Department of Electrical
and
Computer Systems Engineering

Technical Report
MECSE-10-2003

An Evolutionary Computing Approach to Generating Useful and Robust Robot Team Behaviours

K.W. TANG, R. JARVIS

MONASH
UNIVERSITY

An Evolutionary Computing Approach to Generating Useful and Robust Robot Team Behaviours

By:

Kai Wing TANG

Supervisor: Prof. Ray JARVIS
Co-supervisor: A/Prof. Andy RUSSELL

Abstract

Designing control processes for mobile robot teams is a very difficult task. Consequently, methods for the teams to learn or adapt by their own are highly desirable. Genetic algorithm (GA), which is a class of techiques inspired by biological evolution, is a mainstream method to accomplish this self-adaptation objective. However, a major problem of the genetic algorithmic approach is the brittleness of the evolved solution(s) : The solutions provided by GAs can only work properly in those environments closely resembling the training environment. A slight change in the environment can render a very poor performance. In short, the solutions lack of robustness / generality and scalability. This paper discusses a design methodology to obtain robust solutions with the GA approach. The effectiveness of this methodology is demonstrated by an example: we have designed a mobile robot team with stable performance in a variety of environments.

1 Introduction

The application of evolutionary computation techniques to the design of robotic controllers has been evident for more than a decade. The reason for widespread use of genetic algorithms (GAs) and genetic programming (GP) is simple : Hand coding controller processes and parameters is a very tough task and GAs / GP provide(s) a mean to automate this tedious job. However, the time consuming characteristic of real environment evolution always forces the design process be split into two phases: The evolution is run in simulation first and then the evolved results are transferred to real robot(s). Unfortunately, published works from different researchers report that the transference from simulation to the real environment is not a straight-forward procedure [Brooks 1991] [Miglino, Lund and Nolfi 1996]. It is recognized that the evolved results lack of robustness. A successful transference is based on a very close resemblance of simulated and real environments. The evolved results fail to work even for slight deviations during the operation [Tommaso and Jason 1997].

For the projects involving mobile robot teams, this problem of specialty in the evolved results is more serious. Lots of factors like the number of robots in a team, the initial positions of the robots, and the variations in environments etc. all influence the performance of an evolved result. One approach (sequential approach) to cater for this problem is to take a result evolved successfully in a set of factors to a new environment and continue to evolve it. Another approach (compound environment approach) is to group

the different sets of factors together as a compound environment for evolution (Figure 1). However, since the space of variations in those factors are huge, both approaches require extremely long time to evolve, even in simulation. Furthermore, none of these approaches will work if a general, robust, solution does not exist at all. The non-existence may be due to two possibilities : (1) physical constraints disallow a feasible solution (2) limitations in chromosome representation make a feasible solution impossible. In either of these two cases, neither of the cited approaches works. The sequential evolution approach will appear as changes of phenotype values in the subsequent evolutions. In short, the results work properly in a latter evolution environment will no longer work for those previous environments. Whereas, for the compound environment approach, the best solution evolved from a compound environment will perform with a below average in anyone of the environments comprising the compound one. If the bad performance is due to reason (2), it is likely that by modification of the chromosome representations, a more general solution may be found. However, while the evolved results perform poorly, it is difficult to distinguish between cause (1) and (2).

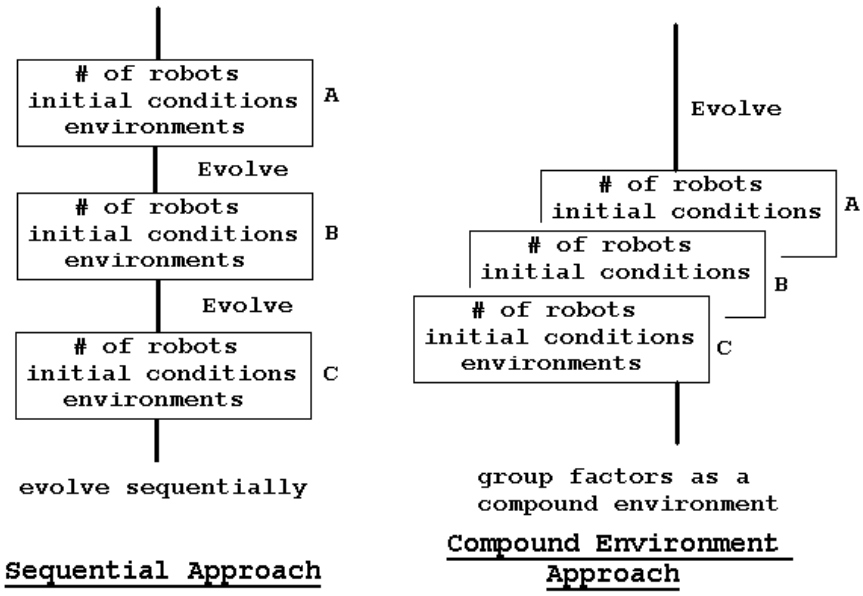

Figure 1: Two approaches to evolve a more general solution

In this paper we present an iterative design methodology to resolve these

difficulties. By using an environmental exploration project as an example, we demonstrate, step by step, how the procedures work. Also, we present the resultant behaviour in various environments (other than the ones used for evolution) to prove its robustness.

This paper consists of five other sections excluding this introduction. The next section describes the design methodology, followed by the setup of multiple robots experiments. The subsequent section narrates the steps we have taken by following the design method. Later, performance comparisons of the results are presented, and the conclusions follow.

2 Design Methodology

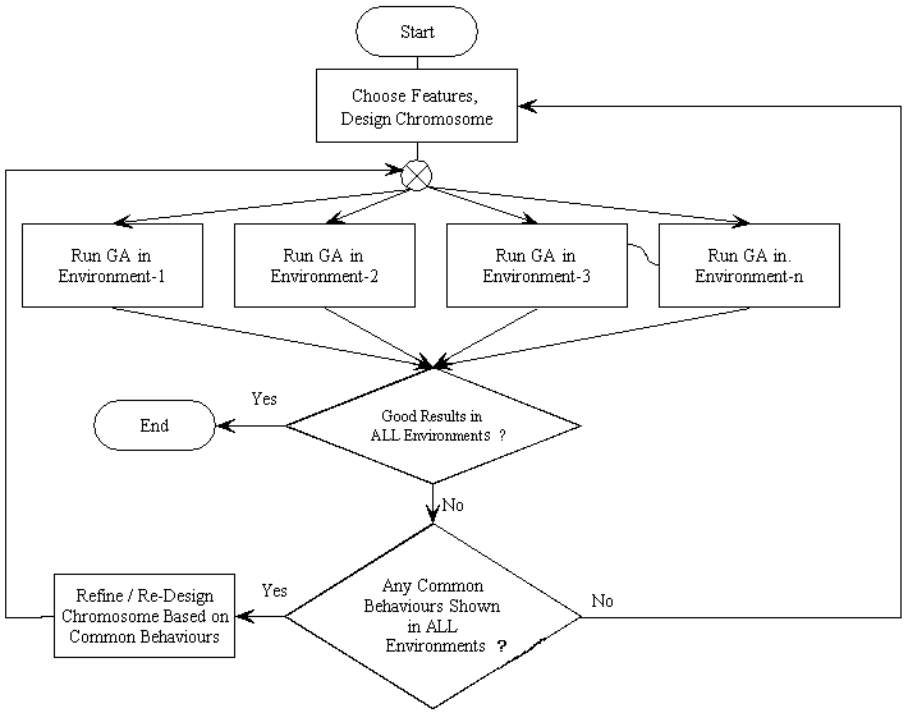

Figure 2: Design Flow

Figure 2 is the flowchart of the proposed design methodology. The hypothesis is such that some general, robust *behaviours* can be deduced from the *actions* in various environments. Here, we borrow the definition of be-

haviour from [Mataric 1994] : *A behaviour is a control law that clusters a set of constraints in order to achieve and maintain a goal*, whereas an action is the component that aggregates to form a behaviour. Our approach is a bottom-up one which infers some more suitable actions to form a behaviour in each iteration.

1. Choose feature and design chromosome.
   This is the basic step of any GAs related project. It requires certain craftiness, sometimes a bit of luck, to get a representation appropriate to the problem.

2. Run GA experiments in some different environments.
   Use the chromosome designed in the previous step, run GA experiments in various environments. For a problem related to the multiple robots, the factors mentioned in the introduction are some guidelines to create variety.

3. Check if a single chromosome instance runs satisfactorily in ALL environments.
   If the answer is positive, then this is very lucky because a good solution is obtained in first iteration. Basically, 99% of the cases are not so smooth. Consequently, we have to move to the next step.

4. Check if certain behaviour(s) emerge.
   Based on the result(s) obtained by the best instance in each different environment, see if there is / are any common denominator(s). In brief, see if some patterns can be formed by different instances in different environments. If the answer is negative, this implies that the chromosome representation is extremely poor and a drastic re-design is required. Then the process has to be re-started from Step 1. If the answer is positive, we move ahead to Step 5.

5. Re-design of chromosome based on emerged behaviours.
   Here, we get the same behaviour by different instances in different environments. We ask a question: what makes the same behaviour not possibly materializing on a single chromosome instance ? The answer to this question can help us to invent a more adequate chromosome representation. After the enhancement of the representation, the process jumps back to Step 2 to form an iterative loop.

The beauty of this proposal is the point of incremental refinement. We understand the fundamental restriction of GAs: the correct representation

is unknown in the beginning. Also, we exploit a fact usually ignored, i.e. even an ill-representation can give a specialized, though not robust, but good result. By skillful manipulation of this feature, we can use GAs to design the control laws for mobile robot teams. The next section gives an example.

3 Setup of Experiments

The environmental exploration task is a task that faithfully exhibits the benefits of using a team of autonomous robots instead of a single robot, i.e. utilization of parallelism, fault tolerance and scalability. A brief description of the environmental exploration task is as follows :

A number of robots are placed in an unknown environment. Each robot has a range finder of limited effective range so that each can detect the existence of obstacles or other robots within a close proximity. Once a robot realizes it is too close to the other robot(s), it will stop. As a result, all robots in the congested zone stop at once. Then, randomly, one of them will re-plan a new path and leave the zone. A short while later, another robot follows the same process. Hence, they disperse one by one from the congested zone and a collision is avoided.

The requirement of this task is for all robots working collectively to plot all unknown environment into a known map by moving around, while attempting to complete the task within the shortest period of time.

In order to fulfill the requirement of scalability, the communication among the team members cannot be too high, or the bandwidth will be saturated as the count of team members increases. On the other hand, the requirement of fault tolerance bans the use of a leader in the team (A breakdown of the leader will collapse the whole team). Therefore, the method of using a leader to maintain a collective map is not desirable. To sum up, in the project being described here, the luxury of sharing a common environmental map among the members are not granted. Instead, all members use a common global coordinate system and a robot always broadcasts its new x-y coordinates to the other members once it changes its location. Since an x-y coordinate pair is only four to six bytes in length, a wireless Ethernet medium can support the communication requirement of thousands of members. Furthermore, once two robots are close and visible to each other, i.e. no obstacle in between, they can exchange their explored environmental maps via infra-red signals.

If the evolution is done in a real environment, a minimum of thousands hours' running time is necessary, hence a simulated environment is used

instead. This simulated environment is a grid of 50 X 50 rectangular cells. Each cell represents the class of: space, obstacle, and robot. Every robot has its own version of environmental map, that is also represented by 50 X 50 cells, each in either states of: known or unknown. In the beginning, all cells are in an unknown state. The task of exploration is completed if *ALL* cells of any robot's map are set to the known state. The effective distance of the range finder on each robot is five cells. The number of robots in the team is ten.

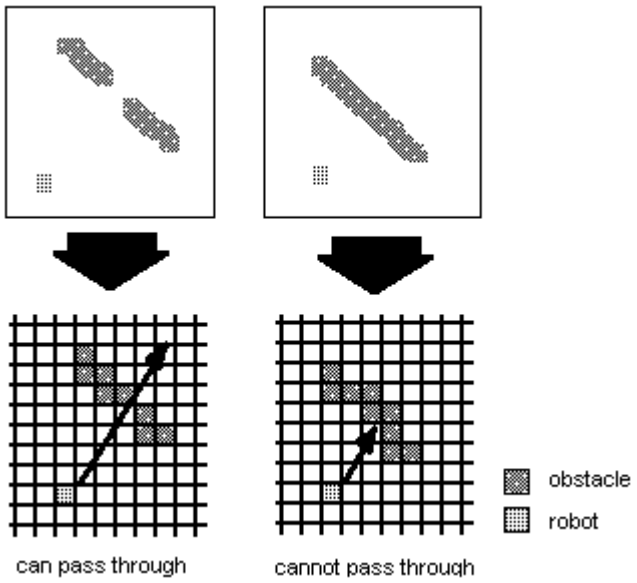

Figure 3: A robot can pass through diagonally placed adjacent obstacles.

For the purpose of simplicity, we make two assumptions in the simulation. (1) Each robot in the team always knows its own coordinate. That is to say, we ignore the localization problem. Since global positioning systems (GPS) are popular now, this assumption is justified. (2) The size of an obstacle is grown before mapping into the grid world, as shown in Figure 3, and a robot can pass through two diagonally placed adjacent obstacle cells. Presumably, these two simplifications only make the simulation easier to implement, but do not make the transference of the results to the real environment more difficult.

Because the searchable size of the state-space for GP is very huge (a range of 10^{30} to 10^{40} is the norm [Koza 1992]), we use GA with a list of production

rules as the chromosome representation instead of using GP in this project. The format of the rules is: if {condition-1,..,condition-n } then { action-1,..,action-m}. There is only AND relation for the conditions, but AND / OR relations for the actions. By using this representation, the size of the search space is only around 10^{12} ([Tang and Jarvis 2003] for implementation details).

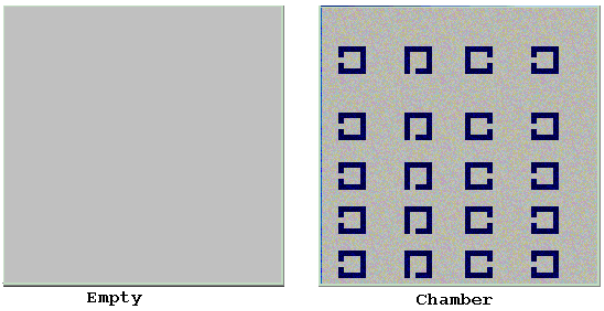

Figure 4: Two evolution environments, one empty and the other with chamber-like obstacles.

Figure 4 are the two evolution environments. For the environment 'Empty', two different initial positions of the robots team are used for evolution: at the central and at the top-left corner. For the environment 'Chamber', only the top-left corner is used.

4 Iterations of GA Trials

In summary, each robot in the team possesses the information of (1) whereabouts of the team members and (2) its map of known / unknown cells. Based on these information, it has to decide either of (1) moving to another robot to exchange the maps so as to get a more accurate scenario of the environment, or (2) moving to some unknown cells to update its own map. The chromosome representation is a list of production rules. The right choices of conditions / actions associations of these rules are the state-space for the GA to explore. The provision of ingredients or features for the GA to choose is our responsibility.
In the beginning, we do not know what features inside a known / unknown map are critical for the robots to make a correct decision, so the process is only started by trial and error. In order to shorten this groping period, we decide to make the robot broadcast the count of unknown cells in its map

as well as its coordinates.

1. First Trial
   The conditions, or environmental stimuli, are:

   - the coordinates of the centroid of the robot team
   - the number of team members at its proximity
   - the coordinates of the nearest unknown cell
   - the coordinates of the farthest unknown cell
   - the coordinates of the team member whose unknown cell count is the highest contrast to its own unknown cell count
   - how long since the current destination has been set

   The actions are:

   - move to the nearest unknown cell
   - move to the farthest unknown cell
   - move to the nearest team member
   - move to the team member with the highest contrast of unknown cell count
   - move to the team member in specific direction
   - change destination
   - change speed (the implementation of different speeds is for the other team member to catch up if it is keen to exchange maps)

   The result of this GA trial is not satisfactory. Even though the evolution gives converged results, the performance is just 30% better than the greedy strategy. (The greedy strategy is: always moving to the nearest unknown cell, and revising the destination instantaneously.)

   After studying the results obtained from this trial, we believe that the culprits are :

   - The contrast of unknown cell count is not a good metric to weigh the appropriateness for approaching. It is because two robots may possess two extremely different maps with the roughly same unknown cells. Since the contrast is low, the robots cannot be aware that a lot of information can be gained by approaching each other.

- Among the list of actions, there is none that can effectively cause spreading or merging.
- The various speeds are not necessary if a robot can stop for a short period at the right moment for the other members to approach.

Based on these findings, we changed the conditions / actions repertoire and run the second iteration.

2. Second Trial

- We designed a new metric named as Depth in Unknown Area. This is:
  Based on its own version of environmental map, a robot counts the number of unknown cells within a certain boundary around the other robots (example: fig 5).

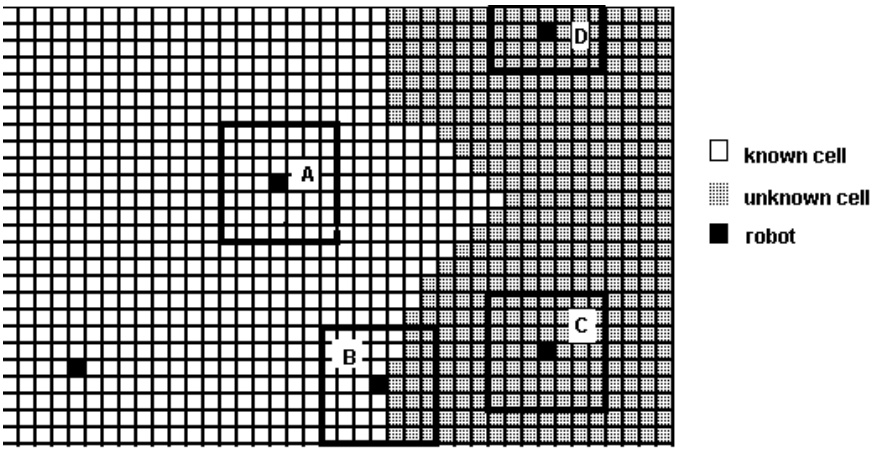

Figure 5: Count of Unknowns for robot A = 0, B = 19, C = 48, D = 27.

- We defined a Spreading operation as:
  Use distance transform [Jarvis 1984] to get the whereabouts of the unknown cell that is farthest from ALL other team members. Use this cell as the destination to plan a path.

- We defined a Merging operation as:
  Move to the centroid of the team.

Consequently, the conditions are:

- the unknown cell count of its own version of environmental map
- the summation of ALL team members' unknown cell counts
- the depth in unknown area
- how long since the current destination has been set
- whether the destination has been reached or not

The actions are:

- move to the nearest unknown cell
- move to the robot with the highest score in metric "depth in unknown area"
- spread
- merge
- stop
- change destination

The results obtained in this trial are good. ([Tang and Jarvis 2003] for results analysis). However, they are some specialized results only. The best chromosome instances for three difference evolution environments are three distinct ones. Nevertheless, we found that all these cases exhibit a similarity:

At first, the robots spread out as far away as possible until roughly maintain an equi-distant from all the other members. Then they used that location as the centre to explore a limited surrounding. Finally, they merged to exchange the maps.

We investigated the inventory of conditions and actions and drew the conclusion:
The merging operation is not efficient enough. If all members run this operation, it is a merging. But if only one or some of them run it, the efforts are wasted.

3. Third Trial

   We added a "merge-request" into the communication protocol among the members. Once a robot issues a merge-request, it checks if any other has sent the similar request or not. If not, that means the robot itself is the only one making this request, it will stop and wait. If some other(s) has / have requested, it will move to the centroid of those requesting members. This request will be cleared after the maps exchanged between two requesters.

   This time, we grouped the three evolution environments as a compound environment to run the GA. A chromosome instance which can perform satisfactorily for all three cases was found. Yet, if some changes like initial positions of robots are introduced, the performance is still not acceptable.

   At this stage, we believed that some constraints in the chromosome representation make the required behaviour impossible to realize. We drastically changed the design and run the fourth trial.

4. Fourth Trial

   The changes are:

   - By using distance transform, a robot detects the existence of an unknown cell in its proxmity whose distance between itself is shorter than those distances to all other members. If such an unknown cell exists, the robot moves to it. If this does not exist, a condition of equi-distant to other members is implied. Then it checks its proximity for unknown cells. If there is / are unknown cells existing, it moves to it / them, otherwise, it stops.
   - Each robot keeps a record of how long since the exploration operation has been started. Once a preset period is expired, it moves to the centroid of the whole team for map exchange.
   - After the exchanges of maps, it moves to the residual unknown cells.

   By using this program, there are only two parameters in the search space: the size of proximity and the expiry period for gathering. This is too simple for the GA to search. Also, now each member broadcasts its

x-y coordinates only (The unknown cell count and the merge-request are no longer required).

This program works adequately. It gives good performance, and it is immunity to environmental changes like changes of initial positions and count of team members, etc.

5 Results

Three different environments named as 'Mixed', 'Random' and 'Maze' as well as 'Empty' and 'Chamber' are used as testing environments (Figure 6).

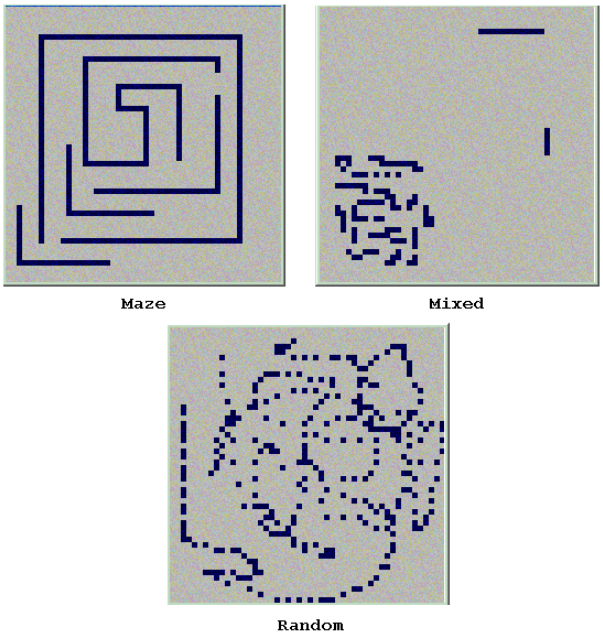

Figure 6: Three more testing environments.

Nine teams of different sizes, whose numbers of robots vary from two to ten, are placed in the five environments (two training and three testing environments), initial positions are choosen randomly. The exploration task is run repetitively in each environment for seven hundred fifty times.

Figures 7 and 8 are the distributions of the completion time of four and ten robots, respectively, in environment 'Empty'. This is measured in a time unit defined as the average time for a robot to move to one of its eight-

neighbouring cells. It shows that the performance is relatively more stable for a team with more robots.

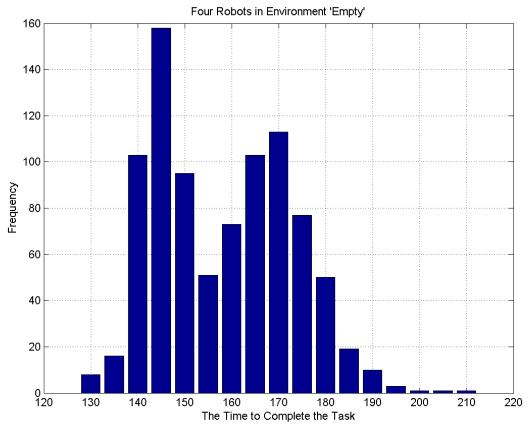

Figure 7: Distributions of complete time (4 robots)

The averages and the standard deviations of the required time are plotted against number of robots for these five environments (shown in fig. 9 to fig 13).

These five graphs show that except for the environment 'Maze', the teams of different sizes perform consistently in four different environments. For the environment 'Maze', the presence of alleys makes it hard to exploit the advantage of parallelism. Accordingly, 'Maze' is not a good environment for utilization of the robotic swarm. Its result with a higher standard deviation is well expected.

The next five figures (fig. 14 to 18) show the efficiency against the number of robots in the five environments. A 100% efficiency means a n-robot team can complete the task using 1/n of the time by a single robot. Likewise, taking the environment 'Empty' as an example, the efficiency of a 4-robot team is 55%. The team can complete the task in:

$$1/4/0.55 = 0.455$$

of the time required by a single robot. With reference to the histogram of 'Time Vs No. of Robots in Empty', the average time required is 157 and 345 for the single and the 4-robot team, respectively.

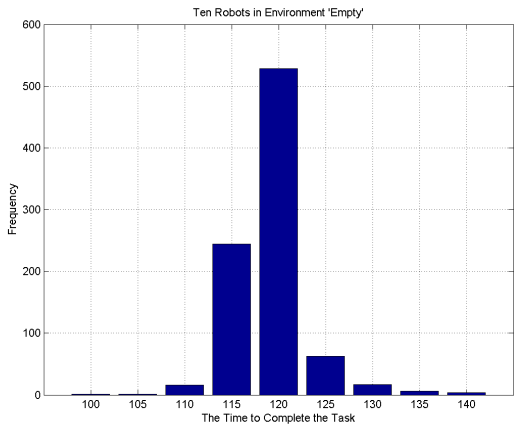

Figure 8: Distributions of complete time (10 robots)

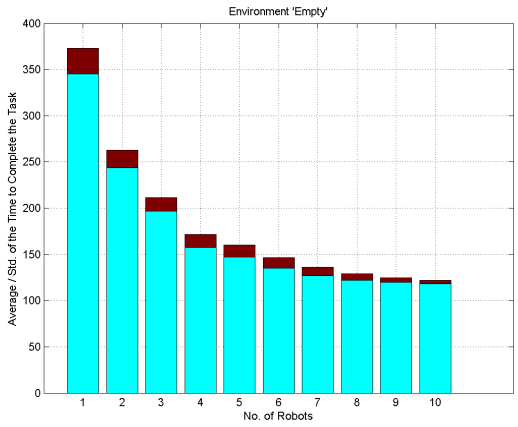

Figure 9: Total time consumed versus No. of Robots (Empty).

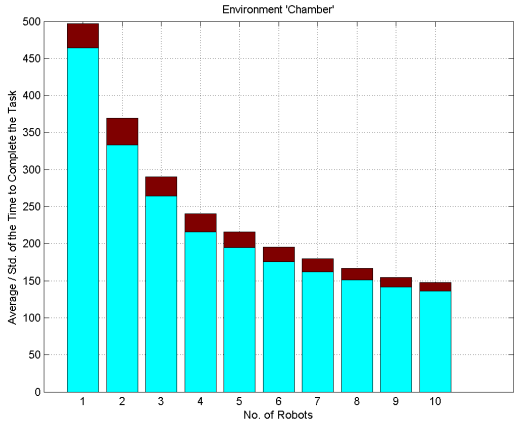

Figure 10: Total time consumed versus No. of Robots (Chamber).

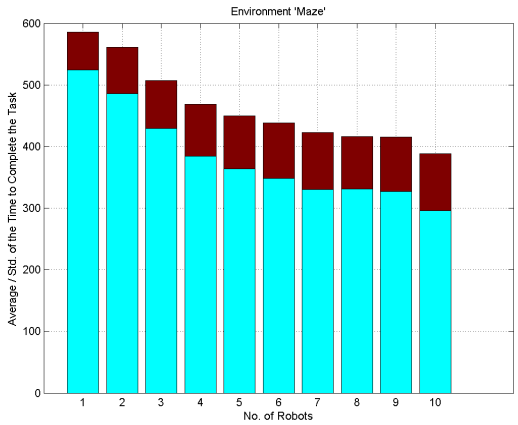

Figure 11: Total time consumed versus No. of Robots (Maze).

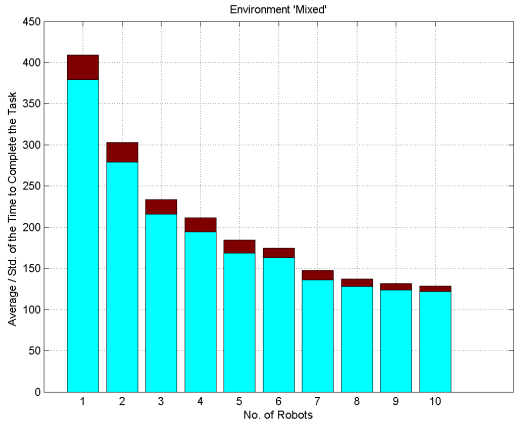

Figure 12: Total time consumed versus No. of Robots (Mixed).

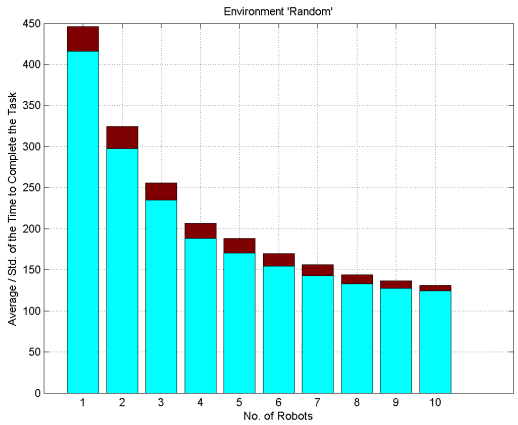

Figure 13: Total time consumed versus No. of Robots (Random).

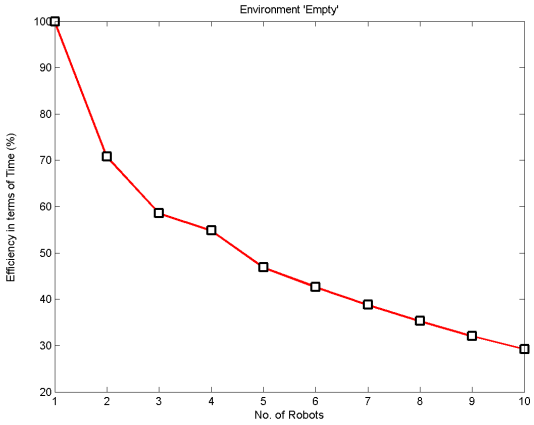

Figure 14: Efficency in percentage versus No. of Robots (Empty).

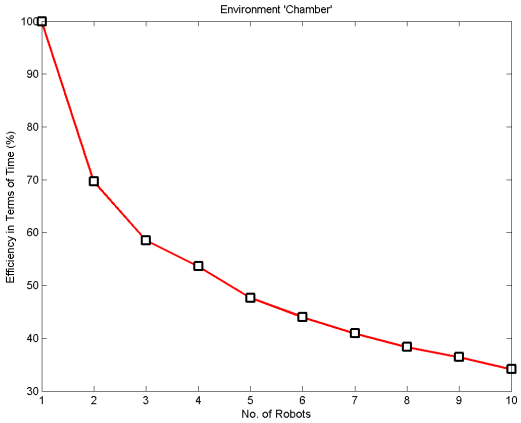

Figure 15: Efficency in percentage versus No. of Robots (Chamber).

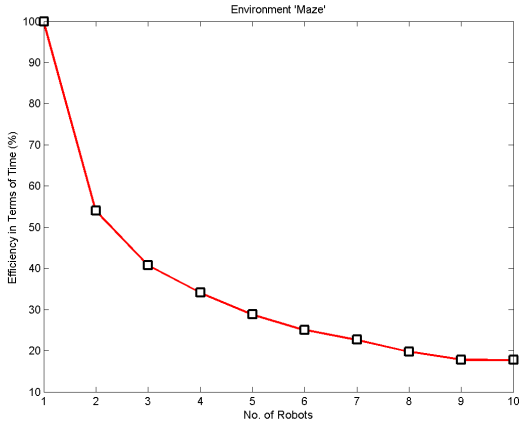

Figure 16: Efficency in percentage versus No. of Robots (Maze).

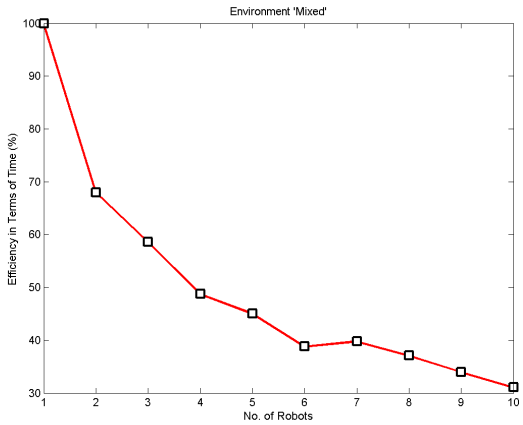

Figure 17: Efficency in percentage versus No. of Robots (Mixed).

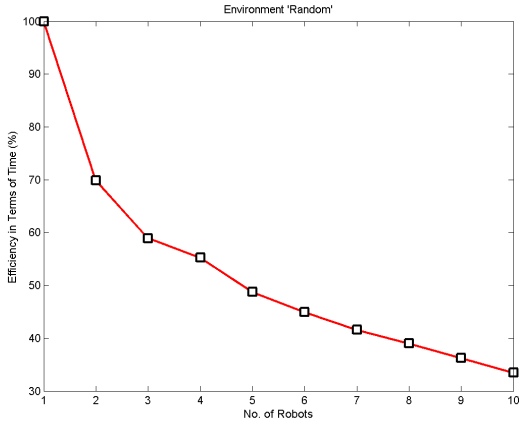

Figure 18: Efficency in percentage versus No. of Robots (Random).

6 Conclusions

The results obtained to date are appealing. Although it has not been implemented in real environment, since no particular assumptions are made in the simulation, we are confident that the transference process will be straight-forward.

The example shown here proves that robust solutions can be found by GAs. Sometimes, even though an adequate chromosome representation is absent, we can still rely on specialized solutions to trace the form of a general solution.

References

[Brooks 1991] R. Brooks *Artificial Life to Actual Robots.*, Proc. of the first European conf. on Artificial Life, MIT Press, Cambridge, Massachusetts. 1991.

[Jarvis 1984] R.A. Jarvis. *Collision-free trajectory planning using distance transforms*, Proceedings of National Conference and Exhibition on Robotics, Melbourne, 1984.

[Koza 1992] J.R. Koza *Genetic Programming, On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Massachusetts. 1992.

[Mataric 1994] M.J. Mataric *Interaction and Intelligent Behaviour*, Tech. rep. AI-TR-1495, Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts. 1994.

[Miglino, Lund and Nolfi 1996] O Miglino and H. Lund and S. Nolfi *Evolving Mobile Robots in Simulated and Real Environments.*, Artificial Life, MIT Press, Cambridge, Massachusetts. 1996.

[Tommaso and Jason 1997] F.B. Tommaso and M.D. Jason *A Discussion on Generality and Robustness and a Framework for Fitness Set Construction in Genetic Programming to Promote Robustness.*, Late Breaking Papers at the 1997 Genetic Programming Conference, Stanford Bookstore, Stanford University, CA. 1997.

[Tang and Jarvis 2003] K.W. Tang and J.A. Jarvis *Production Rules As Chromosomes of GA on Robotic Swarm Application*, Submitted to Australasian Conference on Robotics and Automation 2003.