PARALLEL PLANNING IN PARTITIONED PROBLEM SPACES

Stephen Balakirsky*
Otthein Herzog†

**National Institute of Standards and Technology, Gaithersburg MD*
†TZI – Center for Computing Technologies, University of Bremen, Bremen Germany

Abstract: This paper presents a complete and optimal framework for extending basic graph planning to operate in partitioned problem spaces. These spaces typically occur in systems that implement a hierarchy or contain data of various resolutions. An algorithm for the framework will be presented along with a proof of optimality. Finally, an example implementation for mobile robot path planning will be discussed.

Keywords: planning; parallel algorithms; autonomous mobile robots; graph theory; architectures

1 INTRODUCTION

There are many examples of problem decomposition into distinct planning regions in the current literature. One common occurrence of regions is in the creation of a planning space hierarchy. Fernández and González [2,3] have presented a world view that has been decomposed into the hierarchical graph model. In the graph model, high-resolution nodes are abstracted to form a smaller number of lower resolution super nodes. Each super node is then connected to form a planning graph and planning is performed on this low-resolution representation of the space. Fernández and González present a technique for connecting and searching these graph structures, which may contain any number of hierarchical levels, that is based on the classic refinement method. However, the authors state that this technique is not guaranteed to find the cost optimal path ([3], p. 106).

Another view of hierarchical planning is presented by Lacaze [4] for mobile robot applications. In this paper, the planning space is viewed as having a high node density near the robot with decreasing density as the distance to the robot increases. Forming regions of equal node count as shown in Figure 1 creates the hierarchy. A region of high node density exists around the robot's current location where high-resolution sensor information is available. Further from the robot, the sensors are only able to detect large objects and the corresponding node density decreases. Finally, once the limits of the sensors have been exceeded, the node density drops to correspond to the feature density represented in the a priori dataset. Through this technique an optimal plan may be computed for each individual region. However, the final plan will be piecewise optimal and is not guaranteed to be optimal when taken as a whole.

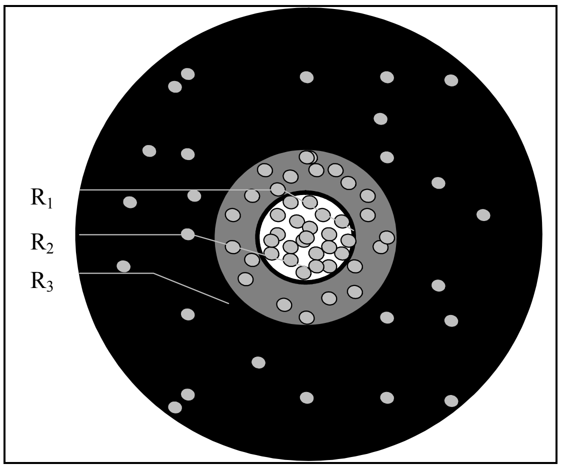

Figure 1: Example of node density for 2-D mobile robot application based on Lacaze [4].

One of the benefits of this hierarchical decomposition is that in time-constrained environments a path may be quickly found through the low-resolution graph and executed as a sub-optimal plan. If more time is available, individual sections of this path are expanded and a new refined plan is constructed that is a refinement of the previous path. Through the use of this technique, this algorithm may act as an anytime algorithm [5].

In this paper, a new technique will be explored that allows for the creation of an arbitrary number of irregularly shaped and placed regions. Regions of this form have been encountered by NIST while developing an autonomous vehicle for on-road driving as part of the ARL Demo 3 and DARPA MARS programs. This technique, which builds upon existing graph-based planning techniques, will be shown to be applicable to multi-agent systems and will be proven to be capable of providing the overall optimal solution to the planning problem. The algorithms discussed in this paper are an extension to thesis work performed at the University of Bremen that may be found in [1].

2 FRAMEWORK DEFINITIONS

Before the framework for planning in partitioned problem spaces may be described in detail, certain basic graph-planning definitions must be established. This section will present definitions that apply equally to traditional hierarchical planning techniques and the partitioned planning framework. Section 2.2 will then extend these definitions to meet the unique needs of the partitioned problem framework.

2.1 Graph-Based Planning Definitions

The basic planning problem may be described as the desire to find a cost optimal sequence of transitions for the task of transitioning from one element of a discrete planning space $\mathbf{S} \subseteq \mathbb{N}^n$ (denoted by $s_{\overrightarrow{start}} \in \mathbf{S}$) to another (denoted by $s_{\overrightarrow{goal}} \in \mathbf{S}$).

A node space **V** that allows for the time sequencing of the elements of **S** by an ordered sequence of points T, the set of arcs A that represents a binary relation on the space **V,** the graph G, and various graph properties may now be defined.

The *node space* V is defined by (SxT) where $S \subseteq \mathbb{N}^n$, $T = (t_1,\ldots,t_m)$, $m \in \mathbb{N}$. An element of V is defined as $v_{k,i} := s_{\vec{k},t_i}$, $\vec{k} \in \mathbb{N}^n$, $k \in \mathbb{N}$, $i \in \mathbb{N}$. (D1)

An *arc* A is defined as a binary relation $a = (v_{k,i}, v_{j,i+1})$. (D2)

The mapping $c: A \rightarrow \mathbb{C}$ is called the *cost function* of an arc. One typical specification for $\mathbb{C}$ is $\mathbb{C} = \Re^+$. (D3)

The *spanning set* of a node $v_{k,i}$ denoted by $SP(v_{k,i})$ is defined as the set of nodes that are reachable through a single arc. Members of the spanning set are also denoted as the successors of the node $v_{k,i}$. (D4)

A *graph* G is defined as G = (V, A). (D5)

Given a graph G = (V, A), and nodes $v_{k,i}, v_{j,i+1},\ldots, v_{m,i+n} \in V$, $i, j, k, n \in \mathbb{N}$. A *path* p is defined by $p = (v_{k,i}, v_{j,i+1},\ldots,v_{m,i+n})$ such that two adjacent nodes of p constitute an arc $a \in A$. P is defined as the set of all paths p in G. The length l of p is defined as the number of arcs of the path. (D6)

The *path cost* associated to a path p is defined as

$$pc = \sum_{j=0}^{m-1} c(a(v_{j+1,i+j}, v_{j+2,i+j+1})). \quad (D7)$$

The system objective of finding the cost optimal path from $s_{\overrightarrow{start}} \in \mathbf{S}$ to $s_{\overrightarrow{goal}} \in \mathbf{S}$ may now be achieved by finding the minimum cost path between the two nodes $v_{start,i}$ and $v_{goal,j}$.

Define the *minimum cost path* p_{min} between the nodes $v_{j,i}$ and $v_{k,i+n}$, $n \neq 0$ and $i, j, k, n \in \mathbb{N}$, such that $\forall p \in P : p = (v_{j,I},\ldots,v_{k,i+n}) : pc(p_{min}) \leq pc(p)$. (D8)

2.2 Framework Specific Definitions

In addition to representing an element of the planning space $s_{\vec{k},t_i} \in \mathbf{S}$, a node may be said to represent the features that are present in $s_{\vec{k},t_i}$. These features are represented by attributes attached to the nodes $v \in \mathbf{V}$.

Define ATT as the set of node ***att**ributes* by $ATT = \{att_{k,i} \mid att_{k,i} \text{ is a attribute of } v_{k,i} \in \mathbf{V}\}$. (D9)

The set ATT defines a set of attributes that may be identified in the given planning problem and mapped onto a single node. An example would be: $ATT_{k,i}$ = {"contains data from sensor 1", "member of area-of-interest 2", "high-probability of obstacles"}.

An annotated node space $\mathbf{V}^a$ may be defined that combines the node space **V** and the set of attribute sets **ATT**.

An *attributed node space* $\mathbf{V}^a$ is defined as $\mathbf{V}^a := \left\{ \mathbf{V} \times \mathbf{ATT} \middle| v^a_{k,i} = \left(v_{k,i}, ATT_{k,i}\right) \right\}$. (D10)

Given the existence of attributes, it is possible to define a unary relation r that partitions **V** according to specific attributes. For the above example, the relation r may define membership in a partition of $\mathbf{V}^a$ by requiring the existence of data from sensor 1. This will define a closed circle of a certain radius centered on the sensor. A *region* $\mathbf{V}^{a,r}$ may then be defined as the subset of $\mathbf{V}^a$ where the relation r holds.

Given a relation $r \subseteq ATT$, a *region* $\mathbf{V}^{a,r} \subseteq \mathbf{V}^a$ is defined by $\{v_{k,i} \in \mathbf{V}^a \mid R(ATT_{k,i}) = true\}$. (D11)

Additionally, a set of regions that contains all of the nodes from $\mathbf{V}^a$ will be defined as a *complete* set of regions R.

The set of n regions $\mathbf{V}^{a,R}$, $R = \{r_1, r_2,\ldots,r_n\}$ is *complete* iff $\bigcup_{i=1}^{n} \mathbf{V}^{a,r_i} = \mathbf{V}^a$. (D12)

It can be shown that any node $v \in \mathbf{V}^a$ is also a member of the complete set of regions R.

Lemma: If $v \in \mathbf{V}^a$ and $\mathbf{V}^{a,R}$, $R = \{r_1, r_2, \ldots, r_n\}$ is a complete set of regions, then $v \in \mathbf{V}^{a,R}$. (L1)

Proof: Assume $v \notin \mathbf{V}^{a,R}$, then by (D12), $\mathbf{V}^{a,R} = \bigcup_{i=1}^{n} \mathbf{V}^{a,i} = \mathbf{V}^a$ and $v \notin \mathbf{V}^a$ which contradicts the original assumption.

Given the above definition of a region, it is possible that graph nodes exist that are members of multiple regions. These shared graph nodes form the boundary between the regions. The boundary nodes may have members of their spanning set that are located in the boundary, or in either region that forms the boundary. In order to provide transitions between regions, the partitioned planning framework requires that boundary nodes contain successors that are uniquely located in each of the regions that form the boundary. The formal requirements on the regions for the partitioned planning approach are defined below.

Given the set of n regions $R = \{r_1,\ldots,r_n\}$, the *boundary set* of any two regions c, d is defined as $B_{c,d} := \{ v_{k,i} \in \mathbf{V}^a \mid v_{k,i} \in \mathbf{V}^{a,c} \wedge v_{k,i} \in \mathbf{V}^{a,d} \}$. (D13)

A boundary set $B_{c,d}$ is called a *partitioning boundary set* iff $\forall\ v_{k,i} \in B_{c,d}, \exists\ v_{m,i}, v_{n,i} \in SP(v_{k,i}) : v_{m,i} \in \mathbf{V}^{a,c}, v_{m,i} \notin \mathbf{V}^{a,d} \wedge v_{n,i} \in \mathbf{V}^{a,d}, v_{n,i} \notin \mathbf{V}^{a,c}$. (D14)

A node space $\mathbf{V}^{a,R}$, $R = \{r_1, r_2, \ldots, r_n\}$ is called a *partitioned planning space* iff $\mathbf{V}^{a,R}$ is complete and $B_{c,d}$ is a partitioning boundary for all $c,d \in \{1,\ldots,n\}, c \neq d$. (D15)

The spanning set of the nodes of the partitioning boundary set may now be specialized on a regional basis. It will be shown that the union of these regional spanning sets is equal to the spanning set defined in (D4).

Define the *regional spanning set* SP^i of a node $v_{k,i}$ as the set of all successor nodes
$SP^i(v_{k,t}) =$
$\{v_{j,t+1} \mid \forall j \in \mathbb{N} : v_{j,t+1} \in sp_{k,t} \wedge r_i(ATT_{j,t+1}) = \text{true}$ (D16)

Lemma: $\forall\ v_{k,t} \in \mathbf{V}^a$, $SP(v_{k,t}) = \bigcup_i SP^i(v_{k,t})$ where $R = \{r_1, r_2, \ldots, r_n\}$ is complete. (L 2)

Proof: (D16) may be rewritten as $\bigcup_i SP^i(v_{k,t}) =$
$\{v_{j,t+1} \mid \forall j \in \mathbb{N} : v_{j,t+1} \in sp_{k,t} \wedge \bigcup_i r_i(ATT_{j,t+1}) = \text{true}\}$

From (D12) and the fact that R is complete, $\bigcup_i SP^i(v_{k,t}) = \{ sp_{k,t+1} \mid sp_{k,t+1} \in \mathbf{V}^a\}$. By (D4) all members of the spanning set are connected by an arc and must therefore be elements of $\mathbf{V}^a$, and the definition reduces to $\bigcup_i SP^i(v_{k,t}) = SP(v_{k,t})$.

3 PARTITIONED PLANNING ALGORITHM

Figure 2 describes the planning algorithm for operation over a partitioned planning space. A separate version of this algorithm will be executed for each of the partitions $r_i \in R$, $i = \{1,2,\ldots,n\}$. These algorithms may be run serially, or may be executed in parallel with no effect on the final complete solution as will be shown in Theorem 1.

Step (1) of the algorithm establishes a maximum planning cost C^{max}. An error and planner termination will result if the planner is unable to find a plan of cost less than C^{max}.

The main change to a basic graph search that is necessary for operation over partitions is the inclusion of the boundary nodes into the algorithm framework. These boundary nodes are made part of the goal set for the partition, and must be reached in regions that do not include the goal before the algorithm can terminate.

In order to include the boundary as part of the goal set, step (3) forms the new goal set G_r^* that includes the systems goals as well as the boundary regions.

Step (4) selects a graph node to be opened. Any optimal and complete graph search algorithm may be applied here.

```
Planning algorithm for partitioned planning space $\mathbf{V}^{a,R}$ where $R = \{r_1,\ldots,r_n\}$.
1) Initialize planning open threshold $C^r$ to maximum allowable plan cost $C^{max}$ (may be $\infty$).
2) If region contains $v_{start,t_0}$, insert this into region's open set $O_r$.
3) If region contains one or more goals, $v_{goal,t_g}$, insert them into the region specific goal set $G_r$.
   a) Form the set
        $G_r^* = G_r \cup B_{r,j}, \forall j \in \{1,\ldots,n\}$.
4) Determine next node to open.
   a) If $O_r = \varnothing$, go to (3a).
   b) Else, using the graph search technique of your choice, select and remove a node $v_{k,t_i}$ from $O_r$.
5) Evaluate $v_{k,t_i}$.
   a) If $PC(v_{start,t_0}, v_{k,t_i}) \geq C^r$, goto (8).
   b) If $v_{k,t_i} \in G_r^*$, remove $v_{k,t_i}$ from $G_r^*$. If $G_r^* = \varnothing$, set $C^r = PC(v_{start,t_0}, v_{k,t_i})$ and go to (8).
6) $\forall\ v_{m,t_{i+1}} \in SP^r(v_{k,t_i})$
   a) Evaluate $arc(v_{k,t_i}, v_{m,t_{i+1}})$ according to graph search algorithm and add $v_{m,t_{i+1}}$ to $O_r$ if necessary.
   b) If $v_{m,t_{i+1}} \in B_{r,j}$, $r \neq j$, $\forall j \in \{1,2,\ldots,n\} \wedge$ $v_{m,t_{i+1}} \in O_r$, add $v_{m,t_{i+1}}$ to $O_j$.
7) Go to (4).
8) Are we finished, or do we have an error?
   a) If $G_j^* = \varnothing$, $\forall j \in \{1,2,\ldots,n\}$ then finished!
   b) Else, if $C^j \geq C^{max}$, $\forall j \in \{1,2,\ldots,n\}$ then no plan found. ERROR.
   c) Else, keep going. Go to (4).
```

Figure 2: Partitioned planning spaces algorithm.

It should be noted that an empty open set causes an idle cycle rather then an error. The reason for this is that the path cost from a boundary node $v_{k,t_i} \in B_{c,d}$ to the start node may be changed by the search algorithm running in either region r_c or r_d. If a change is made, the node is placed on the open list of both regions.

Step (5) of the algorithm begins the evaluation of the selected node. Its cost and membership in the goal set are checked. If its cost is greater than the regional stopping threshold or it is the last member of the goal set, then error and termination checking is performed (see step (8)).

Step (6) continues the evaluation procedure with the cost of achieving each member of the node's spanning set being evaluated. If the graph evaluation algorithm adds the node to the regions open set, and the node is part of a boundary, the node will also be added to the other boundary member's open set. This allows cost to propagate across the boundaries and as will be shown allows for the joint optimal solution to be found.

In order to properly propagate this value to the region not responsible for the value change, the normal graph search stopping criterion must be changed to cause repeated planning cycles to occur. This is accomplished in step (8) of the algorithm. The repeated cycle will quickly terminate (no nodes other then the boundary nodes will be placed on the open list) if no changes have occurred. However, if boundary changes have occurred, they will propagate through the graph.

The algorithm from Figure 2 may be applied to any configuration of partitioned planning spaces. Two possible configurations are shown in Figure 3, where the left hand drawing represents a classical nested hierarchy, and the right hand drawing represents a space designed for parallel implementation of the planning algorithm. A combination of the two partitioning approaches is also possible where regions may be defined based on information content. For example, for a road planning system, cities may represent independent regions where a different planning algorithm may be run than in rural areas. In general, these partitions may exist anywhere in the planning space with a separate graph search system operating on each partition. If the individual plan constructed in each partition is optimal, then the final plan constructed by the algorithm of Figure 2 will also be optimal. This statement will now be formalized.

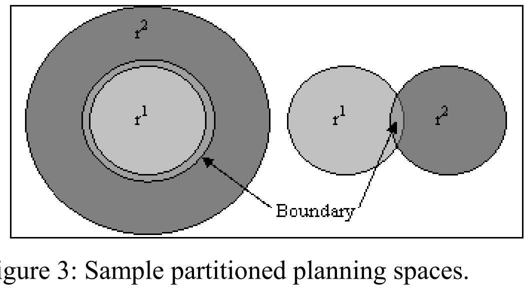

Figure 3: Sample partitioned planning spaces.

Theorem 1: Given the partitioned planning space $\mathbf{V}^{a,R}$, R = $\{r_1,r_2,\ldots,r_n\}$ with a independent version of the planning algorithm from Figure 2 executing in each region and a graph search algorithm in step (4) of Figure 2 that is optimal and complete. Further, given the start node v_{start,t_i}, the goal node v_{goal,t_i+m}, and the path $p(v_{start,t_i},\ldots,v_{goal,t_i+m})$ is the final path produced. Then $p(v_{start,t_i},\ldots,v_{goal,t_i+m}) = p_{min}(v_{start,t_i},\ldots,v_{goal,t_i+m})$ is the cost optimal path through the graph in the space $\mathbf{V}^{a,R}$, and it is cost equivalent to the cost optimal path through the graph in the space $\mathbf{V}^{a}$.

Proof (a): Assume
$p(v_{start,t_i},\ldots,v_{goal,t_i+m}) \neq p_{min}(v_{start,t_i},\ldots,v_{goal,t_i+m})$.
Then $\exists\, p^*(v_{start,t_i},\ldots,v_{goal,t_i+m})$:
$pc(p^*(v_{start,t_i},\ldots,v_{goal,t_i+m})) < pc(p(v_{start,t_i},\ldots,v_{goal,t_i+m}))$.
By (D6), $\exists$ node v_{k,t_a} : $v_{k,t_a} \in \mathbf{V}^{a,R}$,
$v_{k,t_a} \notin p(v_{start,t_i},\ldots,v_{goal,t_i+m}) \wedge pc(p^*(v_{start,t_i},\ldots,v_{k,t_a})) +$
$pc(p^*(v_{k,t_a},\ldots,v_{goal,t_i+m})) < pc(p(v_{start,t_i},\ldots,v_{goal,t_i+m}))$
where v_{k,t_a} is the first node from the path
$p^*(v_{start,t_i},\ldots,v_{goal,t_i+m}) : v_{k,t_a} \in p^*(v_{start,t_i},\ldots,v_{goal,t_i+m}) \wedge$
$v_{k,i} \notin p(v_{start,t_i},\ldots,v_{goal,t_i+m})$.
Without loss of generality assume that $v_{k-1,t_a-1} \in r_c$.
Four different situations exist with respect to boundary set membership for the nodes v_{k,t_a} and v_{k-1,t_a-1}.

Case 1: $v_{k-1,t_a-1} \notin B_{c,d} \wedge v_{k,t_a} \notin B_{c,d}$.

Case 2: $v_{k-1,t_a-1} \in B_{c,d} \wedge v_{k,t_a} \notin B_{c,d}$.

Case 3: $v_{k-1,t_a-1} \notin B_{c,d} \wedge v_{k,t_a} \in B_{c,d}$.

Case 4: $v_{k-1,t_a-1} \in B_{c,d} \wedge v_{k,t_a} \in B_{c,d}$.

For case 1 and case 4, both $v_{k-1,t_a-1}, v_{k,t_a} \in r_c$, $c \in \{1,\ldots,n\}$. In these cases, both nodes are in the same region and a cost optimal algorithm is running in that region. The fact that v_{k,t_a} is not included in the final path contradicts the assumption that the individual region's algorithm is cost optimal.
For case 2, $v_{k,t_a} \in r_c$, $v_{k,t_a} \notin SP^{r_d}(v_{k-1,t_a-1})$, $c \neq d \in \{1,\ldots,n\}$. By (D6), $\exists$ arc(v_{k-1,t_a-1}, v_{k,t_a}), and by Lemma 2,
$v_{k,t_a} \in SP^{r_c}(v_{k-1,t_a-1})$.
Therefore either v_{k,t_a} is a member of the open set O_c, or v_{k,t_a} has already been expanded. Given that the path p is the final path, so all v_{k,t_a} :
$pc(p(v_{start,t_i},\ldots,v_{k,t_a})) < pc(p(v_{start,t_i},\ldots,v_{goal,t_i+m})) \notin O_c$.

This forces the conclusion that v_{k,t_a} has been expanded. However, if v_{k,t_a} has been expanded, then by step (6) of the algorithm from Figure 2, $p_{min}(v_{start,t_i},\dots,v_{goal,t_i+m})$ would include this node. This contradicts the initial assumption.

For case 3, $v_{k-1,t_a-1} \in r_c$, $v_{k,t_a} \in r_c, r_d$, $v_{k,t_a} \in SP^{r_c}(v_{k-1,t_a-1})$, $c \neq d \in \{1, \dots, n\}$. By (D6) $\exists$ arc(v_{k-1,t_a-1}, v_{k,t_a}), $v_{k,t_a} \in SP^{r_c}(v_{k-1,t_a-1})$.
Therefore v_{k,t_a} is a member of O_c or has been expanded for region r_c. If v_{k,t_a} is expanded in region r_c, then by step (6) of the algorithm from Figure 2, v_{k,t_a} will either be a member of O_d, or p_{min} would include this node. p_{min} including this node contradicts the initial assumption. Therefore $v_{k,t_a} \in O_c \vee O_d$. However, since given that algorithm path is final path, all v_{k,t_a} : $pc(p(v_{start,t_i},\dots,v_{k,t_a})) < pc(p(v_{start,t_i},\dots,v_{goal,t_i+m})) \notin O_c, O_d$. This forces the conclusion that v_{k,t_a} has been expanded in both regions which contradicts $v_{k,t_a} \in O_c \vee O_d$.

Proof (b):
From Lemma 1 and Lemma 2, every node and set of arc connections that exists in $\mathbf{V}^a$ also exists in $\mathbf{V}^{a,R}$. Therefore all paths that exist in $\mathbf{V}^a$ exist in $\mathbf{V}^{a,R}$, and all paths that exist in $\mathbf{V}^{a,R}$ exist in $\mathbf{V}^a$.

4 PLANNING EXAMPLE

Figure 4 depicts an example of a planning space that has been decomposed into two partitions. A single cooperative planning agent operates on each partition. In Figure 4(a) the two striped spheres are the start and goal locations, and the dark band represents the boundary between the inner and outer planning regions.

Planning begins in a single region at the start location, and proceeds until a node is expanded whose spanning set contains nodes that are located in a boundary. This is shown in Figure 4(b) where the speckled sphere is the node being expanded.

The boundary nodes will now be placed in both region 1 and region 2's open set. This will cause the planning algorithm of region 2 to begin operation (it now has a node to expand), and both planning operations proceed in parallel.

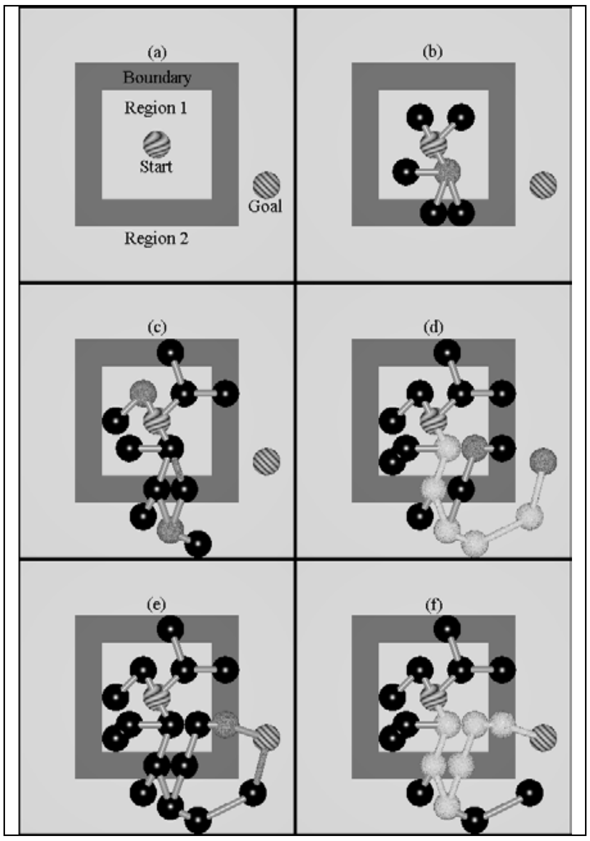

Figure 4: Example of graph expansion.

While evaluating its spanning set (step (6) of the algorithm from Figure 2), it is possible that the planner operating in region 2 will find another path to a boundary node and that this path will be cheaper than the path that was originally found by the region 1 planning system. This is depicted in Figure 4(c) where the specked link represents the older more expensive connection. This will cause the more expensive link to be removed, and the boundary node to be inserted in both regions open sets (step (6b) of the planning algorithm). This procedure does not interrupt the flow of either planning system.

As shown in Figure 4(d) by the white speckled spheres, a valid path exists once the planner operating in the region that contains the goal opens the goal node for evaluation. However, if all of the regions have not finished their planning cycle, this plan is not guaranteed to be optimal. This is seen in Figure 4(e) where the region 1 planning system finds a cheaper path to a boundary node and causes the region 2 planning system to recommence planning. The final optimal path is found when condition 8(a) of the algorithm is satisfied and is displayed in Figure 4(f).

5 CONCLUSIONS AND FUTURE WORK

This paper has presented a new framework for planning in partitioned planning problem spaces. These problem spaces may be of the classical nested hierarchy form or of a more complex form of partially overlapping planning spaces. This framework was proven to provide a complete and optimal solution assuming that a complete and optimal graph search algorithm is utilized.

In the near future, the authors will be implementing this framework for use in an autonomous vehicle's on-road planning system. This system will function much as the planning example in Figure 4 where the exterior region (region 2) will contain a priori data on the basic road network and the interior region (region 1) will contain high-resolution data on obstacles, exact lane trajectories, etc.

The main challenge that is expected in the implementation of this system will be the mapping of the boundary regions and providing a mechanism for assuring that unit cost equivalence is maintained between the two parallel planning systems.

References

1. Balakirsky, S., *A Framework for Planning with Incrementally Created Graphs in Attrributed Problem Spaces*, Akademische Verlagshesellschaft Aka GmbH, Berlin, Germany, 2003.

2. Fernández-Madrigal, J.-A. and González, J., "Hierarchical Graph Search For Mobile Robot Path Planning," *Proceedings of the 1998 International Conference on Robotics and Automation*, Vol. 1, 1998, pp. 656-661.

3. Fernández-Madrigal, J.-A. and González, J., "Multihierarchical Graph Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 1, 2002, pp. 103-113.

4. Lacaze, A., "Hierarchical Planning Algorithms," *SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 2002.

5. Zilberstein, S., "Using Anytime Algorithms in Intelligent Systems," *AI Magazine*, Vol. 17, No. 3, 2002, pp. 73-83.