Non-linear Decision Making for Robust Navigation in Role Based Autonomy

Robert Thome, Christopher Fedor, John Sustersic, Ucheoma Ukah
Autonomous Intelligent Systems Division, Cognitive Perception and Computational Intelligence Lab
The Pennsylvania State University, Applied Research Laboratory
State College, PA 16801 USA
jps263@arl.psu.edu

***Abstract*—*A common approach to the robotic path planning problem is to discretize the search space and plan an optimal path using graph search methods. As the optimality criteria becomes more complex it becomes increasingly difficult to optimize the search with respect to each metric being considered. A hybrid path planning system is introduced that supplements the graph search approach with a cognitive architecture. The selected cognitive architecture, Soar, will provide high level reasoning, guiding the implementation of the graph search and reasoning on the results generated. Given scenario-specific context, the hybrid approach will allow for multiple, complex and possibly conflicting criteria to be considered during path generation.*

Keywords*—*robust navigation, path planning, role based autonomy, cognitive architecture, soar, multiple objectives, decision space

I. INTRODUCTION

Planning a path from a starting location to a goal location is a common problem for autonomous mobile systems and many methods exist for finding an optimal path. A typical approach is to represent the problem as a graph and then use graph search methods, such as A*, to solve for an optimal path using a well-defined cost function. Applications that need a navigation solution that can address unstructured and dynamic environments along with multiple objectives require an extension to traditional planning solutions. This paper proposes a hybrid path planning system that will use scenario specific context, environment inherited costs and the results of traditional planning methods to populate the decision space. Instead of attempting to achieve optimality with respect to every metric defined, we propose a method of path planning that is reliant on a sufficiently populated decision space, from which the path planning solution is derived.

The Soar cognitive architecture is a computational model of human decision making [1]. Soar uses production system rules (roughly in the form "if … then...") to encode knowledge that then governs its behavior. Problem solving in Soar can be described as a search through states of a problem space for a goal state. Each decision cycle in Soar elaborates the current problem state with information from encoded knowledge to then execute a decision procedure most appropriate to the current situation. If the decision is ambiguous, Soar can further reason on which decision aptly applies.

Sponsor: Office of Naval Research SwampWorks

The path planning problem is to find a finite sequence of actions that transform a given initial state to a goal state. The state transition graph (the set of available states to traverse) can be either explicitly or implicitly defined. In most applications, the entire state transition graph is not fully represented because the number of states is prohibitive. Instead it is created incrementally during the planning process [2]. Each state typically encodes problem specific information such as location and costs (such as terrain) inherent in the world. Graph search methods such as A*, attempt to reduce the total states explored by heuristic estimates. A* uses the search evaluation function, *f(s),* for a current state (*s)*:

$$f(s) = g(s) + h(s) \tag{1}$$

Where *g(s)* is the accumulated cost from the initial state to state (*s)* and *h(s)* is the heuristic estimate from state (*s)* to the goal. If *h(s)* is an admissible heuristic, an estimate that is less than the actual cost from *s* to the goal; A* will generate an optimal path. To consider multiple cost criteria requires an extension to (1), usually in the form of a weighted linear-combination of costs. Developing a weighted cost function for graph search methods to consider multiple objectives is difficult to scale and generalize.

The proposed system will treat A* as a black box; a modular piece of the planning solution that will be guided by the operational directives from Soar. The final selected path may not prove optimal with respect to all criteria; rather, Soar selects a satisficing path using cognitive heuristics. The domain used will model the spatial-temporal environment found in ship operations. The planning environment will be limited to two dimensions and is defined as inclusionary and exclusionary zones with further spatial constraints given in the form of cross-track limits (CTL). The system is dependent on a Navigator agent that parallels its real-world counterpart in modeling decision oriented planning.

II. RELATED WORK

The direct solution to the multiple objective path planning problem is to calculate a Pareto optimal surface and select a path with feasible combinations of costs. One such approach made use of gradient descent on a multiple objective cost function to generate the Pareto optimal surface and then used a PDE to determine feasible paths. It was shown that this

978-0-933957-43-5 ©2015 MTS

procedure is only practical for a limited state space dimensionality [3]. Furthermore, this approach became computationally infeasible when the number of cost objective metrics exceeded six.

Even though multi-objective path planning is theoretically NP-Hard, heuristic approaches and problem specific knowledge can provide satisfying solutions. One way to provide this additional knowledge and reduce computational complexity is to compare the current path planning problem with previously solved similar planning problems. The lightning robot path planning framework follows a look up and repair learning strategy while simultaneously planning from scratch and executes whichever solution completes first [5]. While not explored in this paper an advantage of selecting Soar is the architecture's ability to learn about all aspects of the task including performance and apply that knowledge to future path planning problems.

Genetic Algorithms (GA) have also been used to implement Pareto optimality for the path planning problem [6]. A GA is an optimization method that mimics natural selection by modeling a population of solutions. Survival of the fittest allows the best candidate solutions to emerge over each generation. Fitness of a candidate solution is determined by the multi-objective cost function. Genetic Algorithms are an improvement over directly solving the Pareto surface with the added advantage of providing more than one equally feasible solution. However, many GA approaches do not scale well with complexity and it is not clear when to stop searching for a better solution.

To overcome the limitations of a genetic algorithm, a hybrid GA and A* search approach was proposed [7]. In this approach a modified A* algorithm randomly perturbed a single objective cost and heuristic function based on distance to seed the population of candidate solutions for the genetic algorithm. The genetic algorithm's mutation operator was also enhanced to include the traditional A* search in some instances. Simulation results showed the hybrid solution could efficiently find a Pareto optimal solution for a limited test with three objectives in a static obstacle environment.

III. HYBRID SYSTEM OVERVIEW

The hybrid planning system is composed of two primary modules; a path generation module and the Soar cognitive architecture module. The path generation module is responsible for housing various path metrics, graph search algorithms and path geometry used for analysis, verification and repair methods. The Soar module is bidirectional with path generation, receiving input for aiding in decision making and outputting operational directives for constructing a path. To populate the decision space, the system will receive spatial-temporal constraints along with tactical orders to serve as context for decision making. The tactical orders act as a surrogate for the multiple objectives (i.e. to plan considering path length or smoothness and time given a uniform velocity).

Guided by the orders, Soar will direct the path generation module to produce either a single raw path (a path which has yet to be scored by metrics) or a set of raw paths. Soar receives the raw path(s) and further elaborates on if the path(s) meet greater spatial constraints (i.e. CTL) or if a given path exceeds an objective tolerance. The resulting path(s) are scored by permutations of relevant metrics (i.e. distance, risk, energy, smoothness), that assign a score to each path. From the scored set, Soar further elaborates on the constraints and objectives to be met, then selects the most appropriate path considering the scenario-context. The system is designed for the two modules to work interactively, exchanging relevant or required information and results.

A. The Path Generation Module

The module is divided into three individual but interacting components: path metrics, path planner and path geometry (see Fig. 1).

I. Path Metrics

The path metrics component houses various criteria that will be used for two dual purposes. The metric can guide path generation or score a created raw path. Relevant metrics considered are path smoothness, path length, risk accumulated or time feasibility; if the path can be executed given an inclusionary zone and expected time constraint.

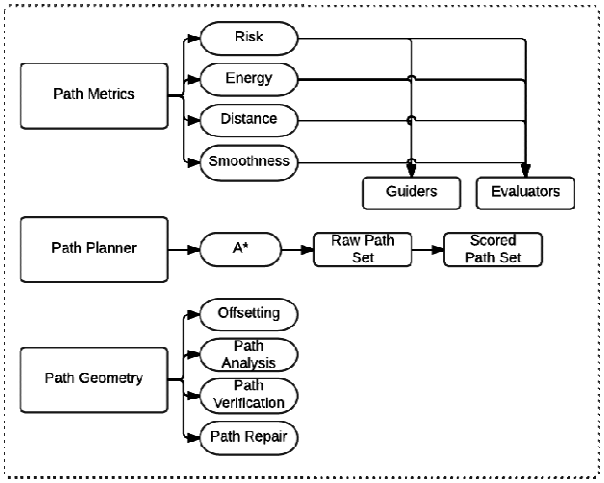

Fig. 1. The Path Generation Module shown composed into three components. The components work to generate and verify the feasibility of the path sets.

II. Path Planner

The path planner module contains graph search algorithms and the set of raw and scored paths when generated. A* is used as the graph search algorithm, traversing a graph that is incrementally generated. The graph is generated by a set of three parameters: velocity, time, rotation. With the parameter selection, the graph is formed by translational extensions (given time and velocity), iterating through rotational increments. Guided by the heuristic estimate, a complete graph representation is not explored; the algorithm traverses amongst the least cost generated states, until it reaches the goal state.

III. Path Geometry

The path geometry component provides four functions. First, path analysis; a set of tools for analyzing the resultant path and its features. This can include intermediary path segment length, path smoothness, or if a given waypoint or set of waypoints are within a distance tolerance from an exclusionary zone. Second, path verification; another set of tools used for confirmation purposes (i.e. the generated path conforming to additional spatial constraints such as CTL). Third is path repair for violations on stricter spatial constraints. The primary repair needed for most paths are for violations in CTL. The final function is inclusionary zone offsetting. This enables redefining the spatial geometry to be further constrained by uniformly offsetting the polygonal-inclusionary zone to limit the search space. The functionality of the path generation module is to provide a selection of metrics and algorithms used for path generation, scoring, verification and repairs.

B. The Soar Cognitive Architecture Module

The Soar module is composed in a similar parallel design. It contains four containers, used for storing pertinent information and decision operations: contextual, path, geometry and planning (see Fig. 2).

I. Contextual Container

The initiation of the hybrid system begins with contextual input in the form of spatial constraints, a goal location and tactical orders. The spatial constraints are inclusionary and exclusionary zones of operation with additional geometric information describing the zone (area, perimeter, number of exclusionary zones in an inclusion zone). The initial information is held within the contextual container and is used throughout the planning process.

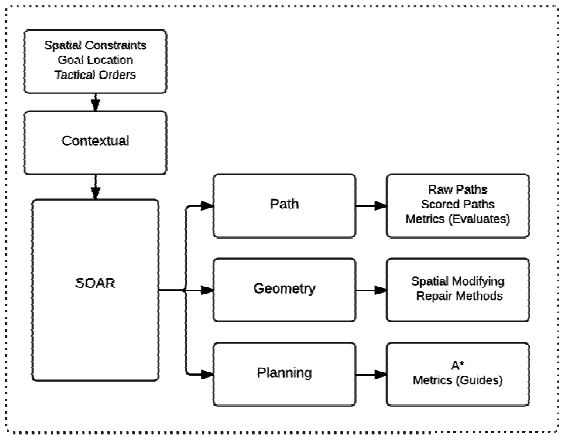

Fig. 2. The Soar Module composed into four containers. The module is bidirectional with path generation, receiving and analyzing paths while directing the overall system operation.

II. Path Container

After path generation procedures have produced either a raw path or a set of raw paths, the path(s) are stored with additional labels, indicating the parameters used to generate each path, a unique identifier and if the path was valid. Metrics used for scoring the path(s) and the decision directives for which metrics should be applied, considering the scenario context, are maintained within this container. After scoring procedures have been used, the final scored sets are stored and another set of decision operators are used to check path validity.

III. Geometry Container

The geometries of the inclusion and exclusion zones are accessed and elaborated upon in this container. Various tools and operators are selected from this container when analyzing or verifying the path feasibility (i.e. CTL are not violated). Decision operators for when to constrain the inclusionary zone, to what extent and for how many iterations are made here.

IV. Planning Container

The planning procedures are maintained within this container; initiated with mission context, in the form of spatial-temporal constraints and tactical orders to reach a given goal location. Decision operators consider the geometric complexity of the operating region by various criteria, such as the density of exclusionary zones, the total numerical area of the operating area or the perimeter bounds and lengths. Depending on the above analysis, the planner will decide whether to produce a single path or a set of paths and then the means to generate appropriate sets. Referencing the tactical orders, the system reasons on parameter selection, tuning and calculation for action-space iterations and operation-area constraining. The planner is enabled to decide, if and when, to interchange the cost function heuristic to consider another metric aside from distance. Path generation can be achieved by the directives described above or a combination of directives, such as interchanging the heuristic, further constraining the inclusionary zone while defaulting the action space parameters to the most ideal calculated values.

IV. Results and Discussion

The hybrid system design enables path planning in varying degrees of complexity while also considering multiple metrics. The system is responsible to mediate on how to best change the input into the path generation module to achieve satisficing paths that meet all constraints and metrics under consideration. The path generation module is designed in such a way to give Soar control and flexibility on how to initialize, limit and increment the action-space parameters, along with methods to constrain the operating area geometry or to change the planning heuristic. Every modifiable parameter in the path generation module produces at least another decision point in Soar. This is the system's purpose, to have a well populated decision space to enable to Soar to produce paths in complex

environments under multiple constraints and metrics. The following results demonstrate various scenarios with either simple or complex planning environments which the system produces paths in light of multiple metrics. A high-level decision tree can be seen in Fig. 3.

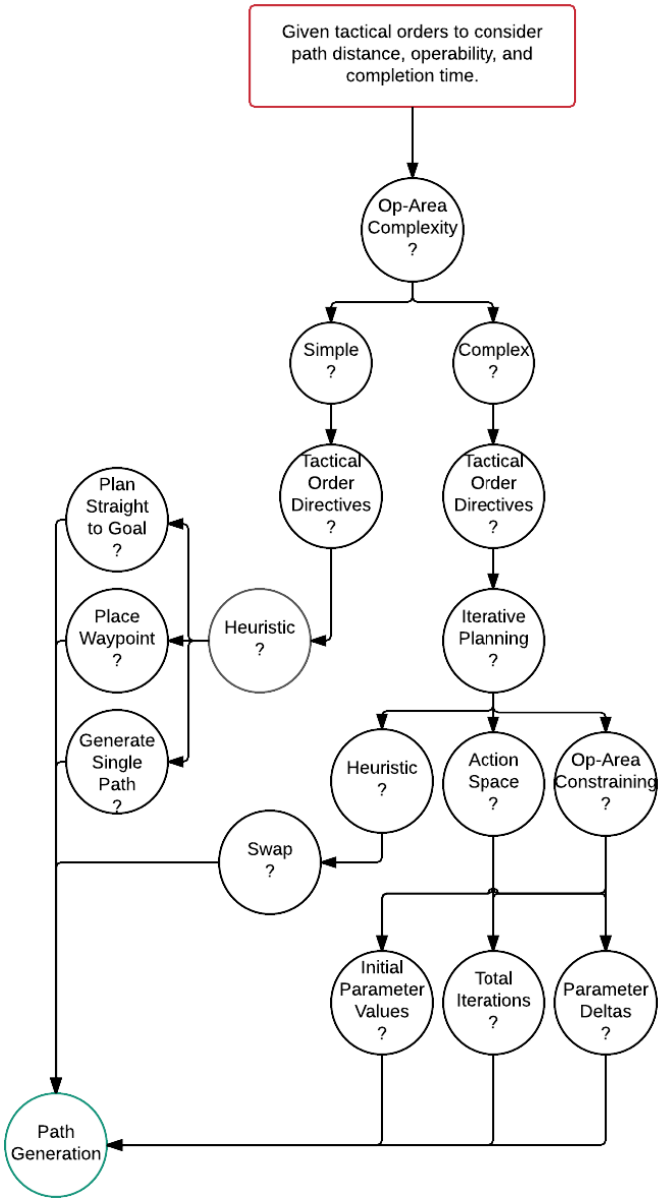

Fig. 3. A high-level view of a subset of the decision space responsible for path generation.

Path generation begins by evaluating the geometry of the inclusionary zone (area, perimeter and verticies defining the space) and the total number of exclusionary zones. The geometry is considered as a precondition to determine if the planning process should result in a single path or multiple paths. A second consideration is made for the tactical orders; which may be either a single order or a combination of orders. For example, given tactical orders to consider path length and a simple geometry will lead to generating a single path. Tactical orders that consider multiple criteria, either in a simple or complex geometry will result in an iterative planning process to produce numerous paths.

For iterative planning, a set of iterations and parameter deltas (i.e. an increment from the intitial velocity parameter) are chosen to iterate through the action-space model of A*.
A similar process for setting iterations and parameter deltas is used to iterate to set the constrained operating area. The action-space model (setting changes in velocity, angle of rotation and time of travel) in A* is flexible enough to plan for complex and dense environments. The Soar module initializes the parameter selection, iterations and deltas for path planning in simple or complex settings. For a simple operating area geometry, defined by four verticies and without exclusionary zones, the module may attempt to plan straight to goal or to use defaulted action-space parameters to generate a single path. If the operating area geometry remains the same but internally, a more dense planning environment with increasing numbers of exclusionary zones occur, the system will choose to use finer-resolution parameters to carefully explore the space. The complexity of the operating area results in a proportional response by the system in setting the action-space parameters. Preliminary results are shown in Fig. 4, for varying numbers of randomly generated exclusionary zones.

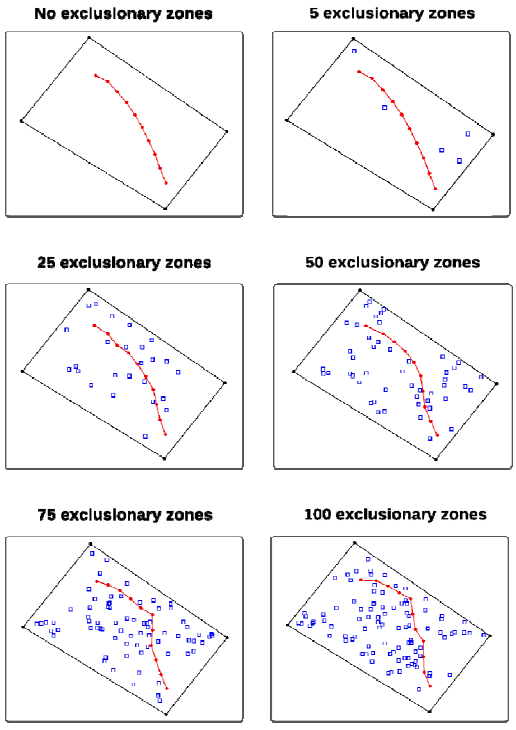

Fig. 4. The hybrid-system reasons on the geometric complexity of the inclusionary zone.

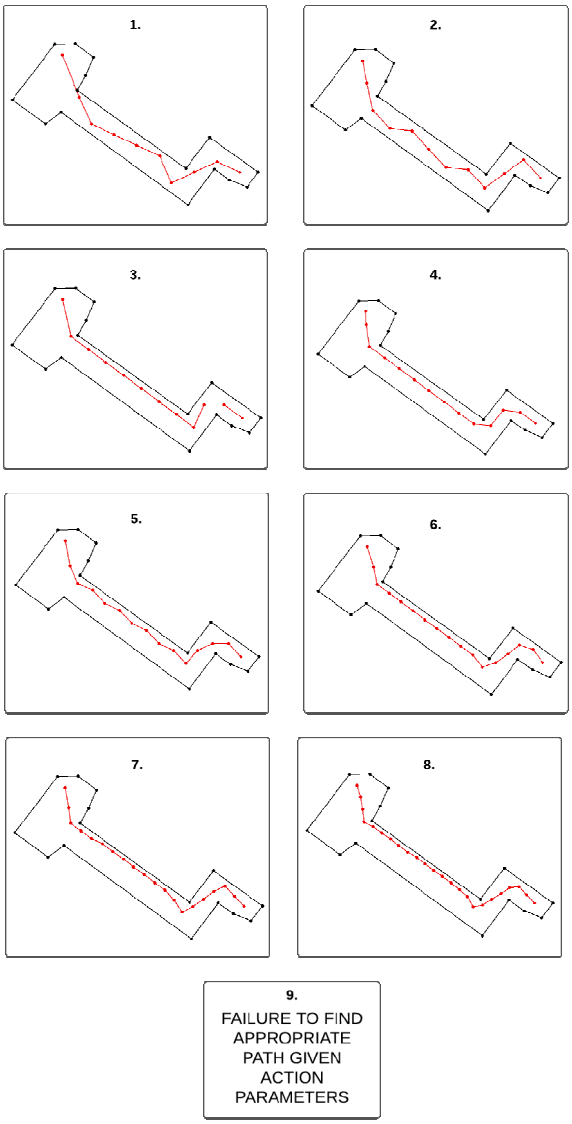

Fig. 5. Resulting paths generated by Soar, given tactical orders to consider path length, smoothness and execution time, Soar reasoned to iterate through action-space to produce eight unique paths, with one failure.

TABLE I. PATH SCORES FOR GENERATED PATHS

Path ID	*Length (m)*	*Smoothness ()*	*Execution Time (s)*
1	74.6924	22.6839	18.6731
2	78.3591	30.5811	19.5898
3	74.7724	20.6300	18.6931
4	73.1028	16.4412	18.2757
5	74.0282	26.7719	18.5070
6	73.4433	14.2364	18.3608
7	74.9683	13.5321	18.7421
8	73.4592	12.4473	18.3648
9	~	~	~

In Fig. 5, tactical orders for path length, smoothness and time of execution are given. The definition of smoothness is given as the interior angle summation in the path. Paths with sharp 90 degree turns, result in poor smoothness scores, while paths that remain linear over several waypoints will result in better scores. The intent is to generate paths that stay on a consistent heading rather than paths that may prove difficult to execute kinematically. To address each metric, the action parameters are iteratively minimized to achieve a finer resolution in creating the graph. For example, by reducing the rotational increment, the system searches for less rigid paths (i.e. 90 or 45 degree sharp turns) through the operating area. Because the orders are not conflicting (i.e. reducing rotational increment will likely reduce both smoothness and length), only the action-space is choosen to be modified. However, a finer resolution while creating the underlying graph does not always correlate to lower metric scores. This can be seen in Table I. The operating area for this scenario along with the generated paths are seen in Fig. 5. The generated failure to find a path results from the parameter selection. If the parameters are initialized at larger values; the system can fail to find a path if the inclusion zone geometry is complex or constraining enough.

The Soar module can also change the heuristic being used within the path generation module. This is primarily used when multiple criteria are being considered that conflict with one another. For example, the order to avoid a designated risk location will conflict with attempting to minimize the path length. To mediate between the conflicting orders, the system will first change the heuristic to drive path generation away from the risk location. Secondly, to consider reducing an opposing metric such as path length, the system can either limit the rotational search (i.e. limit the search to only 120 degree field rather than 360) or constrain the operating area further. By constraining the operating area by uniformly offsetting the geometry, the search will incrementally be shifted closer to the center of the operating area.

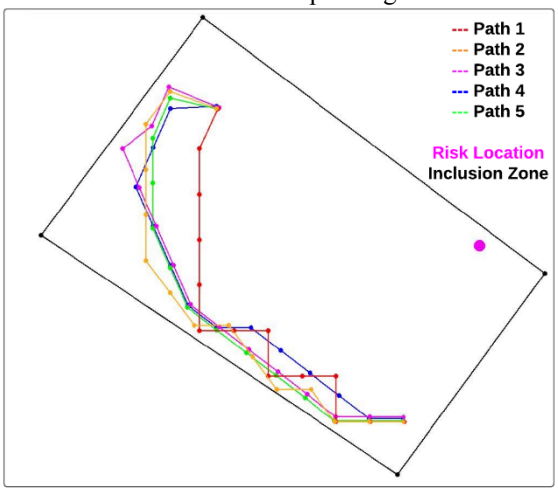

Fig. 6. The system is given tactical orders to consider risk of being observed, path length and path smoothness, as three criteria.

TABLE II. PATH SCORES FOR GENERATED PATHS

Path ID	*Risk ()*	*Length (m)*	*Smoothness ()*
1	76.7811	51.1895	98.1984
2	73.7715	55.3511	46.4473
3	73.3893	55.1928	31.1535
4	74.6422	51.2791	36.0591
5	74.2082	50.6924	36.9710

In Fig. 6, the system was given tactical orders to avoid a risk location with the additional objectives of path length and smoothness. The system reasons to alter the action-space parameters along with changing the distance heuristic to a risk-avoidance heuristic. The generated set of paths and their scores are shown in Fig. 6 and Table II. In this particular scenario and for each metric, a lower score is desired.

V. CONCLUSION & FUTURE WORK

A preliminary hybrid path planning approach, using a cognitive architecture and search methods (A*), demonstrated interaction between a module for high-level reasoning and a traditional path planning solution. The interaction showed that the Soar module can appropriately respond to a range of tactical orders and geometric complexities to generate path sets that aptly respond to the scenario. The system, oriented on a decision space, allows for a generic planning solution that can be easily extended to consider further metrics or other constraints.

Future work will consider using bathymetric data as an additional cost in creating and traversing the graph. Further research will be made on how Soar can reason about real world data to plan in a coarse-grain style then in finer-resolutions depending on the scenario. Additionally, various learning mechanisms in Soar will be explored for the purposes of selecting more appropriate parameter initializations, iterations and deltas during the planning process.

ACKNOWLEDGMENT

The authors would like to acknowledge funding and continuing support from the Office of Naval Research SwampWorks. Additionally, thanks to Ryan Wails for infrastructure support in transferring data between the Soar module and path generation module and thanks to Leslie Aderhold for suggestions in Soar design and implementation.

REFERENCES

[1] J. Crossman, S. Furtwangler, R. Marinier, “Using cognitive reasoning and inorganic data to improve unmanned ground vehicle planning” AUVSI's Unmanned Systems 2013

[2] S. M. LaValle, Planning algorithms. Cambridge university press, 2006.

[3] I. Mitchell, S. Sastry. "Continuous path planning with multiple constraints." Decision and Control, 2003. Proceedings. 42nd IEEE Conference on. Vol. 5. IEEE, 2003.

[4] V. Barichard, X. Gandibleux, and V. T'Kindt, eds. “Multiobjective programming and goal programming: theoretical results and practical applications” Vol. 618. Springer Science & Business Media, 2008.

[5] D. Berenson, P. Abbeel, K.Goldberg. "A robot path planning framework that learns from experience." Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012.

[6] O. Castillo, L. Trujillo, P. Melin. "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots." Soft Computing 11.3 (2007): 269-279.

[7] B. Oleiwi, H. Roth, B. Kazem. "Modified Genetic Algorithm based on A* algorithm of Multi objective optimization for Path Planning." 6th International Conference on Computer and Automation Engineering. Vol. 2. No. 4. 2014.