Bulletin of Electrical Engineering and Informatics
Vol. 8, No. 1, March 2019, pp. 206~213
ISSN: 2302-9285, DOI: 10.11591/eei.v8i1.1437

A genetic algorithm based task scheduling system for logistics service robots

Sariffuddin Harun[1], Mohd Faisal Ibrahim[2]
[1]Electrical and Electronic Engineering Programme, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, Malaysia
[1,2]Centre for Integrated Systems Engineering and Advanced Technologies (INTEGRA) Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, Malaysia

Article Info

Article history:

Received Sep 30, 2018
Revised Oct 16, 2018
Accepted Nov 21, 2018

Keywords:

Autonomous robot
Genetic algorithm
Logistics service robot
Robotic task scheduling

ABSTRACT

The demand for autonomous logistics service robots requires an efficient task scheduling system in order to optimise cost and time for the robot to complete its tasks. This paper presents a Genetic algorithm (GA) based task scheduling system for a ground mobile robot that is able to find a global near-optimal travelling path to complete a logistics task of pick-and-deliver items at various locations. In this study, the chromosome representation and the fitness function of GA is carefully designed to cater for a single load logistics robotic task. Two variants of GA crossover are adopted to enhance the performance of the proposed algorithm. The performance of the scheduling is compared and analysed between the proposed GA algorithms and a conventional greedy algorithm in a virtual map and a real map environments that turns out the proposed GA algorithms outperform the greedy algorithm by 40% to 80% improvement.

Corresponding Author:

Mohd Faisal Ibrahim,
Centre for Integrated Systems Engineering and Advanced Technologies (INTEGRA),
Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia,
43600 Bangi, Selangor.
Email: faisal.ibrahim@ukm.edu.my

1. INTRODUCTION

The demand for autonomous logistics service robots is expected to grow significantly across various applications from healthcare facilities to factory operations. The emergence of automation, artificial intelligence technique and computing power has expanded the efficiency of service robots to deliver logistics-related jobs such as pick up and deliver parcels at various locations or loading and unloading goods at a warehouse.

An autonomous service robot requires an important feature to perform its jobs optimally called as task scheduling system. The goal of the task scheduler is to find an optimal assignment of a list of robotic tasks such that the schedule length or duration is minimised and in the same time the precedence constraints are preserved [1]. The conventional approach to schedule robotic tasks is to use a greedy algorithm in which the robot picks the nearest task or with the lowest cost function [2-4]. This strategy has the advantage of low computational time required but produces a non-optimal path [5]. On the other hand, exact algorithm strategy can be implemented that requires all path combinations to be assessed [6-9]. Such strategy typically produces optimal result but has the drawback that it requires high computational time.

In contrast, an artificial intelligent technique known as Genetic Algorithm (GA) can be adopted to develop a robotic task scheduler that is capable to function in a global optima manner and sensible computational time. With such paradigm, a GA-based task scheduler plans a robot route after taking into account the condition of all current user requests or the cost of moving to all current user locations.

GA has solved task scheduling problems in various applications. Some exemplary works include task scheduling for computer multi-processor systems [10], spacecraft test [11], distributed high-performance computing [12], quality-of-service (QoS) and workflow scheduling in cloud computing [13, 14]. A common attribute of results for abovementioned works is that GA is able to find near optimal solutions and outperforms exact optimal approach. However, the structure of such GA cannot be directly implemented into a robotic task scheduling problems due to different GA search space and fitness calculation needed.

This paper presents an intelligent task scheduling system for a logistics service robot based on GA technique. The goal of the proposed task scheduling system is to formulate the scheduling problem with global optima view such that the overall travelling path of a robot to complete all tasks can be significantly improved. In this study, the proposed method is discussed and implemented on robotic virtual and real world environments. The performance of the proposed steady state GA [15-16] method with two crossover variants was also conducted and compared against a benchmarked greedy algorithm. The contribution of this project is the proposal of GA's chromosome representation and fitness function design to ensure the most optimum route for a service robot to complete its given tasks. This paper is organised as follows. In section 2, the research method is proposed, followed by the results and discussion in section 3.Finally, this work is concluded in section 4.

2. RESEARCH METHOD

This section explains the methodology implemented in this study including robot platform, logistics task scheduling concept and the proposed GA task scheduling algorithm.

2.1. Robot platform

An indoor mobile robot called as Turtlebot2 is used as the robot platform. The robot is equipped with a LiDAR proximity sensor, an odometry and two differential drive wheels to allow autonomous navigation in a flat surface area. Turtlebot2 runs on an open source software Meta operating system for robot called as ROS that provides all necessary navigation packages such as simultaneous localisation and mapping (SLAM), global and local navigation planning algorithms and low level motion control. An open source robot simulator called as Gazebo is used to run the robot in a virtual environment.

2.2. Logistics task scheduling concept

In this study, we assume the robot performs a logistics related task of picking up and delivering items at various locations. The robot has a limit that it can only carry a single item at a time. Thus, when the robot pick up an item from a request location, it must deliver the item to a desired destination location before collecting another item from another users' request.

Such logistics task scheduling can be formulated as an instance of an NP-hard traveling salesman problem (TSP). The general concept of TSP can be defined as the problem of route selection that a traveling salesman t needs to choose to visit each of the listed n cities with the minimum total cost of traveling c. The cost of traveling from city i to city j and vice-versa is defined as undirected graph (symmetrical TSP). Accordingly, the possible number of solutions can be calculated by using (1):

$$Possible\ Solution = \frac{(n-1)!}{2} \tag{1}$$

In this study, t is the robot, n is the total number of user request locations and c is the total path length taken by the robot to complete its tasks.

2.3. The proposed GA task scheduling algorithm

GA relies on a population of individuals i.e. candidate solutions that simultaneously explores a given search space. In this project, steady state GA has been used to develop the task scheduling algorithm where the selection operator is based partial replacement of the parent population. A GA maintains a population of potential solutions that evolves over time and eventually converges to a near-optimal solution. Each potential solution is evaluated for its fitness and then a new population is generated by a set of genetic operations to identify the fittest individual of current population. Thus, steady state GA is differed from generational GA in selection process. For the steady state GA, offspring and parents are both compared and only a number of fittest individual is added into the next population. The main reason that this study choose steady state GA is to cater for real-time GA process execution such that the fitness evaluation time can be cut short as existing evaluated individuals do not have to be evaluated in each generation. Figure 1 is the flowchart of GA process.

A real world problem space can be formulated into GA search space by using chromosome. Chromosome is an individual representative series of numbers such as binary string to represent real world parameters. On the problem of task scheduling, the number of delivery requests from users is the parameters and must be represented into chromosome. For this work, we limit the chromosome at 25 requests or total of 50 locations due to the limitation of hardware to process in real-time.

The next important configuration for the GA is fitness function to evaluate the performance of each chromosome. Fitness function for the task scheduling problem is defined by the total estimated cost of travelling to all requested locations based on the sequence of locations' visit suggested by a chromosome. Details on chromosome representation, fitness evaluation and GA operators are explained in sub-section 2.3.1, 2.3.2 and 2.3.3.

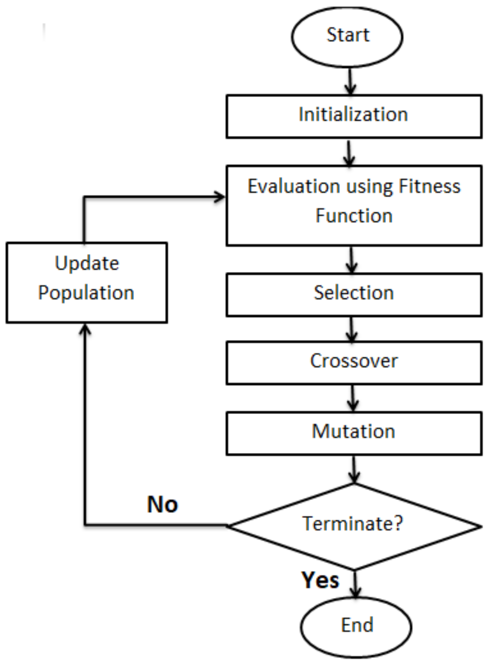


Figure 1. Process of genetic algorithm

2.3.1. Chromosome representation

Each chromosome in the population is encoded into a string of decimal numbers where the length of the string is equal to the number of user requests. Each decimal number represents request identification (ID) for a single delivery request. A request ID can be defined as a pair of source and destination locations of user's delivery request. The sequence of request IDs in a chromosome determine the path that should be taken by the robot to pick up and deliver items for all requests. Figure 2 is an example of a chromosome with 2 request IDs.

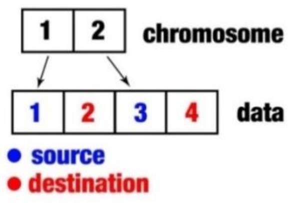


Figure 2. Chromosome and data used in GA

From the figure, it can be interpreted that the chromosome suggests the robot to attempt request ID 1 before completing request ID 2. Thus the sequence of locations to be visited are shown in data string where the robot will move to the source location of request ID 1 (blue colored 1), then delivers item to the destination location of request ID 1 (red colored 2). After that, the robot will pick up another item from the source location of request ID 2 (blue colored 3) before completing at the destination location of request ID 2 (red colored 4). Here, source locations are always in odd index while destination locations are in even index. Chromosome size must always be half of the size of total locations as it is refers to request IDs. Note that the robot location is not included in the chromosome but the cost of travel will be calculated by taking into account the cost of moving from the current robot location to the first location suggested by a chromosome.

2.3.2. Fitness evaluation

A fitness function should be defined to evaluate each chromosome and quality of solution during the evolution process. There are 2 inputs needed to run the fitness evaluation:

a. Chromosome data.
b. Cost of travel from a location to another location including the robot location.

When fitness function receives these two inputs, the process starts with chromosome decoding to extract locations information from request IDs that are source and destination locations. After that, a full path of sequence of locations to be visited is produced starting from the current robot location. Then, the calculation of total cost of travel to complete the path suggested by a chromosome is performed.

To calculate cost between locations, navigation function (navfn) is used to take into account obstacle locations in the robot environment. As the environment is represented as an occupancy grid map, this function computes cost for each grid cell using a wave-propagation technique starting at a selected point. It labels cells in the occupancy grid with the length distance to the selected point, taking into account obstructions by obstacles. The number of cost of travel that must be calculated is (n+1)*(n+1) where n is the total number of locations. Output from a fitness function is an essential element to get good evolutionary result. In this work, a fitness function is designed as in (2).

$$Fitness\ Function = \frac{1}{f(x)} \tag{2}$$

where $f(x)$ refer to the total cost of travel. From the equation, the cost of travel has an inverse proportional relationship with fitness value where it would search for the maximization of fitness value to find the best solution.

2.3.3. GA operators

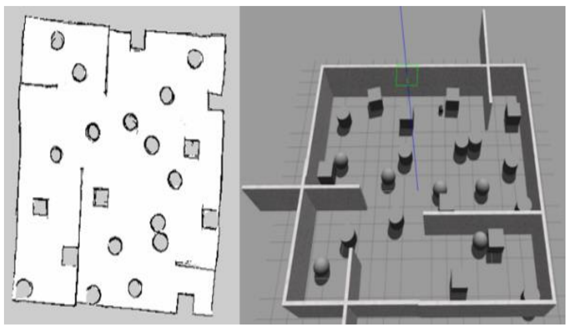
Crossover is the first GA operator to be discussed. The role of crossover in the GA is to combine bits and pieces from fit solutions. Two crossovers were used in this project for performance comparison i) *Edge Recombination Crossover* (ERXover), and ii) *Partial Match Crossover* (PMXover). For ERXover, the edge recombination operator which has been developed using an ”edge map” to construct an offspring that inherits as much information as possible from the parent structures. This edge map stores all the connections from the two parents that lead into and out of a location. Meanwhile with PMXover, two chromosomes are aligned, and two crossing sites are picked uniformly at random along the chromosome strings. These two points define a matching section that is used to effect a cross through position by position exchange operations. Crossover is set with 1.0 crossover probability p_c in this study.

The second GA operator is mutation. Mutation involves the modification of the value of each gene in a chromosome with some probability p_m (the mutation probability). The role of mutation in GAs is to restore the lost or unexplored genetic material into the population to prevent the premature convergence of the GA to local minima solutions. p_m is set with a small value or otherwise the search algorithm will turn into a primitive random search [14]. Thus, p_m is set to 0.1 for this work. Both of crossover and mutation process decimal numbers operation.

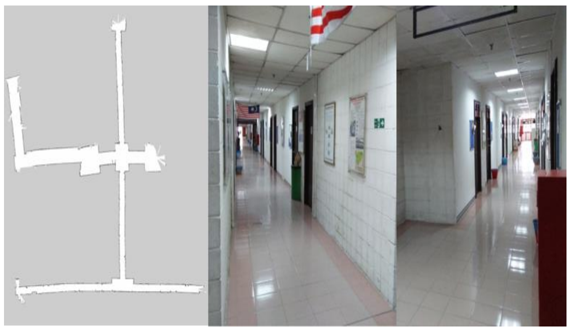
In this study, elitism method for selection process is also added in order to produce the best chromosome. Elitism will copy the best chromosome (or a few best chromosomes) for the next population. Elitism is useful to increase performance of the GA by preventing the best found solution from lost on the next generation.

2.4. Experimental procedures

Experiments to test the proposed GA task scheduling were done by using two environments i) virtual map, and ii) real map. The virtual map is generated by using Gazebo simulator as in Figure 3(a). On the other hand, the real map is acquired by performing real environment mapping by the robot with teleoperation process at one of our faculty's building as in Figure 3(b). On top of this map, a corresponding occupancy grid map was established based on the concept stated in [17]. The function of the grid map is as the input for cost of travel calculation based on navigation function (navfn).



(a)



(b)

Figure 3. (a) A virtual map generated by using Gazebo simulator (b) A real map generated by performing robotic mapping process

For the virtual map environment, four (4) test cases were created where the different between cases is the number of user requests. This variation is set up to show the capability of the GA to handle different numbers of requests. The first case contained 2 user requests, which mean 2 locations of source and 2 locations of destination are defined. The second case has 4 requests. Meanwhile, the third and fourth cases have 7 and 10 user requests, respectively. Similar for the real map environment, the first and second cases also contain 2 and 4 requests, respectively, while the third and fourth cases contain 5 and 7 requests, respectively.

Next, GA parameters need to be set i.e. is the number of generation (*gen*) and population size (*pop*). In this study, it is set heuristically based on the trade-off between the number of possible solutions as in equation (1) and the real-time process. Table 1(a) and 1(b) show the configuration of *gen* and *pop* fort the virtual map and the real map environments, respectively. Each case was executed for 3 times to find the statistical mean and standard deviation for the best result.

Table 1. GA Parameters on the (a) Virtual Map Cases, and (b) Real Map Cases

Case	Possible Solution	Generation	Population
1	3	3	1
2	2520	100	10
3	3133510400	1000	10
4	$6.08\text{x}10^6$	1000	10

(a)

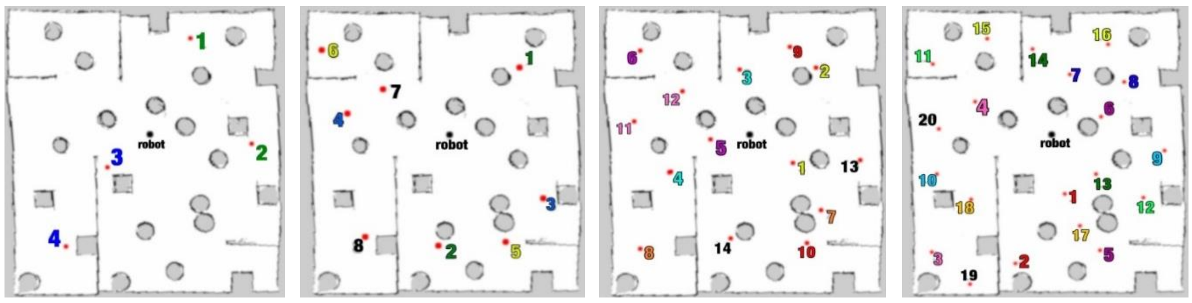
Case	Possible Solution	Generation	Population
1	3	3	1
2	2520	100	10
3	181440	1000	10
4	3133510400	1000	10

(b)

3. RESULTS AND ANALYSIS

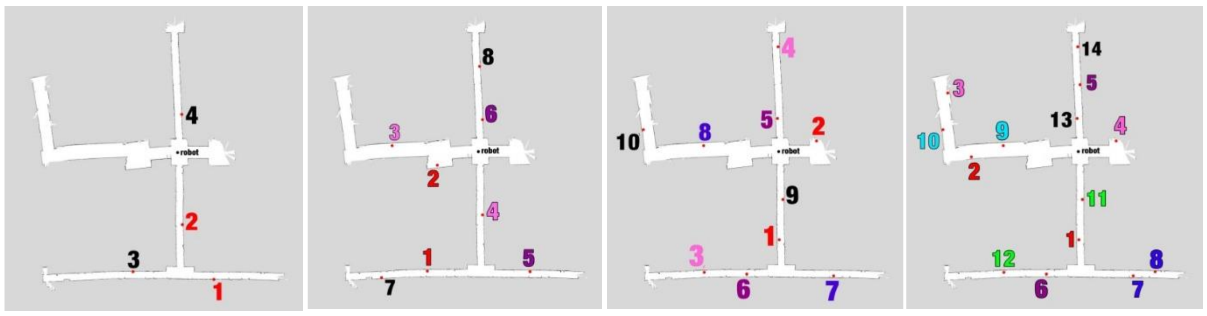
In this section, the result of experiments are discussed for both the virtual map and the real map test cases. Figure 4 and 5 show corresponding maps augmented with users' request source and destination locations for all four (4) cases on the virtual map and the real map, respectively.

The GA execution results for all test cases on the virtual map and the real map is shown in Figure 6(a) and 6(b), respectively. The figure aggregates results from three (3) types of task scheduler for performance comparison i.e. i) the proposed GA with ERXover, ii) the proposed GA with PMXover, and iii) a greedy algorithm. Note that two variants of crossover (ERXover and PMXover) on the steady state GA algorithm were tested to optimise the evolutionary run performance. The figure tabulates the results of the mean value of the best total cost of travel for the two proposed GA variants and also the greedy algorithm.



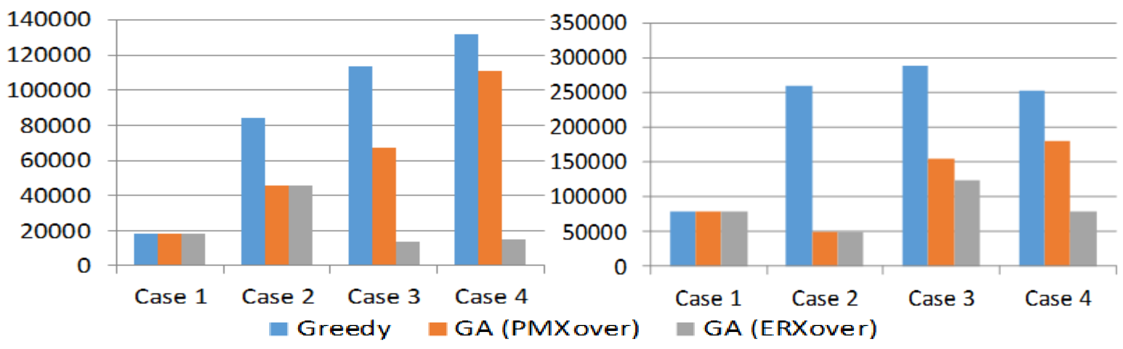
(a) (b) (c) (d)

Figure 4. Coordinates of source and destination locations for each request (a) case 1 (b) case 2 (c) case 3 and (d) case 4, of the virtual map environment. Odd numbers in the figure represent source locations and even numbers represent destination locations. Colors of the numbers represent request pairs



(a) (b) (c) (d)

Figure 5. Coordinates of source and destination locations for each request (a) case 1 (b) case 2 (c) case 3 and (d) case 4, of the real map environment. Odd numbers in the figure represent source locations and even numbers represent destination locations. Colors of the numbers represent request pairs



(a) (b)

Figure 6. Results in terms of the mean of the best cost of travel for all three (3) algorithms vs. all four (4) cases on (a) The virtual map, and (b) The real map

3.1. Analysis for the virtual map test cases

From Figure 6(a), the mean value for the first case is equal for the proposed GAs and the greedy algorithm with the cost of travel at 17,919.8. This result is expected as the number of requests is small, thus no performance variation for all algorithms. All algorithms suggest the same path of R-3-4-1-2. Noted that R is the current robot location, while numbers represent location tags as in Figure 5(a). For the second case, the greedy algorithm suggests a path with the cost of travel at 84,089.5. However, both of the proposed Gas manage to reduce the cost of travel at 45,650.5. The result is a reduction of the cost of travel for about 45.7% or almost half of the original cost of travel.

The result for the third case with 7 user requests shows that the greedy algorithm has the highest cost of travel with 113,714.6. Again, the proposed GAs are able to outperform the greedy algorithm. However, there is a significant performance different between GA (PMXover) and GA (ERXover). GA (PMXover) found the cost of travel at 67,374.0 while GA (ERXover) discovered the lowest cost of travel at 13,799.8 with suggested path of R-5-6-7-811-12-3-4-13-14-9-10-1-2. Thus, GA (PMXover) outperforms the greedy algorithm with 40.8% improvement but GA(ERXover) has better performance at 87.86% improvement.

For the last case of 10 user requests, the results show that the performance of the greedy algorithm decreases proportional to the number of user requests. It found the best cost of travel at the highest cost of 132,129.2. For the proposed GAs, GA (ERXover) maintains as the best algorithm with the minimum cost of travel at 15,112.3 while GA (PMXover) found a solution with the cost of travel at only 110,710.0. GA(PMXover) outperforms the greedy algorithm with 16.2% improvement and GA (ERXover) outperforms the greedy algorithm at a tremendous 88.6% improvement.

From the results of the virtual map cases, it can be concluded that the proposed GAs outperforms the greedy algorithm by being able to suggest a better path with lower cost of travel. GA (PMXover) outperforms the greedy algorithm with an average 40% to 50% improvement on the cost of travel. Similar performance can be found on GA (ERXover) for cases with smaller number of requests. However, GA (ERXover) shows exceptional performance when dealing with higher number of user requests as depicted in case 3 and case 4 where it can optimise the cost of travel up to 88.6%.

3.2. Analysis for the real map test cases

In order to verify the applicability of the proposed algorithms on real world environments, another experiment was conducted with a real map. The real map is differed from the virtual map in that the map is less smooth and distorted with noise due to the nature of sensory noise and mapping algorithm. From Figure 6(b), for the first case, all three (3) algorithms provide the same cost of travel and suggested path that are 79,181.9 and R-1-2-3-4, respectively. This result is consistent on the smallest number of user requests as in the virtual map.

For the second case, the greedy algorithm provides the cost of travel at 259,155.4. Meanwhile, both GAs provide better cost of travel than the greedy algorithm with the same value at 49,082.6 despite the path suggested by both GAs is different i.e. GA (PMXover) with R-3-4-7-8-1-2-5-6 while GA (ERXover) with R-3-4-7-8-5-6-1-2. Both GAs outperform the greedy algorithm with 81.1% improvement.

For the third case, the greedy algorithm suggests the highest cost of travel at 289,005.7. GA (PMXover) provides lower cost of travel at 154,248.0 while GA (ERXover) gives the lowest cost of travel at 124,290.0. Thus, GA (PMXover) and GA (ERXover) supersede the greedy algorithm with 46.6% and 57.0% improvement, respectively.

Last but not least, the fourth case replicates almost the same pattern of the third case where the greedy algorithm suggests the cost of travel at 253,257.5. Meanwhile, GA (PMXover) and GA (ERXover) provide the better cost of travel at 180,390.0 and 78,511.0 respectively. Therefore, GA (PMXover) outperforms the greedy with 28.77% improvement while GA (ERXover) outperforms the greedy algorithm and GA (PMXover) at 69.0% and 43.5%, respectively.

From the real map experiment, the results show that the proposed GAs are capable to maintain the similar pattern of optimisation performance as in the virtual map experiment despite the existance of map noise and distortion. Both GAs can outperform the greedy algorithm up to 81.1% optimization of cost of travel. It is also can be concluded GA (ERXover) provides better solution than GA (PMXover) in all cases. Thus, the usage of as many as possible exisiting edges in ERXover is a better strategy for crossover compared to the usage of partial matching section in PMXover for the robotic task scheduling considered in this work.

4. CONCLUSION

This paper has presented a GA based task scheduling system for logistics service robots. The proposed GA algorithms outperform greedy algorithms by 40% to 80% in terms of total travel paths to complete the robotic task scheduling problem. From the results, the proposed GAs provide optimised solutions for the problem in various numbers of user requests. The performance of the proposed GAs increase significantly with the additional number of user requests. For the future work, the GA algorithm can be extended to be used for more complex logistics tasks such as service robots with multi-loads.

ACKNOWLEDGEMENTS

This research was supported by Universiti Kebangsaan Malaysia under the grant GGPM-2016-003.

REFERENCES

[1] Abraham A, Hassanie AE, Siarry P, Engelbrecht A. Foundations of Computational Intelligence Volume 3: Global Optimization. *Springer*. 2009.

[2] Ostafew CJ, Schoellig AP, Barfoot TD, Collier J. *Speed Daemon: Experience-based Mobile Robot Speed Scheduling*. Proceedings Canadian Conference on Computer & Robot Vision. 2014: 56-62.

[3] Yamauchi B. *A frontier-based approach for autonomous exploration*. Proceedings IEEE Int. Sym. on Computational Intelligence in Robotics and Automation. Monterey, CA. 1997: 146-151.

[4] Lee G, An K, Yun SS, Choi JS. *A simultaneous robot service scheme for Multi-users*. Proceedings 12th Int. Conf. on Ubiquitous Robots & Ambient Intelligence. Goyang, Korea. 2015: 373-374.

[5] Nunes E, Gini M. *Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints*. Proceedings 29th AAAI Conference on Artificial Intelligence. Texas, USA. 2015: 2110–2116.

[6] Coltin B, Veloso M, Ventura R. *Dynamic User Task Scheduling for Mobile Robots*. Proceedings 9th AAAI Conference on Automated Action Planning for Autonomous Mobile Robots. 2011: 27-32.

[7] Lee JH, Park JM. *User-centric Real Time Service Scheduling for Robots*. Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics. Phuket, Thailand. 2011: 1024-1028.

[8] Liu S, Tan PH, Kurniawan E, Zhang P, Sun S. *Dynamic Scheduling for Pickup and Delivery with Time Windows*. Proceedings IEEE 4th World Forum on IoTs. Singapore. 2018: 767-770.

[9] Jeon S, Lee J. *Performance Analysis of Scheduling Multiple Robots for Hospital Logistics*. Proceedings 4th Int. Conf. on Ubiquitous Robots and Ambient Intelligence. Jeju. 2017:937-940.

[10] Bazoobandi HA, Khorashadizadeh M, Eftekhari M. *Solving task scheduling problem in multi-processors with genetic algorithm and task duplication*. Proceedings 2014 Iranian Conference on Intelligent System. 2014: 3–6.

[11] Yang F, Ren L, Yang N. *Task Scheduling Simulation of Spacecraft Test Based on Genetic Algorithm*. Proceedings 10th Int.Symp. on Computer Intelligent & Design. 2017: 267–269.

[12] Gandhi T, Nitin, Alam T. *Quantum GA with rotation angle refinement for dependent task* scheduling on distributed systems. Proc. 10th Int. Conf. Contemporary Computing. 2018: 1–5.

[13] Dai Y, Lou Y, Lu X. *A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization Algorithm with Multi-QoS Constraints in Cloud Computing*. Proceedings 7th Int. Conf. Intell. Human-Machine Syst. Cybern. 2015: 428–431.

[14] Cui Y, Xiaoqing Z. Workflow tasks scheduling optimization based on genetic algorithm in clouds. Proceedings IEEE 3rd Int. Conf. on Cloud Computing & Big Data Analysis. 2018: 6–10.

[15] Corus D, Oliveto PS. Standard Steady State GA Can Hillclimb Faster than Mutation-only Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* 2017; 22(5): 720 - 732.

[16] Xhafa F, Barolli A, Takizawa M. *Steady State Genetic Algorithm for Ground Station Scheduling Problem*. Proceedings *IEEE 27th Int. Conf. Adv. Inf. Netw. Appl.* 2013: 153–160.

[17] Brock O, Khatib O. *High-speed navigation using the global dynamic window approach*. Proc. IEEE Int. Conf. Robot. Autom.1999: 341–346.