Diversity Maintenance for Efficient Robot Path Planning

Petar Ćurković * and Lovro Čehulić

Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Ivana Lucica 5, 10000 Zagreb, Croatia; lovro.cehulic@fsb.hr
* Correspondence: petar.curkovic@fsb.hr

Received: 21 January 2020; Accepted: 28 February 2020; Published: 3 March 2020

Abstract: Path planning is present in many areas, such as robotics, video games, and unmanned autonomous vehicles. In the case of robots, it is a primary low-level prerequisite for the successful execution of high-level tasks. It is a known and difficult problem to solve, especially in terms of finding optimal paths for robots working in complex environments. Recently, population-based methods for multi-objective optimization, i.e., swarm and evolutionary algorithms successfully perform on different path planning problems. Knowing the nature of the problem is hard for optimization algorithms, it is expected that population-based algorithms might benefit from some kind of diversity maintenance implementation. However, advantages and potential traps of implementing specific diversity maintenance methods into the evolutionary path planner have not been clearly spelled out and experimentally demonstrated. In this paper, we fill this gap and compare three diversity maintenance methods and their impact on the evolutionary planner for problems of different complexity. Crowding, fitness sharing, and novelty search are tailored to fit specific problems, implemented, and tested for two scenarios: mobile robot operating in a 2D maze, and 3 degrees of freedom (DOF) robot operating in a 3D environment including obstacles. Results indicate that the novelty search outperforms the other two methods for problem domains of higher complexity.

Keywords: robotics; optimization; path planning; evolutionary computation; diversity maintenance; novelty search

1. Introduction

Robotics expands from traditional industrial environments towards coworking and coexisting with humans in domains like medicine, education, leisure, and the general service domain. Apart from the ethical aspects of robots sharing an environment with humans, there is still a plethora of technical issues limiting the robots’ ability to fully immerse in collaboration with humans.

One of such important problems is the limited ability of robots to perform efficiently in an unstructured environment. Such environments require robots to constantly re-plan, adapt, and optimize their actions. Furthermore, if an objective like time, distance, or collision avoidance is critical for the success of the operation of the robot, then its ability of planning and optimizing according to required objectives becomes of paramount importance.

For the above-mentioned reasons, path planning, meaning finding trajectories optimized under a set of often colliding criteria is a problem of intense interest in the scientific community. It is certain that reliable, safe, and timely path planning presents a stepping-stone in the direction for robots to reach their full potential in collaboration with humans.

There are many important contributions to path planning in recent years. A comprehensive historical survey can be found in [1].
Path planning can generally be classified based on the nature of the environment as static or dynamic. In the case of the static environment, the layout, once perceived by the robot remains unchanged. Dynamic environments include movable objects i.e., other robots.

According to the algorithm used for planning, the planner can either be local—meaning sensor-based and reactive, or global—meaning map-based and planning the trajectory from the initial to the final position at once. Based on the completeness, planning can be classified either as complete or heuristic. In the case of complete planning, the algorithm will always find a path in finite time if the path exists, or report that no feasible path exists in finite time. These algorithms are preferred for the systems with a low number of degrees of freedom over the heuristics algorithms [2–6].

In the case of a 3D problem considered in this paper, both the trajectory and the robot’s body have to avoid obstacles present in the environment and thus checked for collisions. This makes planning significantly more complex when compared to simple point-wise planning. It is known that for similar problems, population-based heuristic planning methods, i.e., swarm and evolutionary approaches are successfully utilized to find high-quality solutions in acceptable time [7–10]. It is also proven that the path planning problem is NP-complete, with the complexity of tasks increasing exponentially with the DOF number [11]. This is an additional motivation of exploring alternative methods to path planning, seeking for high quality (instead of optimal) solutions in limited time.

The majority of papers utilizing instances of population-based heuristic optimization for path planning deal with the case of a mobile robot, presented as a point in the space, aiming to find a set of going-through nodes in 2D space [12–14]. A more realistic scenario is using a 3D model of the robot, with movement in an environment containing obstacles of different shapes and positions. In this case, a simple solution of going-through nodes does not suffice, because the body of the robot plays a critical role in checking for collision-free motion [15,16]. In this paper, an evolutionary algorithm-based path planner is proposed to solve the problem of path planning both for a simplified model of a mobile robot and a realistic 3 DOF stationary robot.

The former will be presented as a point in space, while the later will be actual RRR model including links and presence of obstacles. It is shown that evolutionary algorithms (EA) are susceptible to getting trapped in suboptimal regions of the search space [17]. That means that the best individuals in the population are evolved fitter during the optimization process, but there is still space for improvement, that has to be utilized through the additional exploration of the search space. This usually happens when a local optimum has a large basin of attraction, thus forming a misleading region in the search space that can influence the whole population by trapping it in this suboptimal region. To alleviate this, diversity maintenance methods have been proposed to ensure that potential solutions (that make up the population) remain diverse. This way, EA’s are forcing some of the individuals to explore different regions of the search space with the motivation of eventually finding better optima, or in the best case, the global optimum.

Diversity maintenance (DM) generally can be either implicit or explicit [18], with the latter being of interest in this paper. Explicit DM schemes are based on the idea of forcing the population to maintain different niches when doing selection or replacement. Two of the most common forms are fitness sharing—in which genotypically similar individuals share fitness values with other individuals that are considered similar to them using appropriate metrics. It basically boils down to lowering the fitness of those individuals that have similar counterparts present in the population.

Crowding relies on the scheme that ensures that diversity is preserved by enforcing new individuals to replace similar predecessors in the new population. It is shown, however, that in the case of complex search spaces, after a certain complexity threshold, diversity maintenance may become ineffective [19]. Recent research shows that this problem may be a more serious one than previously thought [20,21].

The problem is that fitness heuristics are based on the assumption that incremental steps to the solution will increasingly resemble it, for the case of complex problems, the steps that in the end lead to the objective may poorly resemble it. So the hypothesis is that as the problem grows in difficulty, the sole gradient defined by measuring the distance to the objective becomes increasingly deceptive.
and thus less informative in an evolutionary context. The consequence is that maintaining diversity on the genotypic level alone does not significantly increase the effectiveness of the search. Despite the diversity explicitly enforced into the population, the bias imposed by the fitness function remains the primary parameter that directs the search. Therefore, individuals in a population, or combinations of steps that could lead to the solution of the complex problem, in the end, can diminish from the current population, and become extinct in an evolutionary context, and thus fail to find high-quality solutions.

Motivated by such results, the novelty method is proposed [22] in which the search process is driven primarily by phenotypic diversity. It is shown that scaling with complex search domains is better than those of traditional objective-based fitness functions. The explanation is that this method does not depend exclusively upon the fitness function to identify the building blocks that lead to a solution. It relies on the idea that already discovered innovations are building blocks for further evolutionary innovations. To summarize, genotypic diversity maintenance methods promote the creation of random raw material, from which objective function selects the most promising ones, novelty search discovers structure in the domain, and thereby making progress possible even when fitness is uninformative.

In this paper, we implement the two most commonly used forms of diversity maintenance, fitness sharing, and crowding. Afterwards we compare it to novelty search using the problem of path planning for mobile and 3 DOF robots. Finally, the three methods are evaluated against a base evolutionary algorithm without any diversity maintenance methods included.

2. Materials and Methods

Two scenarios will be analyzed in this paper: a two-dimensional case, for a mobile robot navigating through a maze, and a three-dimensional RRR robot. In the first scenario, a robot is presented as a dimensionless point in the pace. The second case considers a model of the robot with three degrees of freedom, as seen in Figure 1. Three joints of the robots are of revolute type, allowing the robot to reach an arbitrary position with the tool center point, considering the point falls in the workspace of the robot.

![Figure 1. Robot model used for the 3D path planning problem.](image)

Three versions of the algorithm will be compared to test how different population diversity maintenance methods influence the performance of the algorithm, and then compared to the base algorithm which does not include any DM. The three DM methods are: novelty search, fitness sharing, and crowding.

To keep the results of algorithm testing comparable, every part of algorithms, other than population diversity maintenance, is the same. Each algorithm tested both two-dimensional and three-dimensional...
environment setup, with the same obstacle placement and same starting and goal positions. Examples of 2D and 3D environments used to test the algorithms are shown in Figure 2.

![Testing environments](image)

**Figure 2.** Testing environments. **Left:** 2D maze for a mobile robot. **Right:** 3D environment for a robotic arm. Blue line—the best solution found by the algorithm.

### 2.1. Population

The matrix in Equation (1) represents the whole population. Each row describes one individual member, i.e., one potential solution, in this case, it is one trajectory. The initial population consists of \( m \) random individuals with \( n \) real numbers in the range \([-5, 5]\) on the position of each allele in the gene. So, the population consists of \( m \) genes of length \( n \).

\[
\text{population} = \begin{bmatrix}
    a_{11} = R_{11} & a_{12} = R_{12} & \cdots & a_{1n} = R_{1n} \\
    a_{21} = R_{21} & a_{22} = R_{22} & \cdots & a_{2n} = R_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{m1} = R_{m1} & a_{m2} = R_{m2} & \cdots & a_{mn} = R_{mn}
\end{bmatrix}
\]  \hspace{1cm} (1)

The trajectory is, in theory, a polynomial function of arbitrary order, and every member of the population represents a polynomial function, while each allele matches one of its coefficients. The length of the gene represents the order of the polynomial function. For the three-dimensional environment, the trajectory is composed of two different polynomial functions, in x-y, and x-z planes, respectively.

\[
P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x, \ a_i \in \mathbb{R}, \ i = 1 \ldots n \]  \hspace{1cm} (2)

Equation (2) defines one row of the matrix, this is one individual or candidate solution to the problem. As defined with Equation (1) there are \( m \) potential solutions that form the population. As for the population size, defined by \( m \), in our simulation, the population size is set to 50 individuals. The dimension \( n \) defines the order of the polynomial used to describe a trajectory.

Its size is experimentally determined and set to \( n = 7 \). If the order of the polynomial is set too low, there is not enough flexibility, or inflection points available to evolve a trajectory that avoids obstacles. Too high an order increases processing time significantly, but without any positive impact on the quality of the end solution. The order used in our simulations is determined through trial and error, but might potentially also be a part of the optimization or evolutionary process.
2.2. Fitness Function

Members of the population are evaluated according to their fitness values $F$. Fitness $F$ is calculated using the fitness function, and it depends on the length of the trajectory between the starting and goal positions, and on the penalty for the collisions with the obstacles.

$$F(i) = \frac{C}{1 + \text{penalty}(i) + \text{length}(i)}$$  \hspace{1cm} (3)

Penalty values are 90 for collisions with vertical obstacles, and 100 for collisions with horizontal obstacles, viewed depending on the observed plane. The number $C$ is arbitrarily chosen so that $C > 0$. The form of the objective function defined in Equation (3) performed well for the given examples in terms of discrimination of the candidates forcing them to simultaneously shorten their lengths, but at the same time avoid obstacles. The fitness of the best individual is bounded by the value $C/\text{length}$, which maps the whole population on a finite scale. It is not an easy task to balance out these parameters, since one can get overly dominated over the other, and so limiting the evolutionary potential of the candidates. Also, this approach with penalty enables the population to actually evolve over time, as opposed to the approach where one would simply remove an individual that does not perform well in the current generation. The values used in this study were experimentally determined.

2.3. Diversity Maintenance

The idea is to force the population to maintain diversity when doing selection or replacement. There are two most commonly used methods, fitness sharing, where fitness value of each individual is adjusted before the selection to separate them in niches in proportion to the niche fitness, and crowding, where individuals are distributed uniformly amongst niches using distance-based selection. A niche is defined as a subset of individuals from the population whose distance is below a given threshold, thus making a niche a group of similar individuals. A distance metric and a niche size are explained later in the text.

It should be noted that both methods use global parent selection and there is nothing that prevents the recombination of parents from different niches.

The novelty search is different in that it actually does not rely solely on the fitness function, but rather on a different metric—the novelty of an individual in the population. Novelty is combined with fitness in this study by copying the best, so-called elite member, from each generation directly to the next one. This way, an evolutionary trace or memory is preserved.

2.3.1. Fitness Sharing

This method is based on the idea that “sharing” of individual fitness values before the selection controls the number of individuals in each niche. In this method, every possible pairing of individuals $i$ and $j$ is considered, and then a distance was calculated between them. The fitness $F$ of each individual $i$ is then adapted according to the number of individuals falling within some pre-specified distance $\sigma_{\text{share}}$ using a power-law distribution:

$$F'(i) = \frac{F(i)}{\sum_j \text{sh}(d(i, j))}$$  \hspace{1cm} (4)

$$\text{sh}(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{\text{share}}}\right)^\gamma, & \text{if } d \leq \sigma_{\text{share}}, \\ 0, & \text{otherwise}. \end{cases} \hspace{1cm} (5)$$

The sharing function $\text{sh}(d)$ is a function of the distance. Constant value $\gamma$ determines the shape of the sharing function and it was taken as $\gamma = 1$ to make the function linear in this example. The last parameter was share radius $\sigma_{\text{share}}$, which defines how many niches can be maintained. Common values
are in the range of 5 to 10 \cite{16}, in this algorithm value of 10 was taken to promote more niches and consequently larger diversity.

\begin{equation}
    d_{ij} = \sum_{k=1}^{k=n} \sqrt{(a_1 - b_1)^2 + \ldots + (a_n - b_n)^2}
\end{equation}

It is preferred to use phenotypic distance, but also genotypic distance can be used as a measure of diversity e.g., Hamming distance for binary presentations. Distance \( d \) is defined here as Euclidean distance, see Equation (6), calculated from one individual to every other member of the population, including with itself. With the use of fitness sharing, members with a lower fitness value have increased probability of becoming parents when compared with the basic algorithm. This way we have a larger gene pool which directly increases the diversity of the population.

2.3.2. Crowding

This method relies on the fact that offspring are likely to be similar to their parents. In this algorithm, the parent population is evaluated and then randomly paired. Each pair produces two offspring by recombination, after which offspring are mutated and then evaluated. If parents are denoted as \( p \) and offspring as \( o \), distances: \( d(p_1, o_1) \), \( d(p_1, o_2) \), \( d(p_2, o_1) \) and \( d(p_2, o_2) \) between parents and offspring need to be calculated. After that competition pairs are identified.

\[
    [d(p_1, o_1) + d(p_2, o_2)] \leq [d(o_1, o_2) + d(p_2, o_1)],
\]

If Equation (7) is true, the competition is between \( p_1 - o_1 \) and \( p_2 - o_2 \), otherwise, the competition is between \( p_1 - o_2 \) and \( p_2 - o_1 \). There is no competition between offspring, they only compete with the most similar parent. The competition is based on the fitness of the individual: individuals with higher fitness values stay in the population and losers are discarded.

2.3.3. Novelty Search

This method differs from most EAs in that, instead of tending to converge, novelty is a divergent evolutionary technique. It directly rewards novel behavior instead of progress towards fixed fitness, and thus introduces a constant pressure for finding new and original individuals.

The main idea is that instead of rewarding only the performance of an individual on an objective, novelty search rewards diverging from prior behaviors. It is usually used in combination with fitness function as an additional measure for evaluating solutions \cite{23–25}, with the purpose of preserving a set of solutions for the next generation.

In this paper, we implemented such an approach that utilizes both novelty and fitness function, by copying only one, the best, individual from the current generation directly to the next generation. This enables us to keep the memory trace of quality individuals trans-generationally.

Tracking novelty of the solution requires a small change to any evolutionary algorithm, aside from adding a novelty metric along with a fitness function. This metric evaluates how far away new individuals are from the rest of the population and its predecessors in so-called behavior space, or space of individual behaviors.

It should measure sparseness of a point in behavior space, with denser clusters of visited points being less novel, and thus receiving a lesser reward. A method used for measuring the sparseness of the point is to measure the average distance to \( k \)-nearest members of that point. If this value is large, the space is sparse; if the value is small the area is dense, and hence the novelty level is low. The sparseness \( \rho \) at a point \( x \) is defined as:

\[
    \rho(x) = \frac{1}{k} \sum_{i=0}^{k} \text{dist}(x, \mu_i),
\]
where \( \mu_i \) is the \( i \)th nearest neighbor of \( x \) with respect to the distance metric \( \text{dist} \), which is a measure of the distance between two individuals and is domain-dependent. If novelty is above a certain threshold, then this individual is included in the permanent archive of prior solutions in behavioral space.

The current generation together with the archive is the search space explored and that way search gradient is directed towards new instead of towards a specific objective of a fitness function. The procedure of selecting and creating offspring remains the same in a standard evolutionary algorithm, but now with more novel instead of fitter individuals having higher chances of being selected in the parent population.

2.4. Selection

The parent selection process depends on the used diversity maintenance method. The novelty algorithm evaluates the current population and the archive, attempting to maximize diversity. The fitness sharing algorithm uses the roulette wheel algorithm, but instead of relative fitness values, each member has assigned a shared fitness value. For the crowding algorithm, instead of the roulette wheel algorithm, every member of the population is randomly chosen and paired with another random member to get parent pairs.

The survivor selection process depends on the age-based replacement for the novelty and the fitness sharing algorithms. The offspring replace their parents with the use of elitism where one elite member is preserved in the population. The crowding algorithm uses a fitness-based replacement. The offspring compete with the most similar parent for their place in the population.

2.5. Variation

The variation operators used in the algorithms are arithmetic recombination and nonuniform mutation. During the recombination, each pair of parent genes has a separate probability of 70% for successful recombination. If the recombination occurs, new genes have values equal to:

\[
z_i = \alpha x_i + (1 - \alpha) y_i,
\]

where \( x_i \) and \( y_i \) are parent gene values and \( \alpha \) is a random real number in the range \([0, 1] \).

Mutation happens on each gene separately with a probability of 50%. If the mutation occurs targeted gene value changes by a random amount:

\[
a_i = a_i (1 - \beta + 2\beta \cdot \text{rand}),
\]

where \( \text{rand} \) is a random number in the range \([0, 1] \) and \( \beta \) represents the desired intensity of mutation. The value of \( \beta \) is set to 0.5 in every following algorithm.

3. Results

Matlab environment is used to run the simulations. In order to compare different versions of the algorithm, each algorithm was simulated 1000 times. Testing results are presented using histograms and convergence diagrams and shown in Figures 3 and 4, both for the 2D and 3D cases, respectively.
wheel algorithm, every member of the population is randomly chosen and paired with another random member to get parent pairs.

The survivor selection process depends on the age-based replacement for the novelty and the fitness sharing algorithms. The offspring replace their parents with the use of elitism where one elite member is preserved in the population. The crowding algorithm uses a fitness-based replacement. The offspring compete with the most similar parent for their place in the population.

2.5. Variation

The variation operators used in the algorithms are arithmetic recombination and nonuniform mutation. During the recombination, each pair of parent genes has a separate probability of successful recombination. If the recombination occurs, new genes have values equal to:

\[
    z_j = \alpha x_j + (1-\alpha) y_j
\]

where \(x_j\) and \(y_j\) are parent gene values and \(\alpha\) is a random real number in the range \([0,1]\).

Mutation happens on each gene separately with a probability of 50%. If the mutation occurs, the targeted gene value changes by a random amount:

\[
    a_j = a_j (1-\beta + 2\beta \cdot r_{\text{rand}})
\]

where \(r_{\text{rand}}\) is a random number in the range \([0,1]\) and \(\beta\) represents the desired intensity of mutation. The value of \(\beta\) is set to 0.5 in every following algorithm.

3. Results

Matlab environment is used to run the simulations. In order to compare different versions of the algorithm, each algorithm was simulated 1000 times. Testing results are presented using histograms and convergence diagrams and shown in Figures 3 and 4, both for the 2D and 3D cases, respectively.

(a) 

(b) 

(c) 

(d) 

Figure 3. 2D case: (a) basic algorithm; (b) novelty search; (c) fitness sharing; (d) crowding.
Each histogram represents how many times an algorithm found a trajectory in a particular length range. Convergence diagrams show the convergence of average lengths from the discovered trajectories in a specific generation that does not collide with the obstacles.

Better performance is for those histograms with the bars grouped in the left region—meaning shorter, and thus fitter, individuals are found with increased frequency compared to those histograms whose values are grouped on the right side.

The average length itself is not of crucial importance since a number of poor performers might influence the quality of the whole population. At the same time, the algorithm might find a very fit, even the best individual, while the average quality is low. The average length is included in the figures in order to illustrate the overall performance of the algorithm.

In the table presented above, the bold fields denote the best performance for the given algorithm and fitness sharing method. By simple summation of bold fields, the novelty outperforms other approaches.

It is important to note though, that not all fields have the same importance. For instance, column 5 shows the least fit individual from the population which is definitely less important compared to the best solution found by a certain method. Simple counting of bold fields is thus not the ultimate measure of the quality of a certain method, but has to be rather used as an indication of the quality and interpreted carefully based on additional analysis of the data provided in Table 1.

**Figure 4.** 3D case: (a) basic algorithm; (b) novelty search; (c) fitness sharing; (d) crowding.
Each histogram represents how many times an algorithm found a trajectory in a particular length range. Convergence diagrams show the convergence of average lengths from the discovered trajectories in a specific generation that does not collide with the obstacles.

Better performance is for those histograms with the bars grouped in the left region—meaning shorter, and thus fitter, individuals are found with increased frequency compared to those histograms whose values are grouped on the right side.

The average length itself is not of crucial importance since a number of poor performers might influence the quality of the whole population. At the same time, the algorithm might find a very fit, even the best individual, while the average quality is low. The average length is included in the figures in order to illustrate the overall performance of the algorithm.

In the table presented above, the bold fields denote the best performance for the given algorithm and fitness sharing method. By simple summation of bold fields, the novelty outperforms other approaches.

It is important to note though, that not all fields have the same importance. For instance, column 5 shows the least fit individual from the population which is definitely less important compared to the best solution found by a certain method. Simple counting of bold fields is thus not the ultimate measure of the quality of a certain method, but has to be rather us as an indication of the quality and interpreted carefully based on additional analysis of the data provided in Table 1.

### Table 1. Evolutionary algorithm testing results.

| Algorithm          | Number of Simulations | Number of Unsuccessful Simulations | Shortest Trajectory | Longest Trajectory | Average Length of Trajectories |
|--------------------|-----------------------|-----------------------------------|---------------------|--------------------|--------------------------------|
| Basic 2D           | 1000                  | 153                               | 18.602              | 23.983             | 19.365                         |
| Novelty search 2D  | 1000                  | 0                                 | 18.536              | 19.512             | 18.871                         |
| Fitness sharing 2D| 1000                  | 0                                 | 18.517              | 20.311             | 18.832                         |
| Crowding 2D        | 1000                  | 10                                | 18.915              | 30.043             | 21.415                         |
| Basic 3D           | 1000                  | 0                                 | 15.407              | 135.151            | 18.612                         |
| Novelty search 3D  | 1000                  | 0                                 | 15.319              | 18.522             | 15.649                         |
| Fitness sharing 3D| 1000                  | 0                                 | 15.325              | 291.408            | 23.411                         |
| Crowding 3D        | 1000                  | 0                                 | 15.374              | 20.355             | 16.444                         |

Comparing the test results of the novelty and the fitness sharing 2D algorithms, we can see that they are similarly effective. Both algorithms found a trajectory without collision in every simulation, lengths of discovered trajectories are placed in a narrow span, and the convergence diagrams are similar.

Unlike the previous two algorithms crowding 2D algorithm performed less consistent in terms of quality of results. It failed to find trajectories without a collision in 1% of simulations, lengths of discovered trajectories are approximately 13% longer on average, and convergence to the optimal solution is significantly slower.

Regarding the 3D versions of the algorithms, the novelty algorithm gave the best results with very fast convergence. Testing results of the fitness sharing algorithm are also comparably good as in the 2D example.

In 7.4% of simulations, the algorithm got trapped in suboptimal regions of the search space, with the consequence of the dissipated quality of solutions. Lengths of the trajectories, not shown in the histogram, are in the range $\langle 4, 30 \rangle$. If those simulations were declared unsuccessful and were omitted from the calculation of the average length, the average length would be 1.5659, which is close to the novelty version of the algorithm.
In this scenario, the crowding algorithm was able to find feasible solutions to the path planning problem, without getting trapped in suboptimal regions, but the average length is still larger than that of trajectories provided by the novelty algorithm.

We can also conclude that the basic algorithm, the one that does not include any DM is of lesser quality compared to the other three instances of the algorithm where DM is included. This is an expected outcome since diversity is very important in an evolutionary context.

4. Simulation

Tracking of the trajectory is simulated in both 2D the 3D environment. In order to implement the tracking, inverse kinematics of the 3 DOF robot is solved. Dimensions used to solve the problem trigonometrically are shown in Figure 1.

The process of the evolution of trajectories is illustrated in Figure 5. Fifty individuals are evolved over a number of generations searching for the best solution for the given environment. The whole population is illustrated on the left. In the middle, the best individual from the current population is shown. The right side presents fitness vs. time or generations for the best individual from the current population.

The trajectory is obtained with the use of the novelty 3D algorithm since this algorithm gave the best results for this problem. The optimized polynomial function of the trajectory is discretized in 100 sections, which gave us points that make the polynomial curve. Each point has coordinates that represent external coordinates of robot position. This way, simulation comes down to point-to-point robot positioning which can be easily implemented onto industrial robots. Figure 6 shows the robot following the trajectory at the beginning, after 30%, after 60%, and at the end of the process.

Figure 5. Left: population of trajectories. Middle: best individual. Right: Fitness vs. generation. Top: 2D case, bottom: 3D case.
Figure 5. Left: population of trajectories. Middle: best individual. Right: Fitness vs. generation. Top: 2D case, bottom: 3D case.

The trajectory is obtained with the use of the novelty 3D algorithm since this algorithm gave the best results for this problem. The optimized polynomial function of the trajectory is discretized in 100 sections, which gave us points that make the polynomial curve. Each point has coordinates that represent external coordinates of robot position. This way, simulation comes down to point-to-point robot positioning which can be easily implemented onto industrial robots. Figure 6 shows the robot following the trajectory at the beginning, after 30%, after 60%, and at the end of the process.

Figure 6. Robot motion along the trajectory found by the novelty algorithm. Top left start configuration, bottom right end configuration.

5. Conclusions

The results show that diversity maintenance methods have an impact on the overall fitness of the end solution found, what is expected. The basic version of the algorithm performed less successfully compared to all the three other versions with diversity maintenance included.

Both for the two-dimensional and three-dimensional environment, the novelty algorithm gave solutions that are consistent and converge fast towards the optimal solution.

The fitness sharing algorithm gave comparably good results for the two-dimensional environment but had certain problems with the three-dimensional. The algorithm got trapped in the local optimum in 7.4% of the simulations.

Even though it found trajectories without collisions, the length of those trajectories was such that those simulations could be considered unsuccessful. With the dismissal of those outlying solutions, fitness sharing algorithm has results that are getting close to those of the novelty algorithm. Crowding algorithm gave evidently the least quality results out of the tested algorithms, for both problem domains.

Looking at the results, of this and the past research, the reason for the difference in the performance is not straightforward to explain. It originates from the diversity maintenance methods, which include parent selection models. The novelty and fitness sharing algorithms use a lot more stochastic models when choosing the parent population, which results in greater freedom in the exploration of the environment.

Crowding algorithm is based on the random pairing of parents from the population, and then offspring is compared to parents and fitter individuals are preserved. In the case where the population is sparsely distributed in the large search space, this can become a limitation, since good parents might not be selected to create offspring, and thus their genetic material disappears in the evolutionary process.

Diversity maintenance is not straightforward to implement and requires a significant amount of experimental work to tune the parameters for the algorithm to work efficiently.

It is then a question if this additional work is justified if the increase of the end solution quality is not significant.
The main conclusions regarding the application of different diversity maintenance methods to the problem of path planning drawn from this study are that novelty search equals fitness sharing for simpler scenarios of path planning. The strength of the novelty search arises with the complexity of the environment and path to be found. Also, diversity maintenance methods generally positively influence the quality of solutions to the problem of path planning, when compared to the basic algorithm.

Author Contributions: Conceptualization, P. Č.; investigation, L. Č.; methodology, P. Č.; software, L. Č.; supervision, P. Č.; validation, L. Č.; writing—original draft, L. Č.; writing—review and editing, P. Č. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by grant no 4192 by the Croatian Ministry of Sciences, Education, and Sports.

Acknowledgments: Petar Čurković would kindly acknowledge the hospitality of professor Hod Lipson from Columbia University, NYC, and the Fulbright postdoctoral research grant during which the main idea for this paper emerged.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Latombe, J.C. Robot Motion Planning; Kluwer Academic Publishers: Boston, MA, USA, 1991.
2. Sudhakara, P.; Ganapathy, V.; Priyadharshini, B.; Sundaran, K. Obstacle Avoidance and Navigation Planning of a Wheeled Mobile Robot using Amended Artificial Potential Field Method. Procedia Comput. Sci. 2018, 133, 998–1004. [CrossRef]
3. Duchon, F.; Babinec, A.; Kajan, M.; Beno, P.; Florek, M.; Fico, T.; Jurišica, L. Path Planning with Modified Star Algorithm for a Mobile Robot. Procedia Eng. 2014, 96, 59–69. [CrossRef]
4. Ammar, A.; Bennaceur, H.; Chaari, I.; Koubâa, A.; Alajlan, M. Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments. Soft Comput. 2016, 20, 4149–4171. [CrossRef]
5. Wang, Z.; Cai, J. Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities. Prog. Nucl. Energy 2018, 109, 113–120. [CrossRef]
6. Wang, W.; Deng, H.; Wu, X. Path planning of loaded pin-jointed bar mechanisms using Rapidly-exploring Random Tree method. Comput. Struct. 2018, 209, 65–75. [CrossRef]
7. Zhang, H.; Su, S. A hybrid multi-agent Coordination Optimization Algorithm. Swarm Evol. Comput. 2019, 51. [CrossRef]
8. Čurković, P.; Jerbić, B.; Štipančić, T. Swarm-Based Approach to Path Planning Using Honey-Bees Mating Algorithm and ART Neural Network. Solid State Phenom. 2009, 147–149, 74–79. [CrossRef]
9. Canny, J. The Complexity of Robot Motion Planning; MIT press: Cambridge, MA, USA, 1988.
10. Xue, Y. Mobile Robot Path Planning with a Non-Dominated Sorting Genetic Algorithm. Appl. Sci. 2018, 8, 2253. [CrossRef]
11. Chen, X.; Kong, Y.; Fang, X.; Wu, Q. A fast two-stage ACO algorithm for robotic path planning. Neural Comput. Appl. 2013, 22, 313–319. [CrossRef]
12. Contreas-Cruz, M.A.; Ayala-Ramirez, V.; Hernandez-Belmonte, U.H. Mobile robot path planning using artificial bee colony and evolutionary programming. Appl. Soft Comput. 2015, 30, 319–328. [CrossRef]
13. Lee, J.; Kim, D.W. An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. Inf. Sci. 2016, 332, 1–18. [CrossRef]
14. Čurković, P.; Jerbić, B.; Štipančić, T. Coordination of Robots With Overlapping Workspaces Based on Motion Co-Evolution. Int. J. Simul. Model. 2013, 12, 27–38. [CrossRef]
15. Čurković, P.; Jerbić, B.; Štipančić, T. Co-Evolutionary Algorithm for Motion Planning of Two Industrial Robots with Overlapping Workspaces. Int. J. Adv. Robot. Syst. 2013, 55, 1–11.
17. Oliveto, P.S.; Paixão, T.; Pérez Heredia, J.; Sudholt, D.; Trubenová, B. How to Escape Local Optima in Black Box Optimisation: When Non-elitism Outperforms Elitism. *Algorithmica* 2018, 80, 1604–1633. [CrossRef] [PubMed]

18. Gee, S.B.; Qiu, X.; Tan, K.C. A Novel Diversity Maintenance Scheme for Evolutionary Multi-objective Optimization. In Proceedings of the 14th International Conference on Intelligent Data Engineering and Automated Learning—IDEAL, Hefei, China, 20–23 October 2013.

19. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin/Heidelberg, Germany, 2007.

20. Lehman, J.; Stanley, K.O.; Miikkulainen, R. Effective Diversity Maintenance in Deceptive Domains. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013.

21. Lehman, J.; Kenneth, O. Stanley. Novelty search and the problem with objectives. In *Genetic Programming in Theory and Practice IX (GPPT 2011)*; Chapter 3; Springer: Berlin/Heidelberg, Germany, 2011; pp. 37–56.

22. Kong, F.; Jiang, J.; Huang, J. An adaptive Multi-Swarm Competition Particle Swarm optimizer for Large-Scale Optimization. *Mathematics* 2019, 7, 521. [CrossRef]

23. Lehman, J.; Stanley, K.O. Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evol. Comput.* 2011, 19, 189–223. [CrossRef] [PubMed]

24. Fister, I.; Iglesias, A.; Galvez, A.; Ser, J.D.; Osaba, E., Jr.; Fister, I., Jr.; Perc, M.; Slavinec, M. Novelty search for global optimization. *Appl. Math. Comput.* 2019, 347, 865–881. [CrossRef]

25. Risi, S.; Hughes, C.H.; Stanley, K.O. Evolving Plastic Neural Networks with Novelty Search. *Adapt. Behav.* 2010, 18, 470–491. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).