Co-Evolutionary Algorithms for Motion Planning of Multiple Manipulators

December 21, 2018

Contents

1 Introduction

Robots are making our society more automated, starting from factories to our home and offices. As we expand the use of robots and increase the variations of the robot designs, the motion planning problem for the robots becomes a more significant challenge in robotics field. Over the last decades, engineers have been required to program robotic motion manually for a specific robot and a specific task, and the effort must be repeated every time the environment changes. This time-consuming process will put a limit on potential growth of the robotic products and services unless the researchers develop technique that can automatically generate all required robotic motions for the variety of tasks.

This study deals with the problem of motion planning for multiple robot arms sharing a common workspace. Multiple robots in such a setup can be used to facilitate various operations. For example, the handling operations of flexible material can be more efficiently carried out using more than one manipulators. While multiple cooperating robots can have remarkable benefits, it is more difficult to address the motion planning problem for these robots because the collision immediately happens if one robot does not take into account the motion of the other one. The simplest way to deal with this problem is defining the safety area for one robot and the other robot never enters that area. Another common approach is that one robot constantly tracks the position of the other robot and generates its motions so that no collision occurs. The motions generated by these approaches, however, limits the performance of the robots because of lack of synergy.

In this paper, we formulate this problem as an optimization problem and solve it using a method based on Genetic Algorithms [1],[2], which minimize costs associated with a traveling length and the number of collisions. Our goal is to explore the methods of GA capable to generate well synthesized multiple robots motion more efficiently than conventional GAs. This paper is organized as follows: In Section 2, we review some related works. In Section 3, the problem is formulated and the details of proposed GA are discussed.

The results of our experiments are reported in Section 4. The last section concludes the paper.

2 Related works

2.1 Motion Planning

Motion planning is responsible to finding a kinematic sequence from a start configuration to a goal configuration of the robot without touching any obstacles. To reduce the complexity of the problem, we generally consider this problem in a configuration space (C-space)[3] instead of the physical space where robots and obstacles exist. This turns the motion planning problem into the problem that finds a connected sequence of points $[\boldsymbol{q_{t_0}}, \boldsymbol{q_{t_1}}, \cdots, \boldsymbol{q_{t_f}}]$, where $\boldsymbol{q_{t_0}} \in R^n$ and $\boldsymbol{q_{t_f}} \in R^n$ are the initial and goal configuration of the robot. Each via-point in the path is denoted as a vector $\boldsymbol{q_i} = [q_0, q_1, \cdots, q_n] \in R^n$, where n is the number of degree of freedom (DOF) of the robot. For example, for a 2R robot, the configuration at time t_i is denoted as $\boldsymbol{q_{t_i}} = [q_{i0}, q_{i1}]$, where q_{i0} is the angle of the first joint and q_{i1} is the angle of the second joint (Fig.1).

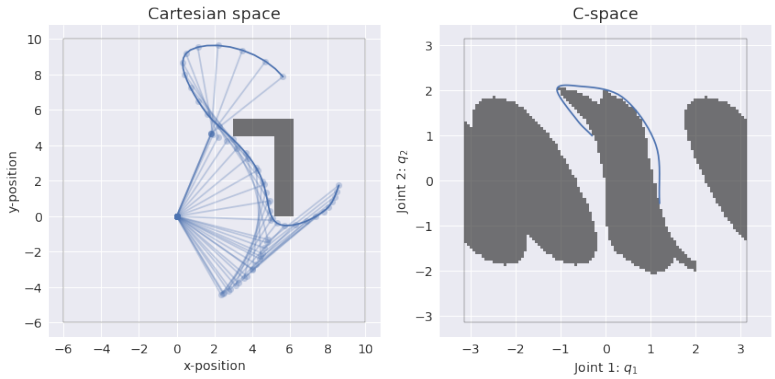

Figure 1: In the left figure, a blue line and a gray object represent a robot configuration in each time step and an obstacle in a Cartesian space respectively. In the right figure, the robot path and the object are mapped in a C-space.

Many important algorithms for motion planning in a C-space fall in the category of graph search algorithms. A graph is a set of nodes connected by edges, where each edge has an associated cost such as a traveling distance. There are some general approaches to create a graph such as visibility graph[4], voronoi diagrams[5], cell decomposition[6], or simply a grid-based map. Once discretizing the C-space and turning it into a graph, graph search algorithms such as Dijkstra[7] or A* algorithms[8] ensure to find the optimal path in a discretized space. Thus, these algorithms give us the reasonable solution if the C-space is sufficiently finely discretized.

The main drawback of the graph search algorithms in a discretized C-space is, however, that the computational cost exponentially increases as the dimensionality of the configuration space increases. For example, the motion planning for a 7-DOF robot arm requires 10^{12} points if representing its C-space with grids for a resolution of 100 point per dimension, which is no longer practical.

Sampling-based planning such as Probabilistic Roadmap (PRM)[9] or Rapidly-exploring Random Tree (RRT)[10] is one of the most practical approaches for exploring the high dimensional C-space. These algorithm does not search a C-space orderly but samples random points, resulting in a relatively small number of points to be explored. These algorithms produces feasible path in a practical execution time. However, they are not guaranteed to find the optimal path.

2.2 Evolutinary Planner

Genetic Algorithm[1],[2] is a method for solving an optimization problem inspired by the process of natural selection. In the context discussed in the previous section, GAs have received a lot of attention from researchers for the following reasons:

- The motion planning problem is known as NP-complete[11], meaning that the complexity of the problem grows exponentially as the number of DOF increases and it is no longer realistic to find the exact solution with deterministic algorithm.

- Genetic Algorithms are classified as a global heuristics search. They are designed to find correct solutions that are not necessarily the best solutions as effectively as possible.

- Genetic Algorithms are well adapted to search for solutions in high dimensionality search space.

- Genetic algorithms can easily be implemented on parallel machines and achieve speed up linearly with the number of processors.

The idea of the GA for the robot motion planning is introduced by Davidor[12] in 1991. Ahuactzin[13] also demonstrates the path planner for both a 2-DOF robot arm and an holonomic mobile robot. Since then, many researchers have contributed to the development of both genetic encoders and genetic operators that are designed to be suitable for the motion planning problems [14],[15],[16],[17].

GA for motion planning of the dual-arm robots are proposed by Rana and Zalzala[18] in 1997. In their paper, paths of two robots are encoded as strings of via-points and combined into a single chromosome. Venegas and Marcial-Romero[19] used the decoupled approach to obtain a plan for the multiple manipulators. The strategy is to combine the exploration of the free collision space and the exploration of the target position from previously explored area. Both solutions are obtained by GA. Curkovic and Jervic[20] evolves the paths for two robots based on Cooperative Coevolution in 2010. The main difference of their work from the others is that they encode the paths of robots as two separate chromosomes and evolve them among separate populations. The communication between two populations take place only

when checking for collisions. Each individual is evaluated with a set of the top 10% collaborators from the other population.

3 Methods

3.1 Problem formulation

Two 2R robots in a 2D space are considered first. The robot arms in the cartesian space are shown in Fig.2. The first robot appears as an obstacle for the second robot and vice versa, but no obstacle exists except the robots.

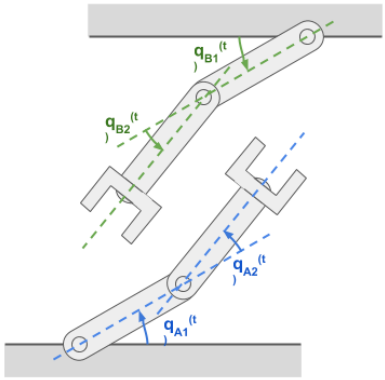

Figure 2: 2D cartesian space for two 2R robots with two links

A configuration of the robot A at the time t is denoted by $\boldsymbol{q_A^{(t)}} = [q_{A0}^{(t)}, q_{A1}^{(t)}]$ and robot B at the time t is denoted by $\boldsymbol{q_B^{(t)}} = [q_{B0}^{(t)}, q_{B1}^{(t)}]$. The paths for the robot A and the robot B are represented by connecting a set of points in the C-space:

$$\left\{\boldsymbol{q_A^{(t_0)}}, \boldsymbol{q_A^{(t_1)}}, \cdots, \boldsymbol{q_A^{(t_f)}}\right\} \tag{1}$$

$$\left\{ \boldsymbol{q_B^{(t_0)}}, \boldsymbol{q_B^{(t_1)}}, \cdots, \boldsymbol{q_B^{(t_f)}} \right\} \tag{2}$$

,where start point $q_A^{(\vec{t_0})}$, $q_B^{(\vec{t_0})}$ and end point $q_A^{(\vec{t_f})}$, $q_B^{(\vec{t_f})}$ are given. The purpose of the motion planning is to find the set of via-points that moves the robot from start to goal as fast as possible. Geometrically, this seems to be achieved with the shortest path obtained by connecting each via-point with a line segment. However, such a path is never feasible in practice because real robots with finite acceleration cannot go through the discontinuous point of the path. The real robots require at least continuous velocity profiles. For this reason, we formulate the parametric curve $\boldsymbol{r}(s)$ with $s \in [0, 1]$, connecting two points $\boldsymbol{q^{(t_i)}}$ and $\boldsymbol{q^{(t_{i+1})}}$ as follows.

$$\boldsymbol{r} = \left[s^3, s^2, s^1, 1 \right] \begin{bmatrix} 2 & 1 & -1 & 1 \\ -3 & -2 & 3 & -1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{q^{(t_i)}} \\ \boldsymbol{v^{(t_i)}} \\ \boldsymbol{q^{(t_{i+1})}} \\ \boldsymbol{v^{(t_{i+1})}} \end{bmatrix} \tag{3}$$

,where $\boldsymbol{v^{(t_i)}} = \frac{1}{2} \left(\boldsymbol{q^{(t_{i+1})}} - \boldsymbol{q^{(t_{i-1})}} \right)$ is a velocity at the time t_i. This curve is called Ferguson spline curve and ensures the continuous velocity profiles for the robotic motion.

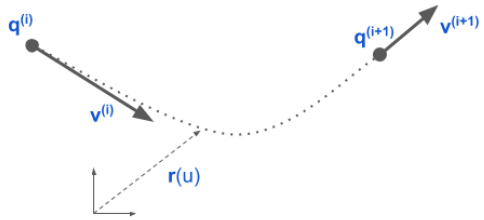

Figure 3: Complete robot path formulated by Ferguson spline curves.

3.2 Co-evolutionary Genetic Algorithm

3.2.1 Representation for robot paths

In Genetic Algorithm, the parameters to be optimized have to be encoded to a string representation. The encoding is a key process for the GA because the optimization will be successful as long as the encoded string contains compact sub-solutions of the problem (Building blocks). The solution of the motion planning problem is the entire path, which can be constructed by the combination of the sub-path segments. Thus, in that sense, it is natural that the solutions are encoded directly as strings of a sequence of the via-points as follows:

$$\left[\left\{q_{A0}^{(0)}, q_{A1}^{(0)}\right\}, \left\{q_{A0}^{(1)}, q_{A1}^{(1)}\right\}, , \cdots, \left\{q_{A0}^{(m-1)}, q_{A1}^{(m-1)}\right\}\right] \tag{4}$$

$$\left[\left\{q_{B0}^{(0)}, q_{B1}^{(0)}\right\}, \left\{q_{B0}^{(1)}, q_{B1}^{(1)}\right\}, , \cdots, \left\{q_{B0}^{(m-1)}, q_{B1}^{(m-1)}\right\}\right] \tag{5}$$

In this study, the number of the via-points m is set to 7.

3.2.2 Initial population

At the beginning of the evolutionary process, an initial population of chromosomes is generated from scratch. This is done by selecting a random via point $\boldsymbol{q_m}$ from the C-space and connecting a given start point $\boldsymbol{q_s}$ and the via point $\boldsymbol{q_m}$, as well as the via point $\boldsymbol{q_m}$ and a given goal point $\boldsymbol{q_g}$ using the Ferguson spline curve described as Eq.(3). Then, the obtained continuous path is divided into $m-1$ sub-path segments.

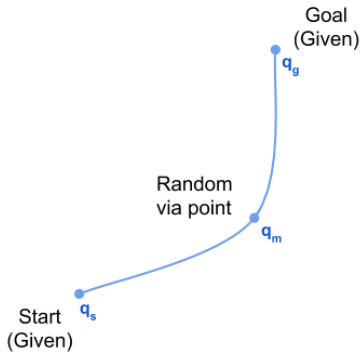

Figure 4: Random initial path.

3.2.3 Genetic operators and selection mechanisms

The progress of the evolutionary process is driven by two key process: Mutation and crossover. The purpose of mutation operators is to find a better solution around the current solution and improve the solution incrementally. Crossover operators are responsible to discovering new areas of the solution space by recombining the current solutions. Recombination of solutions are successful only when the building blocks are unlikely broken by the crossover operation. In this study, the following operators are used .

Mutation

A single point $\boldsymbol{q}$ is randomly picked from the chromosome with the probability $P_m = 0.5$, and the point is slightly shifted by adding a vector $\boldsymbol{\Delta q}$ chosen from the Gaussian distributions with mean $\mu = 0$ and variance $\sigma = 0.1$.

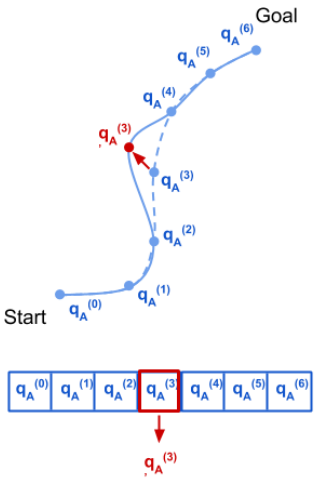

Figure 5: Mutation

Crossover

One point crossover is used. This operator is known as the conventional and simplest crossover operator for GAs, where a cut point is randomly selected from two parents chromosomes and then the tail part from the cut point are swapped between two parents. For the chromosomes encoded for the motion planning problem, however, one point crossover can degrade the quality of the path by inserting the redundant sub-path segment(Fig.6). To avoid this, a pair of the cut points is picked from the nearest points of the parent paths instead of the random points.

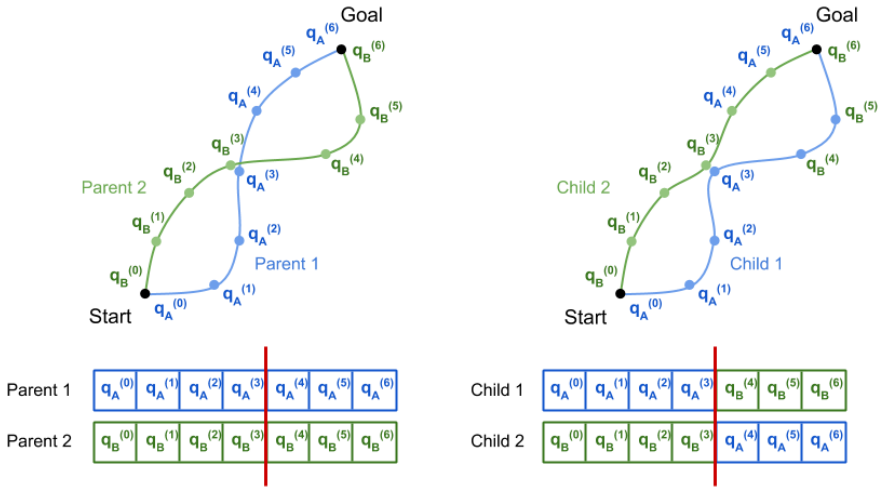

Figure 6: Crossover

Selection is a method for selecting individuals who will survive in the next generation and contribute to the evolution. The following selection mechanism is used in this study.

Tournament Selection

This algorithm randomly picks up N individuals from the population and creates a "tournament". The individual with the best fitness in the tournament is selected as a offspring. This tournament process is repeated until the number of offspring reaches a certain amount. The tournament size $N = 3$ is used for the GA in this study.

Elitist Selection

The top N individuals are selected from the population and retained in the next generation without any operation being applied. N=5 is used in this study. This selection mechanism prevents the best individuals from the destruction and guarantees that the best fitness does not decrease as the generation goes on.

3.2.4 Diversity maintenance

To make the population more diverse, Age-Layered Population Structure (ALPS)[21] is introduced. The key properties of ALPS are follows:

- Population are divided into the 6 age-layers. The age is measured based on how many generations the individual survives since it was born. The age limits for each layer is [5, 10, 20, 40, 80,].
- Every 5 generation, new population are generated from scratch and stored in the youngest layer.
- The next generation in the layer is selected only from the current population in the same layer.
- The crossover happens with individuals in the same layer or younger age layers.

3.2.5 Subjective fitness for co-evolution

The definition of the optimal path in this study is that, first, the number of the collision between two robots F_c are as small as possible, and second, the traveling lengths of the end effector of robot A: d_A, and B: d_B are as short as possible in a Cartesian space. In that sense, the fitness function F is defined as follows:

$$F = f(d_A,\ d_B) + wF_c \tag{6}$$

, where w is a weight on the fitness associated with the number of collision and set to $w = 10.0$. Each component of the fitness function is described as follows:

$$F_c = \sum_{k=0}^{M-1} C_k \tag{7}$$

$$C_k = \begin{cases} 1 & \text{if Robot A and B collide in the } k\text{-th configuration.} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Also,

$$d_A = \sum_{k=0}^{M-1} \| \boldsymbol{p_k}(\boldsymbol{q_A}) - \boldsymbol{p_{k-1}}(\boldsymbol{q_A}) \| \tag{9}$$

$$d_B = \sum_{k=0}^{M-1} \| \boldsymbol{p_k}(\boldsymbol{q_B}) - \boldsymbol{p_{k-1}}(\boldsymbol{q_B}) \| \tag{10}$$

, where $\boldsymbol{p_k}(\boldsymbol{q_A})$ and $\boldsymbol{p_k}(\boldsymbol{q_B})$ are the positions of the end effector of the robot A and B in a Cartesian space at time k. We introduce two types of fitness functions $f(d_A,\ d_B)$: Cooperative fitness and competitive fitness.

Cooperative fitness

The robot A and B cooperate each other, i.e., the fitness value of the robot A decreases not only when the traveling length d_A decreases but also the traveling length d_B decreases.

$$f(d_A,\ d_B) = d_A + d_B \tag{11}$$

Competitive fitness

The robot A and B compete each other, i.e., the fitness value of the robot A decreases only when the traveling length d_A decreases, but the fitness value is capped at the level of the traveling length d_B.

$$f(d_A,\ d_B) = \max(d_A,\ d_B) \tag{12}$$

In this study, each fitness function is applied and compared its performance each other.

4 Simulation results and discussion

The algorithm was implemented with parameters given in Table 1.

Table 1: Parameter for Genetic Algorithm.

Number of Generations	500
Population size in a layer	50
Number of points in a path	7
Number of elites	1
Tournament size	3
Number of layers	5
Max age of each layer	$[5, 10, 20, 40, 80]$
Weight on a collision fitness	10.0
Mutant probability	0.5

The paths were planned with the start and end configurations given in Fig 7. Figure 8 and Figure 9 shows results of the robots' motion by cooperative and competitive GA. The robots seem to overlap in each picture, but they are occurring at different time instants. Therefore collision is not occurring between them.

Figure 7: Start configuration of the robot.

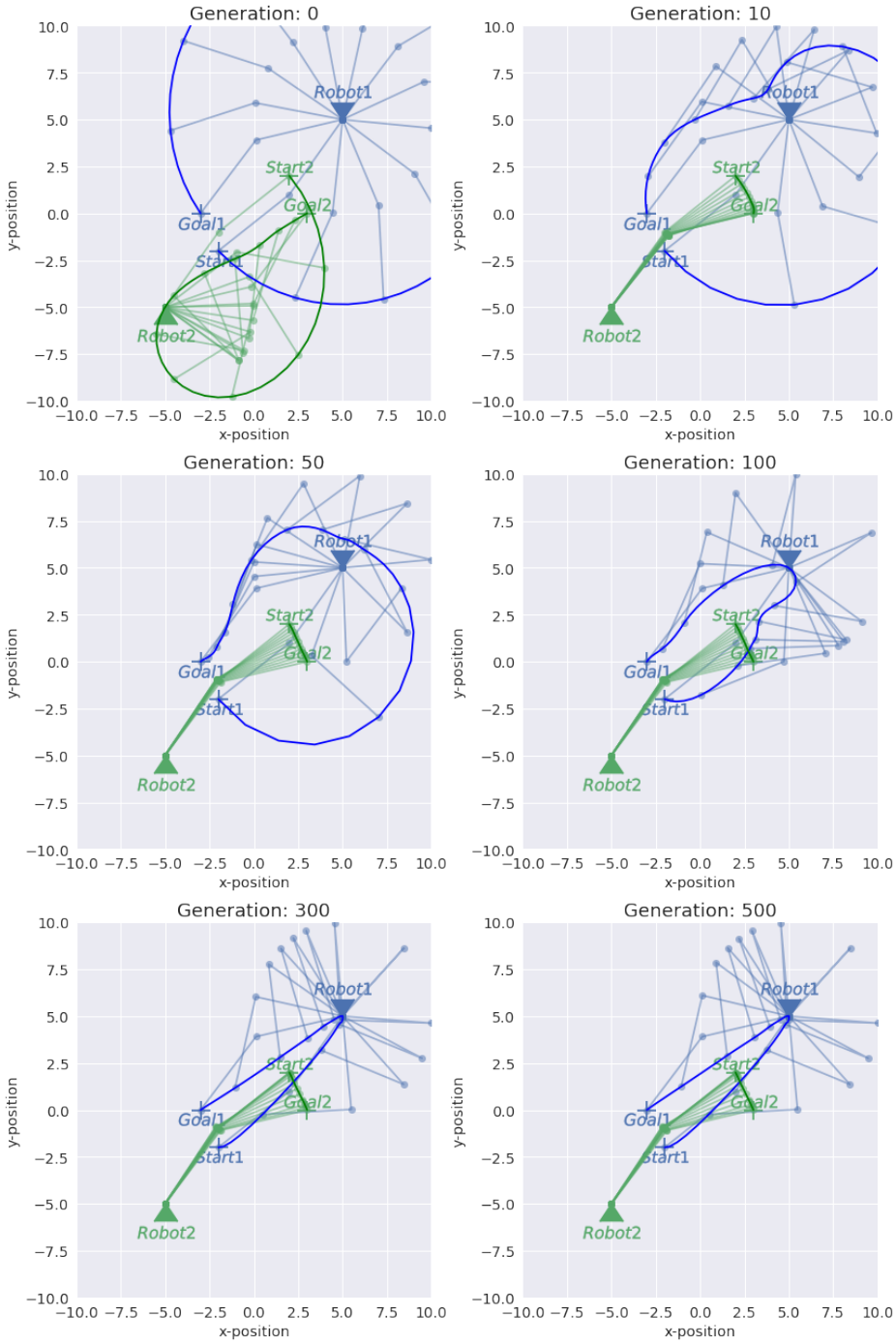

Figure 8: The snapshot of the robot motion evolution with cooperative fitness.

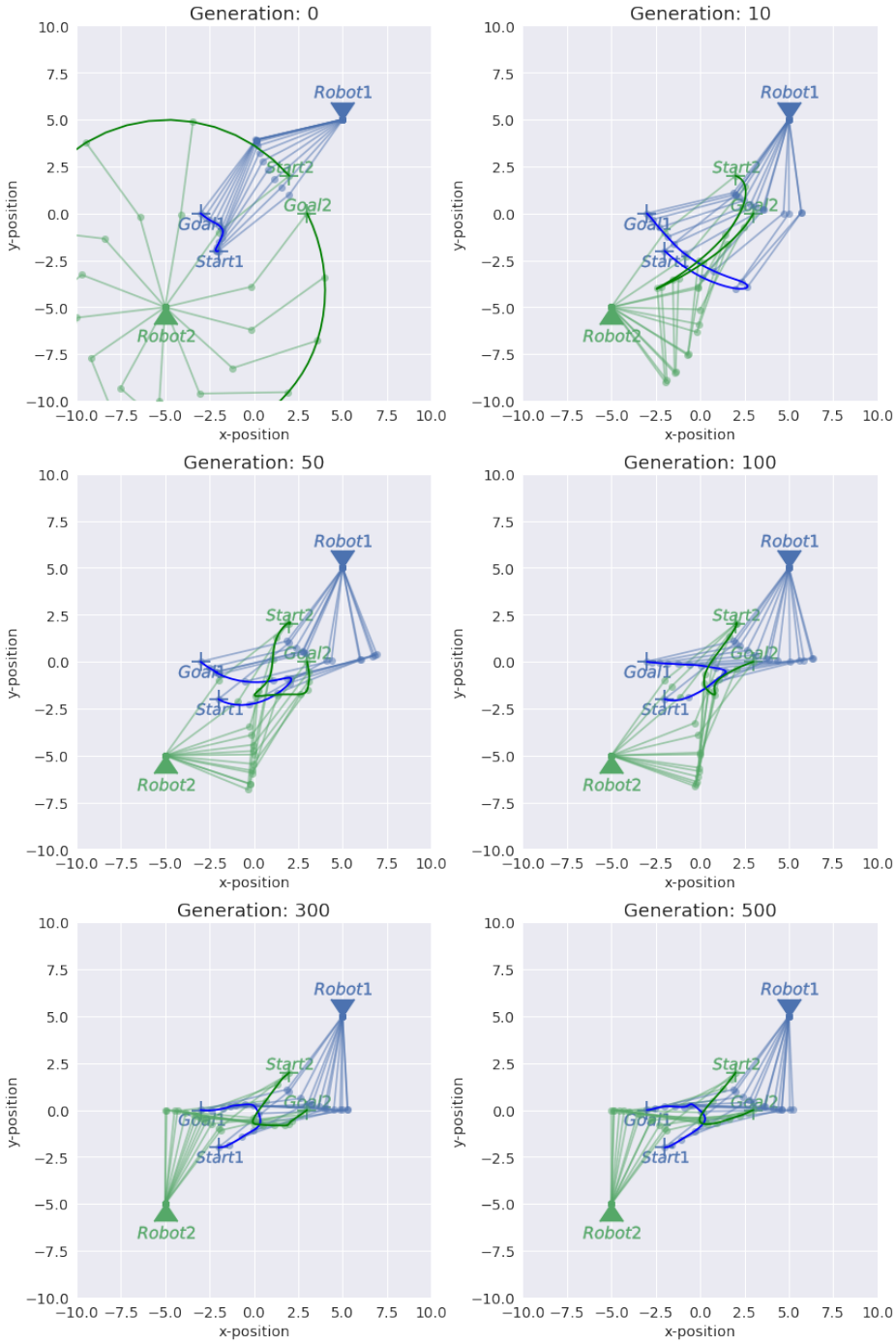

Figure 9: The snapshot of the robot motion evolution with competitive fitness

The shortest path found by each planner is shown in Table 2. Figure 10 shows The learning curve of the GA with cooperative fitness and the competitive fitness. Each is performed 5 times. The solid lines indicate the average traveling distances of the end effectors, and error bars indicate the standard error of these values.

Table 2: The best paths obtained by cooperative and competitive GA.

	Robot A	Robot B
Cooperative Co-evolution	2.2366	19.5419
Competitive Co-evolution	6.5236	6.5240

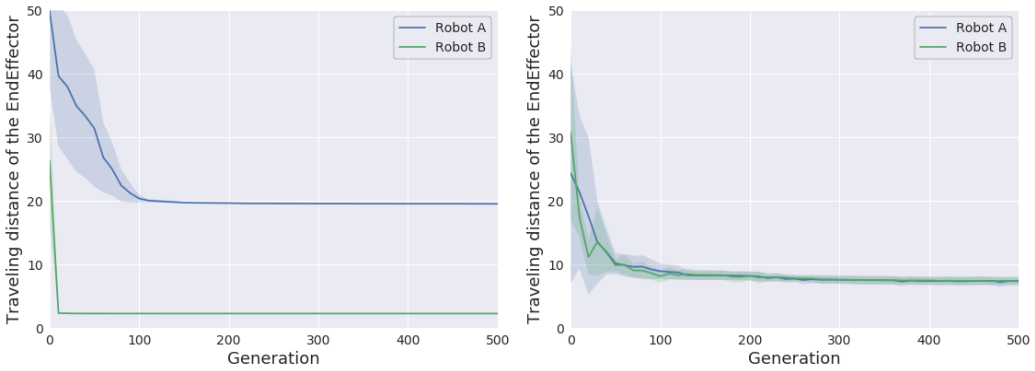

Figure 10: The learning curves of the evolution with cooperative fitness and competitive fitness

In Cooperative Co-evolutionary GA, the Robot A found the path that directly leads it to the goal with the nearly shortest time. Instead of Robot A's straight forward path, the Robot B had to make a detour. On the other hand, in Competitive Co-evolutionary GA, the path lengths of both Robot A and Robot B are approximately equal.

This difference of the multi-arm motions is obviously caused by the difference between the fitness functions of both methods. The fitness function of Cooperative Co-evolutionary GA allows one robot to obtain a good score

as long as the other robot finds a short path, even if the robot itself does not find a short path. Thus, the algorithm tries to find the shortest possible path for one of the robot in exchange for the hard work of the other robot. The fitness function of Competitive Co-evolutionary GA, by contrast, makes the robots compete, which means that if the path of Robot A is shorter than that of Robot B, the Robot A stops evolving until the path of the Robot B reaches the same length as that of Robot A. As a result, both of the path length becomes nearly equal.

The GA with standard population structure was performed as well as GA with ALPS, and the performances are compared. The fitness curves are shown in the Figure 11. Additionally, Figure 12 shows that the diversity of the population in each generation.

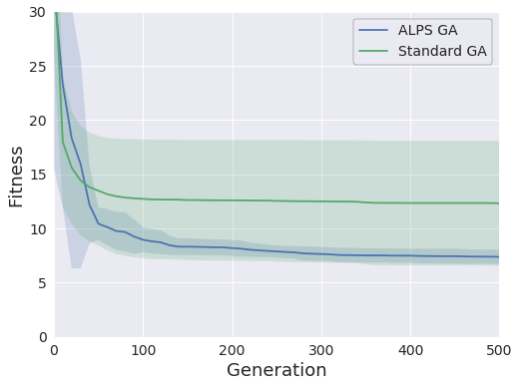

Figure 11: The learning curves of the evolution by standard GA and ALPS GA.

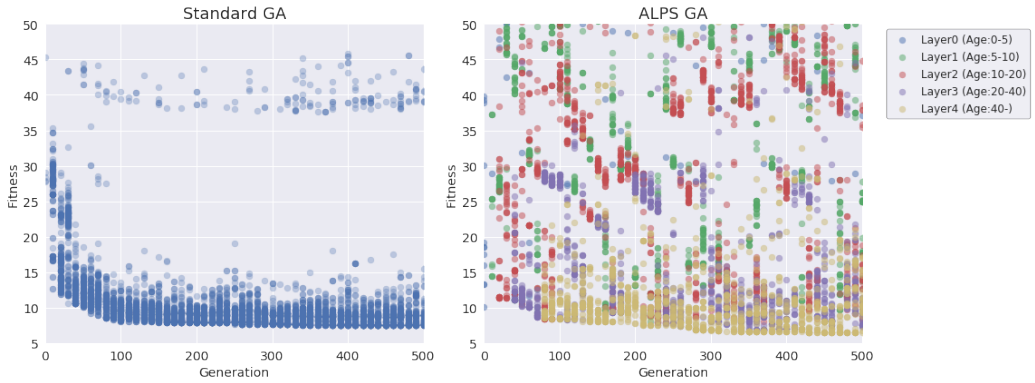

Figure 12: Dot plot of the population with Standard GA and with ALPS GA. Each dot indicates each individual in the population.

In the Standard GA, the population lose its diversity as generation goes on, and fitness of most individuals converges into less than 15.0. By contrast, the ALPS GA successfully generates diverse population, especially in the range of fitness¿15.0. As a result, it achieves better performance compared to the standard GA.

5 Conclusion

In this work, we presented a new strategy for the motion planning of multi robot arms based on Co-evolutionary Genetic Algorithm. In the proposed co-evolutionary algorithm, the paths of two robots are represented as two separate populations and the fitness function for each population are dependent on the other population. We compared the performances of two types of Co-evolutionary fitnesses: competitive fitness and cooperative fitness. In the cooperative process, the sum of the both robots' traveling distances are minimized. This process generates the straight forward path connecting start and goal points directly for one robot, but the other robot has to make a detour. On the other hand, the competitive process keeps the both traveling distances balanced. As a result, in the case that the tasks for the two

robots should be evenly shared, competitive process generates more natural motions.

This study demonstrated the proposed algorithms in the case with two 2R robots. However, considering the GA's potential to deal with high-dimensional problems, it can be adopted to the higher DOF robots and more than two robot arms. These studies need to be done in the future.

References

[1] J.H. Holland: Adaptation in Natural and Artificial Systems, Ann Arbor: University of Michigan Pres, 1975.

[2] David E. Goldberg: Genetic algorithms in search, optimization and machine learning, The University of Alabama, Addison-Wesley publishing company, inc. 1989.

[3] Lozano-Perez, T.: Spatial planning: A configuration space approach. IEEE Trans- actions on Computers 32(2), 108120 (1983)

[4] Lozano-Prez, T.; Wesley, Michael A. (1979), "An algorithm for planning collision-free paths among polyhedral obstacles", Communications of the ACM, 22 (10): 560570

[5] Aurenhammer, Franz (1991). "Voronoi Diagrams A Survey of a Fundamental Geometric Data Structure". ACM Computing Surveys. 23 (3): 345405.

[6] LaValle, Steven M (2006). "Planning Algorithms". Cambridge University Press.

[7] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGrawHill. pp. 595601.

[8] Zeng, W.; Church, R. L. (2009). "Finding shortest paths on real road networks: the case for A*". International Journal of Geographical Information Science. 23 (4): 531543.

[9] Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; Overmars, M. H. (1996), "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, 12 (4): 566580

[10] LaValle, Steven M. (October 1998). "Rapidly-exploring random trees: A new tool for path planning". Technical Report. Computer Science Department, Iowa State University (TR 98-11).

[11] Canny, J.: The Complexity of Robot Motion Planning. MIT Press, Boston (1988)

[12] Davidor, Y.: Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization. World Scientific Publishing Co. Pte Ltd(1991).

[13] Ahuactzin J.M., Talbi EG., Bessire P., Mazer E. (1993) Using genetic algorithms for robot motion planning. In: Laugier C. (eds) Geometric Reasoning for Perception and Action. GRPA 1991. Lecture Notes in Computer Science, vol 708. Springer, Berlin, Heidelberg

[14] Tian, L., Collins, C. Journal of Intelligent and Robotic Systems (2003) 38: 297.

[15] Kazem, B.I., Mahdi, A., Oudah, A.T. (2008). Motion Planning for a Robot Arm by Using Genetic Algorithm.

[16] Marcos, M.D., Machado, J.A., Azevedo-Perdicolis, T. (2009). Trajectory planning of redundant manipulators using genetic algorithms.

[17] Marcos, M.D., Machado, J.A., Azevedo-Perdicolis, T. (2012). A multi-objective approach for the motion planning of redundant manipulators. Appl. Soft Comput., 12, 589-599.

[18] A. S. Rana, A. M. S. Zalzala, "An evolutionary algorithm for collision free motion planning of multi-arm robots," First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, 1995, pp. 123-130.

[19] Venegas, H.A., Marcial-Romero, J.R. (2009). An evolutionary algorithm for collision free motion planning of multi-arm robots. EvoWorkshops.

[20] Curkovic, Petar. (2010). Cooperative coevolution applied to dual-arm robot motion planning. 132-137.

[21] Hornby, G.S.: The Age-Layered Population Structure (ALPS) Evolutionary Algorithm,