INCREASING THE CONNECTIVITY OF PROBABILISTIC ROADMAPS VIA GENETIC POST-PROCESSING

Giuseppe Oriolo [*] **Stefano Panzieri** [**]
Andrea Turli [**]

[*] *Dip. di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Eudossiana 18, 00184 Roma, Italy*
oriolo@dis.uniroma1.it
[**] *Dip. di Informatica e Automazione, Università di Roma Tre, Via della Vasca Navale 79, 00146 Roma, Italy*
panzieri@uniroma3.it, Andrea.Turli@libero.it

Abstract: With reference to the motion planning problem, we present a simple strategy for improving the connectivity of probabilistic roadmaps by genetic post-processing. In particular, our objective is to increase the roadmap density in narrow passages, where many of the existing probabilistic planners perform poorly. To this end, we associate to each individual (i.e., to each robot configuration) an easily computable fitness function based on the distance between disjoint components of the roadmaps. Straightforward selection, crossover and (possibly) mutation operators are then applied to improve the quality of the population. Numerical results in different workspaces, including a well-known benchmark, show the effectiveness of the proposed strategy. *Copyright* ©*2006 IFAC*

Keywords: Motion Planning, Probabilistic Roadmaps, Narrow Passages, Genetic Algorithms

1. INTRODUCTION

In recent years, the concept of Probabilistic Roadmap (PRM) has proved to be a powerful tool for motion planning in the presence of obstacles, particularly for robots with many degrees of freedom (Kavraki *et al.*, 1996; Nissoux *et al.*, 1999; LaValle and Kuffner, 2001). The basic idea of PRM-based planners is to generate random samples of the robot configuration space, check the samples for collision with obstacles in the workspace, and connect feasible samples through a roadmap which captures the connectivity of the collision-free configuration space.

PRM-based planners are only probabilistically complete, but perform very well on the average, chiefly in terms of time efficiency. However, some instances of the motion planning problem are extremely difficult [1]. A critical situation for PRM-based planners is the presence of *narrow passages* in the free configuration space. On the one hand, the small volume of these regions makes their sampling less likely. On the other, the knowledge of the obstacle geometry in the workspace provides no direct clue about the location and shape of these passages in the configuration space, especially when the latter is high-dimensional. Probabilistic techniques aimed at increasing the density of samples in narrow passages have been presented, e.g., in (Boor *et al.*, 1999; Hsu *et al.*, 1999).

[1] Indeed, it should be noted that the motion planning problem is PSPACE-hard (Reif, 1979) and becomes undecidable in the presence of frictional surfaces (Reif and Sun, 2003).

In this paper, we present a simple genetic strategy for increasing the connectivity of PRMs in critical areas. The idea is to use one of the existing PRM-based planner for a limited time so as to obtain a rough roadmap. If this contains unconnected components, it is taken as initial population for the genetic process. A fitness is then computed for each individual (i.e., for each robot configuration), based on the distance between the current and the other components. Straightforward selection, crossover and (possibly) mutation operators are then applied to improve the quality of the population. As experimental results show, the increased computational cost of our particular type of non-uniform sampling is compensated by the reduction in the number of disjoint components, ultimately resulting in a connected roadmap.

The paper is organized as follows. The formulation of the problem and our approach to the solution are outlined in Section 2. The proposed method is explained in detail in Section 3. Section 4 reports the results of planning experiments. A discussion on the genetic aspects of our techniques, given in Section 5, concludes the paper.

2. THE BASIC IDEA

We address the motion planning problem with reference to the classical formulation in (Latombe, 1991):

- Let A be the *robot*, i.e., a single rigid object or a collection of linked rigid objects moving in a Euclidean space W, called *workspace*, represented as $I\!R^3$;
- Let $B_1, \ldots, B_q$ be fixed objects distributed in W, called *obstacles*.
- Assume that the geometry of A, $B_1, \ldots, B_q$ and the locations of the B_i's in W are known. Assume also that A is *free-flying*, i.e., it is not subject to any kinematic constraint.

Given an initial and a goal pose (i.e., position and possibly orientation) of A in W, a collision-free path τ is seeked, i.e., a continuous sequence of poses of A which avoid contact with the B_i's and connect the initial with the desired pose. Failure should be reported if no such path exist.

Our approach prescribes two phases (Figure 1):

(1) In the first phase, we let an existing PRM-based planner run for a short time to obtain a rough roadmap. All such methods try to connect feasible samples by means of a *local planner*. If this roadmap contains unconnected components, it is taken as initial population for the next step. In our experiments we have used various planners, including the basic PRM (Kavraki *et al.*, 1996) and the Visibility PRM (Nissoux *et al.*, 1999).

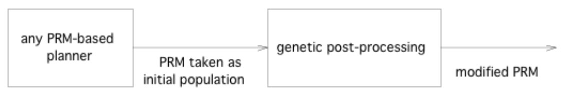


Fig. 1. The proposed two-step approach.

(2) In the second phase, a genetic reproduction process takes place. Each node of the roadmap is an individual of the initial population. A fitness function based on the distance between each node and the other components is used to sort the population before applying selection, crossover and (possibly) mutation operators. The process stops when the roadmap becomes connected or a maximum number of generations is reached.

A solution to the motion planning problem is then seeked by connecting the initial and final configuration to the roadmap.

In the next section, we explain the structure of the second phase.

3. GENETIC POST-PROCESSING

The proposed genetic algorithm works on a population of *individuals* that represent samples of the free configuration space. Each individual is coded through a single *chromosome* of *genes*, i.e., a vector of generalized coordinates corresponding to the particular configuration. The initial population is defined as the roadmap provided by the first step in Figure 1.

A pseudocode of the algorithm follows.

```
the GENETIC POST-PROCESSING algorithm
  $\{P, N\} \leftarrow$ INIT_POP
  $\{P, M\} \leftarrow$ CONNECT($P$)
  $i \leftarrow 1$
  while $M > 1$ and $i \leq I_{\max}$ do
    $\{P, N\} \leftarrow$ FITNESS($P$, $N$, $M$)
    $\{P, N\} \leftarrow$ MERIT_CROSSOVER($P$, $K_{\rm cr}^{\rm m}$, $N_{\max}$)
    $\{P, N\} \leftarrow$ RANDOM_CROSSOVER($P$, $K_{\rm cr}^{\rm r}$, $N_{\max}$)
    $\{P, N\} \leftarrow$ MUTATION($P$, $K_{\rm mut}$, $N_{\max}$)
    $\{P, M\} \leftarrow$ CONNECT($P$)
    $i \leftarrow i + 1$
  return $P$
```

with:

P:	roadmap (population)
N:	number of individuals in P
M:	number of connected components in P
i:	iteration (generation) counter
$I_{\max}$:	maximum number of iterations
$N_{\max}$:	maximum number of individuals in P
$K_{\rm cr}^{\rm m}$:	number of individuals chosen for merit crossover
$K_{\rm cr}^{\rm r}$:	number of individuals chosen for random crossover
$K_{\rm mut}$:	number of random mutation attempts

In the following, we discuss in some detail the single procedures.

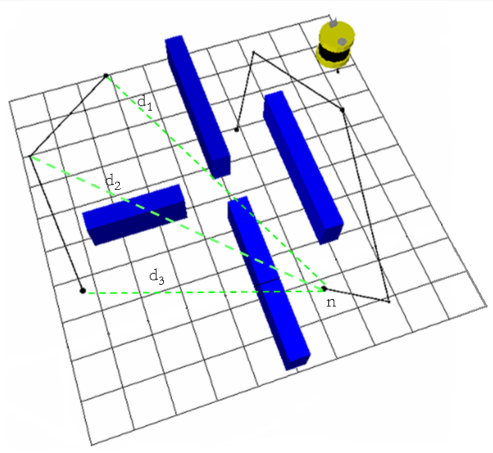


Fig. 2. The fitness of n is $d_3 = \min(d_1, d_2, d_3)$.

3.1 Initialization

The INIT_POP procedure creates the initial population P as a copy of the roadmap produced by the first step (one individual for each node of the roadmap) and computes the population size N. Then, the CONNECT function evaluates the number M of connected components in P; all possible connections so far undetected are investigated at this stage according to the local planner.

3.2 Fitness

The choice of the fitness function is critical because, in a genetic algorithm, it will orient the evolution of the individuals towards a generation with certain characteristics. In our case, we would like to generate configurations that increase the connectivity of the roadmap by sampling narrow passages. In the absence of knowledge about the geometry of the free configuration space, we simply chose to privilege nodes that reduce the distance between unconnected components. In practice, after determining which component of the roadmap each node belongs to, the FITNESS function computes the minimum distance between the node and the other components (Figure 2), and sorts the population in decreasing order of fitness. Note that the fitness function is symmetric, i.e., for each node with a certain fitness there is (at least) another node with the same value.

The effect of this definition of fitness is illustrated in Figure 3, which shows that nodes that close the gap between unconnected components (such as node 2) are preferred over others (such as node 1). In some sense, this fitness function acts as an attractive *potential field* (Khatib, 1986) during the genetic process, pulling new individuals towards the nearest connected components. Note that obstacle avoidance is not obtained by repulsive fields (we assume no knowledge of configuration space obstacles) but implicitly guaranteed by collision checking.

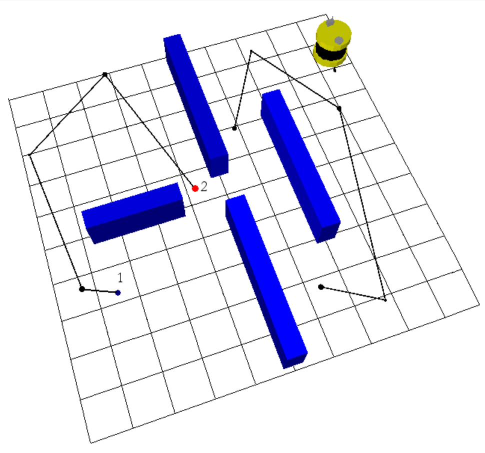


Fig. 3. The fitness of node 2, which reduces the distance between the two unconnected components, is larger than the fitness of 1.

3.3 Crossover

Once the population has been sorted according to the fitness function, crossover takes place. This step, which is aimed at generating new samples approaching unconnected components, takes place sequentially on two subsets of the population: first, on the best $K_{\mathrm{cr}}^{\mathrm{m}}$ individuals (MERIT_CROSSOVER), and then on $K_{\mathrm{cr}}^{\mathrm{r}}$ randomly chosen individuals among the remaining population (RANDOM_CROSSOVER). The objective of random crossover is to preserve the probabilistic nature of the roadmap.

The crossover operator on a generic node n works as follows. First, the component defining the fitness of n is identified (this is the unconnected component to which n is closer). Then, a new sample (a *son*) is randomly placed along the line connecting n to *each* node of that component (recall that we assume a free-flying robot, so that the local planner generates straight lines).

To be added to the population, each son must go through a validation step, which verifies if the following properties are satisfied:

(1) the son is collision-free;
(2) the son can be connected via a collision-free path to one of its *parents* by the local planner;
(3) if $N = N_{\max}$ (i.e., if the population has reached its maximum size), the fitness of the son, which can be computed once the new configuration has been connected to a parent, must be larger than that of the worst *leaf* (i.e., without sons) individual in the population.

While the necessity of the first two properties is self-evident, a comment is in order with respect to the last one. If $N = N_{\max}$, we enforce a 'one birth-one death' policy to contain the population size and guarantee its evolution toward the desired generation. However, to preserve the current degree of connectivity in the roadmap, the death of individuals with sons must be avoided. On the other hand, while the death of low-fitness leaves will not affect the roadmap connectivity, it will decrease its coverage of the free configuration space. For this reason, it is advisable to *merge* the genetically modified roadmap with the initial one before solving a planning query.

If a mutation step is included in the algorithm, all the sons generated during crossover that have not been validated because they are in collision are stored for possible later use (see next subsection).

3.4 Mutation

After crossover, an optional mutation (MUTATION) phase takes place. Unorthodoxly, however, in our algorithm mutation does not occur in individuals that belong to the population; instead, it involves only those crossover samples (sons) that have not passed the collision check. These samples will not be added to the population, but will be used for mutation before being discarded.

The idea is to take advantage of all sons in collision for performing a *bridge test*, as proposed in (Hsu *et al.*, 2003). In particular, denote the generic son in collision by q_1. A second configuration q_2 is generated in a random direction at a given (small) distance from q_1. If also q_2 is in collision, a new sample q_b is generated halfway along the segment connecting q_1 to q_2 (the *bridge*). At this point, q_b goes through a validation phase similar to the one described above, with a key difference: in the second step, since q_1 and q_2 are in collision, one tries to connect q_b to the parents of q_1. If q_b is validated, it is added to the roadmap; otherwise, it is discarded and a new son in collision is chosen for the bridge test.

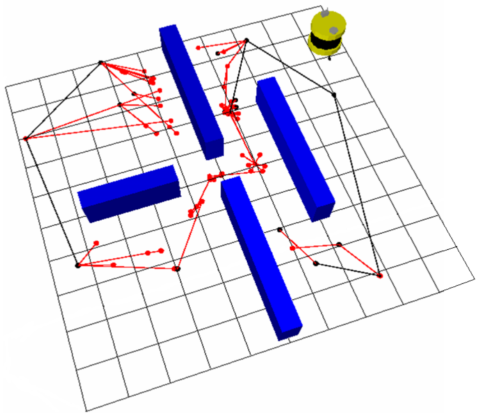


Fig. 4. A typical result of the proposed strategy. Nodes and links in red were added in the genetic post-processing phase.

Similarly to the crossover step, we have chosen to perform an additional mutation phase on a random base. In particular, K_{mut} random samples are generated from scratch and, if found to be in collision, used for performing a bridge test.

3.5 Connectivity test

The final step before starting a new iteration is to check if the newly added individuals allow a reduction in the number M of unconnected components, and to update the roadmap accordingly. This is done by the same CONNECT function used in the initialization phase. If $M = 1$, the algorithm stops. A typical result of the algorithm is shown in Figure 4.

4. PLANNING EXPERIMENTS

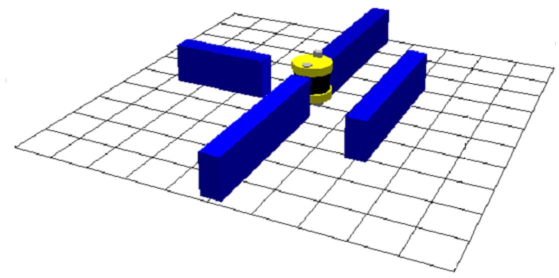
The proposed two-step strategy (PRM generation and genetic post-processing) has been implemented in C on a Pentium IV PC running at 2.80 GHz. In particular, we have integrated our algorithm in Move3D (Simeon *et al.*, 2001), a software platform developed at LAAS-CNRS and dedicated to motion planning[2].

In the presented experiments, the initial PRM is obtained by running the Visibility PRM (Nissoux *et al.*, 1999) for a limited time. This planner has the desirable property of generating a very compact roadmap, which is then given to our genetic post-processing phase. The results (in particular, the CPU time needed to reduce the number of components to one) are then compared with those obtained by letting the Visibility PRM run on its own. All the reported numerical data are averaged over 10 runs of the algorithm.

The first problem we consider is illustrated in Figure 5. The free configuration space of the mobile robot is a copy of the free workspace (a subset of $\mathbb{R}^2$), and includes a narrow passages between two walls. The Visibility PRM has been allowed 100 iterations, while the parameters for the genetic post-processing phase are the following: $N = 100$, $I_{max} = 500$, $K^{\text{m}}_{\text{cross}} = 15$, $K^{\text{r}}_{\text{cross}} = 5$ (see the algorithm pseudocode in Section 3). We have run the proposed algorithm in two versions: in the first, mutation is completely discarded, while in the second it is included with $K_{\text{mut}} = 20$ for the random mutation phase.

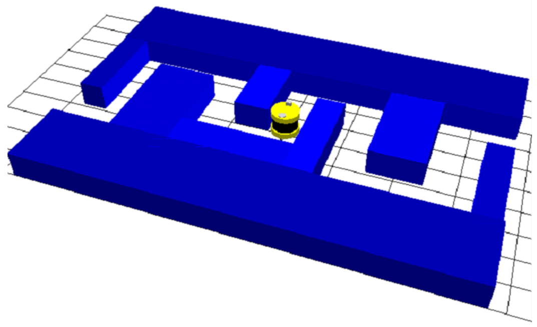
In the result table included in Figure 5, we report the results of the first phase, performed with the Visibility PRM method (V-PRM), and those of

[2] Move3D is at the origin of the product KineoWorks currently marketed by the company Kineo CAM (www.kineocam.com).



Parameter	V-PRM	GPP	GPP_Mut
roadmap size	8	100	61
random confs	100	820	764
free confs	40	105	69
LP calls	525	2868	1169
CC calls	3338	4096	5507
CPU time (sec)	0.32	2.61	1.82
components	2	1	1

Fig. 5. Results in a simple environment.



Parameter	V-PRM	GPP	GPP_Mut
roadmap size	15	200	200
random confs	100	1926	1179
free confs	42	165	148
LP calls	3361	9367	12841
CC calls	30685	40497	51892
CPU time (sec)	4.87	4.92	2.88
components	6	1	1

Fig. 6. Results in a corridor-like environment.

the genetic post-processing phase, without (GPP) and with (GPP_Mut) mutation. The reported parameters for each phase are: the number of nodes in the roadmap, the total number of random samples, the total number of samples that are not in collision, the number of calls to the local planner and to the collision checker, the CPU time and the number of components in the roadmap. While there are two unconnected components in the initial population, both GPP and GPP_Mut are able to join them, although the latter is slightly faster.

Note the following points.

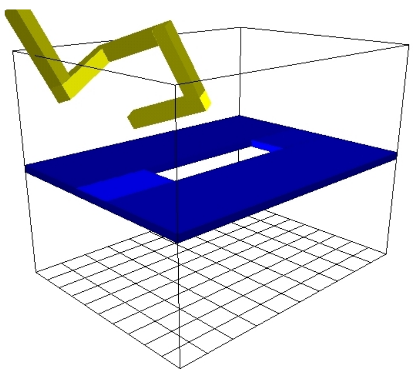
- The size of the initial roadmap is very small (8 individuals) thanks to the V-PRM property of including only configuration that are *guards* of their visibility domain (Nissoux *et al.*, 1999). The population is increased to its maximum size N by GPP, while GPP_Mut increases the number of individuals but does not reach the limit.
- The number of CC calls is much higher than the number of free configurations because the collision checker is also invoked for testing the feasibility of local paths connecting two free configurations.
- The V-PRM method on its own would have built a connected roadmap in 8.79 sec. The total time with our two-step strategy is 2.93 (without mutation) or 2.14 (with mutation) sec.

The environment chosen for the second experiment includes a long, tortuous corridor (see Figure 5). The Visibility PRM has been allowed 1000 iterations, while the parameters for the genetic post-processing phase are the following: $N = 200$, $I_{max} = 500$, $K^{\mathrm{m}}_{\mathrm{cross}} = 10$, $K^{\mathrm{r}}_{\mathrm{cross}} = 5$. For the version including mutation we have used again $K_{\mathrm{mut}} = 20$.

The results show that in this case both GPP and GPP_Mut were able to reduce the number of unconnected components from 6 to 1, increasing the population to its maximum size N in the process. In particular, the total CPU time is 9.79 (without mutation) or 7.75 (with mutation) sec. V-PRM on its own would have needed 15.68 sec to obtain the same result.

The third experiment refers to a ‘puzzle’ problem (Kavraki and Latombe, 1998), which has become a sort of benchmark for probabilistic planners (see Figure 7). The configuration space is 6-dimensional and includes two large free regions connected by a small passage, whose size depends on the relative dimensions of the hole and the object. In particular, we have used the same dimensions adopted in (Nissoux *et al.*, 1999).

The Visibility PRM has been allowed 3000 iterations, while the parameters for the genetic post-processing phase are the same of the second experiment. Both GPP and GPP_Mut were able to reduce the number of unconnected components from 4.5 (remember that results are averaged over 10 runs) to 1, increasing the population to its maximum size N in the process. The total CPU time is 117.83 (without mutation) or 105.43 (with mutation) sec, while V-PRM on its own would have found a solution in 279 sec, a result consistent with the CPU time indicated in (Nissoux *et al.*, 1999).



Parameter	V-PRM	GPP	GPP_Mut
roadmap size	80	200	200
random confs	3000	1366	1131
free confs	1934	5256	613
LP calls	38604	530	39312
CC calls	394923	79917	342085
CPU time (sec)	48.83	69.00	56.60
components	4.5	1	1

Fig. 7. Results in a puzzle example.

5. DISCUSSION AND CONCLUSIONS

We have presented a genetic approach aimed at increasing the connectivity of probabilistic roadmaps in narrow passages of the free configuration space. Our algorithm is intended as a post-processing step to be performed over poorly connected roadmaps. In particular, we associate to each individual (i.e., to each robot configuration) an easily computable fitness function based on the distance between disjoint components of the roadmaps. Straightforward selection, crossover and (possibly) mutation operators are then applied to improve the quality of the population. Numerical results in different workspaces, including a well-known benchmark, have shown that the addition of the proposed post-processing strategy is more efficient than the use of a PRM planner on its own, with a 50% decrease in cost (CPU time) on the average.

This work is to be considered as a preliminary investigation into the potentiality of genetic algorithms in probabilistic motion planning. Much work remains to be done, including a performance comparison with PRM planners especially devised for environments with narrow passages, e.g., those in (Boor *et al.*, 1999; Hsu *et al.*, 1999).

From a genetic perspective, the proposed algorithm is interesting in view of the peculiarity of the application to motion planning. In particular, while genetic algorithms typically aim at generating a *single* individual with outstanding features (e.g., when they are used for solving optimization problems), in our case the objective is to increase the quality of the *whole* population from a connectivity viewpoint. For example, this observation led us to forbid the death of low-fitness individuals whose presence was however necessary to preserve connectivity. Moreover, it could be interesting to allow the birth and growth of new unconnected components by eliminating the connection test in the validation phase.

REFERENCES

Boor, V., M.H. Overmars and A.F. Stappen (1999). The gaussian sampling strategy for probabilistic roadmap planners. In: *1999 IEEE International Conference on Robotics and Automation.* pp. 1018–1023.

Hsu, D., J.C. Latombe and R. Motwani (1999). Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications* **9**, 495–512.

Hsu, D., T. Jiang, J. Reif and Z. Sun (2003). The bridge test for sampling narrow passages with probabilistic roadmap planners. In: *2003 IEEE International Conference on Robotics and Automation.* pp. 421–427.

Kavraki, L. and J.-C. Latombe (1998). Probabilistic roadmaps for robot path planning. In: *Practical Motion Planning in Robotics* (K. Gupta and A. del Pobil, Eds.). pp. 33–53. Wiley.

Kavraki, L.E., P. Svetska, J.C. Latombe and M.H. Overmars (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* **12**, 566–580.

Khatib, O. (1986). Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research* **5**, 90–98.

Latombe, J.-C. (1991). *Robot Motion Planning.* Kluwer.

LaValle, S.M. and J.J. Kuffner (2001). Randomized kinodynamic planning. *International Journal of Robotics Research* **20**, 378–400.

Nissoux, C., T. Siméon and J.-P. Laumond (1999). Visibility-based probabilistic roadmaps. In: *1999 IEEE/RSJ International Conference on Intelligent Robots and Systems.* pp. 1316–1321.

Reif, J.H. (1979). Complexity of the mover's problem and generalizations. In: *20th IEEE Symposium on Foundations of Computer Science.* pp. 421–427.

Reif, J.H. and Z. Sun (2003). On frictional mechanical systems and their computational power. *SIAM Journal on Computing* **32**, 1449–1474.

Simeon, T., J.-P. Laumond and F. Lamiraux (2001). Move3D: A generic platform for path planning. In: *4th International Symposium on Assembly and Task Planning.* pp. 25–30.