From Supervised Ranking to Evolving Behaviours of A Robotic Team

Kai Wing TANG
Intelligent Robotics Research Centre
Department of Electrical and Computer Systems Engineering
Monash University, Victoria, Australia
Kai.Tang@eng.monash.edu.au

Ray A. JARVIS
Intelligent Robotics Research Centre
Department of Electrical and Computer Systems Engineering
Monash University, Victoria, Australia
ray.jarvis@eng.monash.edu.au

ABSTRACT

Using artificial evolution successfully to design behaviours of multiple robot systems has been reported in recent years. Most of such reports are focused on the design of low level controllers. Design of high level team coordination strategies is rarely covered perhaps because the design of an appropriate chromosome representation for a complex multi-agent system is not an easy task. In this paper we propose that by treating the action decisions of every team member as a supervised ranking problem, the chromosome design problem can be solved systematically.

We have tested this approach by dynamically solving the problems in the Solomon's benchmark of Vehicle Routing Problem with Time Windows [1]. Experiments show that our approach can create some simple behaviours which, whilst not optimal, are robust and above average in quality.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Knowledge Acquisition; I.2.11 [**Distributed Artificial Intelligence**]: Coherence and Coordination

General Terms

Design, Algorithms

Keywords

Chromosome representation, free market-based architectural approach, supervised ranking, team coordination strategy

1. INTRODUCTION

The benefits of using a team of multiple, simple, robots instead of a single, complicated robot to accomplish a task have been well discussed in the literature [2]. Recently, many researchers focused on the design of robotic team coordination strategies. For a multiple robot system, either homogeneous or heterogeneous robots work cooperatively to achieve a common goal. The behaviours of each robot interact with one another until the required goal is accomplished. How to learn these behaviours and how to define the effectiveness of some behaviours to achieve a collective goal are central topics of Distributed Artificial Intelligence (DAI) [3].

One popular approach is the free market-based architectural approach [4]. It first defines cost / revenue functions for executing a specific task. Then it further sub-divides the collective task into sub-tasks for individual robots to perform. By using a bid and negotiate mechanism, cooperation or competition materialises and theoretically, the optimal result can be accomplished by all participants maximising their own profits. However, this approach has a prerequisite: accurate revenue and cost functions. These are difficult to determine if the specific task involves many different types of participants whose roles are interlocked. Taking the RoboCup Rescue Problem [5] as an example: A big city is ruinned by an earthquake, many buildings have collapsed or are burning, civillians are buried and roads are blocked. There are three types of rescuers: police cars, fire engines and ambulances. The police cars' role is to clear blocked roads, whereas the fire engines' and the ambulance platoons' are to extinguish burning buildings and save civilians, respectively. The collective goal is to minimise the number of casualities as well as to put out as many burning buildings as possible. Here, the actions of rescuers are interdependent: roads have to be cleared otherwise the fire engines cannot reach the fire sites, ambulances cannot enter a burning building before the fire has been extinguished. How to maximise an individual's profit which in turn can realise an optimal collective goal is not straight forward. In short, the cost and revenue of the same action will vary in different scenarios.

Using figures 1 and 2 as an example: The cost of clearing blockage 1 or 2 are the same because the police car is in the middle between the two blockages. In fig. 1, site 2 is with a higher risk of spreading than site 1 because there are more buildings near the fire. Assuming the bid provided by a fire engine going to a fire site with a higher risk of spreading is higher than that of going to a lower risk site, will the police car make decisions based on the total sum of bids, or on the highest bid only? If it is the total sum, in fig. 1 it will clear blockage 1 first because the sum of two fire engines' bids are higher than the single engine's. This will be a wrong choice because site 2 is more urgent. On the

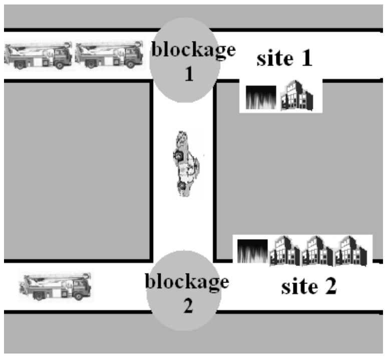


Figure 1: The police car has to decide clearing which blockage, 1 or 2, first. Clearing blockage 2 first is better because site 2 has a higher risk of spreading.

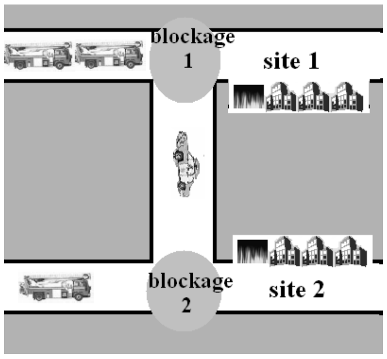


Figure 2: Clearing blockage 1 first is better because there are two fire engines.

contrary, if it is the highest bid, in fig. 2, it is a tied case because only one engine's bid (both sites are with the same risk) are considered. However, it is obvious that blockage 1 is a better choice since site 1 can be put out within a shorter period (two engines work faster than a single engine). This example demonstrates that no single bidding scheme can satisfy both cases. It illustrates how the dynamic and volatile natures of these cost and revenue functions make this market-based architecture difficult to implement.

Another popular approach to design behaviours of multiple robot systems is the utilization of artificial evolution to evolve individual behaviours [6] [7] [8]. Recently, a number of researchers have reported promising results in the field of evolutionary robotics. However, most of the approaches taken are focused on the design of low level controllers or subsumption behaviours. The area of high level coordination strategies for robotic teams is rarely touched. This is due to the paradox of the chromosome design problem: On one hand, the evolutionary computing (EC) approach is preferred to other design methods because it has been proven to be capable of solving problems whose nature is complex and not fully understood. On the other hand, EC requires an appropriate chromosome representation which maps a probable solution to a data structure. The success or failure of an application of EC is highly dependent on the relevance of this representation. If the problem nature is ill-comprehended, how can a good chromosome representation be found?

In this paper we propose a method which, through the conversion of the next actions determination problem into a supervised ranking problem (SRP), can effectively link up short and long-term effects and accordingly use EC to perform the task of team coordination strategy design.

This paper consists of four other sections. Section 2 explains what the supervised ranking problem is and what a suitable chromosome representation to evolve a SRP is. Section 3 narrates how to apply the SRP to design a team coordination strategy. Section 4 is about the results of an experiment: applying this method to dynamically solve the Vehicle Routing Problem with Time Windows (VRPTW). Section 5 is the discussion and conclusions.

2. THE SUPERVISED RANKING PROBLEM

2.1 The Ranking Problem

Basically, the ranking problem is a multi-criteria (MC) decision making problem [9]. Given a finite number, N, of instances $I_1, .., I_N$, each of which is represented by n merit attributes, i.e. $I = \{a_1, a_2, .., a_n\}$. Each merit attribute is either numeric, ordinal or interval. Consequently, each instance can be represented by a n-ordered vector: $I_i = \{a_{i1}, a_{i2}, .., a_{in}\}$. Comparison with a binary relation: better than (>) can be made on any single merit attribute. The ranking problem is to determine whether $I_i > I_j$ based on the relations between a_{ik} and a_{jk}, for k={1..n}. Normally, the dominance rule always applies. It means that if one merit attribute is better than, and all the other attributes are equivalent to or better than, then this instance is better.

$$if \quad \exists k, \quad a_{ik} > a_{jk}, \quad and$$

$$a_{im} \geq a_{jm} \quad \forall m \neq k; \quad then$$

$$I_i > I_j$$

The ranking problem is complicated only if this dominance rule cannot apply. The next two subsections are about applying EC to determine if $I_i > I_j$ when not all $a_{im} \geq a_{jm}$.

2.2 The Supervised Ranking

The supervised ranking problem is about how to use machine learning methodology to induce a multi-criteria preference function from pairwisely presented training sample instances: Given N(N-1)/2 pairs comparison results of the N instances, a MC preference function, which can predict future comparison results correctly, is devised. This MC preference function compares and discriminates between different attributes (fig. 3).

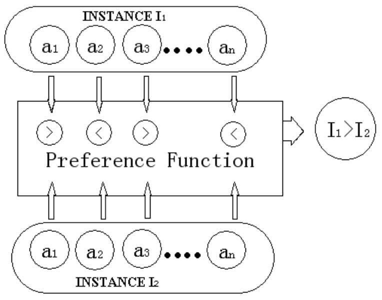


Figure 3: MC preference function discriminates between different attributes.

The MC preference function is an evaluation program which either draws the final conclusion based on some attributive comparisons, i.e. in a form of decision tree (fig. 4); or, gives each result of an attributive comparison a score, and then transforms this set of scores into an ultimate score for making final decision, i.e. as a set of parametric weights.

The format of parametric weights forms the ultimate score by a linear weighed combination of the attributive comparisons.

In the next subsection we will explain that the decision tree representation is more compact for GA chromosome encoding.

2.3 The Chromosome Representation to Evolve the Preference Function of a SRP

The common evolutionary computational methods choose the structure of parametric weights, and encode the weights as the genes of the chromosome. This approach has some shortcomings:

- Since the relations among the attributes are unknown beforehand, the appropriate ranges of the weights are unknown also. Lacking this information may waste the searching effort on some unfruitful areas.
- Different attributes will have different types of values. A weighted sum of different data type may not correctly characterise different significance of attributes.
- The attributes may not be independent to each other
- The relations among the attributes cannot be represented effectively by a linear weighted sum.

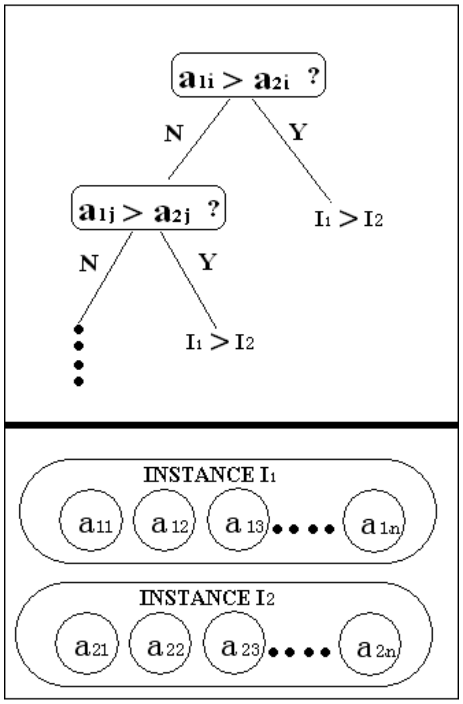


Figure 4: The preference function in a form of decision tree.

Therefore, we propose another chromosome format which can evolve a decision tree effectively.

The chromosome is composed of two portions, the first portion defines the order of significance and the second portion defines the margin of distinction.

Suppose there are n attributes, the first portion will consist of (n-1) genes and the second portion n genes, respectively.

For the first portion, the 1st, 2nd, 3rd, ... (n-1)th gene will have values within the range of {1..n}, {1..(n-1)}, {1..(n-2)}, ... {2}, respectively. This portion represents n! combinations of comparsion sequences of the n attributes.

For example, let n = 6. The first portion will consist of n-1, i.e. 5 genes. The value of the first gene is in the range of {1..6}, and the fifth is {1..2}, etc.

Let the six attributes are named as A, B, C, D, E, F. The value of the first gene defines the place of the most important attribute in this list. According, the value of second gene will be the place of the most important one in the list excluded the first one.

Let the values of the genes be: 5 1 3 2 1

The most important attribute is the fifth, i.e. E. Then E is remove from the list.

init: A B C D E F —-(5::E removed)
left: A B C D F ——-(1::A removed)
left: B C D F ———-(3::D removed)
left: B C F ————-(2::C removed)
left: B F —————-(1::B removed)
left: F

Therefore, the genes of values 5 1 3 2 1 defines the significance order of E A D C B F.

Basically, this is a form of Ordinal Representation of Grefenstette which has been reported [10] that is not very effective in solving Travelling Salesman Problem (TSP). It is because of a major drawback: front parts before the cutoff point are preserved, but back parts of the offsprings of a classical crossover operation are disrupted in a random way. From our observation, influences of the attributes with lower significance orders seem dropping steeply. Accordingly, this representation works satisfactorily for our applications.

The second portion defines the margin of distinction. When two attributes are compared, if the difference is less than or equal to that gene's value, the attributive comparison result is considered to be equal.

By using this form of chromosome, two instances are compared by following the order of the first portion, and using the second portion to determine if the comparisons of attributes are enough to draw a conclusion. If the result of the more significant attribute comparison is equal, the next attribute will be compared until a distinction can be make. If all differences are within the ranges defined by the second portion, then the two instances will be treated as equals.

Using the Olympics Game ranking method as an example: The number of gold medals earned by a country determines the ranking. The number of silvers is considered next and then the number of bronzes. If the preference function is represented by a weighted sum format, then the chromosome will have three genes whose ranges depend on the number of events. Say, there are a total of 300 events, then one gold is equal to 300+1 silvers, and one silver is equal to 300+1 bronzes, which implies one gold is equal to 300 * 300 + 1 bronzes. The three genes' ranges will be 300*300=90,000. On the other hand, first portion of our chromosome representation will be two genes in ranges 3 and 2, respectively; whereas the second portion will be three genes, each with a range of 300. This is much shorter compared with the parametric format.

3. APPLICATION OF SRP TO DESIGN TEAM COORDINATION STRATEGIES

For any member of a team, at any time, there should be a list of advantageous action sequences. The term advantageous means that if any one of these sequences is performed, at least certain part of the collective goal is fulfilled. Again, taking fig. 1 as an example: the actions available for the police car are clearing blockage 1 or clearing blockage 2. The actions available for the single fire engine AFTER blockage 2 is cleared are putting out the fire at site 2 or following the police car, waiting to put out the fire at site 1.

After an action sequence is performed, some immediate, measurable results can be reckoned: For the police car, if clearing blockage 2, a fire engine is accessible to a higher risk site. If clearing blockage 1, two fire engines are accessible to a lower risk site. For the fire engine after blockage 2 is cleared, if putting out site 2, site 2 is extinguished after a period of t_a. If following the police car, site 1 is extinquished after a period of $t_b, t_b \neq t_a$.

The immediate results caused by different action sequences can be reckoned easily. However, what makes the team coordination problems difficult is the intricacy of their long-term effects. Their collective long-term effects (the total number of saved civilians and total number of buildings not destroyed by fire) will only be available after the whole episode (300 seconds after the earthquake) is finished.

Behaviours of choosing one entry from a list of advantageous action sequences can be modelled by a preference function by treating the immediate, short-term results of an action sequence as merit attributes of a SRP (fig. 5).

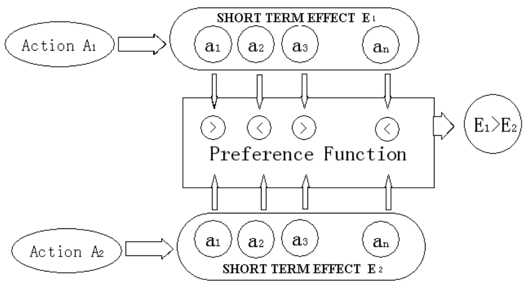


Figure 5: Short-term effects of an action are regarded as merit attributes.

For a normal SRP, ranking orders of the training dataset are given as the N(N-1)/2 comparison results. Here, for the design of coordination strategies, the ranking order is obtained from the global collective result after running a whole episode. The collective result is used as the fitness value of a genetic algorithm (GA). Accordingly, the preference function controls which action sequence to be taken, and the action sequence contributes to determine the group performance. Finally the group performance is used as the fitness to evolve better preference functions (fig. 6).

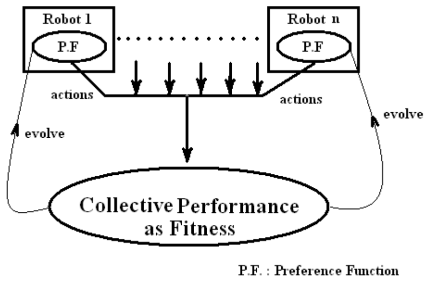


Figure 6: The group performance is used as the fitness to evolve better preference functions of every robot.

Consequently, the task of team coordination strategy design is to:

- partition the episodic run into sub-sections based on environmental states
- define, for each sub-section, and for each type of participants, a list of advantageous action sequences
- define, for each action sequence, its short-term results as merit attributes of a SRP
- encode merit attributes as described in previous subsection

- let GA to evolve the best sets of preference functions

This concludes our proposed method. In next section, we demonstrate its effectiveness.

4. EXPERIMENT

Since most of the researchers on team coordination strategies work on their own problems, there is not a common, popular benchmark to evaluate the performance of our proposed method. Originally, the RoboCup Rescue Problem was a good testing ground. However, certain modules of the RoboCupRescue Simulation System like Fire Spreading, Panicked Pedestrian Behaviours, etc. are still under development and are evolving. It is difficult to get objective comparisons of experimental results in this environment. Consequently, we use a typical vehicle routing problem and the associated well-known benchmark problems to demonstrate the performance of our method. This problem set is chosen because:

- It is NP-hard.
- If it is solved statically, a great combinations of routes have to be tried to obtain the optimal solution. However, if it is solved dynamically, a behavioural approach which makes decisions based on environmental states will be the preferrable method.
- It contains a variety of problem instances. Some algorithms work very well on a subset of these instances may deteriorate significantly on other instances.
- Either optimal solutions or heuristic best results of the problem instances are published. Our method's performance can be quantitatively compared.

4.1 Vehicle Routing Problem with Time Windows

A vehicle routing problem (VRP) requires a number, N, of customers to be served by n, where $N \gg n$, vehicles. All of these vehicles originate from a single depot, and these N customers are scattered geographically. The problem is to design routes which start and end at the depot, and in addition, each customer can only be visited by exactly one vehicle. The problem is to minimise the required vehicles, i.e. n, as well as the total length of the routes taken by the vehicles.

The Vehicle Routing Problem with Time Windows (VRPTW) is a VRP with the added constraint that each customer can only be served within a time window: If a vehicle arrives too early, it has to wait until the time window is open. If it arrives too late, that customer cannot be served.

4.2 Solomon's Benchmark

Each of these problem instances contains 100 customers [1]. The travel velocity of the vehicles is unity, i.e. travel times are equal to their Euclidean distances. Each vehicle has an equal amount of goods which will be spent on serving the customers. The required capacity of goods varies from customer to customer. There are three groups of problems: R, C and RC. The locations of customers in R are randomly placed, whereas those in C are clustered. The RC group is a mixture of R and C, i.e. randomly scattered customers mixed with customer clusters. Each group contains 8 to 12 problem instances, where the customers' time windows vary in length, starting time and ending time, etc.

4.3 Attributes and Behaviours

In our experiments, the VRPTW is solved dynamically. A vehicle, either in the depot, or after serving a customer, has to decide who is the customer to be served next. We assume that there is enough communication bandwidth among the vehicles such that each vehicle always knows:

- Whereabout of the other vehicles.
- The served customers as well as those waiting for services.
- Other vehicles' remaining goods.

Based on these pieces of information, for each vehicle - waiting customer (not served) pair, we define a relation: accessibility. A vehicle is accessible to a waiting customer if the vehicle can reach that customer before whose time window is closed; and it has enough goods to serve that customer. Consequently, each not served, i.e. waiting, customer has a counter holding the total number of vehicles accessible to it, i.e. the accessible count.

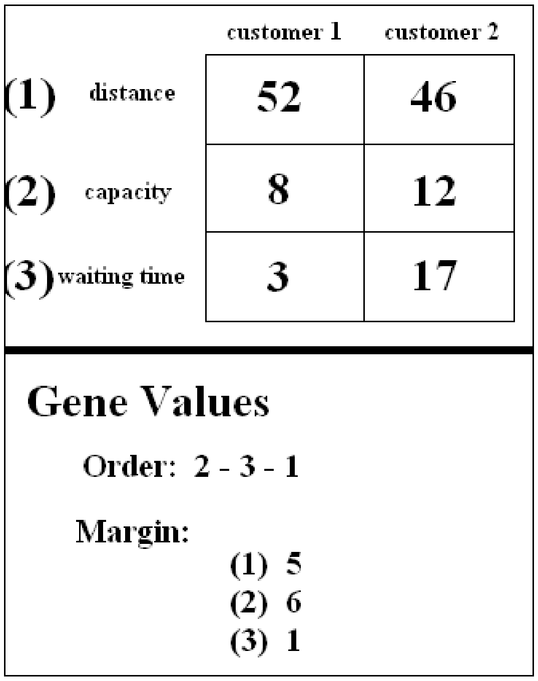
With respect to each accessible vehicle, a waiting customer is represented by three merit attributes:

- Relative distance (route length).
- Waiting time for window opening.
- Goods left after serving it.

The behaviours of the vehicles are encoded as follows:

- If the accessible counts of certain waiting customers drop below some threshold, a new vehicle will leave the depot. A vehicle leaving the depot has to decide whom to serve. It resolves by comparing route length, waiting time and accessible counts of all waiting customers.
- While no new vehicle is required, i.e. the accessible counts of ALL waiting customers are above the threshold, the merit attributes described before will be compared among vehicles accessible to a waiting customer. The best accessible vehicle to a waiting customer, or, the champion, is then defined.
- Each waiting customer will have one and only one champion. However, a single vehicle may become the champions of more than one waiting vehicles. Therefore, a vehicle will have a champion count, which indicates it is the champions of how many waiting customers.
- Another comparison is used to decide which champion gets its best waiting customer first. This comparison is based on the merit attributes of accessible count, champion count, route length, window open waiting time and goods left.
- After decided who is the next customer, a champion with the champion count more than one will release the other customers for next assignments.
- This customer assignment process iterates until all customers are served.

These various comparisons are then coded as SRPs. For example, after a vehicle has just served a customer, it has to decide who is the next to be served. Firstly, based on the precedence order encoded in a gene of the chromosome, it compares the merit attributes like distance, waiting time etc. of each waiting customers against all vehicles to get a subset of customers regarding itself as the champion (fig.7 as an exampe). Secondly, it uses the decision tree encoded in another gene to get the most beneficient customer in this subset.



	customer 1	customer 2
(1) distance	52	46
(2) capacity	8	12
(3) waiting time	3	17

Gene Values

Order: 2 - 3 - 1

Margin:

(1) 5

(2) 6

(3) 1

Figure 7: Since the order is 2-3-1, it compares capacities first. As the difference 12 - 8 $<$ 6, it continues to compare waiting times and concludes customer1 is a better choice.

One of the problem instances of the Solomon's benchmark, i.e. C101 to C109, R101 to R112, RC101 to RC108; is randomly chosen as the evolution environment. In our experiment, R106 is used. A 100 individuals per generation, 40% selection, 60% crossover, 1% mutation rates GA program is coded. The total route length; actually, a constant minus the total route length because we want to minimise this quantity; is used as the fitness. One elite is preserved from each generation. The results are tabulated as in table 1.

4.4 Results

The column of worst distance in the table is the longest route length: using 100 vehicles to serve 100 customers. The performance percentage (optimal is 100%) is calculated by:

(Our Length - Best Length) / (Worst Length - Best Length)

It is shown that our results are not optimal and the average is only 87.1%, which is quite weak compared with those results in the literature. However, our algorithm gets these results responsively. It does not enumerate combinatorial routes, and, it does not back-track. It gets its preference functions from R106 only. On the contrary, the other approaches in the literature run GAs or other heuristics on each problem instances independently. They are specialised results to only one problem instance, whereas the results presented here are general results to ALL problem instances. Therefore, it is more robust. In case of sudden changes like vehicle breakdown or new customer requests, our approach can still handle and can provide a satisfactory service level.

5. DISCUSSION AND CONCLUSIONS

In this paper we propose a method which uses the long delayed collective reward as a ranking result to evolve preference functions. Actually this method is inspired from the market-based architectural approach. We find the bid and negotiate mechanism requires the conversion of different results of an action into a common unit. This conversion will be very difficult, if not impossible, for complex, inter-related multiple agent systems (MAS). Accordingly, we propose to use an EC approach to fill up this gap. To a certain extent, we may claim that this method is a partial integration of the market-based architectural and EC approaches.

The central idea of our method is using GAs to select one among conflicting actions, which can induce the best long-term results. Comparing to the other popular Pareto evolutionary algorithms [11], which are wellsuited for optimization problems involving several conflicting objectives, our method is much simpler. One salient feature of Pareto approaches is they can search multiple solutions concurrently in a single run. However, the long-term results of an agent's action in a MAS are only loosely related to the short-term merits. Consequently, we believe that making decisions based on hierarchical preference functions instead of Pareto front solutions are sufficient,but the save in processing power is significant, for MAS applications.

The results presented in this paper can demonstrate our approach's strength, the generality of this approach is a topic for further evaluations. Our next step is to apply it to a search and rescue operation experiment. And presumbly, real robots will be used to test its validity in the final phase.

Another major contribution of this paper is that it proposes a method which can help researchers to logically design chromosomes of GAs for MAS. It bridges the gap of chromosome design paradox: even though the problem nature is not thoroughly known, GAs can be applied to evolve short-term preferences which result in acquiring good long-term performance.

6. REFERENCES

[1] M. Solomon, Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints, *Operations Research* Vol. 35, pp. 254 - 365, 1987.

[2] R. Arkin and T. Balch, Cooperative Multiagent Robotic Systems, *in David Kortenkamp, R.P. Bonasso, and R. Murphy, editors, Artificial Intelligence and Mobile Robots.* MIT/AAAI Press, Cambridge, MA, 1998.

[3] M.J .Mataric, Coordination and Learning in Multi-Robot Systems, *IEEE Intelligent Systems*, pp 6-8, 1998.

	Distance				Vehicle		
Instance	Worst	Optimal	Our	%	Optimal	Our	Diff.
C101	5771.0	827.3	883.8	98.9	10	12	+2
C102	5771.0	827.3	1251.9	91.4	10	11	+1
C103	5771.0	826.3	1361.0	89.2	10	12	+2
C104	5771.0	822.9	1536.7	85.6	10	12	+2
C105	5771.0	827.3	1081.1	94.9	10	11	+1
C106	5771.0	827.3	1435.6	87.7	10	12	+2
C107	5771.0	827.3	1118.5	94.1	10	10	0
C108	5771.0	827.3	1447.6	87.5	10	11	+1
C109	5771.0	827.3	1374.1	88.9	10	11	+1
R101	4989.4	1637.3	1963.6	90.3	20	21	+1
R102	4989.4	1466.6	1845.0	89.3	18	20	+2
R103	4989.4	1208.7	1830.7	83.5	14	18	+4
R104	4989.4	982.0(H)	1809.1	79.4	10(H)	17	+7
R105	4989.4	1355.3	1841.9	86.6	15	17	+2
R106 (*)	4989.4	1234.6	1411.3	95.3	13	13	0
R107	4989.4	1064.6	1657.4	84.9	11	14	+4
R108	4989.4	960.9(H)	1637.2	83.2	9(H)	14	+5
R109	4989.4	1146.9	1825.9	82.3	13	17	+4
R110	4941.3	1068.0	1890.2	78.8	12	16	+4
R111	4989.4	1048.7	1569.4	86.8	12	14	+2
R112	4989.4	982.1(H)	1652.0	83.3	9	14	+5
RC101	6617.5	1619.8	1997.2	92.4	15	18	+3
RC102	6617.5	1457.4	2001.5	89.4	14	15	+1
RC103	6617.5	1258.0	1955.5	87.0	11	15	+4
RC104	6617.5	1135.5(H)	2193.3	80.7	10(H)	16	+6
RC105	6617.5	1513.7	2168.6	87.2	15	19	+4
RC106	6617.5	1424.7(H)	2351.1	82.2	11(H)	19	+8
RC107	6617.5	1230.9(H)	2148.1	83.0	11(H)	15	+4
RC108	6617.5	1139.8(H)	2098.6	82.5	10(H)	16	+6
	Average			87.1	Average		3.03
	Std.Dev.			4.99	Std.Dev.		2.10
	Max.			98.9	Max.		8
	Min.			78.8	Min.		0

Table 1: The distances with (H) are the heuristic best. The instance with asterisk (*) is the training one.

[4] M. Dias and A. Stentz, A Market Approach to Multirobot Coordination, *Technical Report, CMU-RI-TR-01-26*, Carnegie Mellon University, 2001,

[5] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research, *Proc. IEEE Conf. on System, Man and Cybernetics*, 1999.

[6] M. Quinn, A comparison of approaches to the evolution of homogeneous multi-robot teams, *Proc. the Congress on Evolutionary Computation (GECCO2001)* pp. 128-135, Seoul, S. Korea. IEEE Press, 2001

[7] H. Liu and H. Iba, Multi-agent Learning of Heterogeneous Robots by Evolutionary Subsumption, *Proc. the Congress on Evolutionary Computation (GECCO2003)*, Chicago, USA. IEEE Press, 2003

[8] K.W. Tang and R.A. Jarvis, An Evolutionary Computing Approach to Generating Useful and Robust Robot Team Behaviours, *Proc. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp 2081 - 2086, 2004.

[9] K. Cao-Van and B. Baets, On the definition and representation of a ranking, *Proc. Sixth Int'l. Workshop on Relational Methods in Computer Science*, pp. 316 - 324, 2001.

[10] J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht, Genetic Algorithms for the TSP, *Proc. First Int'l. Conf. on Genetic Algorithms and Their Applications*, pp. 168 - 168, 1985.

[11] E. Zitzler and L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation* Vol. 3, No. 4, pp. 257 - 271, 1999.