An Improved Ant Colony Approach for the Competitive Traveling Salesmen Problem

Xinyang Du[1], Ruibin Bai[1], Tianxiang Cui[1], Rong Qu[2] and Jiawei Li[1*]

1 School of Computer Science, University of Nottingham Ningbo China.
2 School of Computer Science, University of Nottingham, UK.
** Email: jiawei.li@nottingham.edu.cn, ORCID: 0000-0003-4685-2615*

2022 IEEE Congress on Evolutionary Computation (CEC) | 978-1-6654-6708-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CEC55065.2022.9870414

***Abstract*—A competitive traveling salesmen problem is a variant of traveling salesman problem in that multiple agents compete with each other in visiting a number of cities. The agent who is the first one to visit a city will receive a reward. Each agent aims to collect as more rewards as possible with the minimum traveling distance. There is still not effective algorithms for this complicated decision making problem. We investigate an improved ant colony approach for the competitive traveling salesmen problem which adopts a time dominance mechanism and a revised pheromone depositing method to improve the quality of solutions with less computational complexity. Simulation results show that the proposed algorithm outperforms the state of art algorithms.**

***Index Terms*—Ant colony, competitive traveling salesmen problem, heuristic, algorithm**

I. Introduction

As a classical combinatorial optimization problem, the Traveling Salesman Problem (TSP) has been heavily investigated. In a TSP, an agent is to visit a given set of cities with the minimum traveling distance. Each city should be visited exactly once. Both exact and approximate methods have been developed for multiple variants of TSP [2], [12].

A competitive traveling salesmen problem (CTSP) is a variant of TSP first proposed in 2004 [1]. Multiple agents compete in visiting a number of cities to maximize individual payoffs in a CTSP. A agent receives a reward after visiting a city that has not been visited by other agents. The cost of traveling is proportional to the distance travelled. The payoff of each agent is the total reward received in visiting cities minus the cost of travelling. Each agent aims to maximize his/her payoff and they are assumed to be non-cooperative in the competition.

Different from classical TSP, the solution to a CTSP is an equilibrium state in which multiple agents could not improve their payoffs by unilaterally changing their strategies (Nash equilibrium for non-cooperative games). A CTSP is a dynamic decision making problem that is difficult to solve because of the interactions among the agents' strategies [11]. Rather than computing the equilibrium, it is practical to find dominant strategies for an individual agent. Some strategies dominate others as the strategies can be compared according to the payoffs of agents in a CTSP. The reason we do not use the word 'optimal' for strategies is that the strategies of agents are interacted and the best strategy for one agent depends on the strategies of others.

CTSP represents the complicated decision making problems that involves both combinatorial explosion and strategic interactions among multiple agents. So far there is still not any efficient algorithm although some heuristics has been developed for CTSP. A few population based algorithms for CTSP include Gray Wolf Optimization algorithm [3], Salp Swarm Algorithm [4], and Ant Colony System [7], [10]. A hyper-heuristic algorithm is adopted to search the strategies of individual agents by assuming that each agent adopts a heuristic strategy consistently [5], [9].

In this research, we developed an improved ant colony algorithm which outperforms the ant colony system with less computational complexity. We also proposed an evaluation index to compare algorithms for CTSP with each other in competitions.

II. Competitive traveling salesmen problem

A CTSP is a non-cooperative game with multiple agents competing in visiting a set of cities. The agents are self interested and they only care for their individual payoffs. We formulate the following terms about CTSP.

- Benefit
  Every city has a benefit for the first agent who arrives at the city, which is denoted by B_i. Other agents who arrive at the city later receive zero benefit. The visit with zero benefit is called a wasted visit. If two or more agents arrive at the same city simultaneously, these agents will share B_i equally. For example, each of n agents receives $\frac{B_i}{n}$.
- Cost
  Every agent needs to pay for the cost of travel which is proportional to the travelling distance. The cost for travelling from city i to j is denoted by C_{ij}.
- Payoff
  The payoff of every agent is calculated by subtracting the cost from the benefit, and it is denoted by P. If one agent visits m cities and obtains benefits from $m^{'}$ cities, then

This research is funded by an NSFC fund of China (code 72071116) and Ningbo 2025 key technology projects (code 2019B10026, E01220200006).

the payoff of this agent is calculated as:

$$P = \sum_{i=1}^{m'} B_i - \sum_{j=1}^{m-1} C_{j(j+1)} - C_{m1} \tag{1}$$

- Path
The agents are assumed to departure from different cities in order to avoid indifferent choices at the beginning of the game. An agent chooses his/her sequence of destinations independently and travels to them sequentially. A destination of an agent cannot be changed once the agent has started moving towards it.
- Speed of travel
All agents have the same speed of travel.
- Common knowledge
The locations of cities, the agents' current destinations, the speed of travel, the paths travelled, and the information whether or not a city has been visited are known to all agents. Agents will choose their destinations based on common knowledge.

III. An improved ant colony approach

The Ant System (AS) proposed by Colorni, Dorigo & Maniezzo in 1991 as a population-based approach for combinatorial optimization problems. This approach mimics the behavior of ants in collaboratively search for food by using pheromone trails. It has been compared with other local search algorithms like Tabu search and simulated annealing in solving TSP [6].

Inspired by the Ant Colony System (ACS) proposed by Mohannad & Belal & Muamer [10] and other heuristics applied to the CTSP, we have developed an Improved Ant Colony (IAC) algorithm.

The algorithm contains two steps in the decision making process. The first step is called pheromone accumulation procedure, in which the population of ants leave pheromone to guide the search for destinations. In the second step, the destination city is chosen according to the distance between cities and the accumulated pheromone in the first step.

Compared with ACS and other heuristics, IAC has improvements in four aspects: the pheromone depositing method, computational complexity, robustness, and a new mechanism called time dominance.

A. A pheromone reinforcement method

In an ant colony system described in [7], the population of ants will deposit pheromone on each arc they crossed in their route. The pheromone accumulation equation is:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \forall (i,j) \in L \tag{2}$$

where m is the number of ants, and $\Delta\tau_{ij}^k$ is the pheromone deposited by the k-th ant on the arc from i to j, and the value of $\Delta\tau_{ij}^k$ is calculated as:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{C_k}, & \text{if}(i,j) \in T_k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where T_k is the route constructed by the k-th ant, and C_k is the length of T_k.

This pheromone depositing mechanism has been successfully applied in solving TSP. In CTSP, however, probability that the agents gain payoffs from the cities that are far from their departure cities is low so that the agent adopting the above pheromone depositing mechanism may be guided to far cities while those cities have been visited by their opponents. Therefore, the pheromone depositing mechanism needs to be improved. The agents should assign priority to the cities close to their current locations that they have high probabilities to arrive before their opponents.

IAC maintains most components of the conventional ACS. The pheromone evaporation mechanism is also the same as that in the conventional methods. Consider a population of m ants and every ant visits all cities in each round. IAC adopts a revised pheromone depositing method, in which the pheromone deposited by the k-th ant on the arc from i to j, $\Delta\tau_{ij}^k$, is calculated by:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{d_{ij}}(1 - \frac{1}{1+e^{-z+10}}), & \text{if}(i,j) \in T_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where Q is a constant, and d_{ij} is the distance from i to j, and z is the number of visited cities before visiting city j. Let $\sigma(z) = 1 - \frac{1}{1+e^{-z+10}}$ be the visited factor, which denotes the marginal payoff of visiting an extra city. $\sigma(z)$ controls the pheromone deposited on the arc (i,j) and is a strictly monotone decreasing function of z. When an ant visits the first few cities, which are near to the salesman's start location, it will deposit much pheromone on the arc because the value of z is small. As the increase in the number of visited cities, the marginal payoff of visiting an extra city decreases. The relationship between $\sigma(z)$ and z is shown in the Fig. 1.

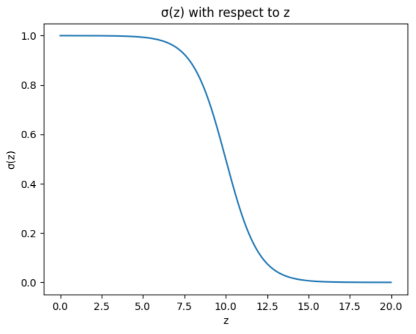


Fig. 1. Diminishing marginal payoff of pheromone deposit

This pheromone depositing function forces ants to deposit more pheromone on arcs to the cities that are close to their departure cities, and less pheromone on arcs to the cities that are far away from their starting locations. The deposited pheromone assigns higher weights to the cities close to them and lower weights to cities that they will arrive later than their opponents.

B. Time complexity

So far, some heuristic approaches have been developed for the CTSP. The time complexity of the GWO, the SSA and the Hyper Heuristics is $O(n^2 * s)$, $O(n^2 * s)$, $O(n^3 * s)$ respectively, where the n is the number of cities, and s is the number of the salesman or agents in the competition. By contrast, this IAC reduced the time complexity into $O(n^2)$, because the pheromone and distance between cities are informative enough to the IAC and there is no need for the IAC to consider potential routes of other salesmen. This improvement is a noteworthy advantage, especially when there exist many salesmen.

C. Robustness

ACS initializes the pheromone on arcs with the values of inverse distance of arcs, and then deposits the pheromone by the equation (2) and (3) after a competition. This method performs well when it is applied repeatedly on the same map so that the locations of cities and distances between cities remained unchanged. In this situation, the ants deposit pheromone on the arcs in the first few competitions. Although the performance may be poor in these competitions, the amount of pheromone in arcs will converge and guide the agent to perform better in later competitions. If the locations of cities and distances between cities change in each competition, the deposited pheromone in previous competitions becomes useless. This is the reason why this method performs poor in the competitions of randomly chosen maps.

By contrast, the IAC finishes depositing the pheromone in the accumulation procedure before the competition, so that there is no need to spend the first few competitions to deposit the pheromone in arcs to a suitable value to guide agents. Even though the locations of cities and distances between cities change and the deposited pheromone is removed, the performance of IAC will not be influenced. This concludes that IAC has better robustness than AC because it adapts to the change of maps.

D. Time dominance mechanism

Every heuristic that does not take into account the opponents' destinations has a serious drawback that it may cause waste moves.

When two agents choose the same destination, it is likely that one agent is dominated by another by doing some waste moves. Consider the scenario in Fig. 2.

In this scenario, Both agents will choose city 2 as the first destination because it is the nearest city for both agents. Agent 1 will arrive at city 2 first because the distance between cities 0 and 2 is shorted than the distance between cities 1 and 2. At the time agent 0 arrives at city 2, agent 1 has moved towards the next destination. If both agents do not change their strategies, agent 0 will follow agent 1's path and make a sequence of waste moves. In order for an agent to avoid making wasted moves, we adopt a time dominance mechanism. The time of an agent arriving at a city matters if this city is also the destination of another agent. We call it time dominance if one agent will arrive at the common destination earlier than the opponent. In case that an agent does not have time dominance than the opponent, the current destination needs to be changed. The flowchart of time dominance (TD) mechanism inside IAC is described in Fig. 3.

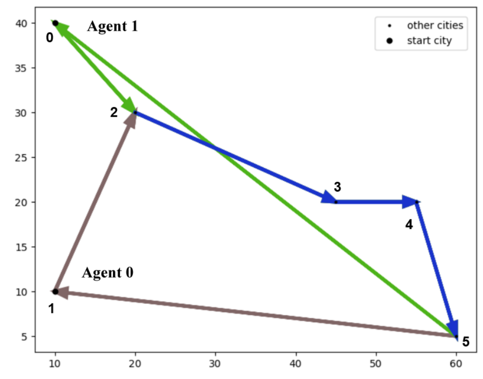


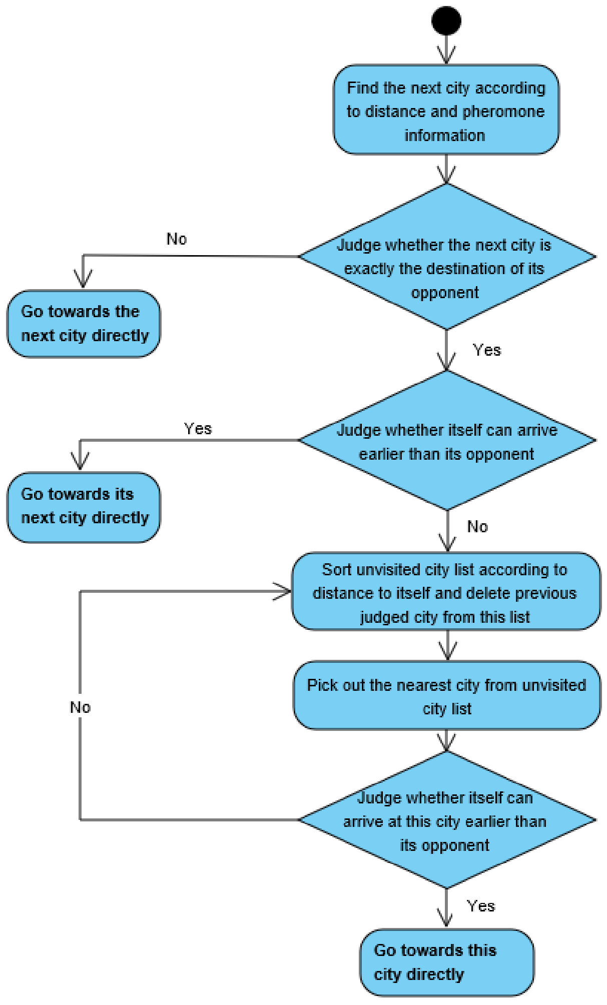
Fig. 2. A Two-agent CTSP in a 2D map. Agent 0 starts from city 1 while agent 1 starts from city 0. Both agents adopt nearest neighbor heuristic. Agent 0 visits cities 2, 3, 4, and 5 in sequence and then move back to city 1. Agent 1 also visits cities 2, 3, 4, and 5 and then return to city 0. The path of agent 0 and 1 are illustrated by brown and green arrows respectively. Blue arrows represent the identical path of both agents.

The first step for the agent is to find the next city by using the proposed IAC. The second step is to judge whether the destination is same destination of the opponent. If it is not, the agent will choose the current destination. Otherwise, the agent needs to judge whether the opponent will arrive at the destination earlier than the agent in the third step. In the case of destination collision, the agent can choose the current destination only if the opponent will not arrive earlier than the agent. Otherwise, the agent is in danger of being dominated and should go to the fourth step. The fourth step is to change the destination to a city that the agent will arrive earlier than the opponent. If such a city does not exist, choose the nearest neighbor city. If there exists more than one alternatives, choose the nearest one.

When TD mechanism is adopted by an agent, some waste moves will be avoided in the competition against the opponent. Assume that agent 0 adopts TD in the previous example of CTSP as shown in Fig. 2. The new result is shown in the Fig. 4. Agent 0 dominates agent 1 in the path in red.

IV. Simulation results analysis

The computation scenario in this simulation is a two-agent CTSP with 20 cities. The locations of 20 cities are randomly chosen in a 100*100 square area. The benefit of visiting a city is set to be 150 and the distance cost is 0.2 times the travelling distance. The moving speed of both agents is 50. To fully verify the reliability of heuristics in multiple competitions, the locations of cities and the departure cities of both agents are randomly set in every competition. To offset the impact of agents' initial locations, each competition will be run twice. Two agents swap their initial locations while keeping locations



Find the next city according to distance and pheromone information
Judge whether the next city is exactly the destination of its opponent
No
Go towards the next city directly
Yes
Judge whether itself can arrive earlier than its opponent
Yes
Go towards its next city directly
No
Sort unvisited city list according to distance to itself and delete previous judged city from this list
Pick out the nearest city from unvisited city list
No
Judge whether itself can arrive at this city earlier than its opponent
Yes
Go towards this city directly

Fig. 3. Flowchart of the TD mechanism

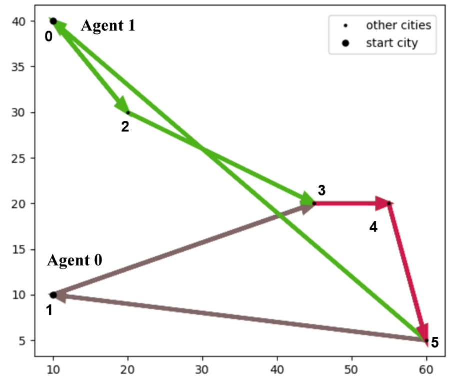


Fig. 4. A two-agent CTSP in which agent 0 adopts TD. Agent 0 starts from city 1 and moves to cities 2, 3, 4, 5, and 1 in sequence. Agent 1 starts from city 0 and moves to cities 2, 3, 4, 5, and 0 in sequence. The path of agent 0 and 1 are illustrated by brown and green arrows respectively. Red arrows represent the identical path of both agents.

of other cities in the second run. Finally, the competition result which includes win, draw, and lose of an algorithm, is determined by the sum of individual payoffs received in two CTSP.

The state of art evaluation mechanism for heuristics in the CTSP only considers the average payoff and best payoff. The payoffs of agents may not reflect the performance of heuristics accurately because these values depend on the locations of cities and they vary in different scenarios. A heuristic algorithm that receives lower payoffs than another heuristic in competitions is not necessarily a poor algorithm. On the other hand, a heuristic may gain high payoffs in some competitions while losing in other competitions. We adopt winning rate together with payoffs to evaluate the performance of heuristics. This evaluation criterion measures the rate of win against the opponent heuristic in competitions. For example, if a heuristic wins 40 times and lose 20 times in totally 100 competitions, then the winning rate of the heuristic is 0.4 and the winning rate of the opponent is 0.2. In this case, the heuristic performs better than the opponent in winning rate.

To compare the performance of IAC and ACS, we run a sequence of 500 competitions with randomly generated locations of cities. The experiment results are shown in Table I.

Table I. Experiment result between IAC and ACS

Heuristics	Winning Rate	Average Payoff
IAC versus ACS	[0.734, 0.266]	[3189.002, 2556.022]

Table I shows that the performance of the IAC is superior to ACS in both winning rate and average payoffs. To further illustrate the performance of IAC and ACS in the competitions, the difference between the payoffs of IAC and ACS are shown by the scatter diagram in Fig. 5.

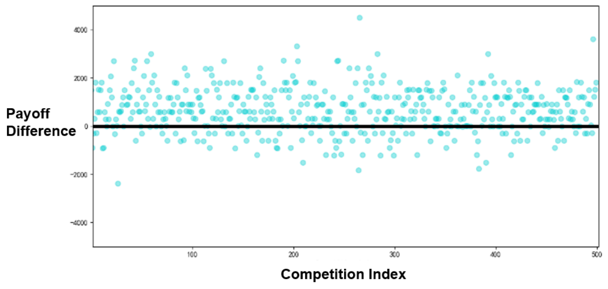


Fig. 5. Payoff difference between IAC and ACS.

Each point denotes the payoff of IAC subtracts the payoff of ACS in one competition. The black horizontal line stands for the situation that the payoff difference is 0. The points above the black horizontal line indicates the situations that the agents who adopt IAC outperforms the opponent adopting ACS.

We also run competitions to compare IAC with the heuristics of nearest neighbour (NN) heuristic and random neighbour (RN). The results are shown in Table II.

Table II. Experiment result between IAC and NN, RN

Heuristics	Winning Rate	Average Payoff
IAC versus NN	[0.568, 0.408]	[3061.997, 2695.163]
IAC versus RN	[0.788, 0.212]	[3172.271, 2566.949]

Simulation results show that IAC outperforms both NN and RN, which denotes the robustness of IAC in competing against common heuristics in CTSP.

To further investigate the effect of the time dominance (TD) mechanism of IAC, one more comparative experiment was conducted. In one set of competitions, both agents adopt NN. The simulation results are shown in Fig 6.

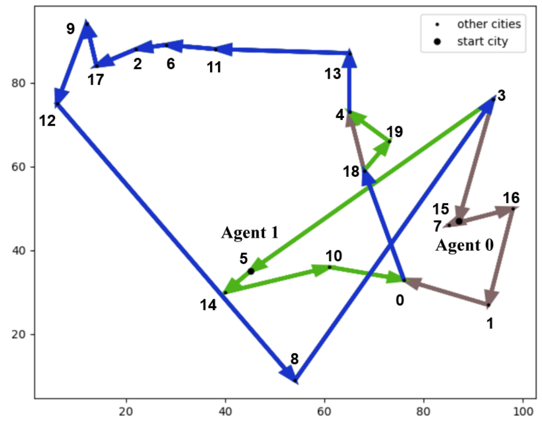


Fig. 6. NN vs. NN. Agent 0 starts from city 5 while agent 1 starts from city 15. The path of agent 0 and 1 are illustrated by brown and green arrows respectively. Blue arrows represent the identical path of both agents.

In the figure, agent 0 at city 1 chooses to move towards city 0. After that, agent 0 will be dominated by agent 1 in the remaining time and make a number of wasted moves. The payoffs of two agents are 523.1091 and 2317.9213 respectively.

In the second set of competitions agent 0 and 1 adopt IAC and NN respectively. The results are shown in Fig 7.

Agent 0 starting from city 1 chooses city 18 as the first destination because agent 1 has time dominance in arriving city 0. According to the TD mechanism, city 18 is chosen to replace city 0. Furthermore, directly moving to city 18 helped the agent 0 to dominate agent 1 in the following moves. The payoffs of two agents are 2173.9057 and 665.9516 respectively in this competition.

V. Conclusion

We have proposed an algorithm for CTSP which reduces the computational complexity of the Ant Colony System by adopting a time-dominance mechanism and the pheromone reinforcement method. In order to evaluate the performance of proposed algorithm, a new criterion, the winning rate, together with the payoffs of agents is adopted in the simulations of a set of two-agent CTSP. Simulation results show that the proposed algorithm outperforms ACS with higher winning rate and less computation time. Agents adopt ACS receive higher payoffs and win rate in competing against the heuristics of NN and RN. The method applies to CTSP with three or more agents as well. Machine learning methods for games have the potential to solve CTSP and it will be our future research.

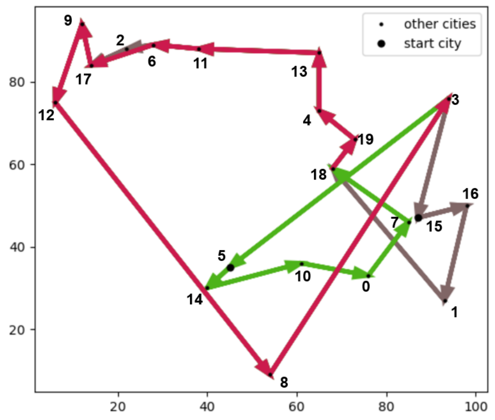


Fig. 7. IAC vs. NN. Agent 0 adopts IAC. Red arrows represent the identical path of both agents.

References

[1] Fekete, S. P., Fleischer, R., Fraenkel, A., Schmitt, M. ”Traveling salesmen in the presence of competition.” Theoretical Computer Science 313.3 (2004): 377-392.
[2] Padberg M., Rinaldi G. ”A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems” SIAM review 31.1 (1991): 60-100.
[3] Taha, Mushreq Luay, et al. ”Solving competitive traveling salesman problem using Gray Wolf Optimization Algorithm.” Periodicals of Engineering and Natural Sciences (PEN) 8.3 (2020): 1331-1344.
[4] Jasim, Taiser Samer, and Esam Taha Yassen. ”An Inspired Algorithm for Solving Competitive Travelling Salesmen Problem.” 7.15(2020): 4188-4198.
[5] Kendall, Graham, and Jiawei Li. ”Competitive travelling salesmen problem: A hyper-heuristic approach.” Journal of the Operational Research Society 64.2 (2013): 208-216.
[6] Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo. ”Distributed optimization by ant colonies.” Proceedings of the first European conference on artificial life. 142 (1991): 134-142.
[7] Dorigo, Marco, and Thomas Stützle. ”Ant colony optimization algorithms for the traveling salesman problem.” (2004): 65-119.
[8] Mohtadi, M., and Kazem Nogondarian. ”Solving the traveling salesman problem in competitive situations using the game theory.” Appl. Math. Eng. Manage. Technol 2.3 (2014): 311-325.
[9] Li, Jiawei, and Graham Kendall. ”A hyperheuristic methodology to generate adaptive strategies for games.” IEEE Transactions on Computational Intelligence and AI in Games 9.1 (2015): 1-10.
[10] Al-Kubaisi, Mohannad and Al-Khateeb, Belal and Mohammed, Muamer. ”An Ant Colony algorithm with dynamic cities allocation for solving competitive travelling salesmen problem.” Journal of Engineering and Applied Sciences. 13.6 (2018): 1400-1406.
[11] Roughgarden, T. ”Algorithmic game theory.” Communications of the ACM, 53.7 (2010): 78-86.
[12] Bektas, T. ”The multiple travelling salesman problem: an overview of formulations ans solution procedures.” Omega, 34.3 (2006): 209-219.