Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, April 1 - 5, 2002.

Hierarchical Planning Algorithms

Alberto Lacaze

Intelligent Systems Division
National Institute Of Standards and Technology

ABSTRACT

A novel method for communicating commands in hierarchical planning control is presented. This new method can be used to guarantee that plans created will be optimal within and across the planning graphs throughout the hierarchy. Boundary conditions are specified so that optimality is guaranteed. Specific hierarchical planning examples are given for off-road autonomous ground vehicles.

Keywords: Hierarchical Control, Planning, Shortest path, Search, A*, Boundary Conditions, Autonomous Ground Vehicle, Robotics, Operations Research, RCS

1. INTRODUCTION

Planning algorithms have been in the literature for many years.[1] Hierarchical approaches to planning have been studied under the label of operations research for even a longer period of time. There are undeniable advantages to organizing the searches in a hierarchical structure.[2] However, with some exceptions, little attention has been paid to how the levels of the hierarchy communicate with each other. The communication between levels is essential to the overall optimality of the solution as well as to reduce the bandwidth requirements of the system. In this paper we will show how by setting up some simple boundary conditions between levels, overall optimality can be guaranteed for the whole hierarchy.

These communication procedures can be utilized in established hierarchical architectures like the Real-time Control System (RCS).[3–5]

2. HIERARCHICAL PLANNING

Although planning is used for a large variety of applications from daily life to complex autonomous systems, there are some main characteristics that can be identified in all of these processes. Let us define planning as a process of finding a solution to move a system from an initial state to a final state. The state representation will change depending on the applications and may include many variables as well as time.

In general,

- The solution (called a plan) is composed of strings of actions and intermediate states or sub-goals.
- The process of planning entails comparing different possible plans or sub-plans.
- Both initial and final state(s) are given as part of the problem.
- A mean of comparing the results of actions and therefore evaluating the alternatives is given or has to be built.

E-mail: lacaze@cme.nist.gov, Telephone: (301) 975 4456

Certain commercial equipment, instruments, or materials are identified is this paper in order to specify the experimental procedure adequately. Such information is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is intended to imply that the materials or equiment identified are necessarily the best available for the purpose.

We will explore and define in more details each of these different activities that are necessary for planning.

Let us assume that the system in question is in a current state S_i. In real systems, there is a finite number of actions that the system can perform (or that we can assign to the system) $a_1 \ldots a_n$. In order to determine which of these actions is taking us closer to our desired state S_f, we need the means for evaluating the result of those actions. In most cases, there is a simulator that can predict the state that the system moves to after the action is applied. Therefore, the chain $S_i \rightarrow a_i \rightarrow S_{i+1}$. Where S_{i+1} is the predicted state that we plan to achieve if action a_i is applied. Once again, from this new intermediate sub-goal state, we can apply actions and predict their effects.

In order to find out which of these strings of states and actions is the most desirable, we need some means to evaluate them.

Planning algorithms vary in the way that:

- They generate these alternatives.
- They evaluate cost of the plans.
- Constrain the search.
- Pick the actions.

In general, most algorithms compare different plans by summing the evaluation of cost of the triplets (S_i, a_i, S_{i+1}) contained in the plan in question.

In most systems, there is an error associated with the predictions of the next state as well as the computation of the cost of achieving it. As a result of the concatenation, we can assume that the error increases as we get further away from the initial state. This is worsened by the fact that in general, we have more knowledge about the world closer to the S_i and our ability for predication and cost evaluation decreases as we get further away from the initial state in space and time.

$$\lim_{i \rightarrow \infty} Err(S_{i+1})\big|_{s_i,\ a_i} = \infty \tag{1}$$

where $Err(.)$ is the prediction error. There is an argument that all complex planning systems follow the above formula, however the purpose of this paper is not to discuss this issue. We will assume that for the systems in question this is the case.

Since planning is a computing intensive activity, especially the cost evaluation portion, most sensible planners try to study only the alternatives that are likely to lead to the best solutions. Since our knowledge of the world and our ability to predict its outcomes is better closer to the initial state, we propose to change the granularity of the planning algorithm in such a way that it is more refined and more detailed closer to the initial state and coarser further away. This change in granularity should be directly proportional to the lessening of our abilities to predict states and their cost evaluation. Although some rules of thumb can be used for certain systems and state representations, this change in granularity will be strongly dependent on the system in question. In simpler systems where representation is either not highly developed or necessary, the change in granularity is replaced with brute force, ignoring the fact that the confidence in the predicted states changes throughout the planning space.

In systems where the state space is partitioned with heterogeneous granularity, changes throughout the planning horizon, there are some problems in the execution of the plan in the sense that we would like the actuators of the system to receive updates at regular intervals. The obvious solution is a system that re-plans. In other words, as a system executes the first step(s) of the plan, which are of very fine granularity, a new plan is being generated from a new initial state. If the re-planning is done (quick enough), the executor of the plan will always be executing the finer resolution parts of the plan.

In order to simulate and evaluate costs, most systems need a system of representation that allows them to generate the prediction necessary for planning. In some cases this representation may be simple. However in

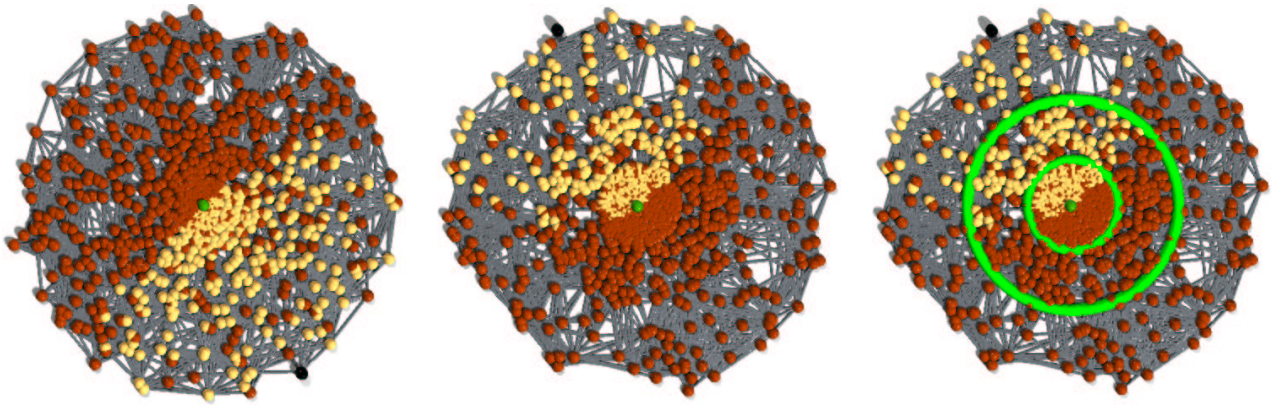

Figure 1. Exponential distribution of nodes within a 2d representation using Dijkstra to search from the periphery to the center

most complex autonomous systems, maintaining and updating a representation suitable for planning is in itself as very complex task. Specifically, the mere fact that these systems make use of re-planning and the granularity of representation changes throughout the space, creates some problems in the representation. In order to solve these problems, control architectures like the Real-time Control System (RCS) create strata or levels of granularity. The advantage of generating theses levels of resolution is that the changes in granularity only occur at the boundaries of the level as opposed to the complete space. In general, this reduces the computational requirements necessary for maintaining a coherent world model representation.

The hierarchical control structure gives the following advantages:

1. Changing granularity between levels to match our prediction capabilities.
2. Fixed granularity at a level that simplifies representation.
3. The ability to plan at different rates at different levels. In general planning at lower resolutions occurs in slower cycles because the predictions are more involved and spaces highly dimensional. Higher resolution levels re-plan quickly and planning spaces tend to be lower dimensional.
4. It is possible to take advantage of the inherent parallelism of the levels and therefore construct distributed computing systems.

In summary, large planning spaces that contain additive costs tend to have an accuracy of prediction and therefore cost evaluation that decreases with the distance from the initial state. Therefore, state selection should be done in such a way that it mirrors the intrinsic granularity of the error propagation. As dealing with a representation that contains constantly changing granularity can become complex, hierarchical system's of decreasing granularity from the initial state naturally emerge as a suitable control architecture. Figure 1 shows an example of this transition. In the first part of the figure, vertices are randomly distributed in a two dimensional space following the assumption that the systems ability to simulate and predict costs is exponential. The second and third part of the figure show an example of how these spaces are subdivided into three levels of resolution. In the last figure, the cross level edges are shown as green rings. In all three cases, Dijkstra's Algorithm is used to search from some node in the periphery to the initial node shown in the center of the screen. The nodes are colored yellow if Dijkstra has opened those nodes and red if otherwise.

The following section of this paper will concentrate on defining some techniques and boundary conditions between different levels of resolution that will help organize the hierarchical planning.

3. HIERARCHICAL PLANNING

3.1. Definitions

Let us define $G^l = (V^l, E^l, \psi^l(.), fr^l)$, $l = 1, \ldots, n$. Be a set of n digraphs of decreasing granularity or resolution. V^l is a finite set of nodes, vertices or states belonging to level l. E^l has ordered pairs, subsets of elements of V^l of cardinality two, called edges ,which represent the actions that take us from state to state in level l. $E^l \subseteq V^l \times V^l$. $\psi^l(e)$ is a function where $e = [v_1, v_2] \in E^l$, v_1,$v_2 \in V^l$ which computes the cost of traversing e. Let us assume that $\psi^l(e) > 0 \; \forall e \in E^l, \; \forall l$. fr^l is the re-planning frequency of level l. In general, $fr^i > fr^{i+1} \; \forall i$.

We can consider that $G^1, \ldots, G^n$ as an approximation of G^* which is the digraph that best approximates the systems decreasing accuracy of prediction and cost evaluation.

A planner is an algorithm $\phi^l(G^l, s, f)$ which returns a directed walk $\mathcal{W}$ through G^l (informally a plan at level l). s is the starting or initial state, and f is the final state. $\phi(G) \triangleq \mathcal{W} = (s, v_1, v_2, \ldots, v_n, f)$ where $v_1, \ldots, v_n, s, f \in V$ minimizing $\sum_i \psi(e_i)$ and $e_0 = [s, v_1]$, $e_1 = [v_1, v_2]$,. . .,$e_n = [v_n, f]$. $\triangleq$ is defined as "returns". $\mathcal{W}$ is an optimal plan if and only if the above summation is smaller than any other walk with the same s and f. $\phi(G) = \emptyset$ if there are no plans from s to f. $\gamma(v)$ is the optimal cost of achieving v from s.

Let us define $EB^{l \to m}$ as the set of boundary edges that connect G^l to G^m. $e \in EB^{l \to m}$ iff $e = [v^l, v^m]$ where $v^l \in V^l, v^m \in V^m$ and $|v^m, v^l| < thr$. These edges connect the graphs created by both levels only in the boundary. One may expect that in general, G^l and G^{l+1} will only overlap in the periphery, as it would not be efficient to represent the same information of the space at different levels of resolution (although it may be done to simplify the representation).

For example, if we imagine that the system that we are talking about is a military hierarchy, we can imagine that a particular level of resolution would be handled by a platoon leader. This platoon leader will generate plans for the movement of the platoon. These plans will consist of very coarse areas where each section may move through. These coarse plans will be given to each section which can be considered another level, and the section leader will create more accurate plans for the vehicles in that particular section. Plans will then once again pass down the hierarchy until a driver will decide how to turn the steering wheel.

The realm of consideration for each level and its supervisor has to include some intersection in the sense that the plans that are sent down the hierarchy must be understood and represented at both levels of resolution. The subordinate is not aware of all the reasons for which the supervisor chooses a particular course of action for his group, or others however it needs a clear understanding of the goal or goals that need to be achieved by its subsystems, as well as the penalties involved in not achieving these goals.

3.2. Optimal Hierarchical Planning

We can create an optimal hierarchical planning algorithm $\phi^{**}(G^{**})$ by defining

$$G^{**} = (\bigcup_{l=1}^{n} V^l, \bigcup_{l=1}^{n} E^l \; \cup \; \bigcup_{l=1}^{n-1} EB^{l \to l+1} \; \cup \; \bigcup_{l=1}^{n-1} EB^{l+1 \to l}, [\psi^1(.), \ldots, \psi^n(.)], fr^1) \tag{2}$$

$s, f \in V^{**}$ are the initial and final states respectively. In general, $s \in V^1$ and $f \in V^n$.

By setting $\psi(EB) = 0$ this multiresolutional graph can then be optimally searched using algorithms like Dijkstra or A*.[6] There are several problems with this approach. First, all levels of resolution are searched at the same time. From experience we know that the general does not need to come up with plans for the whole war at the same rate that the driver of one vehicle controls its steering wheel. Even if the general could re-plan this fast, it would be wasteful to do so. Time would be better spent allowing him/her to try to predict the cost of more alternatives more accurately, or generate plans that have larger time horizons. The inherent parallelism introduced by the hierarchy is not used efficiently by ϕ^{**}.

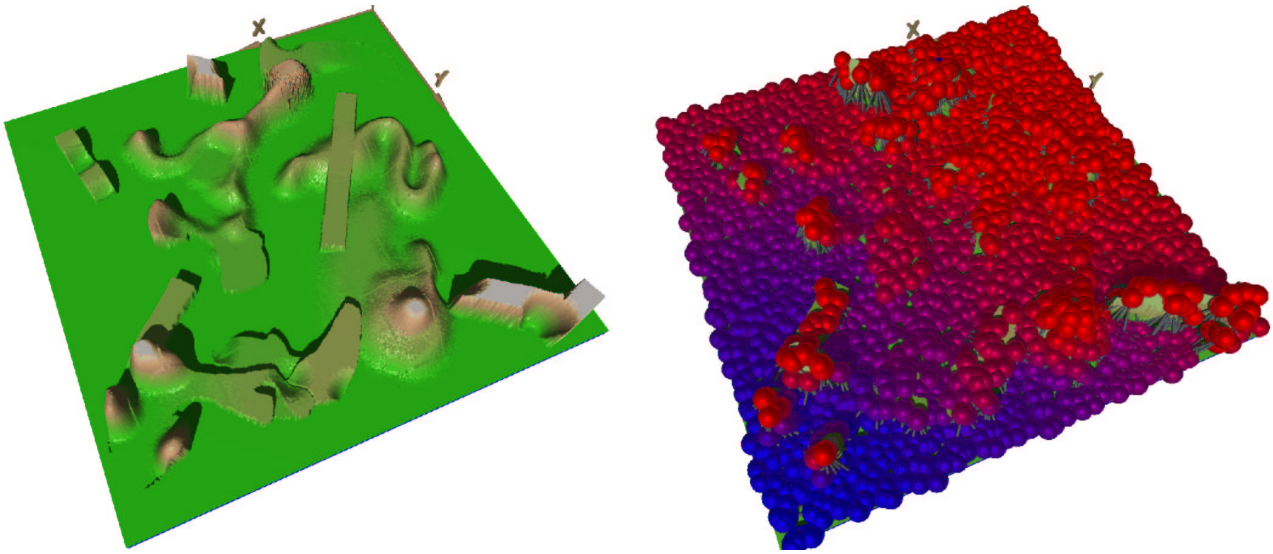

Figure 2: A* planning through an artificially generated state space.

A more reasonable approach is to generate

$$\phi^n(G^n = (V^n, E^n, \psi^n(.), fr^n)) \tag{3}$$
$$\phi^{n-1}(G^{n-1} = (V^{n-1}, E^{n-1}, \psi^{n-1}(.), fr^{n-1})) \tag{4}$$
$$\dots \tag{5}$$
$$\phi^1(G^1 = (V^1, E^1, \psi^1(.), fr^1)) \tag{6}$$

and execute the planning algorithms using different computing facilities. This approach solves the predicament that appeared with $\phi^{**}(G^{**})$ namely,

- Each $\phi^i(.)$ can be executed at its own frequency fr^i, and therefore higher resolution levels are executed at much higher frequencies.
- Each $\phi^i(.)$ can run on a separate computer hardware.
- Bandwidth and data storage requirements are diminished to the resolution required for planning at the level.

However, a new problem appears. Since $s \in V^1$ and $f \in V^n$, algorithms $\phi^2, \dots, \phi^{n-1}$ do not have s and f at their levels to start and stop their searches.

We will now present a method for searching in this structure. We will assume that an algorithm such as A* is being used to search. Other algorithms can be applied in the same fashion. Figure 2 shows an artificially generated terrain. Although this is a two dimensional example, larger dimensionality spaces can be used by the same algorithms. The second part of the figure shows a graph generated on top of the surface, with the color of the spheres representing the cost of achieving this point from the bottom corner (blue is cheap and red is expensive). It is assumed for the example that the cost of traversing is Euclidean distance plus a term that makes slope exponentially more expensive to traverse.

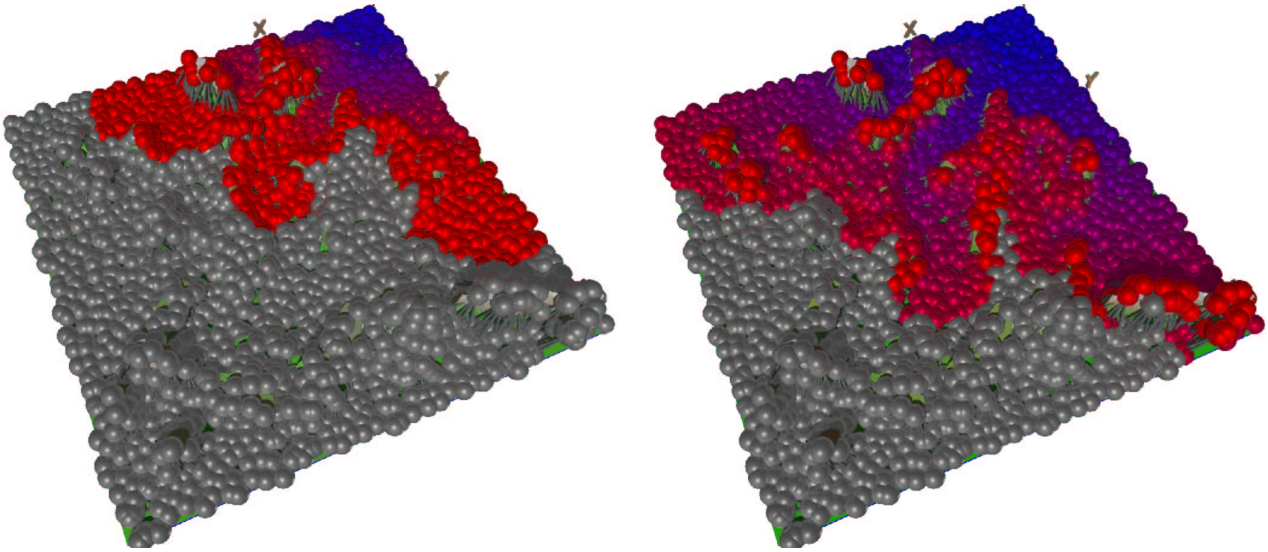

Figure 3: A* planning from a corner to the connections to other levels.

3.3. Lowest Level of Resolution

In the same manner as most hierarchical architectures, planning starts at the lowest level of resolution. At this lowest level of resolution n the final goal $f \in V^n$ is available and we have all the $EB^{n \to n-1}$ that leave this level to their subordinate levels. However, there is no starting state $s \in V^1$. Therefore, A* is run starting from f and stopped iff,

$$\text{open}(v_i) = \text{true, if } \exists\ EB^{n \to n-1} = (v_i, v_j) \in E, v_i \in V^n, v_j \in V^{n-1} \tag{7}$$

In other words, the search will start at f and continue until all the nodes that contain edges connecting it to the higher level of resolution have been explored. Depending on the overlap between the levels this will add some overhead to the overall planning strategy. Figure 3 shows the difference between the planning generated by starting the search at $f = (\frac{length}{2}, \frac{width}{2})$ and stopping it at s (first figure) and stopping it following Equation 7 (second figure). In this case it is assumed that the higher level of resolution will be a square patch 1/10 the length of this levels length (only one level is shown in these figures). s is at the center of both levels at the beginning of the process. Grey nodes show nodes that have not been opened.

In these figures, this level has a large overlap with the higher level of resolution, however, the levels are only connected in the periphery. In other examples we will show cases where the levels only overlap in the periphery.

3.4. Intermediate Levels of Resolution

These intermediate levels $m = 2, \ldots, n-1$ in general do not have neither s nor f, however, they are connected to the lower level of resolution by $EB_{up}^{m \to m+1} = (v_i, v_j) \cup EB_{down}^{m+1 \to m} = (v_k, v_i) \in E, v_i \in V^m, v_j \in V^{m+1}, v_k \in V^{m-1}$.

Since the upper level $m+1$ has already planned, the costs from f to every $v_i \in EB_{down}^{m+1 \to m}$ have been calculated by the upper level. Therefore, A* is initialized as follows:

$$\text{open}(v_i) \doteq \text{true, if } \exists\ EB_{down}^{m+1 \to m} = (v_j, v_i) \in E, v_i \in V^{m+1}, v_j \in V^m \tag{8}$$

$$\gamma(v_i) \doteq \gamma(v_j)\ \forall v_i \in EB_{down}^{m+1 \to m} \tag{9}$$

where $\gamma(v_j)$ is the cost of traversing from v_j to f. "$\doteq$" is defined as assigned. To summarize, the A* search at this level is started by placing in the open list all vertices in the periphery (that are connected to the upper

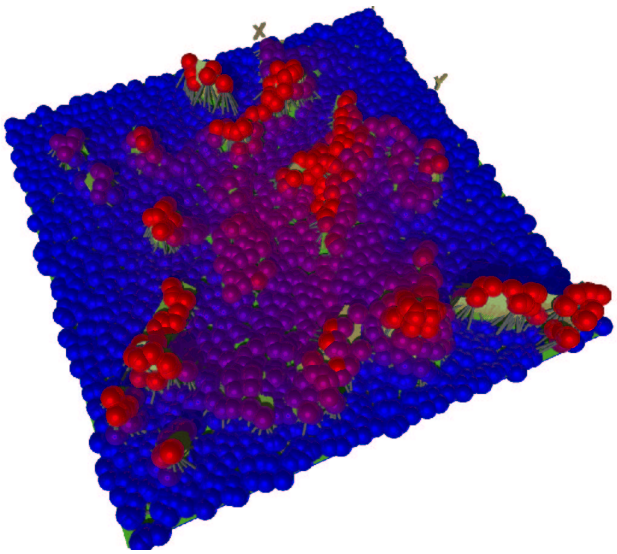

Figure 4: A* planning for intermediate levels.

level), and assigning the cost of achieving that vertex from the results of the upper level planning. The same stopping criteria as shown in Equation 7 is used to stop the search.

Figure 4 shows an example of one intermediate level. In this case all the periphery of the level is cheap (blue upper levels costs where small), and it gets more expensive towards the center (connection to the higher resolution level).

3.5. Highest Level of Resolution

In general s is contained in this level, however, f is not. Therefore, planning at this level is initialized as shown in Equation 8. However, the standard A* termination procedure is used. The planning is stopped when s is achieved opened.

3.6. Optimality

Although the algorithm presented in the previous section solves many of the problems introduced by the hierarchical nature of the problem, as presented the algorithm is not optimal. A case where the algorithm will give different results from the optimum is shown in Figure 5 (left). The example shows two levels of resolution, the road is more accurately detected in the higher level than in the lower level. Let us assume that the cost of traversing the area outside the road is orders of magnitude more expensive than following the road.

Following the algorithm, the search will start at f and continue until all the points in the boundary are opened. In this case, the cheapest vertex in the boundary would be a, however, vertices b and c would be marked incorrectly as being expensive since the low level of resolution is not aware of the segment ab (contained in the high level of resolution). Therefore, the result of planning to s from the boundary will yield a path that starts at a and goes off-road following eg, before achieving s.

In order to guarantee optimality, an extra step must be incorporated to the algorithm. It is clear from the example, that once the higher level of resolution is done searching, the correct cost for b is available. The reason for this is that since a is cheaper than any other vertex in the boundary A* will start expanding towards e along the road. Once e is achieved, A* will continue to go along the road and update the cost of b. Therefore, the cost of b will always be computed before g is reached.

Therefore, if the higher level of resolution passes the results of its search to the lower level of resolution, the value for b can be updated in the lower resolution search (i.e. placing b back in the open list with the updated

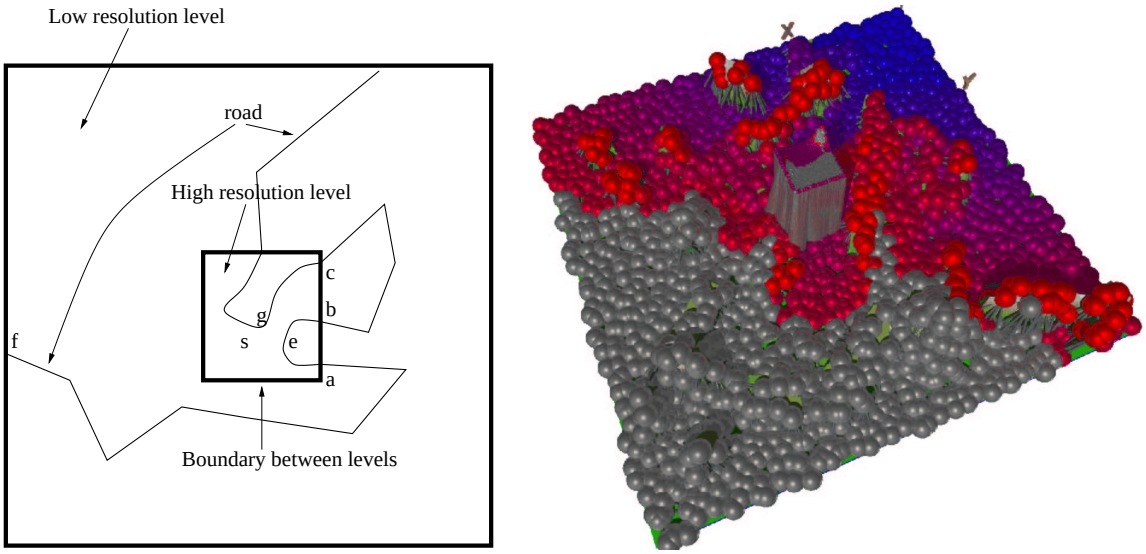

Figure 5: Finding optimal paths across multiple levels

cost), then , the cost for c will be automatically correct. Once this happens, cgs will become cheaper than $aegs$ for the higher level of resolution.

In the case of A*, it is clear that after a few iterations the boundary values will converge to the values that would have been achieved if one large graph including both levels would have been created. The costs in the boundaries cannot oscillate because, A* will only open nodes if they have lower cost, rather than using the previous computation. Therefore, the cost of all boundaries is monotonically decreasing. Note, that it is not necessary to restart the search at the levels, only propagate the newly computed boundary cost calculations. Although the number of computations is larger than in the case of a single search ($\phi^{**}(G^{**})$), the parallel execution of the levels still make the execution time smaller.

Formally, level $m+1$ is restarted with:

$$\text{open}(v_i) \doteq \text{true, if } \exists\ EB_{up}^{m \to m+1} = (v_j, v_i) \in E, v_i \in V^m, v_j \in V^{m+1} \text{and} \tag{10}$$

$$\gamma(v_i) < \gamma(v_j) \tag{11}$$

$$\text{then } \gamma(v_j) \doteq \gamma(v_i) \tag{12}$$

The second part of Figure 5, shows two levels of resolution connected at the boundary. The terrain in the higher level of resolution was elevated in the figure for display purposes. The colors show cost normalized across the levels. Following A* back-pointers gives the optimal path through the multiresolutional graph.

In summary, the supervisor levels send not only a state (goal) to achieve to their subordinates, but a boundary of states with the corresponding costs. In turn, the subordinate levels give to the supervisor a better estimate of the costs of achieving them, and the decision of which path to take is completed at all levels simultaneously.

4. SIMULATED EXAMPLE

Figure 6 shows the progress of re-planning two levels of resolution in an artificially generated terrain. In this case, the lower level of resolution is not re-centered around s. The three pictures display the higher level of resolution connecting to the lower level of resolution in different locations following the path given by the lower level of resolution.

In order to evaluate the optimality of the results, the answer generated by the two level system was compared to a single level system generated with the granularity given by the higher level of resolution. Since this is an

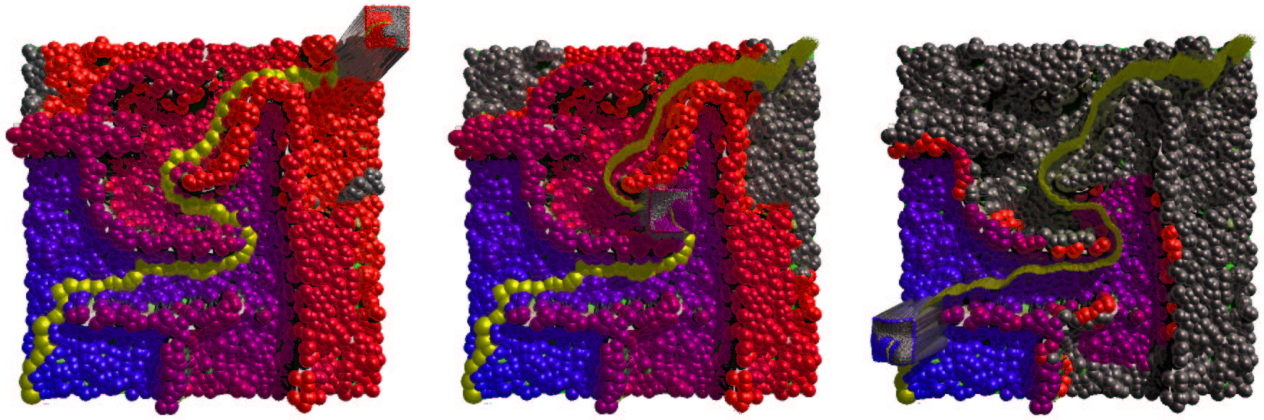

Figure 6: Progress of the re-planning algorithm with 2 levels of resolution

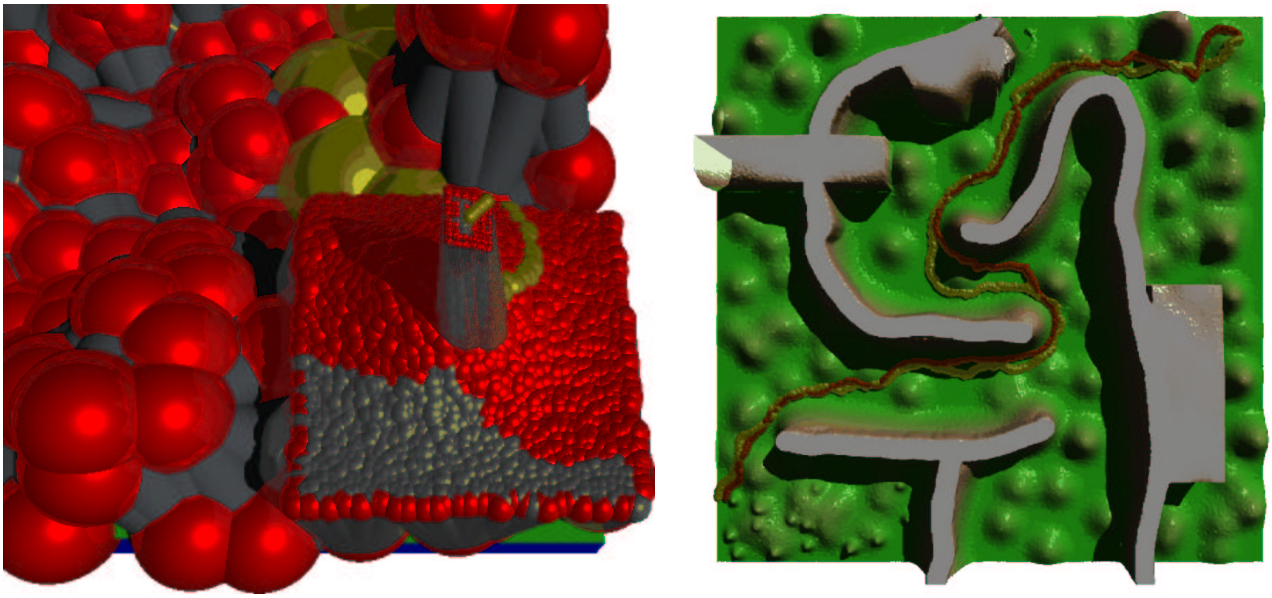

Figure 7. a)A 3 levels optimal hierarchical planning structure, b)Comparing the results of the 3 level system with the maximum allowable single level system

artificially created example, the costs are assumed to be known throughout the graph. The resulting plans are equivalent, with the difference that the execution time as well as memory usage of the two level system was more than an order of magnitude faster(smaller) than the single level system. To generate the single level graph the resources of a Sun Ultra 80 where stretched to the limit (1Gb RAM + 1Gb swap).

Figure 7(a) shows the same algorithm applied to a three level structure. The added level of resolution generates paths that are more accurate than the two level system. The results of the three level system cannot be computed in a single level structure due to the fact that it would require number crunching capabilities that would not be available in many years. However, by dividing the system into 3 levels, the results can be obtained in seconds.

Figure 7(b) compares the paths generated by the largest resolution single level system and the three level system. The difference in cost between the paths paths are about three times the vertex separation for the

Figure 8: XUVs autonomously traversing challenging terrain at Ft. Indiantown Gap, PA

artificially generated terrain. This means that the three level system has $\frac{1}{10}$ the error of the single level system since the resolution of the highest level of resolution in the three level system is an order of magnitude larger than the single level system. These values will change for different cost evaluations.

5. EXAMPLE: THE DEMO III PROJECT

Figure 8 shows eXperimental Unmanned Vehicles (XUVs) traversing unstructured off-road terrain at Ft. Indiantown Gap, PA.

The XUVs use a hierarchical controller that follows the NIST Real-time Control System, RCS, methodology. Sensor processing modules sense the state of the vehicle and the surrounding environment. Sensory data flows to world modeling modules which update the estimated state of the vehicle and its surroundings. The world model is used by behavior generation modules to plan actions and to execute the resulting plan. Planned paths and actions are stored in the world model and can be used by the sensor processing modules to direct sensor attention or processing cycles to locations in the environment that are more critical.

Sensor processing modules at every level gather and process information coming from multiple sensor. The sensed information is fused in world modeling modules which aid the planners at each level to generate behavior. Four levels of the hierarchy are implemented: Servo, Prim, Autonomous Mobility (AM), and Vehicle levels.

A LADAR (a laser scanner) produces a range image from which terrain shape and non-traversable obstacles are detected.

This sensor feeds both the AM and Prim levels with range information populating the terrain and obstacle maps. Due to the nature of the sensor, the information closer to it is more accurate and dense and therefore

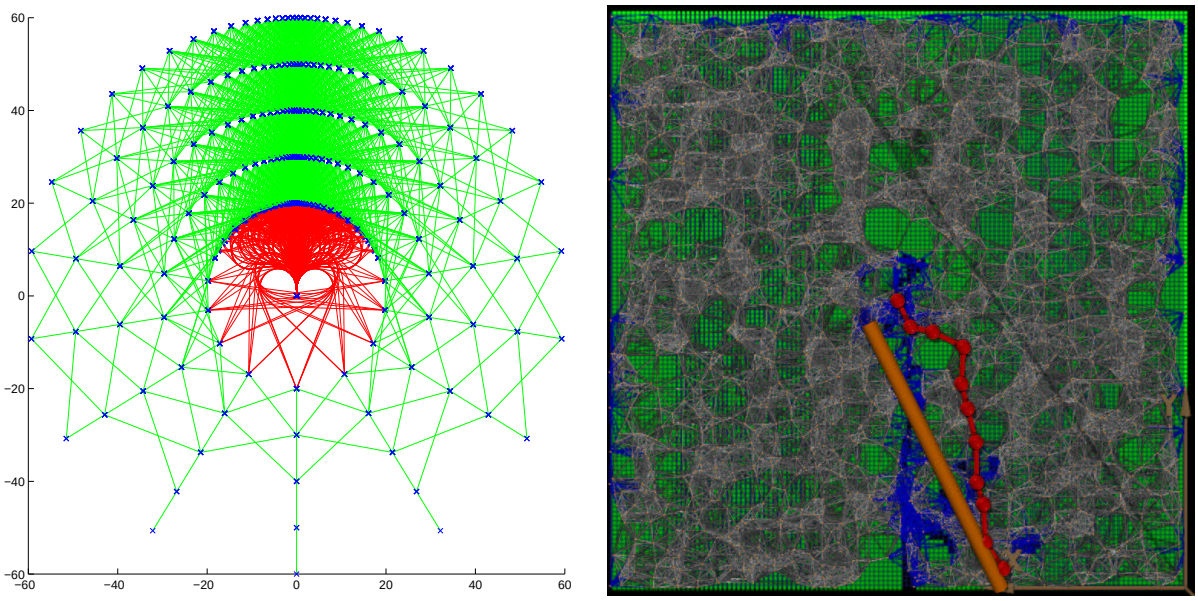

Figure 9: Three levels of resolution used by the XUVs control system

placed at the Prim's world model (within 20m). Further away (past 20m) the information is coarser and no ground returns are received. This information is used at the AM levels. The planners at all levels compute the lowest cost obstacle free path that drives the vehicle to the path commanded by the level above. The AM level cycles at 10 Hz. Lower levels then compute steering and speed commands and servo the electric actuators.

At the vehicle level, sensed obstacles are combined with a priori maps in a 500 m map. The vehicle level planner then selects the lowest cost path that achieves the mission goals that were specified by a human operator. The vehicle level re-plans at 1 Hz.

Figure 9 shows the planning graphs at the Prim (red left), AM (green left), and Vehicle (right) levels. The Prim graph is based on kinematically correct paths computed off-line that take under consideration the initial conditions. Several sets of these graphs that are used depending on the initial conditions (i.e. velocity, current radius of turn). At the AM level, the graph is sparser, although covering more terrain. At this level, the paths are represented as concatenated straight segments. Kinematics and dynamics are not as important since path will be re-planned before the vehicle reaches the boundary between the AM and Prim levels. At the Vehicle level, the graph is built using vertices placed at particular points in the terrain that are of some importance (i.e. roads, buildings, corners), and other points are randomly placed. The placement of the vertices is even sparser than in the AM.

Each level re-plans at different rates as previously shown. The results of using the XUVs for Demo III has been presented in.[7, 8]

6. CONCLUSION

An innovative hierarchical planning algorithm is presented where optimality is guaranteed not only on a level by level basis, but also across levels. The algorithm communicates boundary conditions between levels of resolution which in turn are used to seed and restart the parallel searches at each level.

An artificially generated 3 level system was used to optimize a 2 dimensional shortest path problem. A real world scenario of controlling autonomous vehicles was used to test the planning algorithms.

ACKNOWLEDGMENTS

This research is funded by the Demo III project, Army Research Lab, US Army.

REFERENCES

1. T. Perez-Lozano, "Spatial planning: A configuration space approach," *IEEE Trans. Computers* **32**, 1983.
2. Y. Maximov and A. Meystel, "Optimum design of multiresolutional hierarchical control systems," in *Proceedings of IEEE Int'l Symposium on Intelligent Control*, *Proceedings of IEEE Int'l Symposium on Intelligent Control* , pp. 514–520, (Glasgow, U.K.), 1992.
3. J. Albus, "Outline for a theory of intelligence," *IEEE Transactions on Systems, Man, and Cybernetics* **21**, pp. 473–509, 1991.
4. J. Albus, "RCS: A reference model architecture for intellignet control," *COMPUTER, Special Issue: Computer Architecture for Intelligent Machines* , pp. 56–59, May 1992.
5. A. Lacaze, "4-D/RCS reference model architecture for unmanned ground vehicles," in *SPIE*, **3693**, (Orlando, Fl.), 1999.
6. R. Dechter and J. Pearl, "The optimality of A*," in *Search in Aritficial Intelligence*, L. Kanal and V. Kumar, eds., *Symbolic Computation*, Springer-Verlag, 1987.
7. K. Murphy and A. Lacaze, "Intelligent control for off-road driving," *International Congress on Autonomous Systems* , 2002.
8. A. Lacaze, "ierarchical architecture for coordinating ground vehicles in unstructured environments," *International Congress on Autonomous Systems* , 2002.