A*-based co-evolutionary approach for multi-robot path planning with collision avoidance

Morteza Kiadi[a], Enol García[b], José R. Villar[b]* and Qing Tan[a]

[a]*School of Computing and Information Systems, Athabasca University, Athabasca, Canada;*
[b]*Department of Computer Science, University of Oviedo, Oviedo, Spain*

* Department of Computer Science, University of Oviedo, Faculty of Geology office 5.21, Campus de Llamaquique s/n 33005 Oviedo (SPAIN). E-mail: villarjose@uniovi.es.

A*-based co-evolutionary approach for multi-robot path planning with collision avoidance

In this research, a coevolutionary collision free multi-robot path planning that makes use of A* is proposed. To find collision-free paths for all robots, we generate a route for each of robot using A* path finding but introducing restrictions for each collision found. Afterward, a co-evolutionary optimization process is implemented for introducing changes in the initial paths to find a combination of routes that is collision-free. The approach has been tested in mazes with increasing the number of robots, showing a robust performance although at high time expenses. Nevertheless, several enhancements are proposed to tackle this issue.

Keywords: multi-robot path planning, a* algorithm, evolutionary algorithms, co-evolutionary algorithms

Introduction

This research aims to study optimal path finding for multiple telepresence robots autonomously move from the base station to the lab tables in a controlled indoor environment, a smart laboratory (Tan, Denojean-Mairet, Wang, Zhang, Pivot, & Treu, 2019). Finding the best path from start points to destinations for multiple mobile robots in an indoor environment is defined as Multi-Robot Path Planning (MPP) problem, which is one of the classic research topics of mobile robotics. There are many use cases, such as, indoor multi-robot virtual environments for educational purposes (Solak, Yakut & Bolat, 2020), collaborative tasks in manufacturing, surveillance, and space exploration (Huang, Jiang, Yu, Kang, & Hu, 2018), etc. Multi-robot Path Planning normally includes a local path planning and a global path planning. Global planning produces valid routes for each robot ahead based on the static environment while local path planning is to avoid the collisions based on real-time data received from sensors in the robot and/or the environment.

In our previous researches (Kiadi, Villar, & Tan, 2019) (Kiadi, Villar, & Tan, 2021), we proposed Reinforcement Learning and then Kubernetes and Fog Computing facilities as a global path-planer in an indoor multi-robot path planning problem. This research represents an extension of this latter study that i) proposes A* algorithm to determine the routes for the robots, ii) introduces a simple collision detection method to determine the invalid routes, iii) presents a simple method to seek for alternatives, and iv) designs a co-evolutionary optimization introducing changes in the routes to find collision-free path for each robot. The solution at a high level has three stages as is shown in Figure 1.

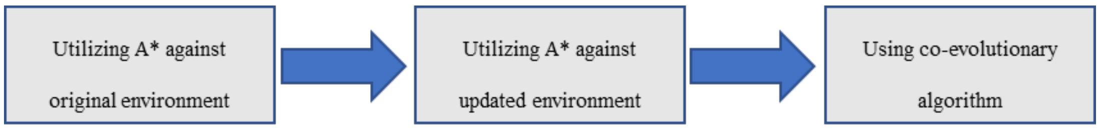

Utilizing A* against original environment
Utilizing A* against updated environment
Using co-evolutionary algorithm

Figure 1: A scheme of the solution as a block diagram.

This study is structured as follows. The next section presents the related works concerning multi-robot path planning, focusing on those solutions using metaheuristics in finding an optimal combination of routes. Then, a complete description of the multi-stage coevolutionary solution proposed in this work follows. Afterward, the experimentation setting, the obtained results, and a discussion on the findings is included. Finally, the conclusions of this research are drawn.

Related Work

Multi-robot path planning (MPP) problem is to find optimal paths for a certain number of autonomous robots moving within an environment without colliding with each other and obstacles. The key issue of MPP problem is to search optimal path, which has mainly been found using heuristic approach and Artificial Intelligence (AI) algorithm. For robot path searching, traditional heuristic approach like A* and Dijkstra as well-known algorithms has been continuously used for global path planning. Recently, a novel algorithm for multiple robots' global path planning was presented as a decoupled approach, which finds the path based on map of working environment and creating the formation of robots using merge and split functions along with an extended version of Dijkstra algorithm for MPP problem (Pereyra, Araguás, & Kulich, 2019). On the other hand, AI algorithm, such as ANN, GE, and EA as new heuristic and biology-inspired approaches have attracted great research interest in academic studies and engineering applications to find collision-free optimal path. The state of the art show that GA (genetic algorithm), PSO (particle swarm optimization) algorithm, APF (artificial potential field), and ACO (ant colony optimization) algorithm are the most used approaches for the path planning of mobile robot (Zhang, Lin, & Chen, 2018). As our research also mainly focuses on using evolutionary algorithm to tackle the MPP problem, more research works in this subject area are reviewed as follows.

Applying the GA in solving a path planning problem means creating a representation of the problem, defining a fitness function and the chromosome structure, and generating initial and subsequent generations of the population by different operations like cross-over or mutation (Wirsansky, 2020). An ant colony optimization has been used for the creation of the Max-Min Ant System for Dynamic Path Planning algorithm for the MPP based on topological maps whose focus is the main reference points of the environment and their connections. The path can be composed by a sequence of state/actions pairs without the information of the complete map (Santos, Otero, Johnson, Osorio, & Toledo, 2020). Another research has developed an algorithm, a modified PSO (MPSO) combined with the η 3 -splines for path planning of mobile robots with kinematic constraints. The MPSO algorithm with adaptive random fluctuations (ARFs) deals with the local convergence issue in the planning. The evolutionary state is classified equally at each iteration according to the evaluated evolutionary factor, and the velocity updating dynamics switches are imposed on the best particles (Song, Wang, Zou, Xu, & Alsaadi, 2019). A research compares the efficiency of two path planning methods, namely probabilistic roadmap (PRM) and genetic algorithm for mobile robots. The research concluded that GA produced smoother paths but consumed more processing time which made it difficult to implement in real-time navigation (Santiago, De Ocampo, Ubando, Bandala, & Dadios, 2017).

The GA has also been suggested in path planning for unmanned vehicles. For example, GA has been utilized with an improved initial population along with adaptive crossover and mutation probability operations, which have produced a shorter and safer non-collision path in different lake environments for lake patrol unmanned surface vehicle (Long, Su, Zhang, & Li, 2018). A path planning for single robot using GA, the solution has tried to optimize the pathfinding through customizing fitness functions and variable-length chromosomes along with three genetic operators namely simplification, revision, and substitution operators (Ni, Wang, Huang, Wu, & Luo, 2016).

Using heuristics and AI algorithm in MPP, it does not guarantee an optimal solution. The algorithms have their own strength and weakness in the applications. A research was conducted to compare the A*, D*, and PSO algorithms for path planning. It concludes that A* algorithm outperforms PSO in terms of processing time of the algorithms and path lengths (Okumuş & Kocamaz, 2019). A co-evolutionary improved Genetic Algorithm shows improvements in fitness function, proposes a new genetic modification operator like selection and crossover operations. It avoids the problem of local optimum and increases convergence rate. The use of a co-evolution mechanism fully enables the cooperation between populations, which avoids collision between mobile robots and generates an optimal or near-optimal collision-free path (Qu, Xing, & Alexander, 2013). Therefore, merging algorithms to improve the MMP solution has been used in our research. We use A* algorithm to create optimal initial paths for each robot in order to obtain the populations of each generation for the co-evolutionary algorithm to find a combination of routes that is collision-free.

A co-evolutionary approach for collision avoided multi-robot path planning

The solution includes several stages: i) obtaining A* routes for each robot, ii) if collisions are detected alternatives A* routes are found introducing fake obstacles and iii) a multi-objective coevolutionary algorithm (CA) to evolve the paths to find a valid combination of routes for each robot. The power of the CA was demonstrated in various applications such as classification, process control, and constraint satisfaction (Liu, 2009). Two-dimensional environments are assumed and through a maze (M) with P rows and Q columns, where each cell has the size of a robot and can be empty or containing one robot at a time.

We assume R robots and a route should be determined for each of them. Let us denote δ is the set of valid movement operations. A movement operation δ_i is each of the unitary movements the robot can perform in a time unit. In this study, for the sake of simplicity, includes only the HALT –(H), making the robot stop for one time slit- and the primitive movements -Up (U), Down (D), Left (L), and Right (R)-. Finally, each robot path is represented with the initial position r_0 and the sequence of movements, each movement belonging to δ. Figure 2 give an example of the route representation.

The initial A* routing stage

For each of the R robots the A* determines the best route from the current position to the corresponding goal. Afterwards, the collision detection is performed to find out if the set of R routes are valid to be deployed. Whenever a collision is detected, then the collision point is marked as an obstacle for the corresponding robots and A* is repeated until a valid solution is found or the limit of repetitions *nA* reached. The algorithm (shown in Algorithm 1) is as follows. Let Lr be the list of A* paths for robot r. In each iteration, the routes in Lr for all the robots are checked so when any combination of routes -one per robot- represents a collision free solution, the whole process is stopped, and this solution is returned. Algorithm 1 explains this initial stage.

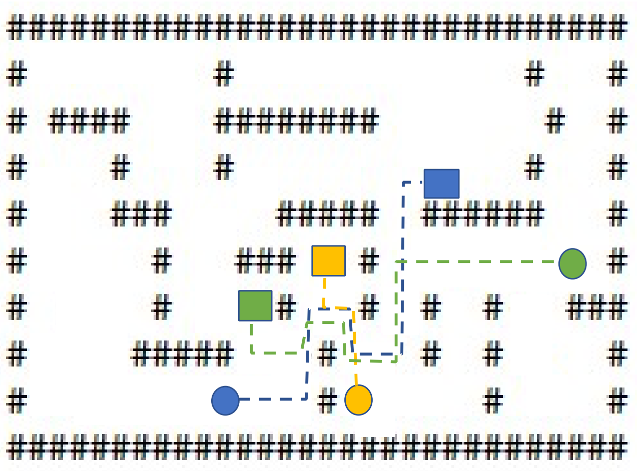


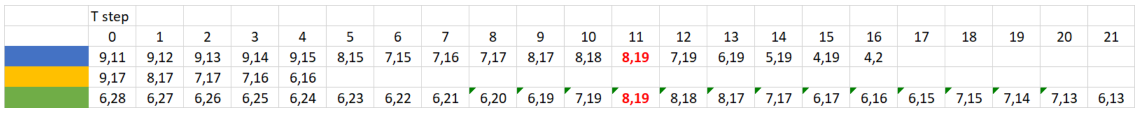
Figure 2. Example of three robots and their A* routes in a character-based maze. The squares are the starting points, the circles are the goals. Then the sequence of movements is RRRRUURRDRRUUUUR for the blue robot, UULU for the orange and LLLLLLLDDLLULDLLU for the green.

```
$M'_r \leftarrow M, \forall r = 1 \dots R$
$Lr \leftarrow [], \forall r = 1 \dots R$
$i \leftarrow 0$
**While** i < nA **and** collisions exist, increment i at the end
    $Lr \leftarrow Lr \cup [A^* \text{ for } M' \text{ and } r], \forall r = 1 \dots R$
    Detect collisions on M
    Update $M'_r$ for each robot r for which a collision exists
**If** no collision detected
    **Return** the solution found
```

Algorithm 1. Initializing the set of A* for each robot r. The initial map M is cloned for each robot (M'_r) and, whenever a collision is detected, updated with the collision point marked as an obstacle on M'_r. Up to nA routes for each robot are determined using this algorithm. Interestingly, all the collisions are evaluated on the real map M.

To detect the collisions among the all the routes we take advantage of the discrete time assumption: each robot moves one cell in each time slot. This might not be true when working with directions beyond the primitive ones, introducing some adjustments that are out of the scope of this work. On the other hand, the possibility of having robots with different speeds would be easily integrated just by considering a different increment in the number of cells. These two issues are left as future work.

The collision detection is performed by determining, at each time step, the current location of each robot and notifying whether robots coincide in a cell or not. This simple procedure requires the exhaustive analysis of all the routes, including routes still in progress in the virtual lab. For the maze with three robots of Figure 2, the process of collision detection is illustrated in Figure 3.



	T step																					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
🟦	9,11	9,12	9,13	9,14	9,15	8,15	7,15	7,16	7,17	8,17	8,18	**8,19**	7,19	6,19	5,19	4,19	4,2					
🟨	9,17	8,17	7,17	7,16	6,16																	
🟩	6,28	6,27	6,26	6,25	6,24	6,23	6,22	6,21	6,20	6,19	7,19	**8,19**	8,18	8,17	7,17	6,17	6,16	6,15	7,15	7,14	7,13	6,13

Figure 3. Illustration of the collision detection. At time slot 11 a collision has been detected for robots blue and green.

At the end of this stage, we can be in one of the following states. The first case is that we have found a valid combination of routes, one for each robot, with no collisions; in this case, this is the solution finally proposed. The second case is that we do not have a solution, so we have R lists of A*-base routes Lr (one for each robot) and length nA; these lists are use in the generation of the initial population of the co-evolutionary process explained next.

Co-evolutionary collaborative phase and mutation operation

The co-evolutionary algorithm is as follows: we generate the initial population P^r_0 for each robot and evaluate both locally and globally each route in P^r_0. Then, detect collisions of the best candidates from each robot. If no solution is found, coevolve until either a solution is found or the maximum number of generations is reached.

Algorithm 2 presents the co-evolutionary procedure, where **PS** is the total size of the population, each robot has **PS/R** individuals. Each individual is encoded using the initial population (which do not vary) and the sequence of movements, each movement being one of the valid movements *δ*.

```
$P^r_0$ ← Generate the initial population, ∀r
Co-evaluate $P^r_0$ locally
Co-evaluate $P^r_0$ globally
Detect collisions on M
i ← 0
While i < nE and no optimal solution found, end with i=i+1
   $P^r_i$ ← Generate next population
   Co-evaluate $P^r_i$ locally using [$P^r_{i-1}$, ∀r]
   Co-evaluate $P^r_i$ globally using $P^r_i$, ∀r]
   Detect collisions on M
```

Algorithm 2. Steps in the co-evolutionary algorithm for finding a set of paths without any collision. Each of the stages are detailed in the text.

The individual representation and the mutation operation

To represent an individual belonging to a subpopulation, that is, a certain robot, we need the initial position r_0 and sequence of movements. We also store the final position r_F for the sake of keeping information; nevertheless, neither r_0 nor r_F are modified: they are included in the representation to simplify the testing. The real chromosomes of the individual are the sequence of movements, each movement belonging to the ser of valid movements *δ*.

The mutation operation is a very simple modification of an individual ***I*** to generate a new route. Firstly, we select a random point within the individual, this is the mutation starting point (MSP). Then, we select a random operation *o* to insert among the valid movements. In case the operation to insert becomes HALT, the mutation ends with a valid routed with the operator inserted as a new node at MSP; see Figure 4 for details.

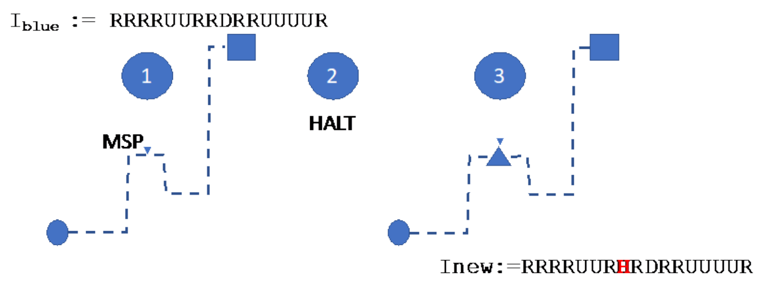


Figure 4. Example of a mutation that introduces the operator HALT.

Otherwise, we need to develop further. Firstly, we select a second random position within the individual, the mutation ending point (MEP). This point could be any of the individuals; however, for the sake of simplicity, we consider the MEP > MSP. Next, we insert ○ followed by all the movements in the individual from MSP to MEP. Finally, we insert the operation that leads to joining the initial route in a single step. The process is illustrated in the next figures, when either an extra operator is needed (Figure 5) or when no extra operator is needed because the two routes coincide (Figure 6).

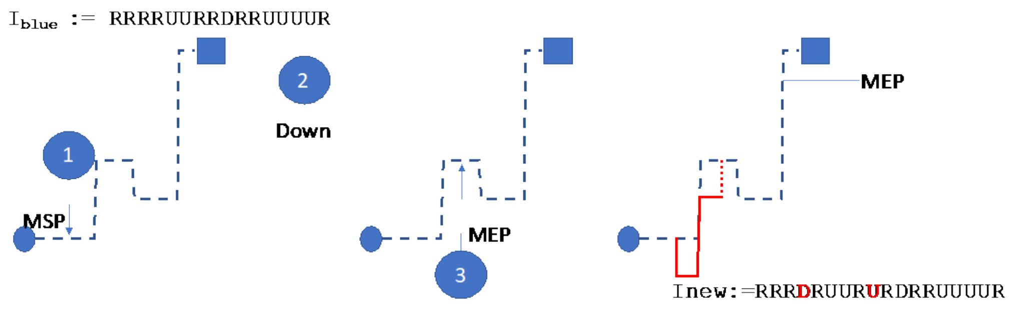


Figure 5. Example of a mutation using the Down operation and using an inverse operation to return to the original path.

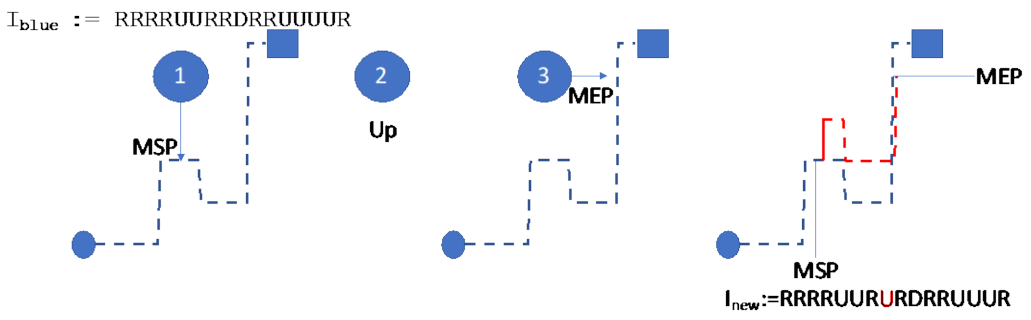


Figure 6. Example of a mutation in which no extra operation is needed to return to the original path

The initial population

The initial population makes use of all the A* candidates generated in the previous stage, that is, `Lr` ∀r. As long as co-evolutionary scheme is used, a subpopulation for each robot is needed. Defining `PS` as the total population size and `R` the number of robots, each subpopulation $\boldsymbol{P_s^r}$ includes `PS/R` robots. In particular, the initial subpopulation for robot r ($\boldsymbol{P_s^{r0}}$) includes the `nA` routes in its corresponding `Lr` plus `(PS/R - 1)` individuals generated using the mutation operator. The individual to mutate is randomly chosen from `Lr`.

Local evaluation of an individual

Each individual needs an evaluation of their performance and validity. The performance is measured using two different objectives: on the one hand, the estimation of the number of collisions the route of the individual induces; on the other hand, the length of a route in the number of operations.

The length of a route is easily measured and relies only on the robot's properties, with a high fitness value close to Infinitum for those invalid routes. With validity, we refer to the ability of the route to advert obstacles. Checking the validity is just analysing the operations included in the route from its origin to its end; if any operation forces to move onto an obstacle the route becomes invalid, and its fitness is set to a very high value.

The estimation of the number of collisions is a more complex task that relies on the selected routes from the other robots involved in the path planning. Therefore, a collaborative co-evolutionary scheme is proposed to estimate the number of collisions. Given an individual $\boldsymbol{I_i^{rn}}$ - individual ***i*** for robot ***r*** from any of its sub-populations at generation ***n*** -, we will estimate the number of collisions using ***Q*** individuals from each of the robots at generation ***(n - 1)***. The process is illustrated in Figure 7.

As mentioned, **Q** individuals are selected from each population $\boldsymbol{P^{a\,(n-1)}}\ \forall \boldsymbol{a} \neq \boldsymbol{r}$ following the tournament selection with **p** initially set to 0.5 and without replacement; we call these pools $\boldsymbol{P_Q^{a\,(n-1)}}\ \ \forall \boldsymbol{a} \neq \boldsymbol{r}$. The value of **Q** is kept small (about 3 or 5) to avoid high combinatorial problems. Then, every single combination of $\boldsymbol{I_i^{rn}}$ with an individual from each pool to form a solution to the multirobot path planning is checked for collisions using the procedure described before. The average (or the minimum, we need to check which one works better) number of collisions represents the estimation of the number of collisions for $\boldsymbol{I_i^{rn}}$.

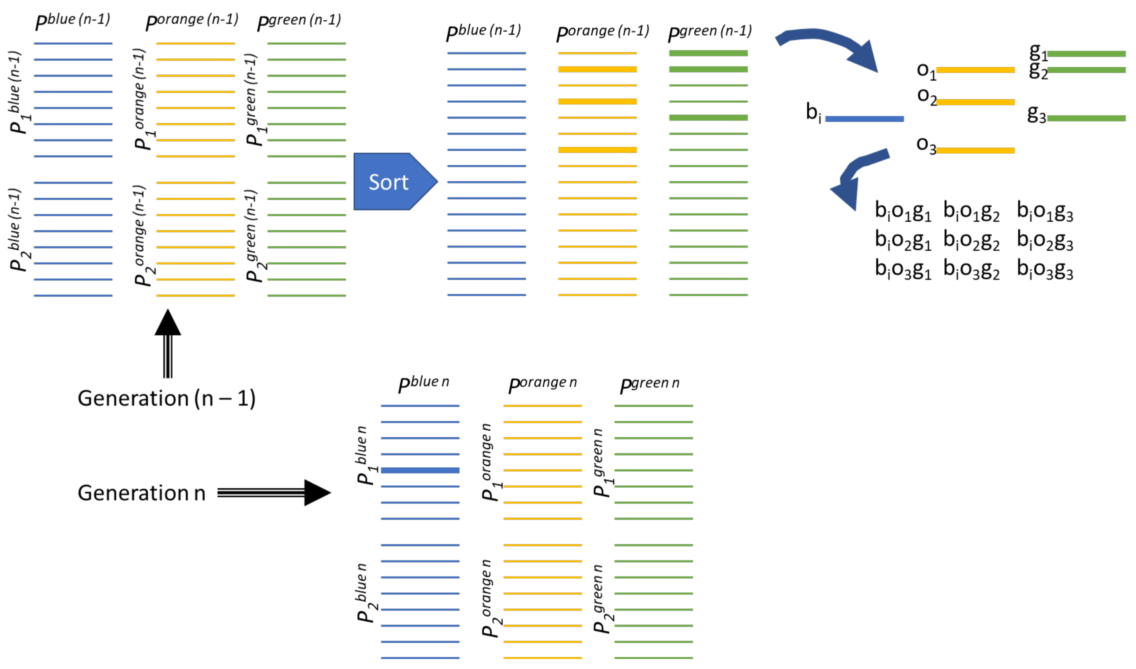


Figure 7. Local evaluation of an individual. The individual i within the blue population at generation n is evaluated for collision detection in collaboration with Q individuals from the other populations at generation (n – 1), with Q=3. All the possible combinations of b_i with the selected individuals are examined to detect collisions.

It is worth noticing that we use the population in the generation `(n - 1)` to estimate the number of collisions of any individual in generation n. This scheme works well for all the generations but the first one because this latter does not have any previous generation to count with. Therefore, for the first generation, only the length of the individual will be used in the tournaments and the local evaluation of any individual at generation 0 is performed using the populations in this initial population.

Co-evolutive evaluation

In this step, we obtain a measure of how good the solutions drawn by the evolutionary algorithm are. This process involves sorting the individuals in each population using the local evaluation. This local evaluation includes two measurements: the estimated number of collisions and the length of the individual; therefore, we need an extra criterion to sort them. In this study, we will sort first according to the estimated number of collisions and then using the length of the individual.

In a second step, we select the $\boldsymbol{W}$ best-ranked individuals from each population at the current generation. Afterward, all the possible solutions to the multi-robot path-planning problem are tested to detect collisions. The minimum number of collisions found represents the co-evolutionary fitness value; when this value equals 0 a valid solution has been found. Figure 8 illustrates this evaluation.

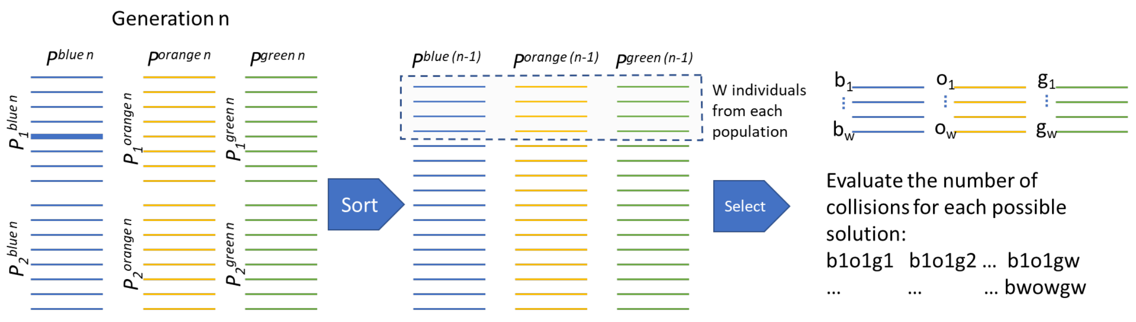


Figure 8. The co-evolutive evaluation. The minimum among all the possible combinations of the W best solutions from each populations represents the measurement of the goodness of the process.

Intermediate and final population generation

We propose to use elitism; therefore, we keep the best $\mathbf{K}$ individuals from each subpopulation $\boldsymbol{P}_s^{r\,(n-1)}$ into the next generation subpopulation $\boldsymbol{P}_s^{r\,n}$. We then complete the sup-population size up to `(PS/nA − K)` using the selection and mutation operators. The mutation operator is the same explained for the initial population; the selection follows a tournament with **p** initially set to 0.35.

Experiments and Results

The experimentation description

To evaluate the coevolutionary collision free multi-robot path planning we propose two different scenarios for which the number of robots will vary; in this way, it would be possible to assess the effectivity and efficiency of the solution.

The first maze is a toy problem of 10x30 tiles, where each cell has the robot size; when occupied, the cell can hold only one robot at a time. We refer to this maze as MAZE1, which is depicted in Figure 9. For this MAZE1, the number of robots will vary from 3 to 15, so the different cases are analysed: from the case for which the A* routes do not generate any collision to those where the number of collisions gets increased. A comparison with the state-of-the-art research proposed in (Qu, Xing, & Alexander, 2013) is included for this maze. We will refer to this study, from here and after, as QXA2013.

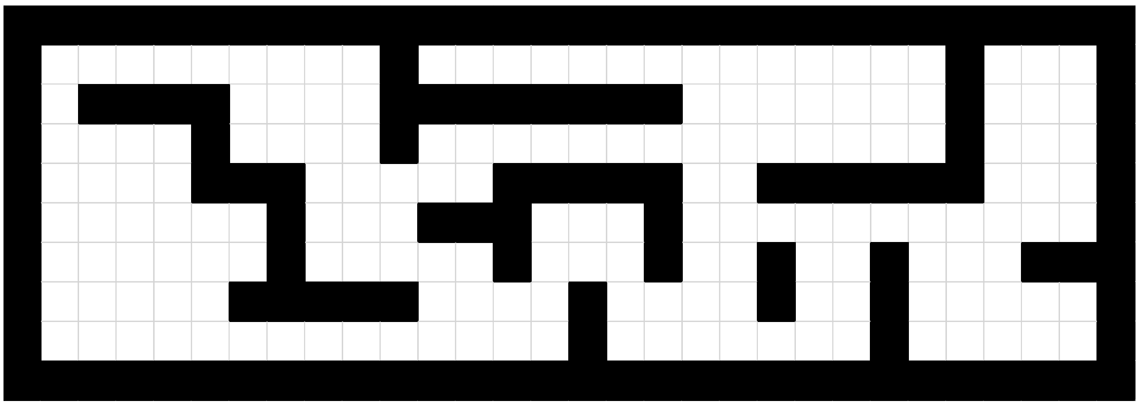


Figure 9. The MAZE1 maze. Each cell can hold a robot at a time. There are several areas in this maze where only a robot at a time can go through, but the total size of the mase is small.

The second maze includes four 10x30 empty tiles mazes that are interconnected to their two closest areas though bridges of one cell wide; the whole maze is presented in Figure 10 and is referred hereinafter as MAZE2. For this maze, three robots were in each room of the maze, with different goals set to each of the other rooms, so to increase the difficulty of the problem. The initial position of each robot is as closer as possible to the corner that is opposite to the bridge between rooms.

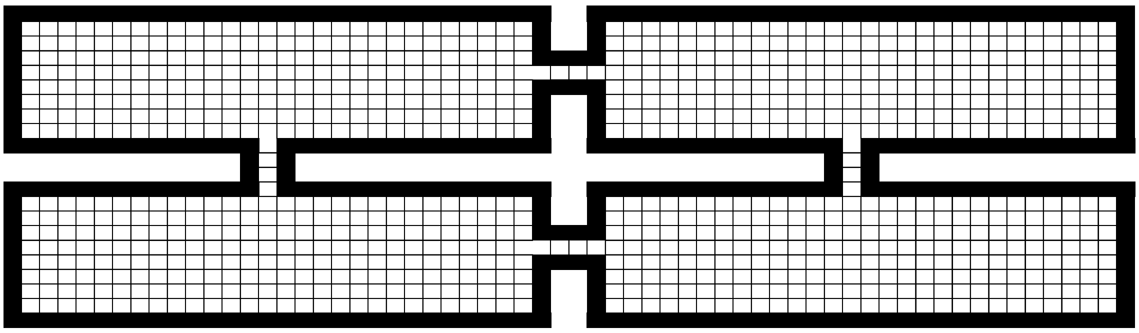


Figure 10. The MAZE2 maze. The robots will depart from the opposite corner to the bridges, although each one has its own position due to the physical restrictions set on each cell.

Obtained results

The results are presented using the following tables: Table 1 shows the number of coevolution generations that were needed to obtain a valid solution; it also shows the average number of generations with the QXA2013 for each problem. Table 2 depicts the length f of the path for each robot in each of the runs of the algorithm, comparing our solution with that proposed in QXA2013. Finally, Table 3 includes the time spent in calculations by each of the methods in this comparison for each of the different number of robots.

As can be seen in Table 1, the number of generations that our method needed kept increasing with the number of robots in the game. Nevertheless, this number of generations was found valid because only 23 generations were needed in the worst case.

Interestingly, when the number of robots was smaller or equal than 5, the multiple A* route finding and the combination of them was enough to find a collision-free proposal. When the number of robots higher is 12 or higher, the time spent in the calculations increased so much and we opted for stopping the process. On the other hand, the number of iterations highly increases with the number of robots for the QXA2013, although the required time is significantly shorter.

Table 2 shows that our proposal does not suffer on introducing large routes. Furthermore, for most of the cases the standard deviation of the length of the routes was kept reduced; only for the higher number of robots the standard deviation became somewhat higher. However, this is understandable due to the size of the maze and the increment in the density of robots. For the case of QXA2013, the length of the paths was always much longer and with a high variability from one run to another.

Table 1. Number of iterations needed to obtain a solution for MAZE1. The first collision that requires a co-evolutionary search is with 6 robots for our research; therefore, no results are shown for a smaller number of robots. MN stands for average.

	Experiment repetition for our solution										**Our**	**QXA2013**
NR	1	2	3	4	5	6	7	8	9	10	**MN**	**MN**
6	18	3	21	19	18	25	11	12	8	20	**15.5**	23.6
7	15	27	8	12	9	6	15	11	12	7	**12.2**	40.3
8	25	16	48	6	22	21	10	22	19	17	**20.6**	44.5
9	16	19	20	15	13	24	5	45	50	13	**22.0**	96.1
10	13	24	16	10	15	28	32	13	31	11	**19.3**	147.9
11	23	12	11	26	29	26	6	9	14	19	**17.5**	236.5

Finally, Table 3 states the number of seconds consumed in finding suitable solutions. As expected, the higher the number of robots, the higher the time consumed.

In this case, QXA2013 defeats our solution as it shows fast convergence. For our solution, the time consumption grows exponentially with the number of robots, which is a real drawback. These results suggests that a better co-evolutionary solution should be developed, and some ideas can be deduced. In the first hand, those robots that do not collide must not be introduced in the co-evolutionary process. This solution would substantially reduce the time in those cases where only a few robots collide. On the second hand, the co-evolutionary search can be restricted to the proximity of the collisions but always before of the collision point. With this enhancement the search space is reduced, and the time spent can be reduced. Finally, this proximity can vary to focus on wider areas when no enhancement of the global fitness is observed.

Table 2. Average and standard deviation -within parenthesis- of the length of the routes for the robot (R) in each scenario -with different number of robots (NR)-. Our method beats QXA2013 in terms of the average path length and in robustness measured as a reduced standard deviation.

NR	6		7		8		9		10		11	
R	**Our**	**QXA2013**	**Our**	**QXA2013**	**Our**	**QXA2013**	**Our**	**QXA2013**	**Our**	**QXA2013**	**Our**	**QXA2013**
1	**16.20 (0.6325)**	63.60 (30.5876)	**17.20 (1.0328)**	45.00 (14.1342)	**17.20 (1.0328)**	42.60 (12.8858)	**16.80 (1.0328)**	38.40 (15.6929)	**17.20 (1.0328)**	41.60 (15.6858)	**19.60 (3.0984)**	40.00 (15.2898)
2	**4.00 (0.0000)**	53.60 (18.0874)	**4.00 (0.0000)**	51.00 (20.4287)	**4.00 (0.0000)**	59.10 (14.3639)	**4.00 (0.0000)**	42.30 (15.2392)	**4.00 (0.0000)**	38.60 (17.9703)	**4.00 (0.0000)**	33.70 (16.4388)
3	**24.60 (0.6992)**	58.60 (17.8463)	**24.30 (0.6749)**	57.10 (15.1030)	**24.60 (0.8433)**	55.30 (15.9238)	**24.20 (1.0328)**	43.90 (13.5519)	**24.30 (2.8694)**	51.40 (19.3861)	**24.00 (1.4907)**	47.40 (18.2647)
4	**24.30 (0.4830)**	68.30 (27.5441)	**24.70 (0.8233)**	51.80 (17.1968)	**24.40 (0.5164)**	53.60 (19.0566)	**25.40 (1.4298)**	39.30 (12.8154)	**28.60 (2.5473)**	42.50 (18.5966)	**29.50 (1.9579)**	33.10 (14.8881)
5	**21.00 (2.1082)**	63.30 (28.5970)	**20.50 (1.2693)**	55.40 (26.0520)	**23.40 (5.8157)**	56.40 (24.8828)	**24.00 (2.5386)**	46.70 (19.6867)	**24.60 (5.4406)**	46.50 (18.2041)	**23.80 (5.4528)**	37.40 (16.7876)
6	**33.90 (1.3703)**	51.00 (10.5093)	**33.90 (0.7379)**	56.80 (18.0234)	**33.30 (1.4181)**	56.30 (15.9237)	**33.30 (1.0593)**	43.40 (12.2854)	**33.40 (1.1738)**	49.10 (17.6915)	**33.10 (0.8756)**	40.50 (9.5714)
7	-	-	**18.00 (0.0000)**	54.50 (22.5351)	**18.00 (0.0000)**	60.60 (17.2123)	**18.00 (0.0000)**	41.90 (18.0213)	**18.00 (0.0000)**	43.80 (15.7042)	**20.20 (0.6325)**	39.20 (21.9636)
8	-	-	-	-	**20.90 (1.1005)**	49.10 (18.2236)	**20.40 (0.6992)**	37.00 (11.0252)	**27.10 (9.1827)**	41.60 (21.7521)	35.20 (9.1263)	**32.50 (12.4387)**
9	-	-	-	-	-	-	**17.40 (0.6992)**	50.70 (19.9857)	**17.20 (0.6325)**	43.40 (15.9248)	**17.00 (0.0000)**	34.20 (15.4833)
10	-	-	-	-	-	-	-	-	**32.30 (0.8233)**	40.50 (13.1930)	**32.30 (0.8233)**	37.40 (9.5475)
11	-	-	-	-	-	-	-	-	-	-	**28.40 (0.6992)**	33.80 (15.3319)

For MAZE2, the number of coevolutionary generations and the path length are included in Table 4. Again, the number of generations is relatively small, and the path length is kept small. Nevertheless, the time consumed in finding solutions is too high and that is the reason we do not study this solution with a higher number of robots per room. The same ideas found for MAZE1 can be repeated here so to improve the performance of the system.

Table 3. Average (MN), median (MD) and standard deviation (SD) of the time spent in finding a solution in each scenario -with different number of robots (NR)-. QXA2013 performed much better in terms of time consumption.

	Our solution			**QXA2013**		
NR	**MN**	**SD**	**MD**	**MN**	**SD**	**MD**
6	3,8319	1,4861	4,4080	4.7479	3.5449	3.2720
7	11,2127	5,4879	10,6205	9.6416	4.3064	9.7090
8	75,8119	41,5637	74,2549	14.3116	8.0994	11.8045
9	303,3489	203,8299	242,1621	32.8010	12.7689	32.0580
10	1255,6156	459,5965	1404,6725	59.7161	24.0424	66.6075
11	3958,5166	1954,4660	3982,3627	95.2412	46.1279	95.2854

Table 4. MAZE2 scenario: number of coevolutionary generations and the length of the route for each robot. REP, MN and SD stand for repetition of the experiment, average and standard deviation, respectively. Rx refers to the corresponding robot in the maze.

GEN	17	24	13	18	27	16	10	15	13	15	16,8	5,2
REP	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**MN**	**SD**
R1	65	65	90	65	65	90	89	91	65	89	77.4	13.1
R2	58	58	44	58	58	44	44	44	58	44	51.0	7.4
R3	79	78	104	79	78	104	105	105	79	106	91.8	13.9

R4	103	103	78	103	103	78	78	78	103	78	90.5	13.2
R5	45	45	58	45	45	58	58	58	45	58	51.5	6.9
R6	87	87	63	87	87	63	63	63	87	63	75.0	12.6
R7	105	107	80	105	107	80	78	77	105	78	92.2	14.4
R8	89	88	64	89	88	64	64	64	89	64	76.3	13.0
R9	44	42	52	44	41	52	52	55	44	52	47.9	5.1
R10	110	109	45	111	109	45	45	45	11	45	77.6	34.4
R11	63	64	89	65	64	89	88	91	65	88	76.8	12.9
R12	74	74	100	74	74	100	100	101	74	100	87.1	13.8

Conclusions

This study focuses on finding collision-free routes for each robot in an known indoor multi-robot laboratory. The proposal includes an initial step to determine A* routes for each robot, detecting collisions and finding alternative suboptimal A* routes, and a co-evolutionary stage that introduces changes in the routes to avoid the collisions.

The experimentation makes use of two simple mazes, one as a rectangular patter of tiles and another as four rooms interconnected by single-robot bridges. For the first maze, different number of robots were evaluated and for each scenario 10 runs have been considered. For the second maze, only 3 robots per room were introduced. The result from the experiments shows that the co-evolutionary solution can easily find collision-free routes for all the robots, with a reduced number of generations. However, there are problems due to the time of service, which increases with the number of robots and the maze complexity.

To tackle this issue, we propose several alternatives. Firstly, only robots that collide should be considered in the A* alternative route finding and the co-evolutionary process (although all the robots must be considered in the evaluations). Secondly, the initial mutation point should be selected from the neighbourhood of the collision point and, more precisely, from points in the route before the collision. In this way, the problem is solved locally, and the search space is highly reduced. Finally, the neighbourhood can be widened with the number of generations: the larger this latter the bigger the neighbourhood. This solution allows to increase the search space when the problem becomes harder due to restrictions in the domain.

Funding

This research has been funded by the Spanish Ministry of Science and Innovation under projects MINECO-TIN2017-84804-R and PID2020-112726RB-I00.

Disclosure Statement

The authors report there are no competing interests to declare.

Biographies

Morteza Kiadi. Currently, a Ph.D. student at the University of Oviedo (Spain) and graduated from Athabasca University with a Master of Information System (Canada). Morteza teaches different courses in different subjects such as Machine Learning, Data Science, Deep Learning, Cloud Computing, and Python programming. Morteza has recently published papers in international conferences and journals.

Enol García. Computer Science degree at University of Oviedo (2020), Enol is currently enrolled in the Computer Science Department as a PhD student. He has published several journal papers and international conference contributions. His research interest are Deep Neural Networks and their application in BioMedicine.

José R. Villar. Professor of the University of Oviedo, Spain. In 1992 he received the B. Sc. Electronics and Control Engineering degree from University of Oviedo. In 2012, he received his Ph. D. in Computer Science from University of León (Spain). He has worked in several companies dealing with control and instrumentation projects. From 1998 to 2004 he was with the Electric and Electronic Department of University of León. Since then, he is a member of the Computer Science Department at University of Oviedo. His research topics includes Hybrid Algorithms applied to Real World Problems, Human Activity Recognition and Event Detection. He has published over 40 papers in JCR journals and more than 100 contributions to international conferences.

Qing Tan. Associate Professor in School of Computing and Information Systems at Athabasca University, Canada. He earned his PhD in Cybernetics Engineering for Robotics from the Norwegian Institute of Technology in 1993. The Japan Atomic Energy Research Institute invited him in 1994 as a foreign senior research fellow. He did his post-doctoral fellowship at University of Alberta. He joined Athabasca University in 2007. His research interests include Location-Based Technology, Adaptive Mobile Learning, Internet of Things, Cloud Computing, Cyber-Physical System, and Telepresence Robotics. His research work also involves emerging technologies, such as, Artificial Intelligence, Machine Learning, and Blockchain. His research interests are also included the studying on the social impact of advanced computing and information technologies. He received many research grants and published more than 60 papers on international journals and conferences.

References

Estefanía Pereyra, M., Gastón Araguás, R., & Kulich, M. (2019). Sequential path planning for a formation of mobile robots with split and merge. *2017 IEEE Latin*

American Conference on Computational Intelligence (LA-CCI), 2017, pp. 1-6, doi: 10.1109/LA-CCI.2017.8285722.

Huang, D., Jiang, H., Yu, Z., Kang, C., & Hu, C. (2018). Leader-following Cluster Consensus in Multi-agent Systems with Intermittence. *International Journal of Control, Automation and Systems*, *16*(2), *437–451. https://doi.org/10.1007/s12555-017-0345-2*

Kiadi, M., Tan, Q., & Villar, J. R. (2019). Optimized Path Planning in Reinforcement Learning by Backtracking. *Current Trends in Computer Sciences and Applications 1(4), 80–90. https://doi.org/10.32474/CTCSA.2019.01.000116*

Kiadi, M., Villar, J. R., & Tan, Q. (2021). Synthesized A* Multi-robot Path Planning in an Indoor Smart Lab Using Distributed Cloud Computing. *Advances in Intelligent Systems and Computing, 1268 AISC,* 580–589. https://doi.org/10.1007/978-3-030-57802-2_56

Liu, J. G. (2009). Competitive coevolutionary genetic algorithms for multiobjective optimization problems. *2009 International Conference on Artificial Intelligence and Computational Intelligence, AICI 2009*, *3*, 594–597. https://doi.org/10.1109/AICI.2009.405

Long, Y., Su, Y., Zhang, H., & Li, M. (2018). Application of improved genetic algorithm to unmanned surface vehicle path planning. *Proceedings of 2018 IEEE 7th Data Driven Control and Learning Systems Conference, DDCLS 2018*, 209–212. https://doi.org/10.1109/DDCLS.2018.8515966

Ni, J., Wang, K., Huang, H., Wu, L., & Luo, C. (2016). Robot path planning based on an improved genetic algorithm with variable length chromosome. *2016 12th*

International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016, 145–149. https://doi.org/10.1109/FSKD.2016.7603165

Okumuş, F., & Kocamaz, A. F. (2019). Comparing Path Planning Algorithms for Multiple Mobile Robots. *2018 International Conference on Artificial Intelligence and Data Processing, IDAP 2018*, (2013), 18–21. https://doi.org/10.1109/IDAP.2018.8620785

Qu, H., Xing, K., & Alexander, T. (2013). An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. *Neurocomputing*, *120*, 509–517. https://doi.org/10.1016/j.neucom.2013.04.020

Santiago, R. M. C., De Ocampo, A. L., Ubando, A. T., Bandala, A. A., & Dadios, E. P. (2017). Path planning for mobile robots using genetic algorithm and probabilistic roadmap. *HNICEM 2017 - 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management*, *2018-Janua*, 1–5. https://doi.org/10.1109/HNICEM.2017.8269498

Santos, V. D. C., Otero, F. E. B., Johnson, C., Osorio, F. S., & Toledo, C. F. M. (2020). Exploratory path planning for mobile robots in dynamic environments with ant colony optimization. *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, (June), 40–48. https://doi.org/10.1145/3377930.3390219

Solak, S.; Yakut, Ö.; Dogru Bolat, E. (2020). Design and Implementation of Web-Based Virtual Mobile Robot Laboratory for Engineering Education. *Symmetry 12, 906.*

https://doi.org/10.3390/sym12060906

Song, B., Wang, Z., Zou, L., Xu, L., & Alsaadi, F. E. (2019). A new approach to smooth global path planning of mobile robots with kinematic constraints. *International Journal of Machine Learning and Cybernetics*, *10*(1), 107–119. https://doi.org/10.1007/s13042-017-0703-7

Tan,Q., Denojean-Mairet, M., Wang, H.X., Zhang, X.K., Pivot, F. & Treu, R. (2019). Toward a Telepresence Robot Empowered Smart Lab. *Smart Learning Environments, (2019) 6: 5. https://doi.org/10.1186/s40561-019-0084-3*

Wirsansky, E. (2020). *Hands-On Genetic Algorithms* with Python, Book, Packt, ISBN: 9781838557744, January 2020

Zhang, H. Y., Lin, W. M., & Chen, A. X. (2018). Path planning for the mobile robot: A review. *Symmetry*, *10*(10). https://doi.org/10.3390/sym10100450