Resilient Movement Planning for Continuum Robots

Oxana Shamilyan[a,*], Ievgen Kabin[a], Zoya Dyka[a,b] and Peter Langendoerfer[a,b]

[a]IHP – Leibniz-Institut für innovative Mikroelektronik
[b]BTU Cottbus-Senftenberg
ORCID (Oxana Shamilyan): https://orcid.org/0000-0002-0961-9059

Abstract. This paper presents an experimental study of resilient path planning for continuum robots taking into account the challenge of multi-objective optimisation. To do this, we used two well-known algorithms for path planning, namely Genetic algorithm and A* algorithm, and modified them by adding the Analytical Hierarchy Process algorithm for paths' evaluation. In our experiment the Analytical Hierarchy Process considers four different criteria, i.e. distance, motors damage, mechanical damage of the robot's arm and accuracy, each considered to contribute to the resilience of a continuum robot. The use of different criteria is necessary to increasing the time to maintenance operations of the robot. The experiment provides us two insights. First, the improved algorithms with multi criteria path-planning show better performance than their classical versions. Second, when comparing improved GA and A* algorithm, A* shows superior performance.

1 Introduction

Continuum robots are a new type of robots providing a high degree of flexibility when it comes to reaching tricky positions, as they are not limited in their movements by joints. Instead, they have infinite degrees of freedom. On the one hand, this makes them ideal candidates for application areas such as inspection tasks, handling of unstructured objects, and minimally invasive surgery [18, 19]. On the other hand, the high flexibility of these robots presents a challenge in terms of controlling their movements. [19] provides a comprehensive survey of the most popular continuum robots types, motion control models and examples of research groups that are focused particularly on the study of motion control problems of the continuum robots. One of the most popular motion control model is the Cosserat theory [4]. Many researchers [12, 13, 14, 15, 16] utilize it for modelling motion control, which considers different kinds of deformations of physical objects, but requires a long time and significant computation costs. To reduce computation costs, [14] used a discrete Cosserat approach, means they processed the continuum robot prototype as a finite set of constant pieces, making the solution discrete and easy to process. [3] provides a comparison of different motion models and demonstrates that the Cosserat theory can be used for the tasks where the high accuracy is needed, but is not suitable for real-time applications.

Path planning is essential for the continuum robots, especially when deployed as autonomous systems in unknown environments. Under these conditions the path-planning should take into account not only the start and the end point, but also the state of the environment and the state of the system itself. Such awareness helps the system to make a decision and generate the optimal path. Here we mean by optimal not essentially the shortest path but the one with the least negative impact on the continuum robot. In order to enable a robot to properly assess more conditions than just the path length, we modified path planning algorithms and added an Analytical Hierarchy Process (AHP) decision-making algorithm. The latter generates the weights to evaluate the generated path, to find the optimal solution. In our experiments we used Genetic algorithm and A* algorithm.

The rest of this paper is structured as follows. In section II we provide information about the methods we used, namely AHP, Genetic algorithm and A* algorithm. Section III describes our experimental setup. In section IV we provide information about our experiments and results. This paper ends with a short conclusion.

2 Methods

This section describes the methods used in our experiment, which aims at comparing two path planning algorithms: Genetic algorithm and A* algorithm, when more criteria than the path length have to be considered. We made changes to the classical representations of both algorithms to suit our needs, primarily by adding multiple criteria for assessing the generated path. The criteria weights were generated using the AHP decision-making method.

2.1 Analytical Hierarchy Process

AHP is the most common multi-criteria decision-making algorithm developed by T.L. Saaty in 1980 [17]. It makes decisions based on pairwise comparison of alternatives or criteria depending on their relative importance levels (Table 1). AHP is applied in various fields such as business, marketing, risk analysis for safety enhancement, environmental location selection, performance evaluation, and path planning.

A full description of the algorithm can be found in [8]. Below we briefly describe each step:

1. Problem analysis;
2. Generation of the hierarchy structure;
3. Definition of the relative importance of the objectives (Table 1);
4. Weight importance calculation;
5. Evaluation of the consistency indexes for objectives;
6. Evaluation of alternatives with reference to each objective.

For the analysis, we established four criteria that are crucial for enhancing the resilience of a continuum robot. This includes reducing or avoiding mechanical damage and increasing the time to the

* Corresponding Author. Email: shamilyan@ihp-microelectronics.com

next maintenance operation. Table 2 provides information about all four criteria, including its name, description of what each criterion estimates, and description of resilience properties that are reasoning the choice of criteria.

Table 1. The pairwise comparison scale

Importance	Definition
1	Equal importance of elements
3	Moderate importance of one element over another
5	Strong importance of one element over another
7	Very strong importance of one element over another
9	Extreme importance of one element over another
2, 4, 6, 8	Intermediate values between two adjacent judgment

2.2 Genetic Algorithm

The genetic algorithm (GA) is a global search and optimisation technique. It was first introduced by John H. Holland in [8]. The algorithm is inspired by Darwin's theory and works on the principle of natural selection. GA can be applied to a wide range of disciplines, such as mathematics, medicine [5], and engineering [2].

The algorithm includes five main steps: initialisation and genetic operators – evaluation, selection, crossover, and mutation. GA continues to repeat genetic operators, creating new generations, until the terminal condition is met. For our experiment we chose population size 50 and generation number 3 as optimal values to conduct the experiment.[1] Further we provide details about each of the steps of the algorithm and specify which techniques we used for the experimental part.

2.2.1 Initialisation of Population

The population has a fixed size and consists of *chromosomes*, each of which consists of *genes*. The chromosome length is not necessarily fixed. Each chromosome represents a possible solution to the problem. During the initialisation step, genes are generated and combined into chromosomes until the whole population is ready.

In our implementation we used an improved random technique to initialize the population. A database of possible connections between nodes was created and used for the initialisation. A starting node is always defined. Each next node n_i is randomly chosen among the possible connections of the previous node n_{i-1}.

2.2.2 Evaluation

Each chromosome in the population is evaluated using the *fitness function* that indicates how well each chromosome fits the current problem. A correctly composed fitness function increases the algorithm's output quality. The fitness function can contain one assessment function or set of them.

For our experiment we use a *multi-fitness function* which is calculated as a sum of the fitness functions for each criterion.

$$F = \sum_i w_i f_i \tag{1}$$

where i refers to the criterion from Table 2, w_i refers to the criterion weight obtained by the AHP algorithm and f_i refers to fitness costs.

[1] We run a few experiments to determine reasonable parameters which we omit here for better readability. The number reported provided the best result.

Table 2. Defined criteria

#	Criteria name	Description	Resilience properties
1	distance	estimates length of the generated path	energy and time savings
2	motor damage	estimates damage of setup's motors	extended time to maintenance of the motors
3	mechanical damage	estimates damage will be caused to robot tendons after the robot follows the generated path	extended time to maintenance of robot wire cables
4	robot's tip accuracy	estimates how accurately the robot's tip is at the goal point	accurate work

2.2.3 Selection

During the selection step, the algorithm selects *parent chromosomes* for *crossover* and *mutation*. The selection algorithm picks chromosomes to act as parents for the *offspring population*. In our implementation of GA we use the Roulette Wheel Selection algorithm that is one of the most common and optimal solutions [11].

2.2.4 Crossover

The *crossover* operator manipulates chromosomes in the population to create a diversity for future generations. In our implementation we use a single-point crossover, where two parent chromosomes are used to produce two offspring chromosomes. A separator (single-point) is randomly chosen to divide each parent into two parts. The first parts of both parents' chromosomes remain unchanged, while the second parts are swapped between each other to create the offspring.

2.2.5 Mutation

At the *mutation* step, only a small part of the chromosome is changed. For our implementation we use a random mutation. A single gene of the parent chromosome is randomly changed to a randomly generated value. The parent chromosome to which the mutation operator is applied, also is randomly selected, so only a few chromosomes in the entire population are mutated. Whether or not a chromosome is mutated depends on the mutation rate value, which is set manually by the user before the algorithm begins to work.

2.3 Algorithm results

After 3 generations (algorithm repetitions), the algorithm provides a final population that contains 50 paths. The path with the best fitness is considered as the resulting path.

2.4 A* Algorithm

The A* algorithm is a well-known path planning algorithm. It was proposed by Peter Hart, Nils Nilsson, and Bertram Raphael [7]. It is widely used in various domains, i.e. game development, logistics, and path planning for mobile robots [1, 6, 9, 10, 21].

The core idea of the classical algorithm is to find the minimum distance between the starting and the goal nodes. A* achieves this by using heuristic search. It calculates the cost of moving from the current node n to its neighbour nodes, and creates a queue based

on the cost results. The node with the lowest cost is given a higher position in the queue. The cost function is presented below:

$$f(n) = g(n) + h(n) \tag{2}$$

where g(n) denotes the cost from the starting node to the current node n and h(n) denotes the cost from the current node n to the goal node.

The classical A* algorithm involves the following steps:

1. Mark the starting node *s* as "open" and calculate its cost *f(s)*;
2. Select the node *n* from the open list that has the smallest value of *f(n)*;
3. If node *n* is the goal node, terminate the algorithm;
4. Otherwise, mark node *n* as "closed" and obtain a list of the neighbour nodes $n_0 \ldots n_i$;
5. Calculate the costs $f(n_i)$ for each neighbour node;
6. Mark all neighbour nodes as "open" except for those already in the closed list;
7. Remark as "open" any neighbour nodes that are marked as "closed" and have a smaller value $f(n_i)$ than *f(n)*;
8. Return to the step 2.

In this context, "open list" refers to a list of nodes that are yet to be evaluated, while "closed list" refers to a list of nodes that have already been evaluated. The cost function *f(n)* calculates the distance between nodes using either Euclidean or Manhattan distance [20]. Based on the result of the cost function A* provides a path that is considered to be the best under current environmental conditions.

The classical approach takes into account only distance between nodes, while we are interested in multi-criteria assessment. So, we changed the assessment function *f(n)* in Equation (2) to the multi-fitness function in Equation (1) for our experiment.

Using the same evaluation function i.e. Equation (1) for both path planning algorithms ensures a fair comparison of the two algorithms applied.

3 Experimental Setup

For our experiments we use a self-built tendon-driven continuum robot prototype (see Figure 1). Our prototype consists of 15 disks, one central flexible backbone tendon, and 8 side tendons. The disks are rigidly fixed on the central backbone tendon at an equal distance from each other. The prototype has a fixed base and two mobile sections. Side tendons pass through each disk along both sections (4 side tendons for each section), so that each section can be controlled independently [18, 19]. The fixed base does not allow the robot itself to move, but the mobile sections can move in all possible directions. The lower section determines the starting point of the upper section, which covers a larger area, than the lower section does.

The robot arm of our prototype is actuated by 4 stepper motors (2 motors for each section). Using of stepper motors helps us to reduce the flexibility of the robot and make it more controllable. Each motor needs 800 steps per revolution which makes robot's movements discrete. The stepper motors are equipped with absolute encoders that ensure the control of the number of steps taken by each motor. The stepper motors are connected to a microcontroller board that controls rotations of the stepper motors.

Path planning calculations are executed on a laptop equipped with an Intel Core i5-10310U CPU @ 1.70GHz Processor and 8 GB RAM. The microcontroller board and the laptop communicate via the UART protocol, while the encoders and board utilize the SPI protocol.

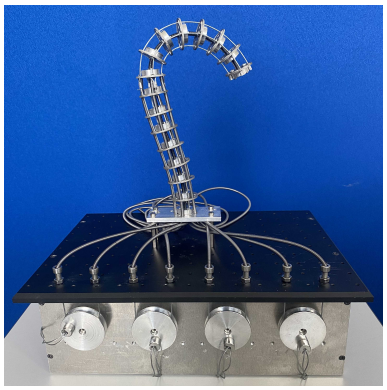

Figure 1. Self-built tendon-driven continuum robot prototype.

4 Experimental Evaluation

The main idea of our experiment is to evaluate the paths generated by GA and A* under different criteria. We analyse the output of both algorithms to discover which one manages to get better results for multi-criteria optimization problems. To do this, we evaluated the time performance of the algorithms and the quality of the generated paths. The quality value is represented by the value of the fitness function. To conduct the experiment, we implemented a virtual environment to simulate the algorithms' results.

4.1 Simulated Environment

The environment of the prototype is discrete, due to the mechanical abilities of the stepper motors that control the prototype's movements. Thus, the environment consists of a finite number of points that the robot can reach.

Both sections are identical, and therefore, the environment for each section is also the same. Figure 2 shows the environment of one of the sections. The simulated environment created in Python and has been drawn using the matplotlib library. The local environment of the section resembles half of a sphere, due to the robot's physical peculiarities. The blue and red curved lines represent the routes along the X and Y axes, respectively. The straight green line indicates the position of one section of the robot. The section can move along grey dashed lines and stop at the blue nodes, each with its own unique id number. Each pair of adjacent nodes along the axes is separated by a distance of 70 motor steps. The smaller the distance between the nodes, the less discrete the map and the higher the map density. With a 70-step interval, the robot's environment includes 61 nodes for one section and a total of 3721 nodes (61 section nodes * 61 possible section positions) for all the possible positions of the upper section. Both, A* and GA, use the simulated environment to generate the path. To provide a small example of how paths are generated, let us take point 4 as the start node and point 6 as the goal node. In this case the shortest path will be 4-5-6, but there are also many other options, for example 4-10-11-12-6, 4-1-2-6 etc.

4.2 Alternative Paths

We implemented an algorithm to search for alternative goal points. For a given goal point the algorithm searches in the robot global environment for the three nearest neighbours. The usage of this algorithm increases the choice of possible endpoints and, as a consequence, increases the diversity of possible paths generated by path planning algorithms. The increased set of available paths gives more options

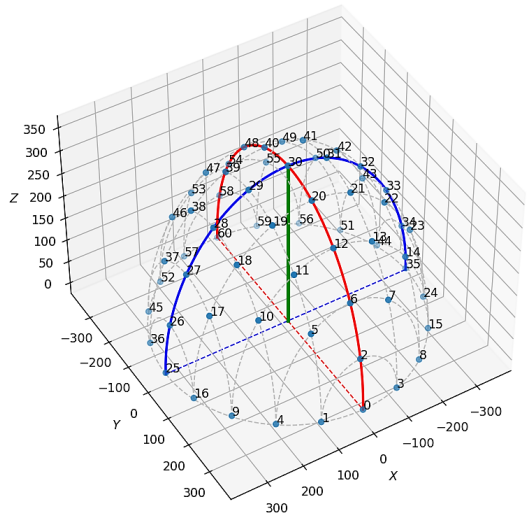

Figure 2. Robot's environment for one section.

to choose the best path for the robot, given the current state of the robot, of its environment and given criteria weights. The alternatives search algorithm provides the same output data for GA and A*. The distance between the initial and alternative points is calculated using the Euclidean distance. Figure 3 shows an example. Here the goal point is marked green and the nearest neighbours are marked orange. Red lines represent the position of the lower section of the robot, and green lines represent the position of the upper section. For a clearer image we omit to display the upper section´s environment.

4.3 Criteria Analysis

The AHP algorithm calculates weights for each criterion based on their relative importance and uses these weights to evaluate the generated paths. We conducted multiple experiments for each of the possible combinations of the criteria given in Table 2. Table 3 displays the criteria combinations that were used, with each line representing a single combination and each cell indicating the weight of the corresponding criterion. The green cells indicate the prioritised criteria, i.e. criteria which have more weight. The case in which no criteria were prioritised is not included, as the weight values are the same as in the case number 15. Prioritizing no or all criteria means they have

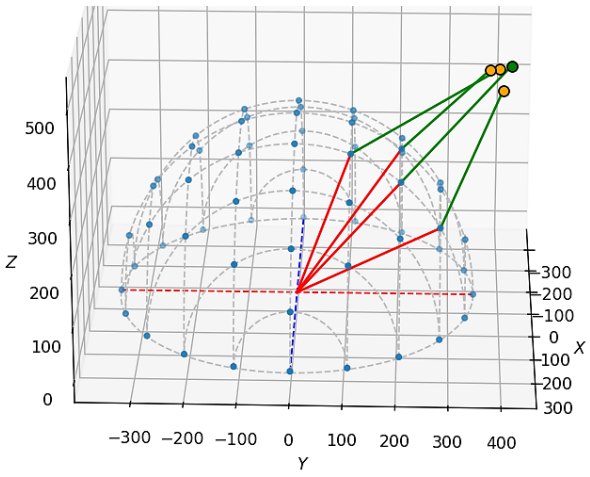

Figure 3. Alternative goal points.

Table 3. Weights for different combinations of criteria (prioritized criteria are highlighted in light green)

Group index	Distance	Motors damage	Mechanical damage	Accuracy
Single criterion				
1	0.75	0.083	0.083	0.083
2	0.083	0.75	0.083	0.083
3	0.083	0.083	0.75	0.083
4	0.083	0.083	0.083	0.75
Two criteria				
5	0.45	0.45	0.05	0.05
6	0.45	0.05	0.45	0.05
7	0.45	0.05	0.05	0.45
8	0.05	0.45	0.45	0.05
9	0.05	0.45	0.05	0.45
10	0.05	0.05	0.45	0.45
Three criteria				
11	0.321	0.321	0.321	0.036
12	0.321	0.321	0.036	0.321
13	0.321	0.036	0.321	0.321
14	0.036	0.321	0.321	0.321
Four criteria				
15	1	1	1	1

equal relative importance levels which means weight calculation has no effect. Criteria weights from the Table 3 are used in Equation (1) to calculate the total fitness value of the generated path.

There are four different fitness functions for each of the criteria. The fitness value "distance" is calculated as the sum of Euclidean distances between each node in the generated path. "Motors' damage" fitness function calculates how many steps each motor will make if the robot follows the generated path. The "Mechanical damage" fitness function calculates how many times each segment between two nodes was used. In order to evaluate the "accuracy" the fitness function calculates the Euclidean distances between the planned/intended goal point and actually reached point. Quantitative characteristics for calculating the fitness functions of motor and mechanical damage criteria are stored in an external sqlite3 database.

We also conducted an experiment to analyse how the generated paths vary depending on prioritised criteria. For this experiment we used just one path option without alternative goal points. The start and goal points for the lower section are 50 and 3, for the upper section 47 and 14. The start and goal points of both sections are randomly selected.

Figure 4 shows the results of the criteria analysis. The "group index" column refers to the corresponding criteria combination from the Table 3. "GA path" and "A* path" columns contain paths generated by the two path-planning algorithms. Paths are represented as 2D arrays. The first part of each array refers to the path, generated for the lower section of the robot, and the second part – for the upper section. Different paths are marked in different colours.

The results show that GA is more sensitive to criteria variations than A*. For the given start and goal points, GA generated different paths for each criteria group, whereas A* generated only two different variants of paths.

4.4 Performance Analysis

In this section we present results of the performance analysis of GA and A*. Both algorithms performed the same task under the same conditions and for the same input data. We divided the experiment into two parts: "single goal point" and "multiple goal point". For the "single goal point" experiment, both algorithms generate paths using only the given start and goal points, without using alternative

Group index	GA path	A* path
1	[[50, 49, 41, 31, 21, 20, 12, 13, 7, 3], [47, 39, 40, 30, 31, 32, 22, 14]]	[[50, 42, 32, 31, 21, 13, 7, 3], [47, 48, 40, 30, 31, 32, 22, 14]]
2	[[50, 42, 32, 31, 21, 13, 7, 3], [47, 39, 29, 19, 11, 12, 13, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
3	[[50, 49, 41, 31, 21, 13, 7, 3], [47, 39, 29, 30, 31, 21, 22, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
4	[[50, 42, 32, 31, 21, 20, 12, 6, 2, 3], [47, 39, 29, 30, 31, 21, 22, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
5	[[50, 42, 41, 31, 32, 22, 23, 15, 14, 8, 7, 3], [47, 39, 29, 19, 20, 21, 13, 14]]	[[50, 42, 32, 31, 21, 13, 7, 3], [47, 48, 40, 30, 31, 32, 22, 14]]
6	[[50, 42, 32, 22, 14, 8, 7, 3], [47, 39, 29, 19, 20, 21, 13, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
7	[[50, 42, 41, 40, 30, 31, 21, 22, 14, 8, 7, 3], [47, 48, 40, 41, 31, 32, 22, 14]]	[[50, 42, 32, 31, 21, 13, 7, 3], [47, 48, 40, 30, 31, 32, 22, 14]]
8	[[50, 42, 32, 22, 14, 8, 7, 3], [47, 39, 29, 30, 20, 12, 13, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
9	[[50, 49, 41, 31, 32, 22, 21, 13, 7, 3], [47, 48, 40, 41, 31, 32, 33, 23, 15, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
10	[[50, 42, 32, 31, 21, 13, 7, 3], [47, 46, 38, 28, 18, 19, 20, 21, 13, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
11	[[50, 42, 32, 31, 21, 20, 12, 13, 7, 3], [47, 39, 40, 41, 31, 21, 22, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
12	[[50, 42, 32, 22, 14, 8, 7, 3], [47, 46, 38, 39, 29, 30, 20, 21, 13, 14]]	[[50, 42, 32, 31, 21, 13, 7, 3], [47, 48, 40, 30, 31, 32, 22, 14]]
13	[[50, 51, 43, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 21, 22, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
14	[[50, 42, 41, 31, 30, 20, 12, 13, 7, 3], [47, 39, 40, 41, 31, 21, 22, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]
15	[[50, 49, 48, 40, 41, 31, 21, 13, 7, 3], [47, 39, 29, 19, 20, 21, 13, 14]]	[[50, 42, 41, 31, 21, 13, 7, 3], [47, 39, 40, 41, 31, 32, 22, 14]]

Figure 4. Criteria analysis results. Each line shows the generated path for a particular criteria combination group (Group index). Different paths are highlighted with different colors. Column "GA path" shows final results of the GA. Column "A* path" shows final results of the A* algorithm. Comparing both columns clearly shows that GA provides a higher variety of paths.

goal points. For the "multiple goal point" experiment, the alternative search algorithm is used. For both experiments all time measurements are shown as an average value of 100 algorithm runs. The axis "group index" refers to values from Table 3.

The results of the "single goal point" experiments are illustrated in Figure 5. Figure 5a shows the time measurements of both algorithms for each combination of criteria. A* algorithm takes twice less time than GA regardless of the criteria combination. Figure 5b shows which algorithm achieved more often a superior fitness function value than the other, displayed are the percentages for both algorithms. From the results in Figure 5b we see that even though GA creates more path variability for the different criteria groups than A*, it does not show efficiency in terms of time and fitness performance. For all 15 criteria groups A* generates better paths than GA in almost 60% of the cases (blue bars). In other cases GA and A* have same fitness values (black bars) means both algorithms generated the same paths.

The "multiple goal point" experiment involves the use of the alternative search algorithm. The output of this algorithm are three additional goal points representing the closest neighbours of the initial goal point. Using alternative search, the path planning algorithms generate paths to 4 different goal points (1 initial + 3 alternatives) and then choose the one with the lowest fitness value. On the one hand this approach leads to an increased processing time, but on the other hand it provides a large variety of choices. The results of the "multiple goal point" experiments are illustrated in Figure 6.

Figure 6a shows a comparison of the execution times for both algorithms. As expected, the execution times of both algorithms regardless of criteria is proportionally increased by 4 times in accordance with the increase in the number of goal points. Figure 6b shows the

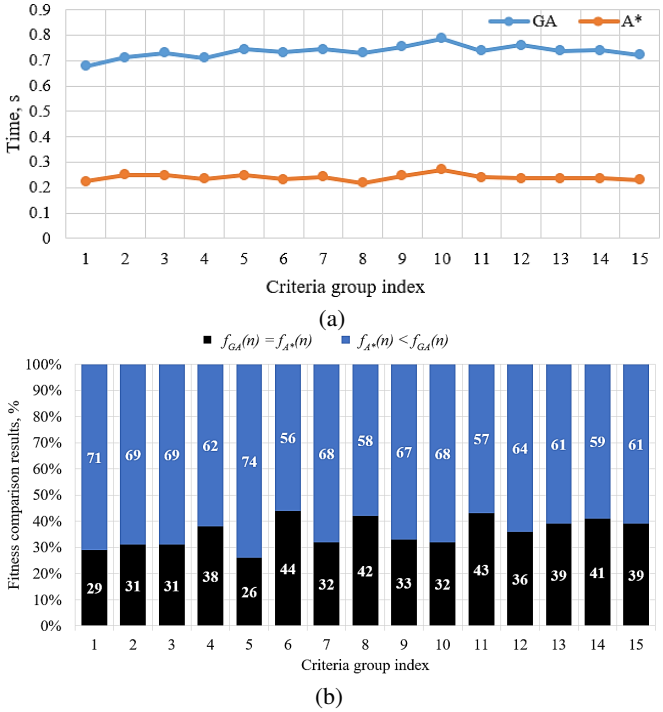

Figure 5. "Single goal point" Experiment: a) Processing Time b) Relation of solutions with better fitness: A* better than GA (blue), both yielding the same results (black).

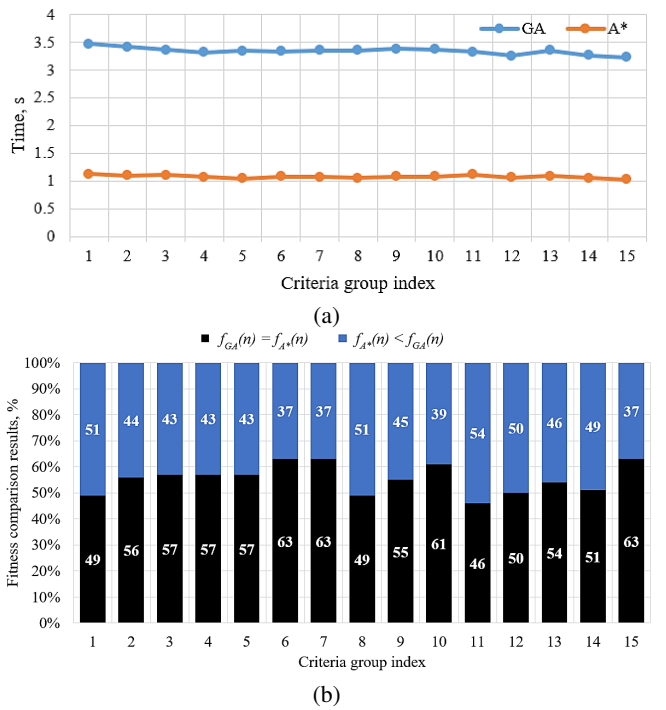

Figure 6. "Multiple goal point" experiment: a) Processing Time, b) Relation of solutions with better fitness: A* better than GA (blue), both yielding the same results (black).

comparison of the fitness values. As well as in Figure 5b, GA does not show better fitness results than A*, but the number of cases where GA and A* generated paths with same fitness value is increased almost 1.5 times (black bars). This can be explained by the fact that if there are more paths to be generated, the likelihood of generating identical paths increases.

The performance advantage of the A* over GA can be explained by the difference in algorithms structure. A* calculates the fitness function during path generation, and GA first generates the entire path and then calculates its fitness.

4.5 Comparison of improved and classical implementation of path-planning algorithms

To ensure that our improved versions of the algorithms provide enhanced performance, we provide a time and fitness comparison with the classical version of GA and A* algorithm (see Figure 7 and Figure 8). The main difference between classical and improved versions is that the classical algorithm generates the path based only on the calculation of the path distance. Even thought classical implementation of both algorithms work significantly faster (see Figure 7), it does not show performance superiority (see Figure 8) and does not reflect criteria priorities. Figure 8 shows how many percents of the paths generated by each algorithm achieved the best fitness value. To calculate these results we run each algorithm 100 times for each criteria group using the same goal and start points, in order to provide clear result data. For example, for the first criteria group A* generated the best path in 100% of all cases, while classical A* yielded the same results as A* in 80% of all cases, GA achieved this in 26% of cases, while classical GA did so in 23%. It is important to emphasise that neither the classical A* algorithm nor GA nor the classical GA demonstrated results, which were better than the improved A* algorithm. At best they generated the same path as A* did.

5 Conclusion

Recently, resilience became a very important feature, due to the growing demand for autonomous systems. Such systems should be self-aware and have the ability to adapt their behaviour [19]. In our research of continuum robots, path planning is considered as an essential task for the system. Therefore, the path should be generated according to the system and environment state.

This paper discusses an experiment with improved versions of GA and A* algorithm. We took classical versions of both algorithms and modified them by adding decision-making algorithm AHP to allow for more criteria than the path length to be considered when assessing the quality of the path calculated. In addition we evaluated the algorithms if they may use alternative goal points to better adapt to the criteria. Both modifications are aiming to increase the resilience properties of the system. The AHP algorithm generates weights to the criteria of choice, thus helps to generate the path that fits the most. The alternatives search gives the opportunity to generate multiple paths to create more variety of choice. The paper provides a comparison between classical and improved versions of GA and A* as well as a comparison of two improved algorithms. Results show that improved versions of algorithms show better performance than their classical versions, and in particular, the improved A* algorithm outperforms all others.

Our experiments clearly show that it is feasible to enable an autonomous robot to consider complex parameters when making decisions. Here we showcased this for example by taking mechanical

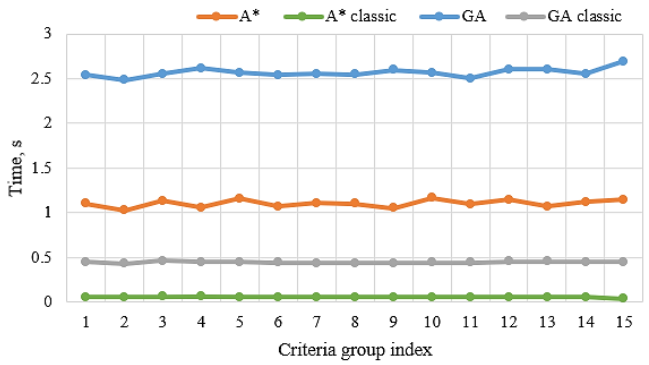

Figure 7. Comparison of processing time between A* (orange), classical A* (green), GA (blue) and classical GA (gray).

damage of its motor and tendons into account. This led to calculating paths completely different from the shortest ones, but with less negative impact on the robot.

In our future research we are planning to provide feedback about how the "system aging" can influence the decision-making to adapt path-planning results, i.e. with increased percentage of mechanical and motors damage, the robot should follow the path different from the shortest one, based on the current state of the robot in order to minimize the load and increase its remaining useful life. Furthermore, we intend to investigate the potential scalability of the developed algorithms and apply them to a larger robot with an increased number of sections.

References

[1] N. H. Barnouti, S. S. M. Al-Dabbagh, and M. A. Sahib Naser. Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm. *Journal of Computer and Communications*, 04(11):15–25, 2016. doi: 10.4236/jcc.2016.411002.

[2] M. T. Bhoskar, M. O. K. Kulkarni, M. N. K. Kulkarni, M. S. L. Patekar, G. Kakandikar, and V. Nandedkar. Genetic algorithm and its applications to mechanical engineering: A review. *Materials Today: Proceedings*, 2(4):2624–2630, 2015. doi: https://doi.org/10.1016/j.matpr.2015.07.219.

[3] M. T. Chikhaoui, S. Lilge, S. Kleinschmidt, and J. Burgner-Kahrs. Comparison of modeling approaches for a tendon actuated continuum robot with three extensible segments. *IEEE Robotics and Automation Letters*, 4(2):989–996, 2019. doi: 10.1109/LRA.2019.2893610.

[4] E. Cosserat and F. Cosserat. Théorie des corps déformables. *Nature*, 81 (2072):67–67, Jul 1909. ISSN 1476-4687. doi: 10.1038/081067a0.

[5] A. Ghaheri, S. Shoar, M. Naderan, and S. S. Hoseini. The Applications of Genetic Algorithms in Medicine. *Oman Medical Journal*, 30(6):406–416, Nov. 2015. doi: 10.5001/omj.2015.82.

[6] A. K. Guruji, H. Agarwal, and D. K. Parsediya. Time-efficient A* Algorithm for Robot Path Planning. *Procedia Technology*, 23:144–149, Jan. 2016. doi: 10.1016/j.protcy.2016.03.010.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. doi: 10.1109/TSSC.1968.300136.

[8] J. H. Holland. Adaptation in Natural and Artificial Systems. https://mitpress.mit.edu/9780262581110/adaptation-in-natural-and-artificial-systems/, 1992. [Accessed 08-01-2024].

[9] L. Jianqin and G. Xiao. Research on improved A-star algorithm for global path planning of unmanned logistics vehicles. In *2022 14th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, pages 44–47, Aug. 2022. doi: 10.1109/IHMSC55436.2022.00019.

[10] C. Kim, J. Suh, and J.-H. Han. Development of a Hybrid Path Planning Algorithm and a Bio-Inspired Control for an Omni-Wheel Mobile Robot. *Sensors*, 20(15):4258, July 2020. doi: 10.3390/s20154258.

[11] G. D. Luca. Roulette Selection in Genetic Algorithms. https://www.

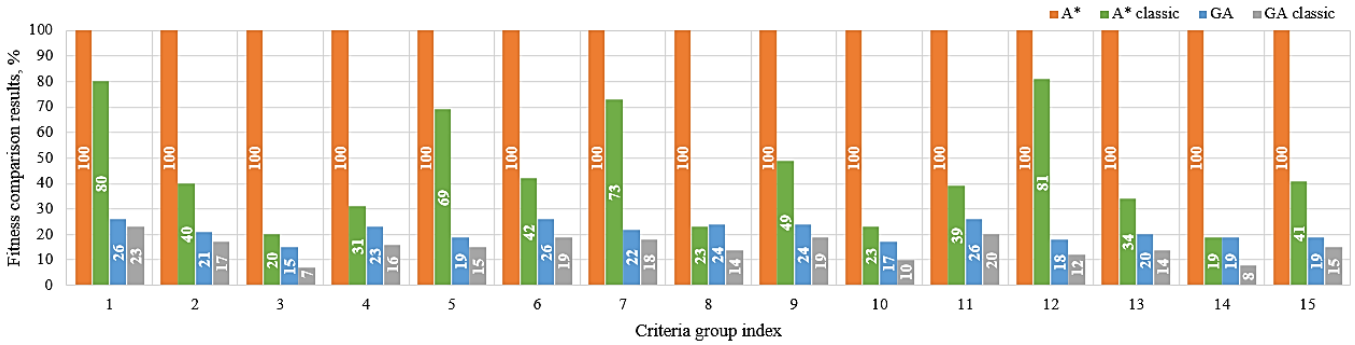

Figure 8. Relation of solutions with better fitness: A* (orange bars), classical A* (green bars), GA (blue bars) and classical GA (gray bars).

baeldung.com/cs/genetic-algorithms-roulette-selection, 2020. [Accessed 27-02-2024].

[12] M. Neumann and J. Burgner-Kahrs. Considerations for follow-the-leader motion of extensible tendon-driven continuum robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 917–923, 2016. doi: 10.1109/ICRA.2016.7487223.

[13] T.-D. Nguyen and J. Burgner-Kahrs. A tendon-driven continuum robot with extensible sections. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2130–2135, 2015. doi: 10.1109/IROS.2015.7353661.

[14] F. Renda, F. Boyer, J. Dias, and L. Seneviratne. Discrete cosserat approach for multisection soft manipulator dynamics. *IEEE Transactions on Robotics*, 34(6):1518–1533, 2018. doi: 10.1109/TRO.2018.2868815.

[15] D. C. Rucker and R. J. Webster III. Statics and dynamics of continuum robots with general tendon routing and external loading. *IEEE Transactions on Robotics*, 27(6):1033–1044, 2011. doi: 10.1109/TRO.2011.2160469.

[16] D. C. Rucker, B. A. Jones, and R. J. Webster III. A geometrically exact model for externally loaded concentric-tube continuum robots. *IEEE Transactions on Robotics*, 26(5):769–780, 2010. doi: 10.1109/TRO.2010.2062570.

[17] T. L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill International Book Company, 1980.

[18] O. Shamilyan, I. Kabin, Z. Dyka, and P. Langendoerfer. Distributed Artificial Intelligence as a Means to Achieve Self-X-Functions for Increasing Resilience: the First Steps. In *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–6, June 2022. doi: 10.1109/MECO55406.2022.9797193.

[19] O. Shamilyan, I. Kabin, Z. Dyka, O. Sudakov, A. Cherninskyi, M. Brzozowski, and P. Langendoerfer. Intelligence and Motion Models of Continuum Robots: An Overview. *IEEE Access*, 11:60988–61003, 2023. doi: 10.1109/ACCESS.2023.3286300.

[20] P. Sharma. Understanding Distance Metrics Used in Machine Learning. https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/, 2024. [Accessed 07-03-2024].

[21] L. Zuo, Q. Guo, X. Xu, and H. Fu. A hierarchical path planning approach based on a$*$ and least-squares policy iteration for mobile robots. *Neurocomputing*, 170:257–266, 2015. doi: https://doi.org/10.1016/j.neucom.2014.09.092.