Multiobjective Motion Planning for a Nonholonomic Vehicle

Vasilios A. Spais
Dept. of Electrical and Computer Engineering
Aristotle University of Thessaloniki
54124, Thessaloniki, Greece
vspa@acm.org

Loukas P. Petrou
Dept. of Electrical and Computer Engineering
Aristotle University of Thessaloniki
54124, Thessaloniki, Greece
loukas@eng.auth.gr

Abstract- A technique is proposed for integrating a probabilistic graph construction algorithm with an evolutionary multiobjective optimizer. A hybrid planner (EvoVBPR) for a nonholonomic robotic vehicle is then presented. It integrates a probabilistic roadmap construction method (VBPR) with the SPEA2 evolutionary multiobjective algorithm and an additional deterministic graph pruning step. The result is a Pareto set of roadmaps that represent different tradeoffs between length of path and obstacle clearance.

1 Introduction

Motion planning problems can be quite difficult to solve effectively using deterministic algorithms [Lat91]. Nonholonomic motion planning is particularly trouble prone. Non evolutionary stochastic methods have been used quite extensively for these problems [SO98] and in some cases evolutionary computation based methods have been proposed [HS01][XMZT97][SM00]. Notwithstanding the evolutionary and multimodal method of [HS01] that is based on speciation, it was not possible to find major previous work where an evolutionary multiobjective algorithm in the sense of those in [Deb01], was applied to nonholonomic motion planning or to motion planning in general. In [CVL02], Coello et. al. discuss generic multiobjective evolutionary algorithms as well as planning applications.

In this work the Visibility Based Probabilistic Roadmap [NSL99] stochastic method is integrated with the SPEA2 [ZLT01] evolutionary multiobjective algorithm and applied to the kinematic motion planning problem for a nonholonomic vehicle. This is not a straightforward operation since the nonholonomic constraints of the vehicle motion and constraints on the structure of the graph representation impose special requirements. Thus the algorithms were augmented with a deterministic graph pruning operator, a stochastic repair operator and custom mutation and recombination operators.

The resulting algorithm is applied to multiobjective motion planning for nonholonomic vehicles. The Pareto front defines the tradeoff between (a) the shortest path length subject to obstacle imposed constraints and (b) the obstacle clearance, which is defined as the minimum distance between any point on the vehicle and any obstacle, at any time during the path traversal.

A set of concepts and guidelines are presented in section 2 that generalize this effort into a technique that is applicable to a class of optimization problems that have a natural phenotypic representation encoded in the form of a graph.

The application details concerning nonholonomic motion planning and probabilistic roadmaps are presented in section 3 while the details of the actual integrated algorithm in section 4. The results are presented and discussed in section 5. The conclusions and suggestions for further work are stated in section 6.

2 Integrating Graph Construction and Evolutionary Search

A number of optimization problems can not be readily represented using a fixed length encoding. A particular case is planning, where the size of the plan is not known in advance. An evolutionary optimization method where the representation is inherently of variable size such as Genetic Programming may be used in some cases. Generally though the representation is dictated by application requirements. In planning, a parameterized graph representation is a natural choice.

Assuming that the phenotypic representation is a graph, a number of choices exist for encoding the graph in a genotypic representation. This is unfortunately not possible if constraints on the phenotype are to be taken into account when applying the search (solution modification) operators. In this case the genotype and the phenotype are identical. If a graph representation is used, a set of graph based search operators (mutations and recombinations) have to be defined.

A graph construction algorithm operates by adding nodes and edges to a graph that is initially empty until some criteria are satisfied. Such an algorithm may take various "local" constraints into account when modifying the graph. In this instance the term "local" refers to constraints that a partially constructed graph must satisfy and that limit the values of the local parameters of the new graph element

(node and/or edge) being inserted.

A probabilistic graph construction algorithm incorporates random elements when inserting nodes and edges and when setting the node and edge parameters. Given an evolutionary optimizer that operates directly on the phenotype, the construction operator(s) is equivalent to a (generalized) mutation operator and may be used as such.

Since the construction operator may by itself deliver a solution if applied an appropriate number of times, it may be used for generating the initial population for the evolutionary algorithm. This population satisfies the constraints on the graph, since the elementary graph modification operators do so. The initial graph construction phase may continue until the graph reaches a specified size or until some "global" appropriateness criterion is satisfied. This criterion usually incorporates solution feasibility as its main constituent.

The design of recombination (crossover) operators is more involved. These operators can be designed so that the resulting graph satisfies the "local" constraints mentioned. This may be done by individually inserting the graph elements of the parent graph fragments, using the graph construction algorithm. In this instance no new probabilistic elements are introduced; those in the parents are used.

The resulting graph is generally a partially constructed graph that may not (and usually does not) satisfy the global appropriateness criterion. This deficiency may be corrected in a *stochastic repair* phase that repeatedly applies the construction (mutation) operator until the graph satisfies the global appropriateness criterion. This is the same as the initial population element construction method starting from a non empty graph.

Given the availability of mutation and recombination operators, the random search phase of an evolutionary optimization step (generation) is readily implemented. Evaluating the solution may require a graph traversal using a deterministic graph search algorithm such as that of Dijkstra which generates an optimal path within the graph (critical path). The resulting optimal path is evaluated and the population element is assigned a set of values for its objective functions. The part of the graph that is not part of the optimal path, is equivalent to the concept of introns in genotypically coded evolutionary computation.

It was experimentally found that the use of this optimal path as the basis for evaluating the population element, leads into insufficient evolutionary pressure, i.e. propensity for introducing new solutions. This is due to the fact that the element modifying mutation or recombination does not affect the optimal path but some other part of the graph. A suggested solution is stripping the graph of the nodes and edges that do not belong to the optimal path. This greatly increases evolutionary pressure since generally every search operator application produces offspring with different objective function values. The resulting lack of redundancy in the coding (intron removal) makes the use of an elitist evolutionary computation method critical for preserving good solutions, particularly if a generational evolutionary optimizer is used. An alternative that was not explored is the use of a steady state (or partially steady state) method. It must be noted that an elitist algorithm presents significant additional benefits for multiobjective optimization.

With population element objective functions computed, the selection method of any elitist evolutionary multiobjective optimization algorithm can be used for generating the elite population that encodes the Pareto set of solutions and the new main population.

3 Probabilistic Roadmaps for Nonholonomic Motion Planning

Probabilistic roadmaps have become a method of choice for motion planning. This is particularly the case when the motion space includes a large number of possibly irregular obstacles, is of a high dimension and/or the motion is subject to nonholonomic constraints. These issues are well discussed in [KSLO96] and references therein.

A constraint on a state variable of a dynamic system (e.g. the position of an object) is nonholonomic when it incorporates derivatives of the variable that can not be integrated out of the differential equation set describing the system dynamics. A system that incorporates nonholonomic constraints may be called a nonholonomic system and if it encompasses moving objects, the movement is also identified as nonholonomic.

A vehicle may be inherently nonholonomic when its dynamics are subject to nonholonomic constraints. This is the case of mobile robots that have the Ackermann steering system of a common car or of tricycles with an actively steering wheel. The nonholonomic constraint that can not be integrated out is that the derivative of the lateral displacement (i.e. the lateral velocity) of a non skidding car is zero.

A nonholonomic constraint may be externally imposed upon a moving object by obstacles. This happens when the completion of a desired maneuver requires that the object move tangentially to and near the boundary of an obstacle (compliance). This requires that component of the velocity of the object that is normal to the tangent of the obstacle and intersects the obstacle be equal zero. Furthermore, the derivative of one or more of the Euler angles of the orientation of the object must also be zero.

A nonholonomic system of moving objects is by definition under actuated, at least for portion of the movement during which the nonholonomic constraints are active. An inherently nonholonomic vehicle is always under actuated.

When planning motion in the presence of obstacles, it is not possible to directly derive a path from the initial to

the target configuration. The path is broken down into local paths that are determined by a local planner that operates in the absence of obstacles. The intermediate configurations (positions and orientations of the vehicle) where the local paths are connected are called waypoints. When using a roadmap representation of the motion plan, these local paths and waypoints are encoded explicitly. The waypoints are mapped to the nodes of a graph and the local paths are mapped to the graph edges. Particular values of variables that are applicable to each local path (e.g. steering settings) are mapped as parameters of the graph edges.

A constructive global planner may select waypoints and then use the local planner to attempt to connect them to the graph as constructed to that point. The graph is initially empty. When querying the plan for a solution to a movement problem, the planning algorithm attempts to connect the start and target configurations of the vehicle to any nodes in the graph. If this succeeds, a deterministic query (e.g. a Dijkstra search) is used to connect the waypoints found. This is referred to as the *precomputed roadmap* approach to roadmap based planning. Alternatively the initial and target configurations may be preinserted as nodes in the roadmap and the waypoints determined afterwards. New waypoints are generated and connected to the graph until the start and target configurations are connected. The deterministic graph search is still necessary. This is referred to as the *online* approach to roadmap construction.

A probabilistic roadmap (PRM) [SO98] selects waypoints randomly within the vehicle working area. If the selected waypoint is not inside some obstacle, an attempt is made to connect it to the graph. The PRM forms the basis of search algorithms that are complete in a probabilistic sense. That is, if enough nodes are added to the graph, given two vehicle configurations that are located in the same connected subset of the free space, the probability of connecting these configurations via the graph may approach one. Various strategies are proposed in the literature (e.g. [SO98] to select waypoints in such a way that the free (not inside the obstacles) space is well explored and that known difficult cases (e.g. corners, long corridors) are handled efficiently. Other methods are proposed to limit the size of the graph which tends to expand quickly.

The Visibility Based Probabilistic Roadmap (VBPR) was proposed in [NSL99] and is the one used in this work. It defines a coloring on the graph, labeling the nodes as of *guard* and *connection* type. Guard nodes define a visible region around them that is the subset of the free configurations (i.e. those belonging to the free space) that are connectable to the guard node using the local planner. A guard node may not be inserted within the guard (i.e. visible) region of an already inserted guard node. By definition any two guard nodes are not directly connectable using the given local planner. Connection nodes are inserted to connect two or more guard nodes that do not already belong to a connected subset of the graph. Connection nodes are by definition within the intersection of two or more guard regions and the insertion of a connection node connects two previously unconnected subgraphs.

The VBPR greatly reduces the number of nodes (waypoints) that must be inserted in the graph (roadmap) to render most of the free space reachable from the graph. This means that a randomly selected configuration that belongs to a connected component of the free space can be connected with a high probability with the subgraph of the roadmap that is located in this free space component. If there is only one connected free space component in the vehicle working area then there is a high probability that two randomly chosen configurations anywhere within the free space are reachable from the graph and therefore may be connected via the roadmap.

In the variation of the algorithm used in this work we use the online approach to roadmap construction and the VBPR method to limit the number of nodes.

Additionally, when a feasible path has been found (via Dijkstra search) from the start to the target configuration, a deterministic graph pruning step is applied. This removes the nodes and edges that are not part of the optimal path while preserving the visibility constraints that are applicable to the nodes that remain in the graph.

This step was found to be necessary for effective differentiation among the evolving population elements. Otherwise the graph elements that are not part of the critical path act as an abundance of introns that have no effect on the element fitness. Therefore the mutation and recombination operators mainly modify the introns without changing the element fitness. Since most elements are phenotypically identical, the selection algorithm becomes mainly random sampling. As a result the population overconverges to a phenotypically degenerate state, even though the genotype is still differentiated.

The pruning step modifies the function of the VBPR construction algorithm from that of a construction method of a roadmap used for later batch querying, to a step in a construction method that produces a *path* (with extraneous branches) that is the result of a specific query.

The local planner used was the Arc-Line-Arc (ALA) local planner for vehicles with Ackermann steering, as presented in [Sve96] for Reeds-Shepps [RS90] cars. It is assumed that the vehicle can move bidirectionally (i.e. both forwards and in reverse). The PQP version 1.2 library was used for collision detection and for determining the clearance of a given configuration form the nearest obstacle.

4 The EvoVBPR Multiobjective Motion Planner

The VBPR algorithm and its associated local planner and graph query produce a path considering only a single criterion, namely the path length. This is a sum along the path and has the characteristics of a low order norm.

When applying the VBPR, the intent is not to produce a complete roadmap as e.g. in [KSLO96], even though this may well happen if sufficient effort is expended at the initial step of the algorithm. The roadmap is used as a probabilistically complete set of paths out of which a specific path is selected for single query motion planning as in [Hsu00]. This specific path constitutes a single point (or tradeoff) on the Pareto front and its length is the first of the objectives being minimized.

The obstacle clearance criterion is handled *exclusively* by the evolutionary optimizer and is the minimum of the clearances of the local paths and has the characteristics of an infinite norm. This is the functionality supplied by the evolutionary component of the EvoVBPR beyond that of the VBPR.

Since the baseline VBPR considers only path length, the search component (initial population setup, mutation, recombination and repair) of the evolutionary optimization system is biased towards producing shorter paths. It is therefore a challenge for the evolutionary multiobjective optimization algorithm (EMOO) to produce a distribution along the clearance dimension. The clearance of a path is used as a proxy for the risk of colliding into an obstacle because of errors inherent in the execution of any plan. As the clearance increases, the collision risk decreases. Even a small difference in clearance is significant since the risk increases nonlinearly (exponentially if a normal noise distribution is assumed) as the clearance approaches the magnitude of the mean positioning sensor noise and of the mean actuator error.

The EMOO algorithm chosen was initially SPEA [ZT98][Zit99] and was later upgraded to SPEA2 [ZLT01]. This choice was initially driven by the excellent at the time performance of SPEA and the advantages of having an explicit separate archive containing the Pareto front. SPEA2 was a natural choice for improving the algorithm performance which was experimentally determined to be satisfactory. Alternatively any modern elitist EMOO algorithm could have been selected (e.g. NSGA-II [DAPM00] or a derivative of PAES [KC00]), although its performance in this specific application would still have to be experimentally determined.

The main algorithm parameters were as follows:

- Number of generations : **50**
- Main population size : **20**
- Elite population size : **6** to **10**
- Maximum attempts to insert nodes (application of repair operator) in an initial population element : **100**
- Nodes attempted to insert for each insertion attempt (initial or repair) : **50**
- Maximum number of failed attempts when applying mutation or recombination : **10**
- Mutation probability : **0.25** for each of 2 operators
- Recombination probability : **0.50**

The parameters above were determined experimentally and chosen so that the solution to the most time consuming of the worlds (obstacle sets) considered (as shown in figure 6), was derived in under an hour of computational time (worst case) on a 1 GHz Pentium 3 personal computer.

The guidelines proposed in section 2 were all applied in the design of the EvoVBPR planner. In particular:

- The roadmap representation is naturally that of a graph. The nodes must be located in the free space and the structure of the graph is subject to the visibility constraints. The edges must not collide with the obstacles.
- Custom graph based search operators (mutation and recombination) were implemented so that the above constraints on the graph are not violated.
- The elementary VBPR step (i.e. the insertion of a waypoint) functions as a constructive method and is used as a mutation operator. It is also used to generate the initial population and as the elementary step of the repair operator. In these operations a waypoint is repeatedly inserted until a path is found between the start and target configurations.
- The crossover operator operates upon the optimal path as computed by a Dijkstra search on the graph. It also uses the elementary VBPR node insertion step to construct the new graph without violating the constraints.
- A deterministic graph pruning operation as already mentioned in section 3 is used to increase evolutionary pressure and accelerate the solution.

The graph based search (mutation and recombination), initial construction and repair operators were defined as follows:

- *Mutation:* A guard or a connection node is removed from the pruned roadmap (2 types of mutation). A pruned roadmap is one where only the nodes and edges that belong to the optimal path have been retained. This operator produces a non feasible path and forces the application of the stochastic repair operator described below. Its intent is to keep the optimization going even if premature population convergence has

occurred.

- *Recombination:* Starting from a pair of pruned roadmaps, a single point crossover is applied to different points on each roadmap. It must be noted that after pruning, the roadmaps have been converted to chains. The guard and connection nodes of the contribution from the first parent are inserted. Then an attempt is made to insert the guard nodes of the contribution from the second parent and finally the connection nodes from the second parent. By recombining partial paths previously generated (a form of schemas), this operator accelerates the algorithm as experimentally determined by comparing with experiments where the mutation operator was used alone.
- *Initial Construction:* Starting with an empty graph, the elementary VBPR step is repeatedly applied until a feasible path connecting start and target configurations has been found. This is done by repeatedly applying the repair operator below.
- *Stochastic Repair:* The elementary VBPR step is repeatedly applied until a feasible path connecting start and target configurations has been found. This operator is applied starting from a partial roadmap, i.e. one that does not yet connect start to target. This operator ensures that a feasible path is generated after the application of the stochastic variation operators (mutation and recombination) and at the initial construction step. This allows the evaluation function to compute the values of the multiple objectives used in selection, which in the case of a non feasible path are undefined.

The EvoVBPR algorithm steps may be summarized as the following sequence:

1. Initialize population by repeatedly applying the repair operator on an empty roadmap (Initial Construction operator)
2. Prune graph and convert it to a chain of nodes
3. Apply stochastic search operators (Mutation and Recombination) as described above
4. Apply Stochastic Repair operator. If a feasible path from start to target is not found, revert the graph to its state before the search step and go back to step 3
5. Compute the multiple objectives that are to be optimized
6. Apply the selection method of the SPEA2 EMOO
7. If the maximum number of generations is not exceeded go to step 2

5 Results

Representative results are presented for three different environments. The first is an easy environment that is representative of a typical indoors arrangement *(Indoors, figures 1 and 2)*.

The second environment is very difficult for deterministic planners but of medium difficulty for stochastic planners *(Triangles, figures 3 and 4)*. This is due to the large number of obstacles.

The third environment is very difficult for stochastic planners (such as PRMs and the VBPR) when the vehicle is nonholonomic *(Corridor, figures 5 and 6)*. The difficulty is due to the fact that this environment is a long (difficult to connect), twisting (many changes of motion direction are necessary) corridor and narrow when compared to the vehicle turning circle (the local planner fails very often). Furthermore, a large portion of the working area is covered by obstacles (lower probability of selecting a random configuration in the free space).

In figures 1, 3, and 5 the points on each Pareto front (shown as markers) are connected by straight line segments. In the 3D drawing the vertical axis represents the generation of the evolution in increasing order and the Pareto front is drawn along the other two axes. The objectives are to be minimized and are positive values, i.e. the algorithm attempts to drive them to zero (at the rear left of the drawing space). The top view of this drawing rotated 90 degrees counterclockwise is presented for clarity in the 2D drawing. In both drawings, each alternate Pareto front is drawn in the same color (blue or red). The marker that is used to denote the points on the Pareto front that correspond to the specific tradeoffs found is rotated through four shapes (circle, square, diamond and triangle) every four generations.

It is evident from the results that Pareto fronts (or indifference fronts) do arise in path planning. The performance of the proposed solution is quite good in common environments and in environments where the baseline PRM methods perform well. The environments that are difficult for PRMs are naturally difficult for the EvoVBPR method as well, since the elementary search step in the algorithm is the insertion of a random configuration (graph node).

The above experiments were repeated 20 times for each environment. The aggregate results of these experiments are presented for the more difficult environments *Triangles* and *Corridor* in tables 1 and 2 respectively. The entries *Mean of Best*, *Mean of Worst* and *Mean of Median* refer to the average of the values of the objective functions *Length* and *Inverse of Clearance* over the 20 experiments. The values *Best of Best* and *Worst of Worst* refer to the extreme values that the objective functions assumed again over all the experiments. On the Pareto front the best values of *Length* correspond to the worst values of the *Inverse of Clearance*

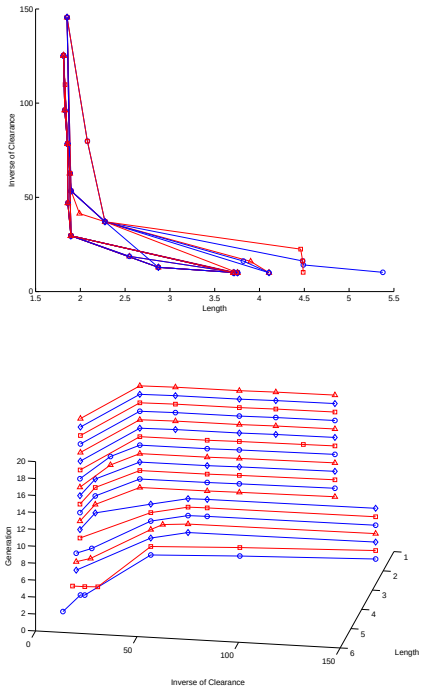

Figure 1: Progression of the Pareto front by generation in the *Indoors* environment

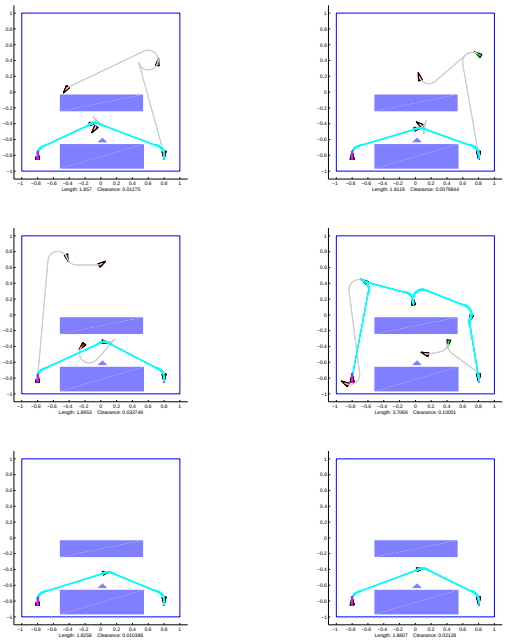

Figure 2: The final elite population in the *Indoors* environment

and vice versa.

To see the detail in the figures, this paper may be read in its electronic form (Adobe PDF), where a zoom feature is available.

6 Conclusions

A probabilistic graph construction algorithm was integrated with an evolutionary multiobjective optimizer and successfully applied in a multiobjective nonholonomic motion planning setting (sections 3 to 5). The resulting algorithm is quite different from related work in the literature.

A set of guidelines was defined to assist future applica-

	Length	Inverse of Clearance
Mean of Best	2.4036	44.7865
Mean of Worst	3.6841	308.0225
Mean of Median	2.7025	91.7446
Best of Best	2.3626	29.4097
Worst of Worst	6.3976	919.4420

Table 1: Aggregate results of multiple experiments in the *Triangles* environment.

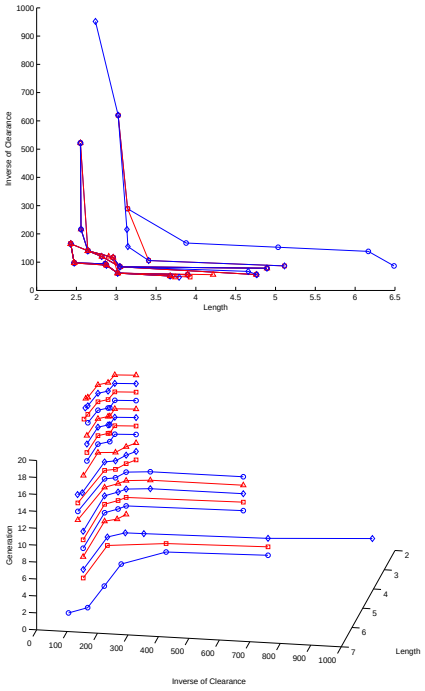

Figure 3: Progression of the Pareto front by generation in the *Triangles* environment

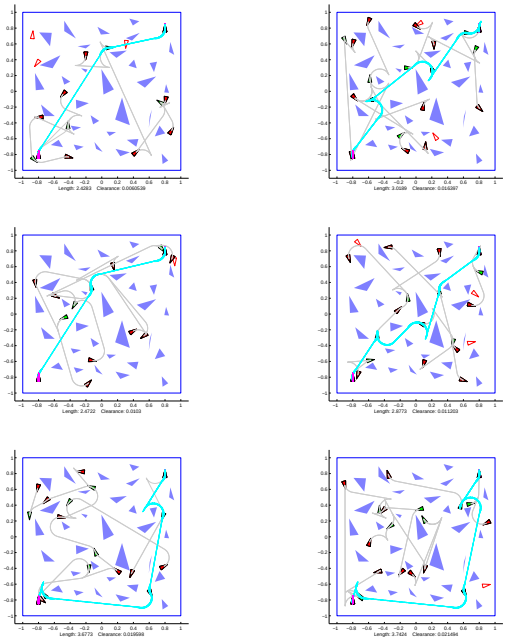

Figure 4: The final elite population in the *Triangles* environment

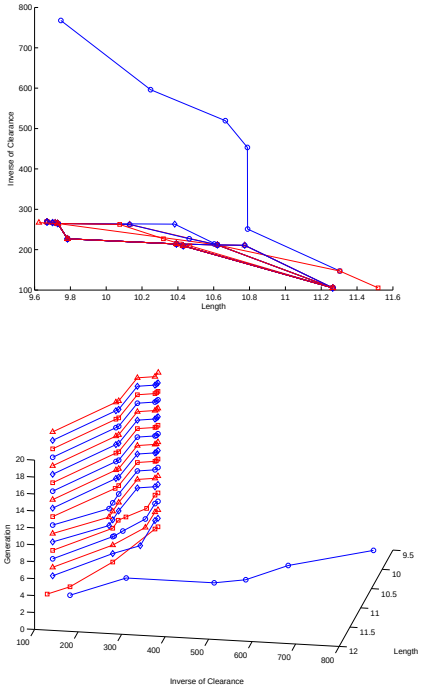

Figure 5: Progression of the Pareto front by generation in the *Corridor* environment

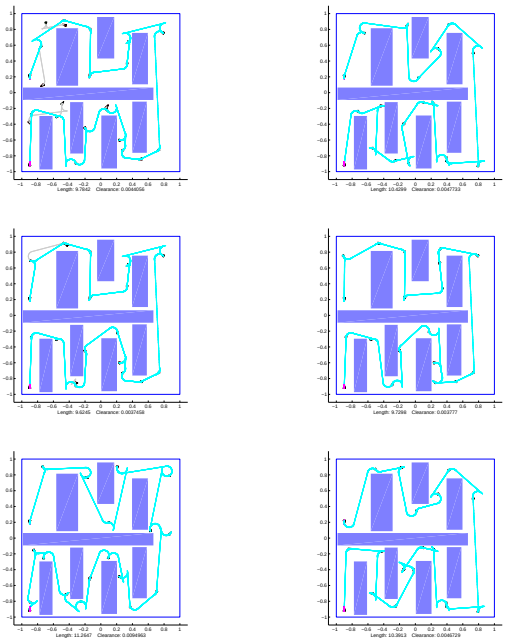

Figure 6: A subset of the final elite population in the *Corridor* environment

	Length	Inverse of Clearance
Mean of Best	8.9775	103.8056
Mean of Worst	9.9358	467.6432
Mean of Median	9.4345	168.6460
Best of Best	8.2409	61.8132
Worst of Worst	11.6129	811.1340

Table 2: Aggregate results of multiple experiments in the *Corridor* environment.

tion of the same integration concept to other settings (section 2). This technique is also novel.

These results may be generalized to any setting where a graph based phenotypic representation subject to constraints would be an appropriate representation and the search operates on the phenotype. Planning is a natural domain for applying this technique and further work is concerned with applying a similar methodology to kinodynamic nonholonomic motion planners.

Bibliography

[CVL02] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[DAPM00] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[Deb01] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, West Sussex, England, 2001.

[HS01] Cem Hocaoglu and Arthur C. Sanderson. Planning Multiple Paths with Evolutionary Speciation. *IEEE Transactions on Evolutionary Computation*, 5(3), June 2001.

[Hsu00] David Hsu. *Randomized Single-Query Motion Planning in Expansive Spaces*. PhD thesis, Department of Computer Science, Stanford University, May 2000.

[KC00] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[KSLO96] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4), August 1996.

[Lat91] Jean-Claude Latomb. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[NSL99] C. Nissoux, T. Siméon, and J-P. Laumond. Visibility Based Probabilistic Roadmaps. In *Proceedings of the 1999 IEEE International Conference on Intelligent Robotics and Systems*. IEEE, 1999.

[RS90] J.A. Reeds and L. A. Shepp. Optimal Paths for a Car that Goes both Forwards and Backwards. *Pacific Journal of Mathematics*, 145:399–409, 1990.

[SM00] Roman Smierzchalski and Zbigniew Michalewicz. Modeling of Ship Trajectory in Collision Situations by an Evolutionary Algorithm. *IEEE Transactions on Evolutionary Computation*, 4(3), September 2000.

[SO98] P. Svestka and M. H. Overmars. Probabilistic Path Planning. In Jean-Paul Laumond, editor, *Robot Motion Planning and Control*, pages 255–304. Springer-Verlag, 1998. Published as Lecture Notes in Control and Information Sciences 229.

[Sve96] Petr Svestka. A Probabilistic Approach to Motion Planning for Car-Like Robots. Technical Report RUU-CS-93-18, Departmetnt of Computer Science, Utrecht University, April 1996.

[XMZT97] Jing Xiao, Zbigniew Michalewicz, Lixin Zhang, and Krzysztof Trojanowski. Adaptive Evolutionary Planner/Navigator for Mobile Robots. *IEEE Transactions on Evolutionary Computation*, 1(1), April 1997.

[Zit99] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[ZLT01] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 2001.

[ZT98] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.