2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Evolving A Diverse Collection of Robot Path Planning Problems

Daniel A. Ashlock
Mathematics and Statistics
University of Guelph
Guelph, ON Canada N1G 2R4
dashlock@uoguelph.ca

Theodore W. Manikas
Department of Electrical Engineering
University of Tulsa
Tulsa, Oklahoma, 74104-3189
theodore-manikas@utulsa.edu

Kaveh Ashenayi
Department of Electrical Engineering
University of Tulsa
Tulsa, Oklahoma, 74104-3189
kash@utulsa.edu

Abstract

This study presents an evolutionary computation system that can generate grid robot path planning problems. An evolvable cellular representation that specifies how to build a PPP is used. Also presented is a technique for taxonomizing path planning problems so that the vast number of problems that can be generated with the evolutionary computation system can be subsequently winnowed into a collection of substantially different problems of specified size. In this study the most difficult path planning problems, according to three different criteria, are evolved and those results are used to demonstrated the taxonomic technique. The hardness criteria are (i) the minimum number of turns a robot must make, (ii) the minimum number of forward moves it must make, and (iii) the sum of these quantities. A dynamic programming algorithm is used to compute these quantities for a given path planning program. The technique can be generalized to find cases of a specified hardness. The size of the board and maximum number of obstacles used are transparently specifiable.

I. Introduction

An important task for mobile robots is autonomous navigation, where the robot travels between a starting point and a target point without the need for human intervention [21]. While basic information may be available to the robot about the navigation area boundaries, unknown obstacles may exist within the navigation area. This is called an uncertain environment: the robot must be able to maneuver around these obstacles in order to reach its target point. The world space refers to the physical space in which robots and obstacles exist - the free space is the subset of the world space that is not occupied by obstacles. A path between the starting and target points that avoids collisions with obstacles is said to be feasible - this is a path that lies within free space. Thus, robot navigation methods need to solve the path-planning problem, which is to generate a feasible path and optimize this path with respect to certain criteria. In this study an evolutionary computation system that automatically generates path planning problems is presented together with a technique for selecting a good collection of path planning problems from those that evolved.

A variety of approaches have been used to solve the path-planning problem, such as analytical methods [7], [10], [6], [15], [9], artificial intelligence [3], [18] and neural networks [30], [4], [17], [19], [28]. However, the path-planning problem has been shown to be NP-hard [16], which means that many of the traditional methods can become computationally intractable [23]. Genetic algorithms have been shown to be effective in solving NP-hard problems, thus they are often used for path planning in contemporary robot navigation algorithms [1], [14].

There have been several contemporary applications of genetic algorithms to the robot navigation problem. One approach is to combine fuzzy logic with genetic algorithms [23], [2], [20]. In this approach, the genotype structure represents fuzzy rules that guide the robot navigation, so the genetic algorithm evolves the best set of rules. The rules are generated during the projective component of the navigation algorithm, and then the reactive component follows this path until the discovery of new obstacles requires a new set of rules. While this approach can produce a feasible path through an uncertain environment, the genotype structure becomes very complex, as it needs to represent a variety of fuzzy rules. A complex genotype structure can take a long time to process in a genetic algorithm, which affects the real-time performance of the robot during navigation. The experimental results of these methods on simulated world spaces demonstrate that feasible paths can be found. However, there are no reported results on a real-time robot.

Another approach is to use genotype structures that represent local distance and direction, as opposed to representing an entire path [5], [11], [8], [27]. While these are simple to process and allow for faster real-time performance, the local viewpoint of these methods may not allow the robot to reach its target. Some methods have relatively simple genotype structures that can represent feasible paths, but require complex decoders and fitness functions [1], [14], [25], [29]. This can also affect real-time response.

The remainder of the study is organized as follows. Section II gives a representation for evolvable grid robot path planning problems. Section III specifies the dynamic programming algorithm used to compute the fitness of the PPPs. Section IV explains neighbor-joining taxonomy, the classification tech-

0-7803-9487-9/06/$20.00/©2006 IEEE

nique used classify PPPs. Section V gives the experimental design of the evolutionary computation trials done to locate robot path planning problems. Section VI gives the results of the experiments and presents an example taxonomy of path planning problems. Finally Section VII outlines next steps for this research.

II. A Representation for Evolvable Path Planning Problems

A *grid robot* is a virtual agent that is situated in an environment made of square grids. An agent's presence in the environment is defined by its position (the grid square it occupies) and the direction, up down, left, or right, that it is facing. A *path planning problem* (PPP) for a grid robot is a specified by giving the size of the grid, the position of obstructed grids within the environment, and an initial and goal position for the grid robot. For simplicity the PPPs in this study will have an initial placement of the agent in the upper left corner, facing left, and a goal of the lower right square. An example of an evolved PPP is given in Figure 1.

```
##########################################
#->......[].............[].............#
#........[].............[].............#
#.......................[].............#
#.......................[].............#
#[][][][][][]...........[].............#
#.......................[].............#
#.......................[].............#
#.......................[].............#
#.......................[].............#
#....................[][][]............#
#..................[][]..[]............#
#................[][]....[]............#
#..............[][]......[]............#
#............[][]........[]............#
#..................[]..[]..............#
#..................[]..[][]............#
#..................[]..[]....[][]......#
#..................[].......[][]......#
#..................[]....[][].........#
#..................[]..[][]..........**#
##########################################
```

Fig. 1. An example of an evolved 20x20 planning problem with 50 obstructed squares, denoted by pairs of brackets. This example is taken from an evolutionary run maximizing the number of turns the agent must make. The agent starts in the upper left corner and has a goal of the lower right corner. This path planing problem requires, at a minimum, 15 turns, 46 forward moves, and 61 total moves by the agent.

A representation that places each obstacle individually might be cumbersome and would potentially have an enormous search space. This study uses a *cellular*[12], [13] representation for PPPs. In a cellular representation, directions for constructing an object of interest are evolved, rather than direct descriptions. The cellular representation for PPPs is constructed of *single descriptors*. A single descriptor specifies a group of obstructions laid out according to a simple rule. It has four parameters (x, y, l, t) which specify its starting grid (x, y), the maximum number of blocks l the descriptor is permitted to specify, and t the *type* or pattern of the descriptor. Six types of single descriptors were used in this study. The first four place blocks in a line left, right, up, or down from (and including) the starting position. The last two types place blocks in a zig-zag with an obstacle at (x, y) and then left, down, left, down, ... or left, up, left, up, ...

A PPP is made of some number, in this study 4, 8, 20, or 50, single descriptors. A PPP has a maximum number of obstructions that are permitted *at all* in addition to the limits within the single descriptors. They are used to construct the path planning problem according to the following rules.

1) The single descriptors are processed in the order they appear in the PPP's chromosome.
2) Each single descriptor places its first obstruction at (x, y) and follows its pattern thereafter.
3) If an obstruction's position is outside the grid it is not placed and placement of obstructions for the current single descriptor are terminated.
4) The number of obstructions placed for a single descriptor may not exceed is own maximum number of obstructions.
5) If placement of an obstruction is attempted where one already has been placed then processing of the obstructions for the current single descriptor not only continues but that obstruction is not counted against the single descriptor's total l.
6) if the number of obstructions placed reaches the maximum permitted for a PPP then obstacle placement terminates.

This set of rules yields an unambiguous map from a list of single descriptors to obstacle placements for a PPP. Together with the fixed initial and goal positions for the grid robot already given a data structure for PPPs is now specified. The rule set given above, and the list of six types of single descriptors, are both *ad-hoc* choices made from a rich set of possibilities. Additional exploration of the design space of cellular representation for PPPs is left for future studies.

In order to evolve PPPs we must also specify the variation operators used. These are two point crossover of the list of single descriptors, treating single descriptors as atomic objects, and a mutation operator that functions in the following manner. With equal probability the mutation operator (i) generates a new x-coordinate for the starting position, (ii) generates a new y-coordinate for the starting position, (iii) lengthens a single descriptor by one obstruction, (iv) shortens a single descriptor unless this would make its length zero, (v) changes the type of the descriptor uniformly at random.

III. Dynamic Programming Fitness

The PPPs in this study are evolved to maximize three measures of fitness; the minimum number of turns, advances or forward moves, and the total moves (turn left, turn right or advance) that an agent must make to reach the goal. The evolutionary algorithm is configured to use any one of these three measures as the fitness function for evolving PPPs but all three are computed simultaneously during any execution of the dynamic programming algorithm. Once the obstacles in

a PPP are placed these three measures are computed via the *dynamic programming* algorithm given as Algorithm 1.

Algorithm 1: **Dynamic Programming Algorithm**.
Input: A PPP.
Output: the minimum number of turns, advances, and moves=turns+advances a grid robot agent requires to move from the PPPs initial position to the goal position.

Details:

1) initialize a state space array of all possible triples (x, y, h) representing the possible positions (x, y) and headings $h \in \{up, left, right, down\}$ that the agent can occupy. For each state three values a stored: the smallest number of turns, advances, and total moves found *so far* to reach that state. The initial value for each state is (∞, ∞, ∞) for all possible states except the agent's starting state, set to $(0, 0, right)$ in this study, which has an initial values of $(0, 0, 0)$.
2) Place the agent's initial state in a queue Q.
3) While Q is not empty, remove the first state S. Generate all possible moves from that state (turn left, turn right, and advance if possible) and compare one plus the value at state S to the value at the new state generated for each of the three statistics. If a better value is found for any of the statistics record it and place the new state in Q.
4) Examine all four states with position in the lower right corner; since the final heading is *not* specified the smallest value for each of the three statistics in these four states is reported as the score for the PPP for that statistic.

As a side effect the dynamic programming algorithm detects boards in which the goal cannot be reached. Such PPPs are assigned the lowest available fitness and quickly winnowed out by evolution.

IV. Neighbor Joining Taxonomy

A *taxonomy* is a hierarchical classification of a set. Linnaeus established the first definite hierarchy used to classify living organisms. Each organism was assigned a kingdom, phylum, class, order, family, genus, and species. This hierarchy gave a tree structure to the taxonomy for all living creatures. Modern taxonomy has nineteen levels of classification, extending Linnaeus' original seven. The reader should see [22] for details on modern taxonomic procedures for living organisms. An example of a taxonomy of PPPs is shown in Figure 4. Making such a tree requires that we extract *taxonomic characters* from the PPPs. A taxonomic character is simply a measurable or computable quantity such as number of legs or maximum number of teeth in a healthy adult. With taxonomic characters in hand, neighbor joining taxonomy, described subsequently, is then used to produce a "family tree" of a set of PPPs. PPPs closer to one another within the tree are considered more similar, those more distance are less similar.

The choice of taxonomic characters used is critical. They must avoid bias, they must vary across the set of PPPs, and they must avoid arbitrary judgments to the greatest degree possible. Using color in a numerical tree building algorithm, for instance, requires numbers be assigned to colors in a fashion that arbitrarily ranks some colors as closer to one another than others. The preceding brief discussion gives only a taste of the difficulty of choosing good taxonomic characters. Readers familiar with automatic classification, decision trees, and related branches of machine learning will recognize that those choosing decision variables face similar issues. Any taxonomic character or decision variable must be relevant to the decision being made, vary across the set of objects being classified, and be cleanly computable for all members of the set of objects being classified. For PPPs we will use four taxonomic characters: the three numbers computed by the dynamic programming algorithm together with the number of obstructions actually placed. All four of these are normalized linearly to the range 0-1 to prevent characters with a larger natural numerical range from unfairly dominating the classification.

For each pair of PPPs P and Q, the Euclidian distance $d(P, Q)$ between the vectors $m(P)$ and $m(Q)$ of normalized characters was then computed by the formula

$$d(P, Q) = \sqrt{\sum_{i=1}^{4} [m(P)_i - m(Q)_i]^2}.$$

$d(P, Q)$ was interpreted as the distance between the problems P and Q. An "UPGMA" (neighbor joining) tree was used to describe the relatedness of the the PPPs.

UPGMA is a clustering method commonly used to transform distance data into a tree. It received attention in [24], and a good recent description may be found in [26]. It is especially reliable if the distances have a uniform meaning. Normalization of the numbers in $m(P)$ makes the widely different numerical ranges for the statistics comparable so that the inferred distances are appropriate for analysis by UPGMA.

UPGMA is an acronym for "Unweighted Pair Group Method with Arithmetic mean." Given a collection of taxa and distance d_{ij} between taxa i and j, the method first links the two taxa x and y that are least distant. The taxa x and y are merged into a new unit z. For all taxa i other than x and y, a new distance d_{iz} is computed as the average of d_{ix} and d_{iy}, and it is noted that the new taxon z really represents the average of two original taxa. Henceforth, x and y are ignored, and the procedure is repeated to find the next pair of taxa that are least distant. When two taxa u and v are combined into a new taxon w, the new distance d_{iw} is the average of d_{iu} and d_{iv}, weighted according to the number of original taxa in u and v respectively; w contains all the original taxa in both u and v. The procedure ends when the last two taxa are merged.

V. Experimental Design

The evolutionary algorithm used to evolve PPPs operated on populations of size 60. The model of evolution used was steady-state size-seven tournament selection. In this model of evolution a group of seven chromosomes are selected. The two most fit are copied over the two least fit, breaking ties

uniformly at random. The copies are subjected to the crossover operator and one mutation each. One such tournament with selection, variation, and replacement is called a *mating event*. A run consists of generating a random initial population of PPP chromosomes followed by 10,000 mating events. For each set of parameters a set of 100 runs were performed. The parameters used are given in Table I. Initial populations are created by generating the number of chromosomes, consisting of lists single descriptors used in a given run. The values in an initial descriptor are selected uniformly at random for x, y, and t from the valid values. The initial value of l in a single descriptor is generated by averaging two random numbers, one in the range [0,L-1] and the other in the range [0,H-1] and then adding 1 where L is the horizontal dimension of the PPP, H is the vertical dimension. Fractions are discarded.

PPP Dimensions	Single Descriptors	Fitness	Maximum Obstructions
12x12	4	Turns	20
12x12	4	Adv.	20
12x12	4	Moves	20
12x12	4	Turn	30
12x12	4	Adv.	30
12x12	4	Moves	30
20x20	8	Turn	50
20x20	8	Adv.	50
20x20	8	Moves	50
50x50	20	Turns	200
50x50	50	Turns	500

TABLE I
PARAMETERS FOR EVOLUTIONARY RUNS PERFORMED.

VI. RESULTS

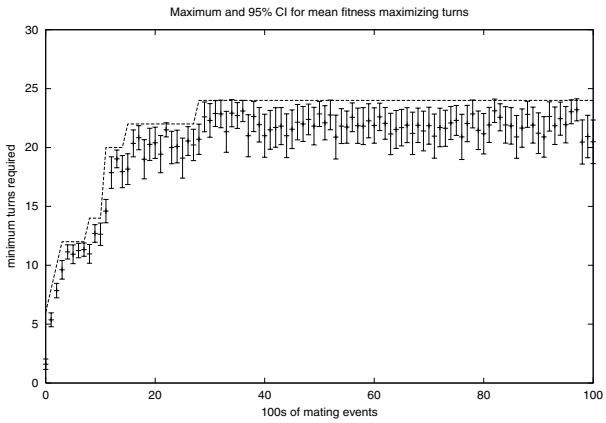

Fig. 2. An example of the fitness, maximizing the minimum number of turns, over evolution for one run. This is the 6th run of 100 performed for 20x20 PPPs.

Figure 3 shows the first 12 PPPs generated in the runs for 12x12 PPPs with four single descriptors and a maximum of 30 obstructions. A taxonomy of these PPPS is shown in Figure 4. Juxtaposition of the statistics (turns, advances, obstructions) with the position in the taxonomy shows that the result is not

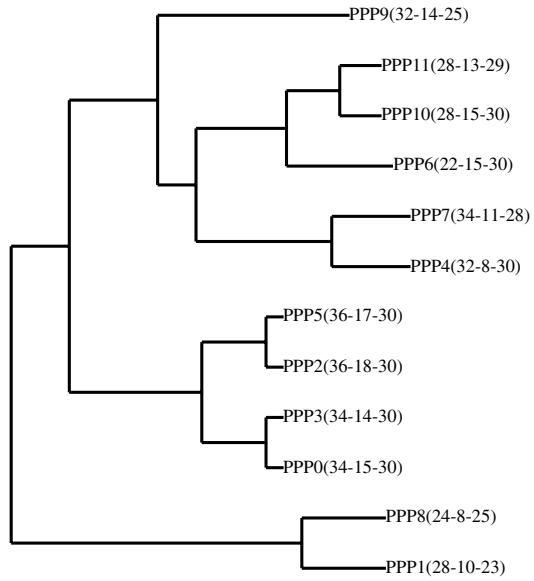

Fig. 4. A neighbor joining taxonomy for the boards in Figure 3. The characters used are the minimum number of required turns, advances, total moves, and blocks present for the path planning problem. The number of advances, turns, and obstructions (adv-trn-obs) are given at the end of the PPP names.

unreasonable and that the taxonomic technique can be used to divide the boards into groups from which representatives can be selected to create benchmark sets of PPPs of desired size with controllable diversity and difficulty.

The evolutionary algorithm functioned nominally in all 11 collections of 100 runs performed. Figure 2 shows the fitness as a function of evolutionary time for one of the runs performed to maximize the minimum number of turns required on a 20x20 PPP with 8 single descriptors and a maximum of 50 obstructions.

Examining the final best-of-run boards showed a wide diversity of boards and statistics. The character of the boards differed between the fitness functions with the zig-zag patterns of obstacles, which excel at forcing the grid robot to turn when used correctly, being more common in the runs where turns and total moves were used as the fitness function. Figure 5 gives the distribution of minimum number of turns and advances required as well as obstructions placed for the 100 12x12 PPPs evolved with 4 single descriptors and at most 30 obstacles. The distribution of these statistics suggest that the search space has many optima.

The dynamic programming fitness function was gratifiyingly fast; even the populations of 50x50 PPPs with up to 500 obstructions ran in less than a minute per run on an inexpensive desktop machine. An example of one of these PPPS is show in Figure 6.

VII. DISCUSSION AND FUTURE DIRECTIONS

The evolved character of the PPPs is apparent in Figure 6 and to a lesser degree in the other PPPs presented. The 50x50

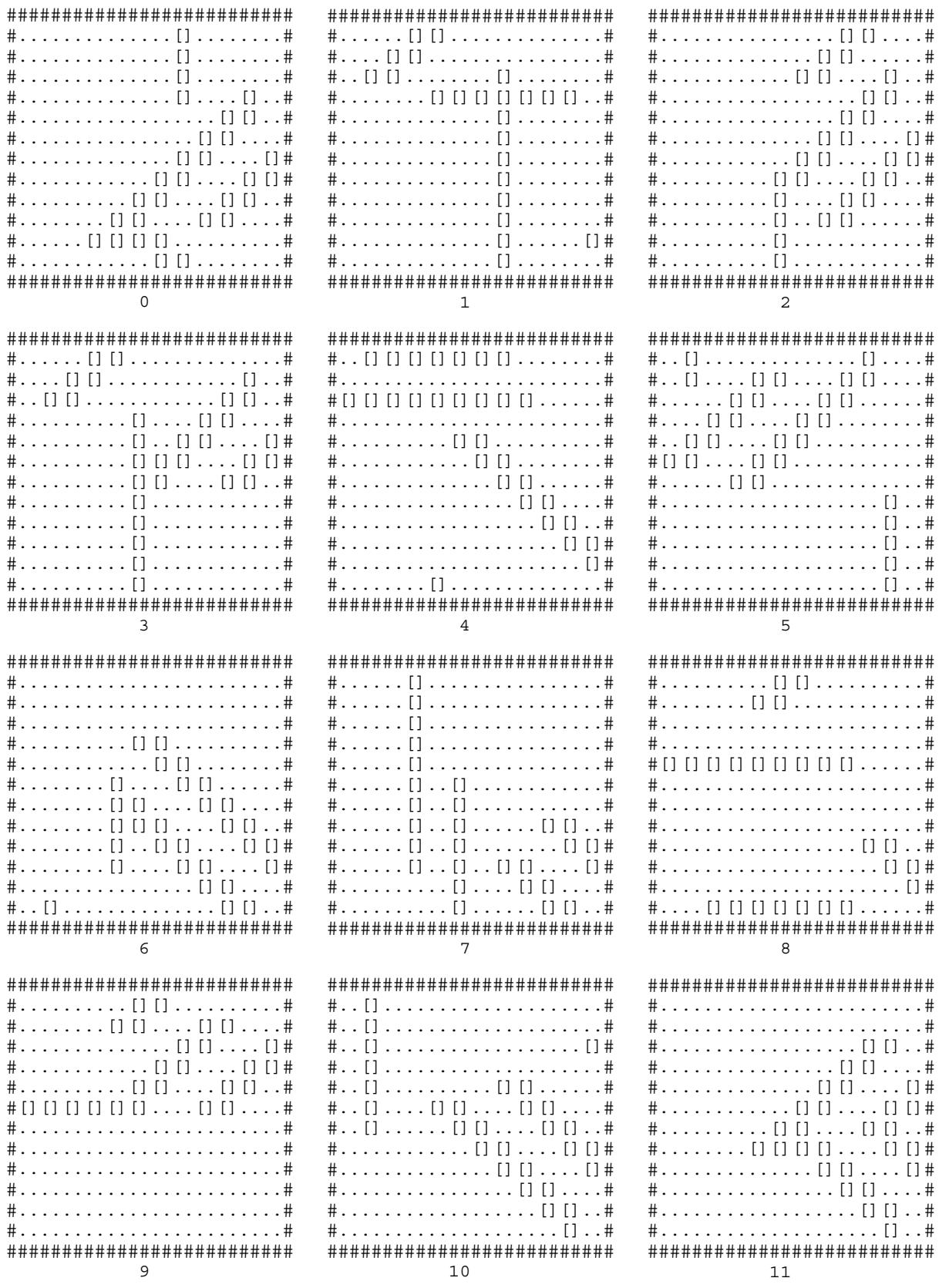

Fig. 3. The best-of-run path planning problems for the first twelve runs using a 12x12 planning space with 4 descriptors and at most 30 blocks optimizing for the maximum number of turns. Robots start in the upper left corner and are to reach the lower right corner.

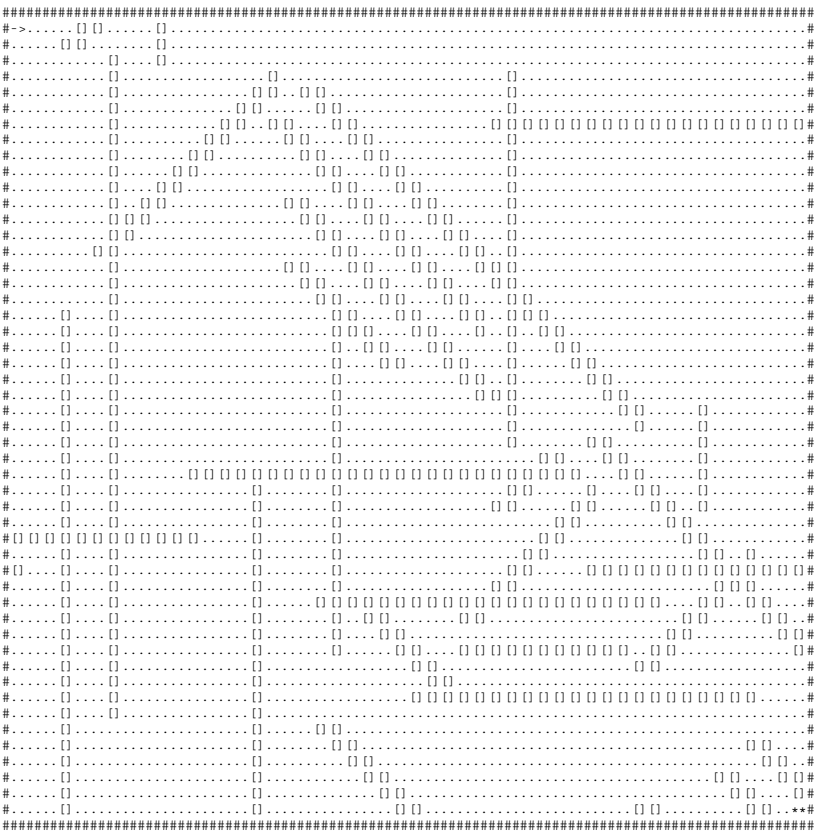

Fig. 6. An example of an evolved 50x50 PPP with 50 single descriptors and up to 500 obstructions. The agent's initial position is denoted with $-$ $>$ and the goal is denoted **

PPP shown has many clever traps that the agent cannot reach and others that can be ignored by an efficient agent because they are not on its shortest path to the goal. This suggests the presence of a nontrivial evolutionary lineage. The traps that the agent cannot reach or need not deal with may be residual from an earlier evolutionary epoch in which an optimal path did interact with those obstructions. This suggests both that this technique for creating PPPs yields interesting mazes and that rewarding simplicity during the evolution of PPPs might yield higher levels of difficulty by causing "vestigial" obstacles to be under selection pressure to become active obstructions again. On the other hand, this study may have inadvertently discovered a technique for automatically generating virtual architectures for video games.

Examine the PPPs shown in Figure 7. These are typical boards from the 12x12 runs with 4 single descriptors. One is from a run where the total obstructions were limited to 20 the other the limit was 30. As in the example shown, many of the PPPs evolved with at most 20 obstructions produces boards that did not require all their single descriptors (whether they used them or not). The top board in Figure 7 could be created with two descriptors. The average number of obstructions permitted *per descriptor* in the initial population for 12x12 PPPs was 12. Since a line of obstructions must reach the edge of the board to function efficiently in many cases, it may be that a limit of 20 total obstructions caused all fitness to rest on the first two single descriptors. In any case there appears to be a phase change in behavior (and fitness) of the evolved PPPs when the number of total obstructions was changed from 20 to 30. This suggests that parameter tuning for this system is not smooth or incremental.

Figure 6 is drawn from tests that showed that the system for evolving PPPS scales nicely. The system can evolve relatively large, complex PPPs in modest wall-clock time. Likewise, it can produce very large collections of PPPs as an initial step toward building diverse collections of such problems. While it was not tested in this study, which focused on *maximizing* the difficulty of PPPs, modifying the fitness function to reward a particular difficulty value can further increase the diversity of PPPs located; at odds with the author's initial expectations the system already locates a large diversity of difficulties of PPP while attempting to maximize their difficulty. The histograms given in Figure 5 are typical of the diversity of outcomes for the eleven sets of runs performed.

The taxonomic technique for PPPs, introduced in this study but not fully exploited, can be used to divide PPPs into similar groups. In this case the taxonomy is functioning as a hierarchical clustering technique. The taxonomy software can be modified to automatically search for sets of PPPs of desired size and with the large diversity relative to the size of the set requested.

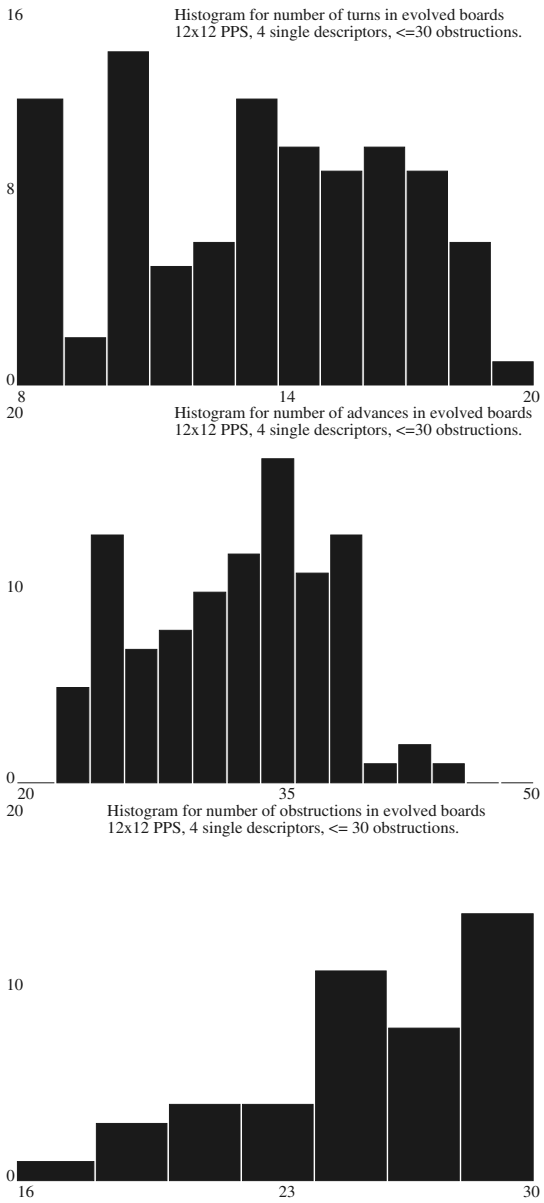

Fig. 5. The distribution of turns, advances, and obstructions places for the best-of-run 12x12 PPPS in the 100 runs performed with 4 single descriptors and a maximum of 30 obstructions.

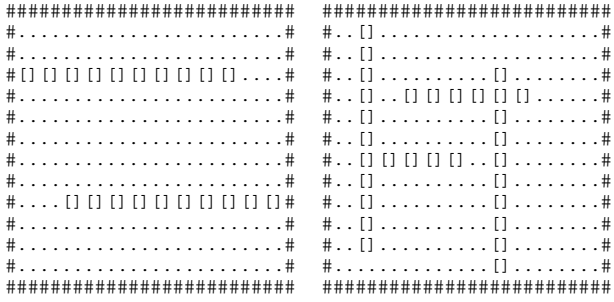

Fig. 7. Exemplary best-of-run PPPs from a run with at most 20 obstructions placed (left) and 30 (right), from runs maximizing forward moves.

A. Possible Applications

The intended application for this work is to creating a tool with which to explore the space of path planning problems. There are other possible applications, including automatic generation of levels or boards for a computer game. Evolution could rapidly produce boards that have a desired character from a relatively open "cathedral" to a complex, twisty maze. Another application is to homeland security where evolution could generate thousands of floor plans so that anything from toxin detectors to guard deployment could be gamed thousands of times and tested for robustness. For both computer games and homeland security applications the cellular representation could be extended to embed threats, traps, and other task-relevant features.

Another potential application is to provide a nontrivial environment for robot ecology studies. Virtual robots that attack one another, forage for food, or that must find mates are used in ecological simulation. Typically they use a single simulated environment. Using evolved boards with very different levels of obstruction and connectivity could be used in experiments that test the hypothesis that different hunting, foraging, or mating strategies will arise in different environments.

B. Additional Characters

One reason for the lack of emphasis on the taxonomic technique is the need to develop a richer set of taxonomic characters for use in classification of PPPs. At present the characters are the number of turns, forward moves, total moves, and obstructions. A natural extension to this character set is to add the length and width of the grid; in this study the example taxonomy was done entirely on 12x12 PPPs. Nevertheless this set of characters lacks independence: total moves is the sum of advances and turns; size is highly predictive of the number of advances and total moves required. The following additional characters could be used.

Agent derived characters are those derived from simple heuristics. The amount of time an agent performing left wall following, right wall following, or other simple heuristics might make a valuable taxonomic character for classifying PPPs. Care must be used since some of these heuristic may fail on some PPPs but, once noticed, simply assigning a computationally sensible value for "infinite number of moves to solution" would make heuristic failure another useful piece of information for classification. Evolved agents, selected for their ability to behaviorally distinguish PPPs, might also serve as a useful source of taxonomic characters.

Geometric characters are those extracted from the pattern of obstructions within the PPP. The characters used in this study are all geometric characters, but many others exist. The *mean diameter* of a PPP is the average, over all non-obstructed squares, of the number of unobstructed squares visible in the horizontal and vertical dimension in both directions. This character would capture the degree to which the PPP is broken into compartments.

Representationally derived characters are extracted from the cellular encoding of the PPP. These characters are less

generally useful because they can only be computed when some specific representation is used. If all the PPPs being taxonomized are derived from, for example, the cellular representation presented in this study then the number of vertical, horizontal, or zig-zag single descriptors executed during the construction of the PPP are a potentially informative taxonomic character.

C. Different Cellular Representations

The six types of single descriptor used in this study could be extended to dozens with very little effort, yielding a far richer design space for evolved PPPs. Additional types of descriptor might include rectangular shapes with openings (in effect rooms) or more complex types of zig zags. it is not clear, however, that increasing the complexity of the descriptors is the only interesting possibility. The zig-zag descriptors were of great utility in forcing turns - note the zig-zag "corridors" in Figure 6. It might be interesting to see which types of PPP would arise if the zig-zag descriptors were not used.

It is also possible to construct entirely different types of cellular representations for PPPS. A representation that fills in sub-squares of the grid from a pallet of smaller PPPs while maintaining connectivity for example, would yield a simple, evolvable representation. The representation would consist of an ordering of the sub-squares. The expression of such a representation would use the next available sub-square in the evolvable ordering that maintained connectivity. Evolution of the order in which sub-squares are considered yields a search of a rich space of PPPs.

VIII. Acknowledgments

The first author would like to thank the Department of Mathematics and Statistics at the University of Guelph for its support of this research.

References

[1] A. Alvarez, A. Caiti, and R. Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429, 2004.

[2] C.T.C. Arsene and A.M.S. Zalzala. Control of autonomous robots using fuzzy logic controllers tuned by genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 428–435, 1999.

[3] M. Beetz and T. Belker. Learning structured reactive navigation plans from executing mdp navigation policies. In *Proceedings of the fifth international conference on Autonomous agents*, pages 19–20, 2001.

[4] A. Berlanga, A. Sanchis, P. Isasi, and J.M. Molina. A general learning co-evolution method to generalize autonomous robot navigation behavior. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 769–776, 2000.

[5] R.R. Cazangi and M. Figueiredo. Simultaneous emergence of conflicting basic behaviors and their coordination in an evolutionary autonomous navigation system. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 466–471, 2002.

[6] J.L. Diaz, S. De Leon, and J.H. Sossa. Automatic path planning for a mobile robot among obstacles of arbitrary shape. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 23(3):467–472, 1998.

[7] H.J.S. Feder, J.J. Leonard, and C.M. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Researc*, 18:650–668, 1999.

[8] D. Gallardo, O. Colomina, F. Florez, and R. Rizo. A genetic algorithm for robust motion planning. In *11th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 115–122, 1998.

[9] S.S. Ge, X. Lai, and A.A. Mamun. Boundary following and globally convergent path planning using instant goals. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 35(2):240–254, 2005.

[10] M. Gerke and H. Hoyer. Planning of optimal paths for autonomous agents moving in inhomogeneous environments. In *Proceedings of the 8th International Conference on Advanced Robotics (ICAR '97)*, pages 347–352, 1997.

[11] V. Di Gesu, B. Lenzitti, G. Lo Bosco, and D. Tegolo. A distributed architecture for autonomous navigation of robots. In *Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, pages 190–194, 2000.

[12] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, France, 1994.

[13] Frederic Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–183, 1995.

[14] C. Hocaoglu and A.C. Sanderson. Planning multiple paths with evolutionary speciation. *IEEE Transactions on Evolutionary Computation*, 5(3):169–191, 2001.

[15] J.Y. Hwang, J.S. Kim, S.S. Lim, and K.H. Park. A fast path planning by path graph optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 33(1):121–129, 2003.

[16] Y.K. Hwang and N. Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[17] K-Y. Im, S-Y. Oh, and S-J. Han. Evolving a modular neural network-based behavioral fusion using extended vff and environment classification for mobile robot navigation. *IEEE Transactions on Evolutionary Computation*, 6(4):413–419, 2002.

[18] J.S. Jennings, G. Whelan, and W.F. Evans. Cooperative search and rescue with a team of mobile robots. In *Proceedings of the 8th International Conference on Advanced Robotics (ICAR '97)*, pages 193–200, 1997.

[19] T. Kondo, A. Ishiguro, S. Tokura, Y. Uchikawa, and P. Eggenberger. Realization of robust controllers in evolutionary robotics: A dynamically-rearranging neural network approach. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 366–373, 1999.

[20] N. Kubota, T. Morioka, F. Kojima, and T. Fukuda. Perception-based genetic algorithm for a mobile robot with fuzzy controllers. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 397–404, 1999.

[21] J. C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[22] E. Mayr and P. D. Ashlock. *Principles of Systematic Zoology*. McGraw-Hill, New York, 1991.

[23] D.K. Pratihar, K. Deb, and A. Ghosh. Fuzzy-genetic algorithms and mobile robot navigation among static obstacles. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 327–334, 1999.

[24] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy; the Principles and Practice of Numerical Classification.* W.H. Freeman, San Francisco, 1973.

[25] K. Sugihara and J. Smith. Genetic algorithms for adaptive motion planning of an autonomous mobile robot. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97)*, pages 138–143, 1997.

[26] D.L. Swofford, G.J Olsen, P.J. Waddell, and D.M.Hillis. Phylogenetic inference. In D. Hillis, C. Moritz, and B. Mable, editors, *Molecular Systematics, second edition*, pages 407–514. Sinauer, Sunderland, MA., 1996.

[27] P. Vadakkepat, K.C. Tan, and W. Ming-Liang. Evolutionary artificial potential fields and their application in real time robot path planning. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 256–263, 2000.

[28] F. Wang and E. Mckenzie. A multi-agent based evolutionary artificial neural network for general navigation in unknown environments. In *Proceedings of the third international conference on Autonomous agents*, pages 154–159, 1999.

[29] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski. Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation*, 1(1):18–28, 1997.

[30] S.X. Yang and C. Luo. A neural network approach to complete coverage path planning. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(1):718–724, 2004.