A neutral evolutionary path-planner

Eivind Samuelsen, Kyrre Harald Glette, and Kazi Shah Nawaz Ripon

Department of Informatics, University of Oslo, Norway
{eivinsam,kyrrehg,ksripon}@ifi.uio.no

Abstract: This paper explores methods for path-planning using evolutionary algorithms. Inspired by research on neutral mutations in evolutionary algorithms, we propose an algorithm based on the idea of introducing redundancy in the solutions, adding explicit neutrality to the evolutionary system. The algorithm introduce explicit neutrality by evolving roadmaps rather than single paths. Since some of the mutation and crossover operators used in conventional evolutionary path-planners are not well suited for this representation, appropriate evolutionary operators will also be explored. The performance of this algorithm on shortest distance path planning problems is compared to a known good genetic algorithm in three different static environments.

Keywords: genetic algorithm, neutrality, path-planning, roadmap

1 INTRODUCTION

Path planning is the problem of finding an optimal obstacle-free path through an environment. In a shortest distance problem, the optimal path is the one going from a start point to a goal point in the shortest possible distance travelled. Many methods have been proposed for solving this optimization problem. They all have certain trade-offs between planning time, robustness, requirements on environment representation and so on. An overview of the most common no-evolutionary methods can be found in [1].

Probabilistic roadmaps (PRMs) are one of the more common methods in path planning. A PRM samples random points in order to simplify the problem and scale well with environment dimensionality. However, it will seldom find optimal solutions unless the amount of samples taken is made impractically large. They also have problems with finding solutions at all in narrow passages and maze-like environments, a problem that to some extent can be remedied by sampling adaptively [2].

Several path-planning methods using evolutionary algorithms (EAs) have also been proposed. A straightforward approach to the shortest distance problem is presented in [3]. Once a good set of solutions has been found by an EA path-planner, it can easily adapt to changes in the environment simply by running the a few more iterations of the algorithm on the updated environment data. However, finding a good set of initial solutions from a set of random solutions can take some time. A remedy for this is to initialize the population with results from a PRM planner has been proposed in [4].

There has recently been some interest in EAs using neutrality and neutral mutations. Neutrality is having redundancy or extra information in the chromosomes so that they can change in ways that do not affect fitness. It is claimed that this can be greatly beneficial to EAs, making them both converge faster to the optimum and also escape local optima more easily. This is inspired by similar theories in biological evolution and some experimental findings, though many of these findings are considered inconclusive by others. A summary of this research can be found in [5].

Although the effects of neutrality in EAs in general are still inconclusive, it is an interesting avenue to explore in solving path-planning problems. One such method is proposed in this paper. Instead of trying to evolve good paths directly, this method tries to evolve good roadmaps for finding paths. We will rank the roadmaps only by the best path found in them by a graph traversal algorithm. Since this ranking does not consider the points in the roadmap that are not part of the resulting path it has explicit neutrality.

2 PROPOSED METHOD

Starting out with a set of PRMs as initial population, our algorithm evolves these roadmaps over many generations. We call this method a roadmap evolver. Roadmaps well suited to find a good path are more likely to survive to the next generation. The nodes that are not on a roadmap's best path can mutate without affecting the roadmap's fitness. This can enable the population to escape local optima more easily.

2.1 Representation and evaluation function

Each chromosome is a set of floating-point vectors that together with the start point and the goal constitute the nodes, or milestones, of a roadmap. If one can draw a straight line between two nodes without intersecting any obstacle, they are connected. A graph traversal algorithm is run to evaluate the chromosomes, finding the best path through it from the start to the goal node.

When an optimal path through the roadmap has been found, the fitness of that path is taken as the fitness of the roadmap. In the experiments done in this paper the path's are optimized for the shortest path. So the fitness of the path is the path's curve length, and the graph traversal is done by A*.

Since not all nodes are necessarily visited during the traversal, the connectivity of the roadmap graph is done dynamically during the graph traversal in order to reduce the number of connectivity checks needed. Still, in general, $O(n^2)$ connectivity checks will be needed in the traversal, where n is the number of nodes in the roadmap.

2.2 Evolutionary operators

The algorithm proposed has a single crossover operator and three mutation operators: nudge, insert and reduce size.

In order to emphasize average-sized children we use a crossover operator similar to uniform crossover: First each point in parent A is given to either child A or child B with equal probability. Then the points in parent B are distributed in the same way. The size of each child is the sum of n coin flips, so it has a binominal distribution. The chromosomes are sets of points and thus inherently unordered. Therefore, there is no need to try to maintain any ordering.

The nudge mutation goes through all the points in a chromosome. Each point has a probability P_{nudge} of being displaced a small distance in some random direction, but only if that displacement does not move the point into or over an obstacle: A normal-distributed vector is added to the point if the straight line between the original point and the point plus the random vector is obstacle-free. If this fails, it is reattempted up to 6 times with a new random vector with increasingly narrow normal distribution. If all attempts fail that point is left unchanged.

The insert mutation tries to add a single point to an individual. A candidate point is generated randomly with uniform probability within the bounds of the environment. If the candidate is obstructed, a short random walk is performed until an unobstructed point is found, or a maximum of three steps are taken. If the candidate is still obstructed, we try it up to 5 more times with different random start points. If no feasible candidate has been found, the operator leaves the individual unchanged.

The reduce size mutation goes through all the points in a chromosome and removes that point with a probability P_{remove} . Thus, on average $n \times P_{\text{remove}}$ points are removed from the set, where n is the original size of the set.

2.3 Initial population and evolutionary process

The population is initialized with N PRMs with an average of N_g non-obstructed nodes in uniform distribution over the environment. If none of the roadmaps give a feasible path from start to a goal then N_g is increased by a constant and N new PRMs are created. This is repeated until at least one initial feasible solution is found.

Each generation is generated as follows: N new individuals are created by crossover. The parents for each crossover operation are selected by simple tournament selection with a tournament size of 2. This is repeated until two different parents have been selected.

Table 1: Average run-time, including initialization

Environment	Reference path-planner (per iteration)	Roadmap evolver (per iteration)
"Rocks"	19.0s (10.6ms)	78.3s (174.0ms)
"House"	49.0s (27.2ms)	163.6s (363.6ms)
"Spirals"	25.7s (14.3ms)	131.0s (291.1ms)

All the new individuals are then subjected to mutation. The nudge mutation is performed once, then the insert mutation is performed once with probability P_{insert} . Finally the reduce size mutation is performed. After that, new population is merged with the old. To reduce the population size back to normal, N one-round tournaments are held, and the losing individuals are removed.

Each new individual has P_{insert} points added on average, while $n \times P_{\text{remove}}$ points are removed. This helps the average chromosome size to stay relatively stable, while allowing for variations in order to adapt to different environments.

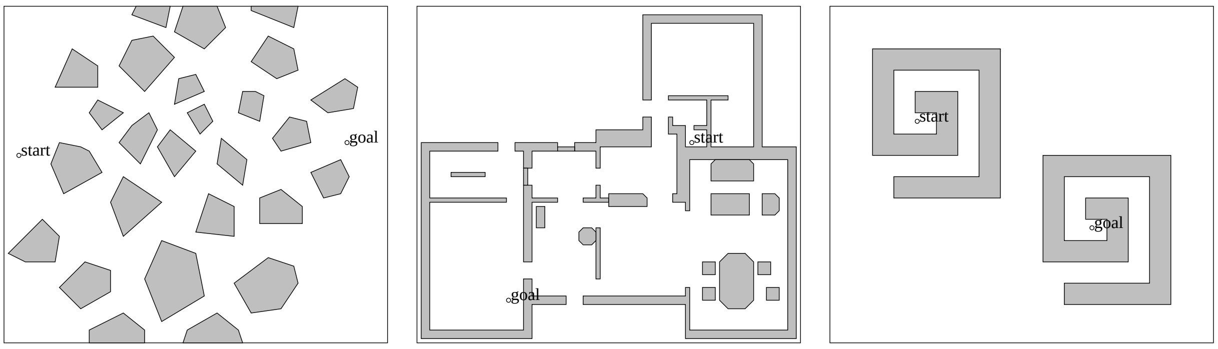
3 RESULTS AND DISCUSSION

The algorithm has been tested against a reference path-planner in different environments. The reference algorithm used is similar to the one described in [4], differing mainly in the parameters and selection operators used. It is run with a population size of 40 and a tournament-based selection. It should be noted that the parameters and implementation of the reference algorithm are not fine-tuned or optimized, and can therefore only serve as a general guideline to the performance of a reasonably robust evolutionary path-planner.

The roadmap evolver was run with a population size of 35 and an average initial number of nodes per chromosome of 35 or more. Mutation parameters were set to $P_{\text{nudge}} = 0.2$, $P_{\text{insert}} = 0.4$ and $P_{\text{remove}} = 0.05$.

Three different environments, shown in Fig.1 are tested. Each environment has different characteristics: The "rocky" environment consists of a number of scattered convex shapes, and has a large number of intersecting local optima. The second environment imitates the interior of a single-floor home, and has wall- and corridor-like obstacles, leading to a few, far-apart local optima. The third environment contains two spiral structures and has only one hard-to-find optimum.

The result of 100 runs of each algorithm in each environment is shown in Fig.2. The reference algorithm is run for 1800 iterations, while the roadmap evolver is run for 450 iterations. The large difference in number of iterations is due to differences in run-time per iteration and to differences in how



(a) The "rocks" environment (b) The "house" environment (c) The "spirals" environment

Figure 1: The different environments tested

quickly the algorithms converge. The average run-times are shown in Table 1. The average iteration time of the roadmap evolver is more than ten times that of the reference algorithm.

The reference algorithm shows very good performance in the "house" environment, with a good start and little variation. In "rocks" it converges quickly, but seems to get stuck in local optima, because the variation between runs stays high. In the last environment it closes in on the optimum with low variation, but at a very slow pace.

The roadmap evolver has good performance in the "rocks" environment, converging quickly towards a near-optimal solution quickly and with little variation between runs. In the other two environments it closes in on the optimal solution at a good average pace, but the pace is very uneven, with large variations in fitness between runs for many generations.

Compared to the reference algorithm in terms of generations only, the roadmap evolver performs as good or better in both "rocks" and "spirals" environments, but needs more generations to reliably produce near-optimal solutions in the "house" environment. In "rocks" it seems to avoid local optima better, while in "house" it shows signs of jumping out of local optima it have been temporarily stuck in. Note that both algorithms fail to ever find the globally optimal path in the "rocks" environment.

The good performance of the reference algorithm on "house" might be explained by the way the populations are initialized - the reference algorithm guarantees N different feasible solutions, while the roadmap evolver only guarantees at least one.

4 CONCLUSION

In this paper, we proposed a path-planning method using a genetic algorithm that has explicit neutrality. The algorithm was compared to an existing genetic path-planning algorithm that has no neutrality. The proposed algorithm closes in on the optimal solution in comparatively few generations, and shows signs of both avoidance and escape of local minima in a small selection of tested environments. However, run-time performance is poor, most likely because of comparatively high complexity in the fitness function.

The algorithm did display properties expected of a neutral evolutionary system. For future work it would be of interest to examine whether these properties are displayed in other situations too, such as dynamic or partially unknown environments, or path-planning problems with multiple objectives. Initializing the population in the same way as the reference algorithm might increase performance in early generations. Another possibility is developing other different and possibly more efficient neutral path-planning EAs and examine if the same properties appear in them.

REFERENCES

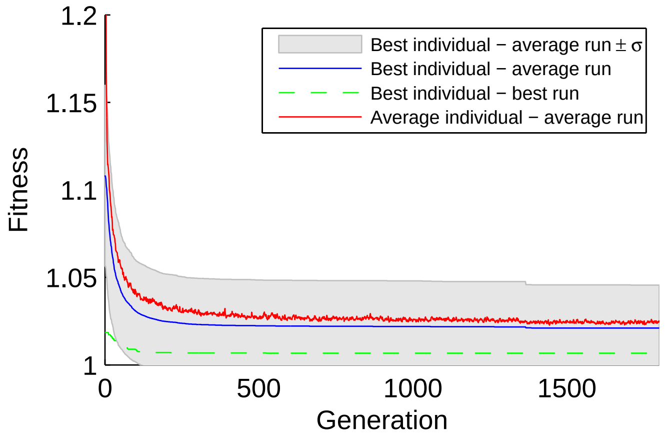
[1] S. LaValle, *Planning algorithms*. Cambridge Univ Pr, 2006.

[2] V. Boor, M. Overmars, and A. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1018 –1023 vol.2, 1999.

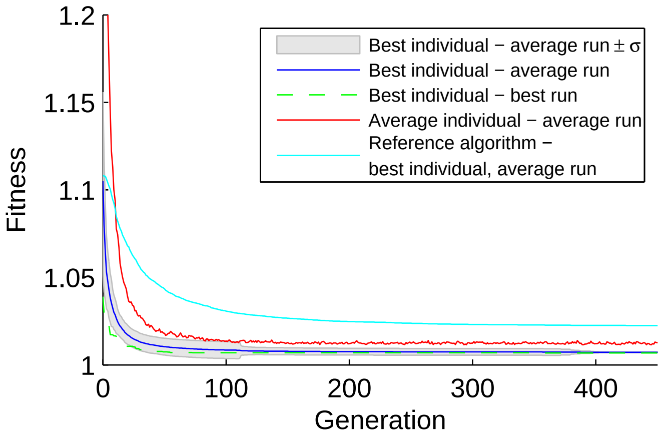
[3] H. Lin, J. Xiao, and Z. Michalewicz, "Evolutionary algorithm for path planning in mobile robot environment," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 211–216, IEEE, 1994.

[4] M. Naderan-Tahan and M. Manzuri-Shalmani, "Efficient and safe path planning for a mobile robot using genetic algorithm," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pp. 2091 –2097, may 2009.

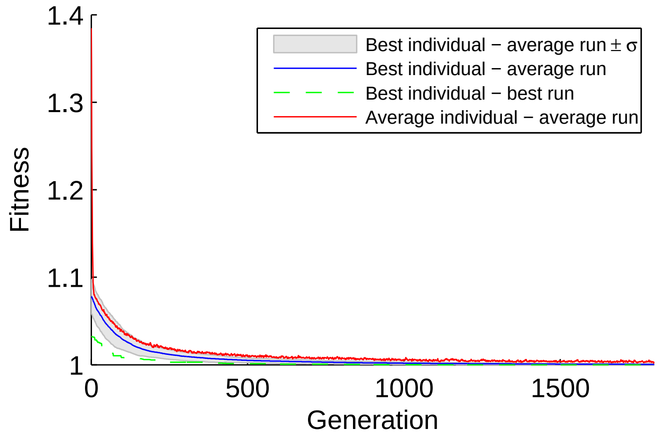
[5] E. Galván-López, R. Poli, A. Kattan, M. O'Neill, and A. Brabazon, "Neutrality in evolutionary algorithms... what do we know?," *Evolving Systems*, vol. 2, pp. 145–163, 2011. 10.1007/s12530-011-9030-5.



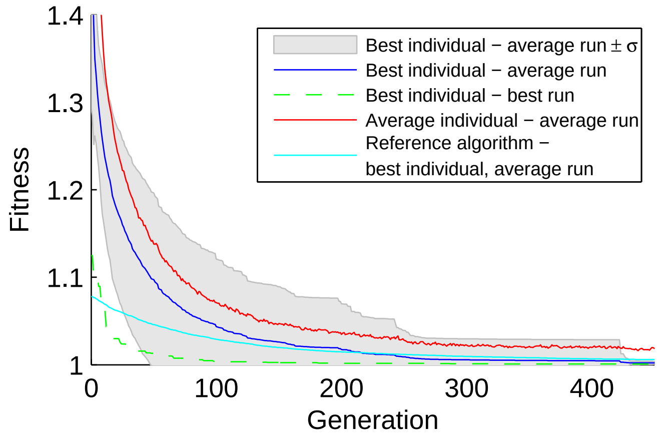
(a) "Rocks" environment - reference path-planner



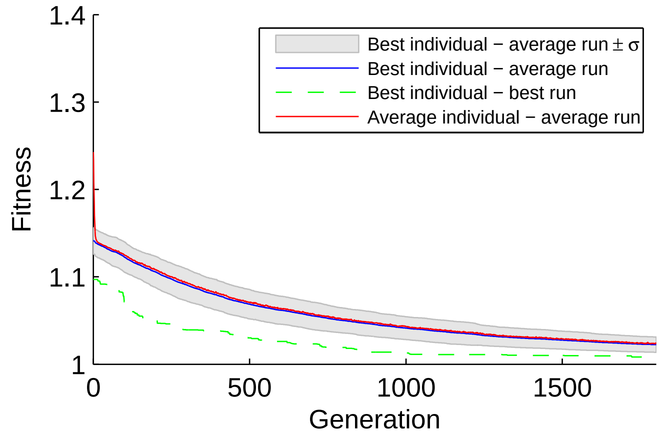
(b) "Rocks" environment - roadmap evolver



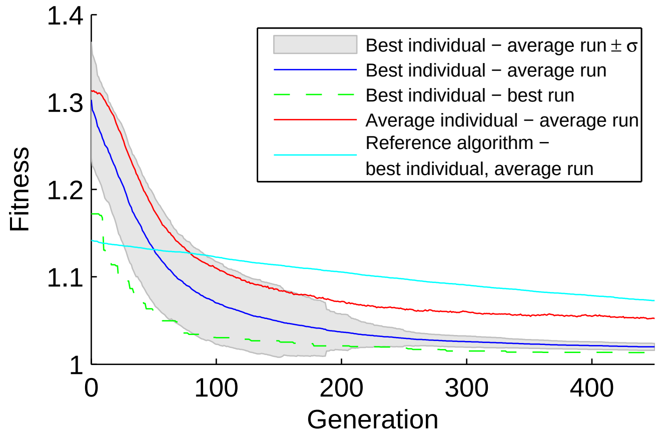
(c) "House" environment - reference path-planner



(d) "House" environment - roadmap evolver



(e) "Spirals" environment - reference path-planner



(f) "Spirals" environment - roadmap evolver

Figure 2: Fitness development for each algorithm in each environment. All fitness values are relative to the optimal solution as found by the vismap algorithm. The blue line is the average best fitness for each run. The light grey area around it signifies one standard deviation from that average. The red line is the average fitness of all feasible solutions in the current generation, averaged over all runs. The cyan line in the roadmap evolver plots shows the average best fitness for the reference algorithm for the same iterations.