ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **15**, Number 2, 2012, 129–145

Optimized Dynamic Task Allocation and Priority Assignments in an Immunized Autonomous Multi-Robot Search and Rescue Operation

M. Tahir KHAN[1], Bessie CHAN[1], Afzal KHAN[2], Zia HAQ[2], Javaid IQBAL[3]

[1]Industrial Automation Laboratory, University of British Columbia, Vancouver, BC, Canada
E-mail: `mkhan@mail.ubc.ca`

[2]University of Engineering and Technology, Peshawar, Pakistan

[3]National University of Science and Technology, Rawalpindi, Pakistan

Abstract. This paper discusses the optimization of the cost function for task allocation and conflict resolution in robots in immune-based cooperative multi-robot search and rescue missions. In this research, robots respond to an emergency situation such as a bomb blast or an earthquake, approach the site, and assemble a structure out of parts collected from the debris in order to evacuate the injured people. A genetic algorithm is used to optimize the cost function that determines the most suitable robot that can transport or help another robot to transport the part to an assembly location in order to build a structure. Further, a priority assignment method is suggested for different parts based on their order dependencies. Decision conflicts among the robots are also discussed and solutions to resolve them are provided. The developed scheme is verified through simulation of immune-based multi-robot cooperation.

Key words: Artificial immune system, cooperative transport, cost optimization, multi-robot systems.

1. Introduction

In recent years, disasters such as 9-11, Hurricane Katrina, and the March 2011 earthquake in Japan have caused researchers to focus on methods to aid in relief operations. Such relief efforts can be aided by multi-robot search and rescue teams. Robots are highly suitable for tasks that one would hesitate to assign to a human. These include handling hazardous material, mapping dangerous or unknown terrain, and de-mining war zones. Robots are also suited for tasks where human supervision is undesirable, as in nuclear waste cleanup, or unlikely, as in planetary exploration. Multiple robot strategies improve upon single robot strategies in many ways. Search and rescue is an excellent example of an application that can be massively parallelized. With traditional single agent solutions, the process can be slow and, if the agent fails, information can be lost. When time is a crucial factor, the scaling properties of multi-robot search and rescue can reduce the time needed to explore the search space.

This work is an extension of a larger project on autonomous multi-robot cooperation for search and rescue mission using an artificial immune system [1]–[5]. In the current work, robots are tasked with retrieving useful parts from the debris in the disastrous area and relocating them to the area where they can be assembled to make a structure that can be used to evacuate the injured people.

The environment is completely dynamic and unknown and has both static and dynamic obstacles. Robots search the area for parts and transport them, if they can, to the assembly location. If a robot is unable to transport a part, because either the robot does not possess the necessary capabilities or the part is too heavy, it calls for help. Based on artificial immune system principles, the robot broadcasts a message to other robots within the environment, detailing the location of the part and the capabilities required to transport it. The other robots then determine their fitness for the task and broadcast that fitness to each other. When the robots receive the broadcast fitness of the other robots, they compare the other robots' fitness to their own. If a robot determines that another robot is more capable for the task, it reverts to the search state and resumes searching for other parts to transport. The task is allocated to the most suitable and capable robot that arrives at the part to transport it.

It is believed that an intelligent and experienced human can make the most suitable decisions in crucial moments of a rescue situation after a disaster. He/she can evaluate the situation based on different conflicting criteria, understand priorities, consider the time criticality, and take into account his/her own and his/her teammates' capabilities. In addition to this, a person's experience increases with each rescue operation, resulting in better decision making capability, which means better trade-offs between conflicting criteria. The general objective of this work is to mimic the evaluation, decision making, and experience of humans to design a mission planning framework for robots, so that the most suitable robot is selected for the task. The selection of the most suitable robot is based on a multi-criteria cost function, which is optimized in this work using a genetic algorithm (GA) offline.

Genetic algorithms are being actively used in multi-robot research. Cai and Peng [6] presented a novel cooperative coevolutionary adaptive genetic algorithm for path planning for a cooperative multi-robot system. Kubota *et al.* [7] developed a perception-based genetic algorithm for mobile robots for acquiring collision avoidance behaviors. Yang and Luo [8] proposed a GA-based algorithm for coalition structure formation in multi-agent systems. Vadakkepat *et al.* [9] investigated an evolutionary fuzzy behavior-based architecture for a robot soccer team. The GA was applied to different aspects of the fuzzy controller at different levels of behavioral architecture.

Pugh and Martinoli [10] used particle swarm optimization to update the parameters of robotic search algorithm. Jennings *et al.* [11] developed distributes search and rescue algorithm and implemented it on a team of mobile robots that search the object and then cooperatively rescue it. Macwan and Benhabib [12] proposed multi-robot coordination method for search and rescue in the wilderness using probability density function.

2. Brief overview of immune-based multi-robot cooperation

The human immune system's antibody structure and the idiotypic network model [13], which are widely used in artificial immune systems, are the theoretical foundation of the immune-based multi-robot cooperation technique developed by us in [1–5]. In our system, a robot is seen as an antibody and its environment, including obstacles and parts to be transported, is seen as a collection of different antigens. Antibodies have to use different strategies to deal with different antigens.

Multiple robots with different capabilities coordinate with each other to search for parts in an unknown, unpredictable, and dynamic environment and to transport them to the goal for assembly into a useful structure. The robots do not know the exact location of the parts to be transported. If a robot detects a part and does not have the capabilities to transport it, then a message about the part is broadcast to other robots in communication range, and the part is left to be transported by another robot. The robots use a global messaging system to send and receive messages about a detected part and signals for help. The robot first tries to tackle the task alone, and if it is unable to do the task, it coordinates with other robots by broadcasting a help signal and then cooperatively transports the part to the goal location.

The simulation ends when all of the parts are at the goal. Search, help, and push are the various states each robot can take on.

3. Multi-robot cooperation

3.1. Decision Conflicts

In the real world, people constantly make decisions that affect their lives – whether to take public transit or drive, whether to drink coffee or tea, whether to take one job or another. In order to enrich the multi-robot cooperative environment, decision conflicts are incorporated into the robots to mimic decisions that a human would have to make if the human were in the place of the robot. Three main decisions are identified: the in-process decision, the in-bound decision, and the aiding decision.

An in-process decision is made when a robot is in the process of transporting a part and receives a message calling for help in the transport of another part. This call may be the result of either the robot finding a part it is not capable of moving or the robot having difficulty transporting a part. Either way, the robot must make a decision about whether it should abandon its current task and fulfill the request in the help message or continue on its current task.

An in-bound decision is made when a robot is in the process of transporting a part and detects another part along its path to the goal. The robot then needs to make a decision about whether it should abandon its current part and transport the detected part or continue transporting its current part. The main factor in this decision is priority. In this case the priorities of the two parts should heavily affect whether the robot chooses the current part over the detected part. The assumption in this decision is that higher priority parts should be transported over lower priority parts even if the time it takes to retrieve both parts will be greater due to the overhead of getting another robot to retrieve the lower priority part, if the robot's current part is the lower priority part and the robot has chosen to abandon it for the detected higher priority part.

The last decision conflict is less about making a choice between the priorities of different parts and instead concerns the capabilities of the transporting robot. The aiding decision is made when a robot is transporting a part with difficulty. Difficulty is defined as transporting a part below a certain speed threshold. When a robot detects that its speed is below this threshold, it sends out a message to the other robots in the environment that it requires aid in the transport of its part. Each robot in the communication range then decides whether it is able to help and sends its decision back to the requesting robot. Whether the requesting robot receives help or not is entirely dependent upon the situation. It can be imagined that the one situation in which the requesting robot receives aid is if there is at least one robot that is neither transporting a part nor en route to transporting a part. This implies that there exists an intricate relation between the number of robots capable of transporting that type of part and the number of parts in the environment that require the united capabilities of those robots. That is, if the requesting robot's part requires capability a, and there are other robots that have capability a and also have capabilities b, c,and d, then these robots may be busy with other parts in the environment that require capabilities *a, b, c,* and *d*. A situation can also be imagined in which an aiding robot has come as a result of abandoning its previous task. It is to be expected that the part to be transported by both robots has a much higher priority than that of the aiding robot's original part—high enough, in fact, to overcome the distance and other parameters as mentioned in equation 2. If a robot chooses to abandon its current part, then either distance isn't much of a factor (that is, the robot was already close to the object) or the difference between the two priorities is great enough to overcome the distance and other factors mentioned in cost function discussed in section 3.3.

3.2. Priorities

We developed a priorities-based method to enrich the decisions made by the robots as they complete the tasks. An example will illustrate the significance of this technique. Suppose a cart for evacuating an injured person is to be assembled, and the parts to assemble the cart are scattered and are to be collected from debris in the disaster area. The order in which the parts are assembled depends on which parts are found and retrieved first. The relation between the parts can be modelled as a graph, where each node represents a part (*e.g.*, a wheel, a handle, a screw), and an edge represents two parts that connect to each other (*e.g.*, a wheel connects to a shaft). At the beginning of the simulation, each node starts with the same priority. Let P be the priority of some node in the graph at the start of the algorithm, and let N be the number of nodes in the system. The total value of priority within the system – that is, the sum of all the priorities of the graph – is constant and is equal to N_p. The fact that the value of priority within the system is constant makes comparisons of priorities meaningful – that is, when the priorities of two nodes are compared, the absolute values of the priorities are compared rather than the relative magnitudes, which are normalized by the amount of priority in the entire system. While the two comparisons will yield the same result, when the neighbours of a task node that has been completed are updated, it is not necessary to recalculate the priorities of every node in the system.

Figure 1 shows an example of how the completion of each part leads to the next part and of how priority can be distributed among the tasks. The nodes labeled A and B represent different parts that can be transported by robots with different capabilities. The values beside each task node represent the task's priority. The priority of the entire system stays constant throughout the simulation. Every time a task node $P$is completed, P is removed from the task list and the priorities of all neighbour nodes of P are updated. The new priority of each neighbour is equal to the sum of the neighbour's priority N_p and the priority of P, P_p divided by the total number of P's neighbour nodes N, as shown in equation 1.

$$p_p = \sum_{k=1}^{N} p_p + \frac{P_p}{N} \tag{1}$$

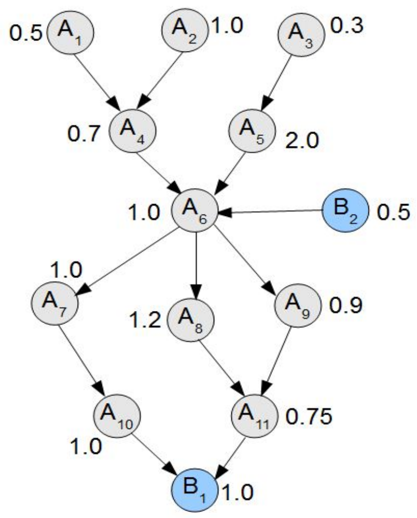

Fig. 1. An example of task distribution in each scenario.

Leaf nodes of the graph act as priority sinks. Priority may flow into a leaf node, but that priority will not flow back into the system. This does not prove to be an issue for the algorithm, since once those leaf node tasks are complete, the task will not be compared to other tasks to determine priority, and thus the relative priority ordering of tasks yet to be complete is still preserved. The algorithm behaves well no matter where in the graph the first task is completed. In fact, the algorithm behaves well even if nodes are randomly completed within the graph. As priorities flow from completed tasks to uncompleted tasks, the priorities of uncompleted tasks rise proportionate to the number of completed neighbours. This increase in priority based on completed neighbours increases the chances that if a node with high enough priority is found, low priority parts will be temporarily abandoned in favour of transporting a high priority part.

To apply the work flow analogy described above, suppose that a wheel and a shaft have been successfully transported and that a screw is required to connect the components together. The priority of the screw increases when the wheel and the shaft retrieval tasks are completed. A robot detects a screw, but since this robot was designed only to push larger parts it does not have the capabilities of picking up and transporting the screw. The robot sends a broadcast message detailing the location of the screw and its priority and asks for a robot that can pick up the screw. If there is a robot nearby that is transporting another wheel, one which cannot yet be made use of, then it abandons the wheel in favour of the screw. While this is a simple example, it illustrates the ability of the robots to complete related tasks local to a region within the task graph.

3.3. Cost Function: Selection by Fitness

The fitness of a robot for transporting a part is determined by seven factors: the linear distance between the part and the robot, d; the speed at which the robot travels to reach the part, v, as it is not necessarily the robot closest to the part that will reach it first; the angle between the nose of the robot and the direction of the part, θ; the effort required to push the part, e, as different parts with different shapes, weights, and softness require different effort to transport; the part's priority, p; the robot's operating cost, r, as a robot with more capabilities costs more to operate; and the obstacles between the robot and the part within detection range, b. The fitness of the robot with respect to a part, c, is the weighted sum of the seven properties, which can be defined as in equation 2, below:

$$Cost\,(c) = \omega_d\,(d) + \omega_v\,(v) + \omega_\theta\,(\theta) + \omega_e\,(e) + \omega_p\,(p) + \omega_r\,(r) + \omega_b\,(b) \qquad (2)$$

In other words, the mindset of decision making of each robot, which is aimed to be optimized in this work, consists of seven weighting factors. Depending on the type of situation the robot is in, the cost function is calculated differently. When the robots are in a conflict situation, the robots calculate cost with priorities first to decide if the cost of transporting the current part is more than that of the conflicting part. If the cost of the other part is less than the current part, the robot broadcasts its cost without priorities, as is shown in equation 3, and compares that cost with the costs of other robots to determines if it is the best robot to transport the part. Once the robot determines that it is the best candidate to transport the part, it drops its current part in favor of transporting the new part. If a robot is not currently pushing any part or targeting any part, it uses equation 3 to decide whether to transport the part. The rationale for setting the value of priority to 0 in such cases is that if a robot is neither in a decision conflict nor comparing two parts to decide if it should carry one or the other, the priority of the part does not matter.

$$Cost'\left(c\right) = \omega_d\left(d\right) + \omega_v\left(v\right) + \omega_\theta\left(\theta\right) + \omega_e\left(e\right) + \omega_p\left(0\right) + \omega_r\left(r\right) + \omega_b\left(0\right) \qquad (3)$$

Let R be the set of robots in an unexplored environment where $R = \{r_1, r_2, ...r_n\}$ has n robots. Let O be defined as the set of parts in an unexplored space where $O = \{o_1, o_2, ...o_n\}$ with n parts to transport to a goal. The following is a typical scenario in which a robot has to handle making a decision between a part it is currently transporting or targeting and another part that was broadcast by another robot:

1. r_1 is transporting or targeting o_1 as the most suitable robot;
2. r_2 detects o_2 but is not capable so it broadcasts the message that o_2 is available. If capable of transporting o_2, r_1 compares the costs of transporting o_2 and o_1 using equation 2.
3. If $cost(o_2) < cost(o_1)$, r_1 proceeds to compete with other robots in $\{R - r_1\}$ for o_2 using equation 3.
4. Once it is established that $cost'_{r_1}(o_2) < \min\{cost'_{r_k}(o_2)...cost'_{r_n}(o_2)\}$ (where r_k represents any robot in set R that is capable of transporting o_2, r_1 leaves o_1 in favor of o_2.

The above decision routine can be generalized into three decision conflict situations as outlined in section 3.1. All three events can happen with any robot on the field at any time simultaneously.

3.4. Prediction of Intuitive Weights

In the environment, robots transport parts that cost the least. Thus a negative weighting in the cost function will maximize the parameter, and a positive weighting will minimize the parameter. This section outlines the predictions for the weights of each parameter in the cost function.

A task that requires more effort causes an increase in cost. An efficient system is one in which parts that take less effort are transported first and parts that take more effort or require more than one robot are transported afterwards. Valuable time steps can be consumed if a robot has to wait for another robot to help transport a part that requires more effort. Thus if a part that requires less effort is available, it is much more effective to leave the part requiring more effort to be transported later. With this reasoning, we give effort a positive weighting. A robot with a higher operating cost has the ability to transport parts that require more effort or that have higher priorities. Intuitively it is preferable to reserve more expensive robots with better capabilities for parts that require them and to use robots with minimum capabilities first. Therefore, operating cost of the robot is assigned a positive weighting. Distance is straightforward in that parts that are farther away from the goal require more time to transport and thus increase the cost. With this reasoning, the predicted value of distance is positive. The more obstacles in the way of the part to be transported, the more time it takes for a robot to transport the part to the goal. As in the case of angle (robot orientation with respect to part), it costs time to make numerous turns and detours before reaching the goal. Moreover, robots transporting parts can get in the way of other robots and can thus become obstacles themselves. Obstacle is predicted to have a positive weighting as well. The predicted weighting for angle is positive. The greater the angle of a part in relation to the robot, the more time it takes for the robot to reorient itself to transport the targeted part. Thus if a part is at an angle that can be easily manipulated for transportation, it is preferred over a part with an angle that requires more time for reorientation.

In determining the weighting of priority, one can imagine a situation in which multiple parts having lower priorities should be transported after a higher priority part if the lower priority parts have some type of dependency on the higher priority part. For example, a cart cannot be built with just wheels and a shaft, if screws are not available to keep the parts together. Thus, it is predicted that priorities should have a negative weighting in preference for higher priority parts. Finally, the faster the robot is able to move, the faster a part can be transported towards the goal. A robot with higher velocity is valuable to the system, as it can transport parts to the goal more quickly. Thus velocity is given a negative weighting. Based on these intuitions values of different weights are predicted as shown in Table 1.

Table 1. Predicted values for the weights in the cost function

Weight	Value
Effort	0.2
Priority	– 0.5
Distance	1
Angle	0.03
Obstacles	0.03
Cost	0.03
Velocity	– 0.25

4. Optimized multirobot cooperation

4.1. GA Implementation

The general purpose of a genetic algorithm (GA) as a search heuristic is to solve optimization problems. Genetic algorithms are inspired by concepts found in natural evolution such as inheritance, mutation, selection, and crossover. This study uses a GA to determine the best set of weights for the cost function and to determine the importance of each parameter.

In the discussion of the fitness function, the parameters of the function were described but not the importance of each characteristic to the overall fitness of a robot for transporting a part from the search area to the assembly location. In pondering the seven parameters of the cost function, it is possible to form a sense of which of those parameters ought to weigh more in determining which robot is most fit to transport a part. To determine the optimized weights, a genetic algorithm can provide a less intuitive and more methodical answer.

During the initialization of the GA, a random weight between 1 and –1 is generated for each weight $\omega_d, \omega_v, \omega_\theta, \omega_e, \omega_p, \omega_r$ and ω_b. The simulation is then run with the randomly generated weight set. The weight set with the best score is noted. After the initialization, the genetic algorithm runs for a number of generations where inheritance, selection, crossover, and mutation of the weights occur. The genetic algorithm terminates if there is only a 1% difference in score of 3 successive generations. Generally, after each generation, a set of five candidate parents are selected from the pool of weight sets produced by running the simulation. A linear probability distribution function is used to select the best parents for each child to inherit the parents' genes (i.e. the weights). After selecting two of the five candidate parents, the crossover operation occurs. For each parameter or weight, the child has a 50% chance of inheriting the weight from one of the two parents. Finally, with the combined weights, there is a 10% probability of mutation in one of the parameters of the weight set produced from the two parents. Fifty different scenarios were created to optimize the weights with the genetic algorithm for the cost function.

Each configuration file for a scenario was constructed by specifying robots and parts to be transported to the goal. Each robot has position in the environment, an angle in relation to the part to be transported, a velocity, an ability to carry parts up to a certain weight, a type of transporting capability, and a particular operating cost.

In testing the developed methodology, a score was assigned to each scenario after the robots successfully transported all the parts to the goal. The score is dependent on the number of time steps it takes the robots to complete the task. The robots were given a limit of 5000 time steps to bring all parts to the goal, and if the limit was exceeded the scenario was penalized 50 000 time steps. The total score Sassigned to each trial is the sum of the time steps to complete all fifty scenarios, and t_k is the total number of time steps taken to complete the k^{th} scenario as:

$$S = \sum_{k=1}^{50} t_k. \tag{4}$$

4.2. Testing Procedure

This section outlines the procedure for testing the algorithm. Two tests were run to test the validity of the genetic algorithm and weights. First we wanted to determine how the predicted weights performed compared to the weights produced by the GA. The second test was to determine whether robots that are able to resolve conflicts were more efficient in completing a task compared to robots that do not have the ability to resolve conflicts. The steps of the testing procedure are as follows:

1. Run the genetic algorithm on the robot code with conflict resolution and obtain the set of optimal weights.
2. Use the optimal weight set for conflict resolution, run the simulation across all fifty scenarios, and record the scores over 1000 trials.
3. Run the genetic algorithm on the robot code without conflict resolution and obtain the optimal set of weights.
4. Use the optimal weights for the code without conflict resolution, run the simulation across all fifty scenarios, and record the scores over 1000 trials.
5. Run the genetic algorithm on the robot code with the predicted weight set across the fifty scenarios with no conflict resolution and record the scores over 1000 trials.
6. Run the genetic algorithm on the robot code with the predicted weight set across the fifty scenarios with conflict resolution and record the scores over 1000 trials.

5. Results and discussion

After implementation of the java code with conflict resolution, we ran the simulation against the genetic algorithm and obtained fifteen weight sets. Table 2 shows the weight set with the lowest score of the fifteen. Table 3 shows the standard deviation, median and average of all fifteen runs of the best scores produced by the GA.

Table 2. Values for the best weight set generated by the GA

Weight	Value
Effort	-0.72
Priority	– 0.15
Distance	0.42
Angle	0.99
Obstacles	– 0.44
Cost	0.07
Velocity	– 0.63

Table 3. Standard deviation, median, and average of the best scores for 15 runs of the GA

Standard Deviation	1368.7
Median	90623
Average	90535.63

In Table 2, some of the values in the weight set produced by the GA do not line up with our predicted values. For this weight set, the value of the effort weight is negative compared to the predicted value of +0.2. The negative value could be attributed to the fact that the parts requiring more effort to transport also have higher priority than the less effortful parts. For example, if a part that has higher priority and requires more effort to push is not transported to the goal before a lower-priority and lower-effort part is transported, efficiency decreases since the lower priority part depends on the higher priority part. Both distance and angle were positively weighted as predicted. However, angle had a much higher weighting compared to distance. This indicates that distance is not as important, because robots prefer to transport parts that do not require reorientation in order to transport them. This could be due to the fact that the field space was not large enough for distance to be a large enough factor.

Table 4. Summary of the weights

	Conflict resolution		**Without conflict resolution**	
	Intuitive weights	**GA weights**	**Intuitive weights**	**GA weights**
effort	0.2	– 0.72	0.2	0.99
priority	– 0.5	– 0.15	– 0.5	0.9
distance	1	0.42	1	– 0.77
angle	0.03	0.99	0.03	– 0.8
obstacles	0.03	– 0.44	0.03	– 0.2
cost	0.03	0.07	0.03	0.83
velocity	– 0.25	– 0.63	– 0.25	0.11

Table 4 summarize the weight sets generated intuitively and through GA optimization in different situation.

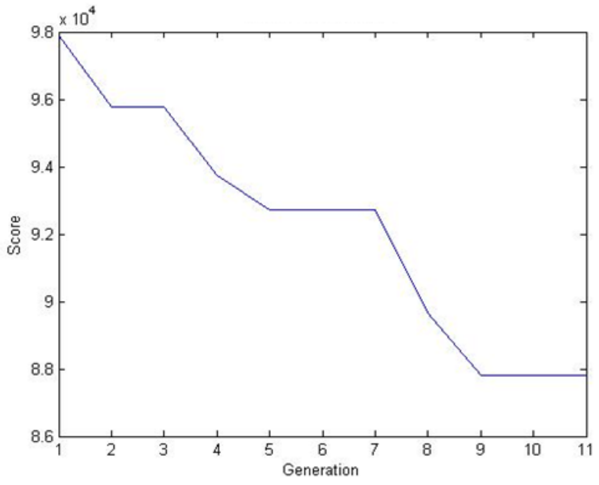

Fig. 2. Improvement in scores as the GA progresses.

Figure 2 shows score as a function of generation. This graph is generated using the selected weight set given in Table 2. It can be seen with each generation that the best weight set extracted from the best scores by the genetic algorithm produces progressively better weights.

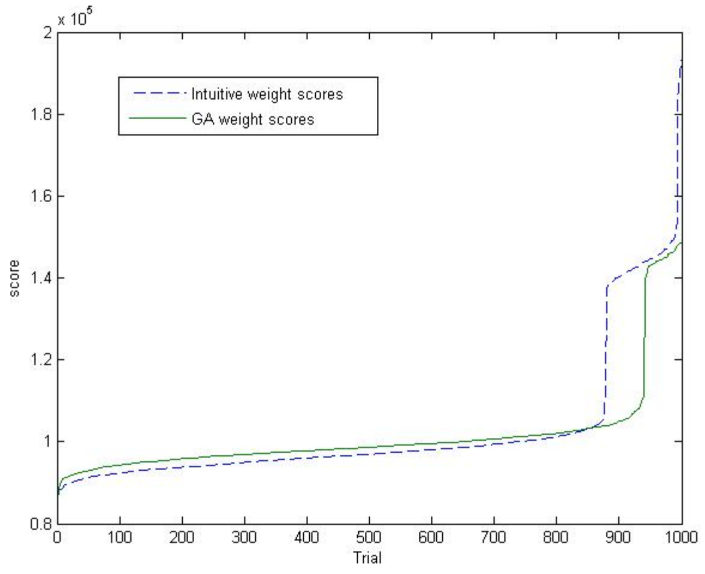

Fig. 3. Comparison of intuitive and genetic algorithm weight scores for no conflict resolution.

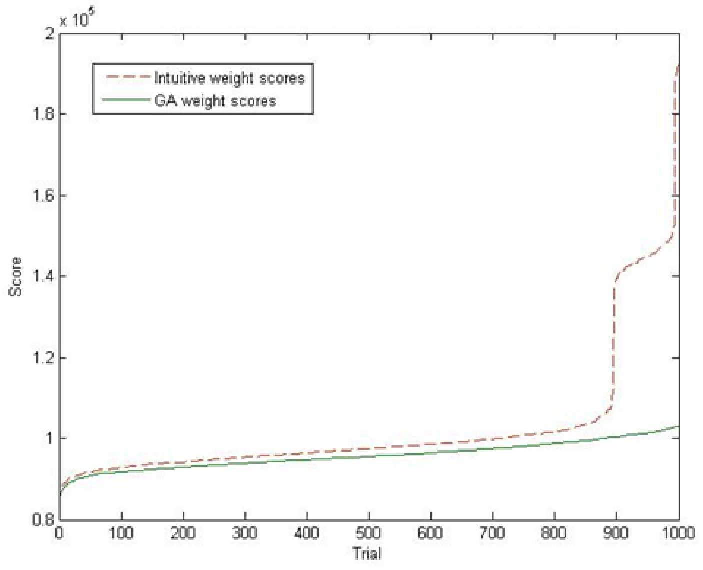

Fig. 4. Comparison of intuitive and genetic algorithm weight scores for conflict resolution.

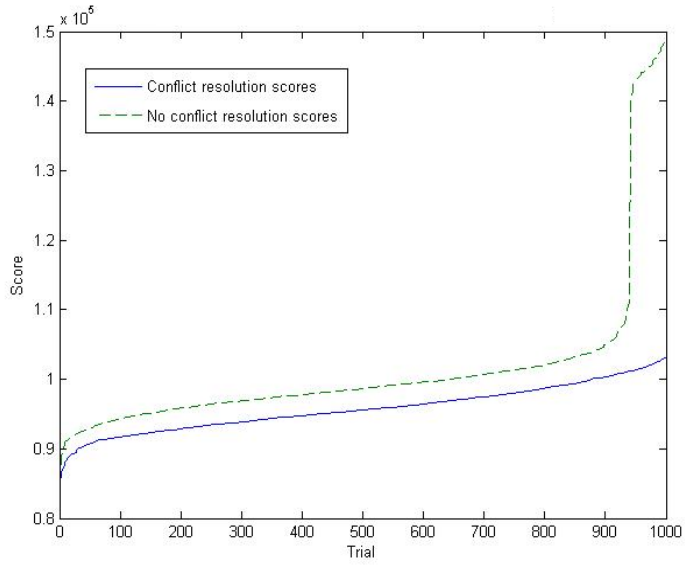

Fig. 5. Comparison of genetic algorithm weights in conflict resolution and in no conflict resolution.

The graphs in Figs. 3 to 5 are generated using the selected weight set given in Table 4. Figure 3 compares scores generated by intuitive weights and GA weights for robots that do not make conflict decisions. An improvement in the scores is seen with the GA weights. It should also be noted that the scores for conflict resolution in Fig. 3 for both intuitive and GA weights do not exceed 20 000 time steps and that scores for no conflict resolution can be as high as 22 000 time steps.

Figure 4 compares scores collected from the intuitive weights and the GA weights for conflict resolution. The scores generated by the GA weights are consistent throughout each run of the simulation. Compared to the scores generated by the GA weights, the intuitive weights are not as consistent. A good portion of the trials produced scores greater than 100 000 time steps. Such high scores indicate that many simulations did not run to completion and were therefore penalized 50 000 time steps. It can be concluded that the genetic algorithm is more reliable for generating more accurate weight sets for the simulations.

Figure 5 compares the scores produced where robots make conflict decisions and where they do not make conflict decisions. A large number of trials either took too much time or the robots did not complete the task within 50 000 time steps. Robots that are able to make conflict decisions in all 1000 trials show the ability of the robots to complete all fifty scenarios efficiently with no score over 120000. It is clearly advantageous for robots to be able to make conflict decisions, as it allows for an improvement in co-operative task completion with multiple robots.

Having ran the genetic algorithm about fifteen times on our implementation of robot co-operation with conflict decisions; we can make further preliminary observations on how the weights vary and how the weights should be interpreted when robots are making decisions. Figures 6 to 11 show how the weights fluctuate between runs of the genetic algorithm. On the x axis is the n^{th} run of the GA, and the y axis indicates the best weight for the parameter in that run. Figures 6 and 7 show that the angle and distance have a common tendency to have a positive weighting in the cost function. This is consistent with our predictions, as they both have positive weightings because the cost of transporting a part should increase as the distance and angle increase. Additionally, in Fig. 8, also conforming to our intuition, velocity tends to have a negative weighting. That is, the faster a robot, the quicker the part is transported to the goal.

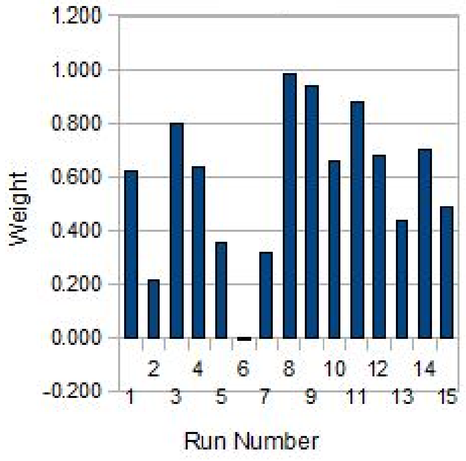

Fig. 6. Weight trends in angle.

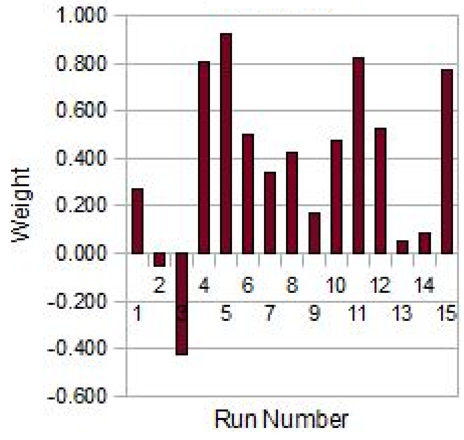

Fig. 7. Weight trends in distance.

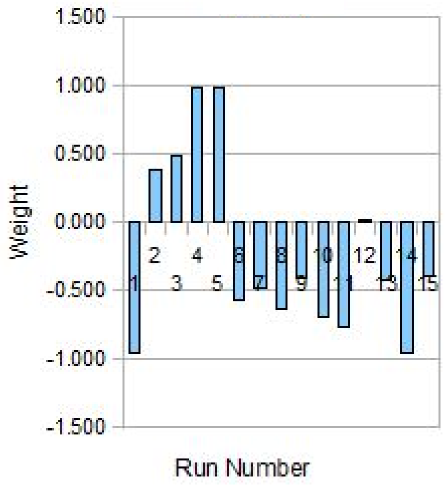

Fig. 8. Weight trends in velocity.

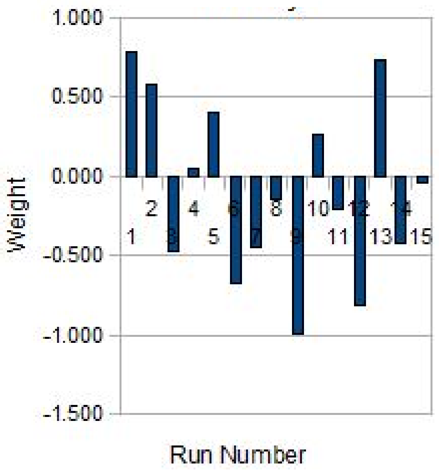

Fig. 9. Weight trends in priority.

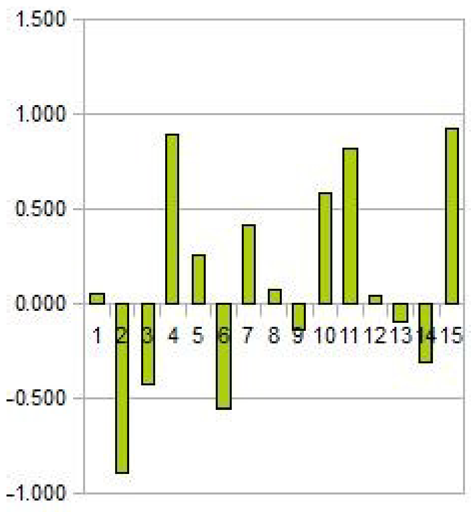

Fig. 10. Weight trends in the operating cost of the robot.

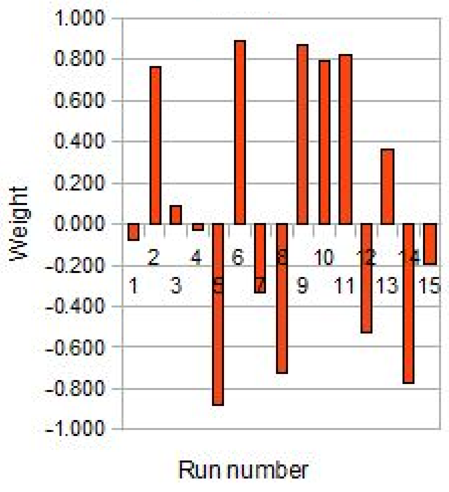

Fig. 11. Weight trends in effort.

In Figures 9, 10, and 11, priority, operating cost of the robot, and effort, there is no general tendency for the weights to be positive or negative. For example, the predicted positive weighting for effort can also be negative in some weight sets produced by the GA. Perhaps the negative weighting is due to the beneficial effects of robots transporting parts that require more effort and are also of higher priority. Completing these tasks enables the system to take on the lighter load tasks more efficiently near the end of the simulations. The same can also be said about priorities. Effort and priority weights may thus fluctuate from weight set to weight set to accommodate the different combinations of effort and priority required to transport a part. Finally, interpreting the negative and positive weightings of the operating cost of a robot can be argued both ways. Lower operating cost robots have fewer capabilities and are only able to transport lighter parts. A higher-cost robot with more capabilities and more ability to transport heavy parts can increase efficiency in completing the global task. Thus the weights for a higher operating cost robot can cause the weight to be negative. It can also be the case that a robot has a high operating cost because its function is inherently expensive. For example, a robot that handles extremely fragile parts may be more costly, but this does not mean that the task at hand is most important in contributing to completing the global task more efficiently. Possible correlations between each parameter require more analysis in the future.

6. Conclusion and future work

The study has presented a robot co-operation strategy following the artificial immune system model with conflict resolution. The aim of this work was to determine whether conflict resolution would increase the productivity of a multirobot system.

Further, we wanted to find the optimal weight set to determine which weights are most important in cooperative robot decision making. In general, robots that were able to resolve conflicts and reconsider decisions allow for greater flexibility in a multi-agent system, increasing overall efficiency. In determining which weights are most important in decision making, some parameters such as distance and angle agree with our intuitions that they should have positive weightings. However, weights for other parameters such as priority and operating cost of the robot are not as obvious as we thought. Further analysis of whether and how these parameters interact with each other is needed.

Future improvements to the implementation of our code will include limiting the robots' communication range and determining the optimal number of robots for the task. Currently, when a robot broadcasts a message, the chance that the message will be lost is only 1%. There should be a limited communication range between the robots. Additionally, the current study does not examine how many of the robots contribute to the task of transporting various parts to the goal. Future work will determine the optimal number of robots required to complete the task.

References

[1] Khan M. T., de Silva C. W., *Autonomous fault tolerant multi-robot cooperation using artificial immune system*, *Proc. IEEE International Conference on Automation and Logistics*, Qingdao, China, pp. 623–628, 2008.

[2] Khan M. T., de Silva C. W., *Autonomous fault tolerant multi-robot coordination for object transportation based on artificial immune system*, *Proc. 2nd International Conference on Robot Communication and Coordination*, Odense, Denmark, pp. 1–6, 2009.

[3] Khan M. T., de Silva C. W., *Immune System-Inspired Dynamic Multi-Robot Coordination*, *Proc. ASME/IEEE International Conference on Mechatronics and Embedded Systems and Applications*, San Diego, CA, pp. 37–43, 2009.

[4] Siriwardana P. G. D., Khan M. T., de Silva C. W., *Object Pose Estimation for Multi-Robot Cooperative Object transportation*, *Proc. ASME/IEEE International Conference on Mechatronics and Embedded Systems and Applications*, San Diego, CA, pp. 449–457, 2009.

[5] Khan M. T., de Silva C. W., *Autonomous fault and robust multi-robot cooperation using artificial immune system*, International Journal of Robotics and Automation, vol. **27**, no. 1, (in press), 2012.

[6] Cai Z., Peng Z., *Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-robot systems*, Journal of Intelligent & Robotic Systems, **33**(1), pp. 61–71, 2002.

[7] Kubota N., Morioka T., Kojima F., Fukuda T., Learning of mobile robots using perception-based genetic algorithm, Measurement, **29**(3), pp. 237–248, 2001.

[8] Yang J., Luo Z., *Coalition formation mechanism in multi-agent systems based on genetic algorithms*, Applied Soft Computing, **7**(2), pp. 561–568, 2007.

[9] Vadakkepat P., Peng X., Quek B.K, Lee T.H., *Evolution of fuzzy behaviors for multi-robotic system*, Robotics and Autonomous Systems, **55**(2), pp. 146–161, 2007.

[10] Pugh J., Martinoli A., *Distributed Adaptation in Multi-robot Search Using Particle Swarm Optimization*, Lecture Notes in Computer Science, vol. **5040**, pp. 393–402, 2008.

[11] Jennings J. S., Whelan G., Evans W. F., *Cooperative search and rescue with a team of mobile robots*, *Proceedings of the 8th International Conference on Advanced Robotics, 1997, ICAR '97*, pp. 193–200, 7–9 July 1997.

[12] Macwan A., Benhabib B., *A multi-robot coordination methodology for autonomous search and rescue*, *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pp. 675–680, 26–27 Sept. 2009.

[13] Jerne N. K., *Towards a network theory of the immune system*, Ann. Immunol. (Inst Pasteur), vol. **125C**, no. 1/2, pp. 373–389, Jan. 1974.