


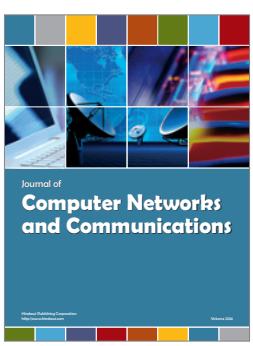












Hindawi
Submit your manuscripts at
http://www.hindawi.com