Multihierarchical Graph Search

Juan-Antonio Fernández-Madrigal and Javier González, *Member*, *IEEE*

Abstract—The use of hierarchical graph search for finding paths in graphs is well known in the literature, providing better results than plain graph search regarding computational costs in many cases. This paper offers a step forward by including multiple hierarchies in a graph-based model. Such a multihierarchical model has the following advantages: First, a multiple hierarchy permits us to choose the best hierarchy to solve each search problem; second, when several search problems have to be solved, a multiple hierarchy provides the possibility of solving part of them simultaneously; and third, solutions to the search problems can be expressed in any of the hierarchies of the multiple hierarchy, which allows us to represent the information in the most suitable way for each specific purpose. In general, multiple hierarchies have proven to be a more adaptable model than single-hierarchy or nonhierarchical models. This paper formalizes the multihierarchical model, describes the techniques that have been designed for taking advantage of multiple hierarchies in a hierarchical path search, and presents some experiments and results on the performance of these techniques.

Index Terms—Graph theory, search, hierarchical graphs, path planning.

1 Introduction

Search is an important problem-solving paradigm. Searching for chains of elements that connect places, concepts, reasonings, etc., appears in different areas of research: In AI, finding sequences of elements can be applied to solve complex tasks or find chains of reasonings ([25], [14], [11], [5]); in robotics, geometric paths are calculated to move mechanical elements in space under kinematic and dynamic constraints (manipulators, mobile platforms, cooperating robots, etc.) ([27], [23], [1], [18], [8]); one of the main issues in networks and geographical information systems (GIS) is to find optimal routes ([17], [12], [3]); etc. These and many other examples highlight the importance of path searching in any computational representation of knowledge.

Hierarchical path search consists of finding paths connecting pairs of elements that are represented in a hierarchical fashion using hierarchical information for reducing the computational cost of conventional plain search ([10], [15], [19], [20], [26], [3]). The pair of elements defines the *search problem* and any path that connects them is a *solution* for the search problem. It has been demonstrated that hierarchical information can reduce the computational cost of path searching from exponential to linear in the best case ([20]). This is an essential issue when there is a large number of elements on which the search has to be carried out. Hierarchical path search has direct applications in GIS, computer networks, large databases, etc.

For performing hierarchical graph search, a well-formalized hierarchical graph model must be provided. Plain graphs can be arranged in hierarchies as follows: Consider a plain graph that models some knowledge (conceptual knowledge, large-scale space, or any other) that can be "abstracted" by reducing the amount of detail it represents. There are several ways of doing this ([2]). We are interested in abstraction by representing a number of nodes of the original graph as a super-node of the abstracted graph.[1]

This model of abstraction produces stacks of plain graphs that are called *hierarchies*. Fig. 1 shows a very simple example of a hierarchy representing the spatial elements of a room. A hierarchy is composed of a sequence of graphs called *hierarchical levels*. The higher the level, the smaller the amount of information it represents. Other hierarchical models can be found in ([6], [4], [33]), but usually either they use more constrained hierarchical structures in the hierarchies (trees instead of graphs) or only a fixed number of hierarchical levels (it is common to use only two). The model we propose is general enough to deal with an unconstrained number of hierarchical levels and any hierarchical structure.

When more than one hierarchy is interwoven in the same model, the representation becomes a *multihierarchical representation*. Multihierarchical representations are more efficient and adaptable for solving different problems than single-hierarchy or nonhierarchical models since a number of hierarchies are available, each one adapted for solving a given class of problems ([7]). More concretely, a graph-based model of knowledge with more than one hierarchy built from certain ground information presents the following advantages in path searching with respect to single-hierarchy representations:

- The most suitable way of solving a search problem can be chosen from the set of different hierarchies. The suitability of a hierarchy for solving search problems highly depends on each specific problem ([7]). Choosing the best hierarchy for each problem tends to optimize the overall performance of the hierarchical path search algorithm.

1. The abstraction of arcs follows directly from the abstraction of nodes, as explained in Section 2.

• *The authors are with the System Engineering and Automation Department, University of Málaga, Campus Teatinos—Complejo Tecnológico, 29071 Málaga, Spain. E-mail: {jafma, jgonzalez}@ctima.uma.es.*

Manuscript received 2 June 2000; revised 18 Jan. 2001; accepted 11 Apr. 2001. Recommended for acceptance by J.R. Beveridge.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 112230.

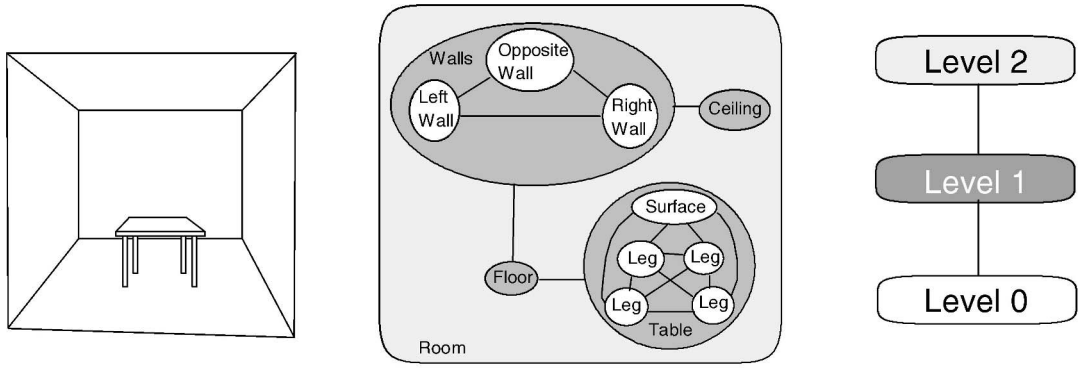


Fig. 1. Example of a hierarchical graph-based model that represents a possible abstraction of the spatial elements perceived in a room. (a) Room. (b) Hierarchical levels that model the room (each of them is a plain multigraph) represented by different shades. (c) Resulting hierarchy (each level contains a plain graph).

- A set of multiple search problems can be solved more efficiently. Given more than one search problem to be solved in the model, it may be the case that they share some abstract concepts (the super-nodes of some hierarchical levels). In these cases, shared paths can be found just once for all the problems, reducing the computational cost of the search with respect to solving each problem separately.
- The possibility of presenting the results of hierarchical path search on different notations, that is, using different systems of concepts and relations (nodes and arcs). For example, a hierarchy can be more appropriate for optimizing the computational cost of a given problem, whereas another one can be used to explain the results more intuitively to humans.

Therefore, a multiple hierarchy can provide more flexibility, can perform better than conventional hierarchical path search, and gives a wider range of possibilities to represent a set of interconnected elements.

This paper is structured as follows: Section 2 formalizes a particular multihierarchical graph representation that has been used elsewhere ([7], [10], [9]): the Multi-AH-graph model. Section 3 presents some basic concepts related to hierarchical path search in single hierarchies. Section 4 shows the utility of the Multi-AH-graph model for performing path searching more efficiently than single-hierarchy or plain-graph models, along with some experiments of hierarchical path search in Multi-AH-graphs. Finally, the conclusions of this work are outlined in Section 5.

2 A Multihierarchical Graph Model

This section presents a graph-based, multihierarchical model of knowledge, called Multi-AH-graph. It has been designed as a skeleton that supports mechanisms of multiple abstraction ([7]).

A Multi-AH-graph is an evolution of simpler relational models: graphs, hierarchical graphs, and annotated graphs. The formalization in this section relies on that incremental enhancement: First, a graph-based, single-hierarchy model called AH-graph is presented ([9], [24], [10]). Then, this model is enhanced to cope with multiple hierarchies.

2.1 Formalization of a Single-Hierarchy Graph Model

An annotated, hierarchical graph (AH-graph) is a linearly ordered sequence of **hierarchical levels**. Formally, it is a quadruple $(L, c, k, annot)$, where L is the sequence of hierarchical levels, c the **arc-cost function**, k the **arc-type function**, and *annot* is an **annotation function** as described later on. Fig. 2 shows two examples of AH-graphs.

The sequence of hierarchical levels is $L = L^0 L^1 \dots L^{r-1}$. The **depth** of the AH-graph equals its number of hierarchical levels and is denoted $|L|$. Each hierarchical level L^i is in turn a quadruple (N^i, A^i, s_n^i, s_a^i), where N^i is a set of nodes, A^i a set of arcs, and s_n^i and s_a^i are the **abstraction functions** for nodes and arcs of L^i, respectively. A hierarchical level is therefore a *plain directed multigraph*.[2]

Level L^0 is the **lowest** or **ground hierarchical level** of the AH-graph. It represents the data with the maximum amount of detail that is available. Level L^{r-1} is the **highest** or **universal hierarchical level** of the AH-graph: It represents the same data with the minimum amount of detail, usually as a single node. Thus, the higher the hierarchical level, the less detailed the information it represents. The way this reduction of data is set up is defined by the abstraction functions for nodes and arcs. The only existing connections between hierarchical levels in an AH-graph are given by these functions: no arc exists between nodes of different hierarchical levels.

The **abstraction function for nodes** s_n^i (see Fig. 2) is formally defined at any hierarchical level L^i except at the highest one, L^{r-1}:

$$s_n^i \ : \ N^i \rightarrow N^{i+1}, \ \ i < r-1.$$

Function s_n^i *abstracts* a node of a hierarchical level up to the next higher level, hence, reducing the amount of detail from N^i to N^{i+1}. Its inverse *refines* a given node. The node that the function s_n^i returns for $n_j (n_j \in N_i)$ is called the

2. A plain multigraph is a plain graph with possibly more than one arc between a given pair of nodes ([29]). In the particular multigraph model used in this paper, a *type* is assigned to each arc. The types of the arcs existing between a given pair of nodes must be different.

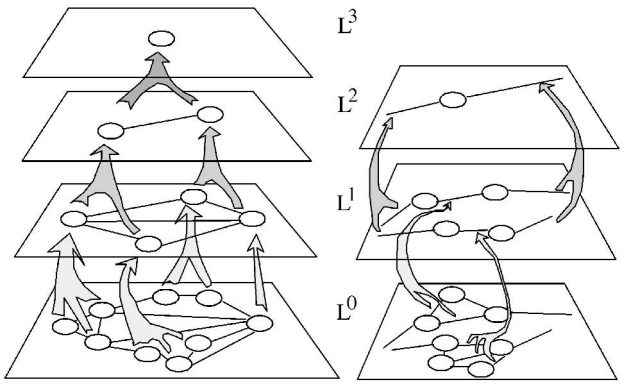


Fig. 2. Two examples of AH-graphs of four and three hierarchical levels, respectively. The AH-graph on the left side illustrates the behavior of the abstraction function for nodes from the lower to the higher hierarchical levels. The AH-graph on the right illustrates the behavior of the abstraction function for arcs (see the text).

supernode of n_j (correspondingly, the nodes that are abstracted to n_k are called the **subnodes** of n_k and the graph they form is the **subgraph** of n_k, also called a **cluster** of the hierarchy). The only restriction defined on function s_n^i is that no node can be mapped into more than one supernode.

Analogously, the **abstraction function for arcs** s_a^i (see Fig. 2) is defined at any hierarchical level L^i except at the highest one, L^{r-1}:

$$s_n^i \ : \ A^i \rightarrow A^{i+1}.$$

It is defined for an arc[3] a (n_s, n_g, t) iff the abstraction function for nodes is defined for both n_s and n_g and $s_n^i(n_s) \neq s_n^i(n_g)$.

The function s_a^i *abstracts* an arc of a hierarchical level up to the following higher level, reducing the amount of detail from the set A^i of arcs of level L^i to the set A^{i+1} of arcs of level L^{i+1}. Its inverse *refines* a given arc. The arc that is abstracted by s_a^i is called the **superarc** of the original one. The arcs that result from refining the superarc are called its **subarcs**.

The properties of s_a^i are derived from the properties of s_n^i since the abstraction function for arcs is completely defined by the abstraction function for nodes: The existence of the former is for mere convenience. The way s_a^i is determined by s_n^i is

$$s_n^i(a(n_s, n_g, t)) = a(s_n^i(n_s), s_n^i(n_g), t).$$

The weights of the arcs of an AH-graph are defined as **cost intervals** ([30]). The set of cost intervals is a partial-ordered set which elements are numeric intervals defined as

$$\boldsymbol{i} = [i^-, i^+], \text{where } i^+ \leq i^- \leq 0, i^+ \neq \infty.$$

The addition operation in cost intervals is defined as $\boldsymbol{i} + \boldsymbol{j} = [i^- + j^-, i^+ + j^+]$.

Given a superarc $a(n_s, n_g, t) \in L^{i+1}$, its cost is deduced from the following expression:

$$w(a(n_s, n_g, t)) = \wedge^* \left\{ w(a(n_r, n_t, t)) \ : \ a(n_r, n_t, t) \in [s_a^i(a(n_s, n_g, t))]^{-1} \right\}.$$

The symbol $\wedge^*$ represents the *propagation* operator. It yields the minimum-width cost interval that contains a given set of cost intervals. Thus, the weight of a superarc is the propagation of the weights of its subarcs. This provides an easy and efficient method of implementing the abstraction of weights of the arcs through the hierarchy. Notice that this procedure generates weights for superarcs that are indistinguishable (incomparable) from the weights of the subarcs.

Nonstructural information is represented by **annotations** both in nodes and arcs of the AH-graph. The number of annotations a node or arc can store is not restricted. The **annotation function** yields a selected annotation of a given node or arc. The use of annotations in mobile robotics can be found in [9], [10], [24], and [28].

2.2 Formalization of a Multihierarchical Graph Model

Broadly speaking, a Multi-AH-graph is a *set of AH-graphs* possibly sharing some hierarchical levels. A Multi-AH-graph can also be seen as a *DAG*[4] *of hierarchical levels*, that is, a graph whose nodes represent hierarchical levels and arcs represent the abstraction functions.

Formally, a Multi-AH-graph is a quadruple $(AHG, c, k, annot)$, where AHG is the set of **hierarchies** (AH-graphs) of the Multi-AH-graph, c the **arc-cost function**, k the **arc-type function**, and *annot* the **annotation function**. Functions c, k, and $annot$ are the same for all the AH-graphs of the Multi-AH-graph. Their definitions have been given previously in the formalization of the AH-graph model.

AHG is a set $AHG = \{L(0), L(1), \ldots, L(n-1)\}$. The ith **hierarchy** $L(i)$ is essentially an AH-graph defined as a set of hierarchical levels: $L(i) = \{L(i)^0, L(i)^1, \ldots, L(i)^{|L(i)|-1}\}$. From now on, the term **hierarchy** will refer to an AH-graph belonging to a Multi-AH-graph.

Extending the notation given in the previous section, the jth hierarchical level of the ith hierarchy of a Multi-AH-graph is formally defined as a quadruple $(N(i)^j, A(i)^j, s(i)_n^j, s(i)_a^j)$, where $N(i)^j$ is the set of nodes of the hierarchical level, $A(i)^j$ the set of arcs, and $s(i)_n^j$ and $s(i)_a^j$ the abstraction functions for nodes and arcs of that level, respectively.

In a Multi-AH-graph, any hierarchical level can be shared by a number of hierarchies. For example, if $L(2)^0 = L(4)^0$, that is, if hierarchies #2 and #4 share their lowest hierarchical levels, the nodes and arcs of this level can be abstracted in two ways: using hierarchy #2 ($s(2)_n^0$ and $s(2)_a^0$) or using hierarchy #4 ($s(4)_n^0$ and $s(4)_a^0$). If, again, $L(2)^1 = L(4)^1$, then nodes and arcs of the next higher level can also be abstracted in two ways; otherwise, each hierarchy will abstract the same elements separately.

2.2.1 Multihierarchies

In general, the hierarchical levels of a Multi-AH-graph and the abstraction links between them define a DAG. Other

3. The notation used for arcs in AH-graphs is $a(n_s, n_g, t)$, where n_s and n_g are the start and goal nodes of the arc, respectively, and t is its type (typically, a natural number).

4. Directed Acyclic multiGraph.

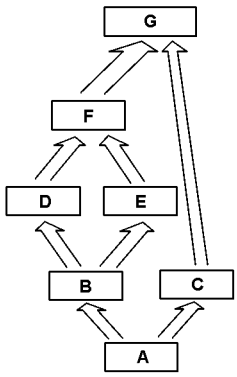


Fig. 3. A typical example of multiple hierarchy. In this directed acyclic graph (DAG), hierarchical levels are represented by rectangular boxes. The thick arrows represent bundles of abstraction links. Level **A** is the only ground hierarchical level of this multiple hierarchy and level **G** the only universal level.

structures may appear, such as *sets of sequences*, *trees*, or *forests*, but all of them can be considered special cases of DAGs. The DAG induced by a given Multi-AH-graph *M* is denoted $\varphi(M)$. From now on, this DAG will be called a **multiple hierarchy**. An example of multiple hierarchy is shown in Fig. 3.

A **multiple hierarchy** $\varphi(M)$ is a plain directed, acyclic multigraph (N, A, k), where N is the set of all hierarchical levels of the AH-graphs contained into M, A a set of **abstraction links**, and k the abstraction-link-type function.

The nodes of $\varphi(M)$ correspond to the hierarchical levels of the AH-graphs of *M*. A hierarchical level which is the lowest level in some hierarchy is called a **ground hierarchical level** of the multiple hierarchy. A hierarchical level which is the highest one in some hierarchy is called a **universal hierarchical level**.

The arcs of $\varphi(M)$ are abstraction links. An abstraction link between two hierarchical levels $L(i)^u$ and $L(i)^{u+1}$ is denoted as $a(L(i)^u, L(i)^{u+1}, i)$. It represents all the values of $s(i)_n^u$ and $s(i)_a^u$. There is a different abstraction link for each hierarchy (AH-graph) of *M* to which both hierarchical levels belong (this is what leads to the multigraph shape of $\varphi(M)$). Each abstraction link has an associated type that equals the hierarchy to which both levels belong. For example, the type of $a(L(i)^u, L(i)^{u+1}, i)$ is *i*; the type of $a(L(j)^v, L(j)^{v+1}, j)$ is *j*. These values are returned by the abstraction-link-type function *k*.

3 Hierarchical Path Search in AH-Graphs

Hierarchical path search in hierarchical graphs consists of finding a path connecting a pair of nodes, using hierarchical information for reducing the computational cost of conventional plain search. The pair of nodes defines the *search problem* and any path that connects them, a *solution* for the search problem. A search problem is denoted (n_s, n_g), where n_s is the start node and n_g the goal node. From the definition of a hierarchical graph, both of them must belong to the same hierarchical level.[5] The *total cost* of a path $P(n_s, n_g)$ is defined as the sum of the costs of all its arcs and denoted $|P(n_s, n_g)|$. If there is no path between the pair of nodes (n_s, n_g) with a total cost smaller than $|P(n_s, n_g)|$, then $P(n_s, n_g)$ is said to be *optimal*.

5. Otherwise, no path exists that connects them.

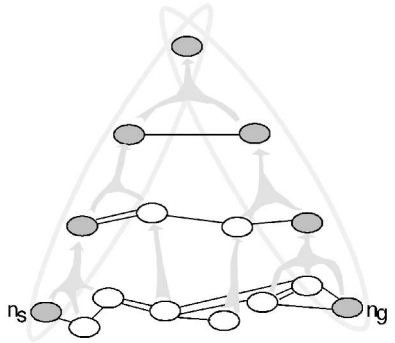


Fig. 4. A search problem (n_s, n_g) is solved hierarchically by solving its abstraction at higher hierarchical levels. In the figure, a pyramid of search is shown for the problem. When a problem is abstracted three times, both nodes become the same and, therefore, this is the end of the pyramid. The pyramid consists of all the hierarchical levels involved in the abstraction chains of the nodes of the problem. The contour of the pyramid is shown enclosed within ellipses. The nodes of the contour are gray shaded.

The set of hierarchical levels involved in a search problem is called the **pyramid of search** for the problem. In order to obtain the pyramid of search, both start and goal nodes must be abstracted until they coincide in a common supernode.[6] The pair of sequences of abstracted nodes are the **contour** of the pyramid of search. An example of pyramid of search and its contour is shown in Fig. 4.

3.1 Hierarchical Path Search Algorithm

The approach used in our work for hierarchical path search in AH-graphs (which is used in Multi-AH-graphs too, as shown later on) is an instance of the classic refinement method ([15]). This method can yield the best results in computational cost and under certain restrictions, also good optimality ratios of the paths that are found. The general scheme of a classic refinement algorithm is shown in pseudocode in Fig. 5.

This scheme does not guarantee obtaining optimal paths when the cost of traversing nodes at a nonground level $L^i(i > 0)$ is discarded. There are three possible ways of addressing this suboptimality problem:

- First, by *materializing* the costs of L^i refining the nodes of L^i ([19]), that is, by storing in the nodes of L^i the costs of traversing their subgraphs at L^{i-1}.
- Second, by identifying and using only those hierarchies which guarantee that the optimal path(s) will not be discarded ([7]).
- Third, by using a searching scheme different from the classic refinement method described above, e.g., heuristic or mixed ([15], [31], [16]).

Since obtaining optimal paths is beyond the scope of this paper, these approaches are not addressed here. For an in-depth analysis of the cases where the classic refinement method guarantees obtaining optimal paths, see [7].

6. It is assumed that there is a common supernode (ancesstor) for any pair of nodes. By default, a universal level which contains a supernode that is an ancesstor for any other node of the AH-graph is assumed to exist.

```
Recursive Hierarchical Path Search Scheme

(at the first call, L^i = Highest hierarchical level of
the pyramid of search)

Find an optimal path at L^i by conventional
plain-graph path search
Until the path has enough detail do:
    For every node in the path do:
      Call (Recursive Hierarchical Path Search
      Scheme for refining the node at
      L^(i-1))  /* that is, traverse the
      cluster -subgraph- of the
      node at L^(i-1) */
    end_for
end_until
```

Fig. 5. Recursive algorithm for hierarchical path search in AH-graphs based on classic refinement.

The hierarchical path search algorithm we have implemented stores incomplete abstract paths at each of the hierarchical levels of the hierarchy. When any of these portions becomes a complete abstract path, an abstract solution can be provided. If such a complete path is obtained at the ground level (the one where the search problem is defined), the final solution is found. The utility of the abstract solutions in real-time implementations is clear: It is not necessary to wait for the complete, final path for using the result. Similar approaches can be found in [6], [31].

When backtracking appears (no path is found at a given stage of the search), the hierarchical path search algorithm looks for another possible solution. For implementing that, a modified plain-graph path search algorithm is included in the hierarchical searching method. That algorithm does not return a single path (typically, the optimal path) connecting two nodes within a cluster, but it is able to deliver several paths ordered with respect to their total costs.[7]

3.2 Goodness of Hierarchical Path Search

When hierarchical path search is to be compared to other techniques, an appropriate measurement of its goodness must be defined. Two factors are essential for this purpose: the computational cost of the search and the optimality ratios of the paths obtained by the search. The optimality ratio of a path is defined as follows: Let (n_s, n_g) be the search problem; let $P(n_s, n_g)$ be the path which optimality is to be calculated; let $P^*(n_s, n_g)$ be an optimal path connecting both nodes. The optimality ratio of the path is defined as

$$opt(P(n_s, n_g)) = \frac{|P^*(n_s, n_g)|}{|P(n_s, n_g)|} \in [0, 1].$$

In [7], [13], mathematical and graphical estimates for the computational cost and optimality of paths of hierarchical path search can be found. The main conclusion is that, although they are symbolically unsolvable expressions, the computational cost and optimality of paths are highly influenced by the size of the clusters[8] of the hierarchy. If those clusters are large, the plain-graph path search performed within the clusters by the hierarchical path search algorithm has a high computational cost. However, in that case, the plain-graph path search finds paths that are close to the optimal ones (the optimal ones are found if the cluster comprises all the nodes of the hierarchical level).

Therefore, the computational cost of hierarchical path search tends to be lower (better) as the size of the clusters in the hierarchy decreases. The optimality of the paths found by hierarchical path search, on the contrary, tends to increase (better) as the size of the clusters increases. The goodness of a hierarchy for hierarchical path search can be defined as a combination of both factors.

4 Hierarchical Path Search in Multi-AH-Graphs

Solving a single search problem in a multiple hierarchy can be carried out by choosing the most appropriate hierarchy for solving the problem and then using the algorithm described in Section 3.1. This approach is studied in Section 4.1. However, this is not the only advantage of using multiple hierarchies for hierarchical path search.

In Section 4.2, it is described how a given set of search problems can be solved by using a multiple hierarchy. This can reduce the computational cost of solving the search problems with respect to both single-hierarchy models and nonhierarchical models.

In addition, Section 4.3 presents an example of the utility of a multiple hierarchy for representing the results of hierarchical path search in a variety of ways.

4.1 Solving a Single Search Problem in a Multihierarchy

The main concern in this case is which hierarchy to select for running the hierarchical search algorithm described previously. The method for selecting the best hierarchy must evaluate the goodness of the hierarchy for solving the search problem. This evaluation should not assume a high computational cost, although it should yield a precise enough estimate of goodness. In addition, since different factors are involved in the goodness of a hierarchy (it can be good with respect to computational cost, or to the

7. This is an implementation of Yen's algorithm ([32]) for obtaining the k-shortest paths between a pair of nodes in a plain graph.

8. Remember from Section 2.1 that a cluster in a hierarchy is a set of nodes corresponding to the subgraph of a given abstract node.

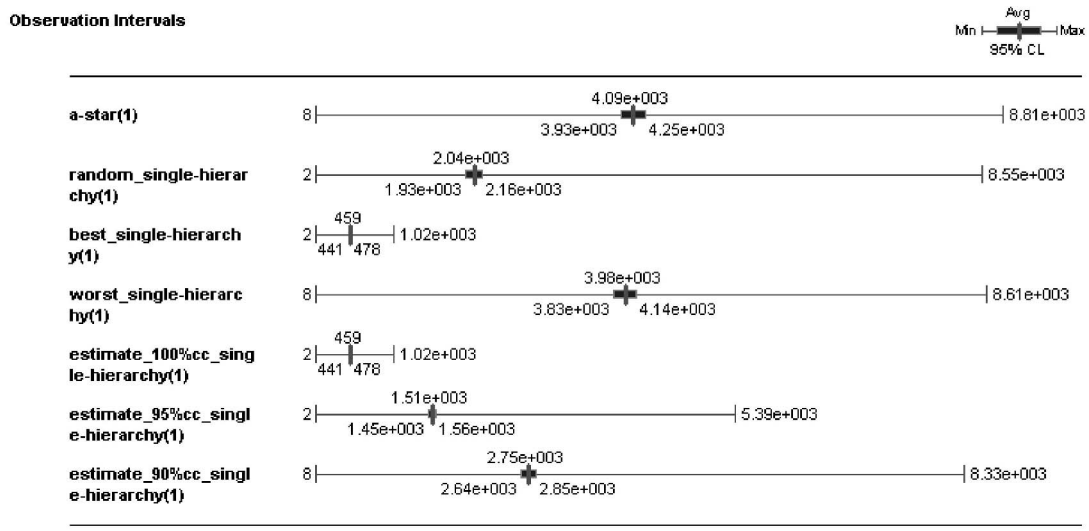


Fig. 6. Computational costs of solving 1,000 search problems in the multiple hierarchy mentioned in the text. (a) A* algorithm. This yields the worst results since it uses no hierarchical information. (b) Randomly choosing a hierarchy for solving each problem. This serves for comparison with other techniques. If a method for selecting a hierarchy for each search problem gives a better (lower) computational cost than this, it is worthwhile. (c) Choosing by visual inspection a hierarchy that seems to yield the best computational costs (it has small clusters). (d) Choosing by visual inspection a hierarchy that seems to yield the worst computational costs (it has large clusters). (e), (f), and (g) Choosing hierarchies by the method explained in Section 4.1 for a weight of computational cost with respect to optimality of paths of 100 percent, 95 percent, and 90 percent, respectively. The large variation in the results by the small variation in the weight (5 percent in each test) is due to the discrete nature of the multiple hierarchy.

optimality of paths that are obtained, or to perhaps other factors, or a combination of them), it is interesting to consider the possibility of varying the importance of each of these factors.

We present an algorithm that follows these guidelines. It is based on the results mentioned in Section 3.2 about the goodness of hierarchical path search. As it is demonstrated there and in other well-known works ([20], [13]), the computational cost and optimality of hierarchical path search are directly influenced by the size of the clusters of the hierarchy.

Our algorithm defines the goodness of a hierarchy in terms of both the computational cost of the search in that hierarchy and the optimality of paths obtained by that search. Estimates of these factors are obtained by functions of the average size of the clusters involved in the pyramid of search of the search problem. The computational cost estimate $(\widehat{cc}(h,p))$ is taken as proportional to the square of the average size of those clusters (since the plain-graph path search performed within a cluster with x nodes is $O(x^2)$), while the optimality of paths estimate $(\widehat{op}(h,p))$ is taken as proportional to the size of the cluster (x). A weighted linear expression combines them in order to obtain a single value for goodness[9]

$$goodness(h,p) = w.\widehat{cc}(h,p) + (1-w).\widehat{op}(h,p), w \in [0,1].$$

The hierarchy h with the greatest value of goodness for a given search problem p is the one selected for solving it. This method allows us not only to select the hierarchy that leads to the lowest computational cost or the highest optimality of paths, but also to select a hierarchy that optimizes a combination of both. It has also yielded goodness estimates close to the real goodness values.

4.1.1 Experiments

The method for selecting the best hierarchy for solving a search problem has been tested with a multiple hierarchy of 10 hierarchies constructed randomly (by the automatic multiple hierarchy constructor presented in [7]) on a ground graph with 1,000 nodes.

In order to simulate a real structured environment, for example, a building containing several rooms or the map of a country divided into regions or provinces, a particular structure has been imposed on the ground graph. Its nodes are conceptually grouped into 100 sets of 10 nodes each, the density of arcs within a set (*internal* arcs) being of 50 percent (there is a probability of 50 percent that an arc connects two nodes within a set), and of $[1..3]$ for arcs connecting different sets (*external* arcs) (a minimum of one arc connecting two given sets, a maximum of three). Similar types of graphs can be found in robotics and GIS applications.

Fig. 6 presents a comparison of different techniques for solving 1,000 search problems with this multiple hierarchy. These techniques include an A* algorithm that only runs at the ground hierarchical level of the multiple hierarchy (it does not use the multiple hierarchy), the hierarchical path search algorithm described in Section 3.1 that has been tested on some hierarchies of the multiple hierarchy, and the selection of the best hierarchy algorithm added to the hierarchical path algorithm in order to choose the best hierarchy for solving the problems. All the computational costs[10] are measured by the number of arcs that are

9. Since the average size of the clusters of the hierarchy can be calculated for each hierarchical level when the hierarchy is constructed, calculating the goodness of a hierarchy is $O(d)$, where d is the hierarchy's depth.

10. The computational cost of selecting the hierarchy for each search problem is not shown in the figure since it is quite small with respect to the computational cost of the search. This hierarchy selection cost is about 50 units (measured in the number of hierarchical levels explored).

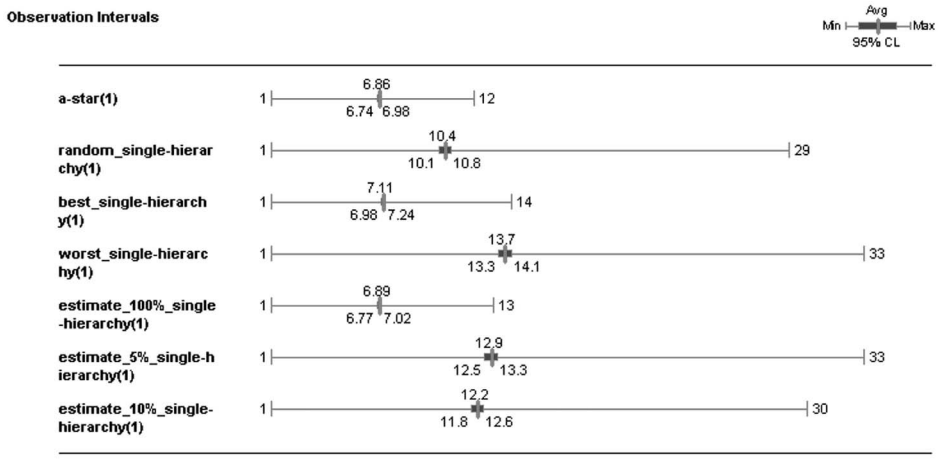


Fig. 7. Total costs of the paths obtained by the same experiments as in Fig. 6. In this case, the A* algorithm yields the best results since it uses all the nodes of the ground graph (without refining higher hierarchical levels). As it is shown, when the weight for the optimality of paths is set to 100 percent with respect to the computational cost, the algorithm for selection of hierarchies finds hierarchies that are even better than those found by visual inspection. When the weight is only 5 or 10 percent, the total costs of the paths are close to the worst.

explored during the search. Fig. 7 shows the total costs of the resulting paths under the same comparison.

As shown in the figures, the selection of the best hierarchy by our algorithm can obtain up to 67.5 percent (on average) of computational cost reduction with respect to choosing hierarchies randomly, up to 88.5 percent with respect to choosing the worst hierarchy, and up to 88.8 percent with respect to the A* algorithm. In addition, it allows us to weight the computational cost factor by less than 100 percent with respect to the optimality of paths, selecting hierarchies that, although they are not the best ones for the computational cost, obtain shorter paths. The reduction in the total costs of the paths when the weight for the computational cost is 0 percent and the weight for path optimality is 100 percent is up to 33.75 percent (on average) with respect to choosing hierarchies randomly, and almost reaches the total costs obtained by the A* algorithm (6.89 average total cost versus 6.86 obtained by A*).

4.2 Solving Multiple Search Problems in a Multihierarchy

A multiple search problem is defined in a Multi-AH-graph model as a set $\{(n_s^1, n_g^1), (n_s^2, n_g^2), \ldots, (n_s^p, n_g^p)\}$ of p different search problems, where (n_s^i, n_g^i) is the ith search problem, consisting of start node n_s^i and goal node n_g^i, both of them belonging to the same hierarchical level.

A multiple search problem in a Multi-AH-graph can be solved in three different ways:

- Solving each problem separately and sequentially, possibly using the approach described in Section 4.1.
- Solving all the problems in parallel, without taking into account the possible dependencies between them.
- By considering those abstract portions of the pyramids of search of the problems that are shared, solving part of the problems separately and another part just once for subsets of the problems. This method has been called **SAC** (Shared Abstract Concepts method).

The first method requires no special analysis. The second one is not addressed in this paper, although an implementation in a parallel machine should not be difficult. The third one consists of taking advantage of the cases where the pyramids of search of several problems are partially shared. In those cases, the shared portions of the pyramids only need to be solved once, reducing the computational cost of solving the multiple search problem with respect to solving it sequentially.

The SAC procedure may reduce the computational cost of solving the multiple problem from order p to order 1 in the best (ideal) case, obtaining results in a conventional single-processor computer that are close to a parallel implementation. The method needs to carry out a previous analysis of the multiple search problem for finding the shared portions of the pyramids of search, as described in the following paragraphs.

A search problem is associated to a pyramid of search, which is derived from a given hierarchy. When more than a hierarchy is available in the model, the search problem can be solved using different pyramids of search, with different depths (number of hierarchical levels), and a different number of nodes at each level. These parameters influence the computational cost and optimality of searching since both the number of hierarchical levels and the number of nodes of each level are influenced in turn by the size of the clusters of the hierarchy. We assume that a given hierarchy is previously assigned to each search problem by some specified method (i.e., the best for a given purpose: computational cost reduction or increased path optimality).

A given assignment of hierarchies to search problems may lead to the appeareance of shared portions of the pyramids of search or not. In the positive case, the abstract representations of the start and goal nodes of more than one search problem coincide at a certain hierarchical level. This level is called a *common hierarchical level* of the assignment (see Fig. 8). The existence of that kind of level is required in order to use the SAC method for solving multiple search problems.

Thus, the solution of the multiple search problem can be obtained following these stages:

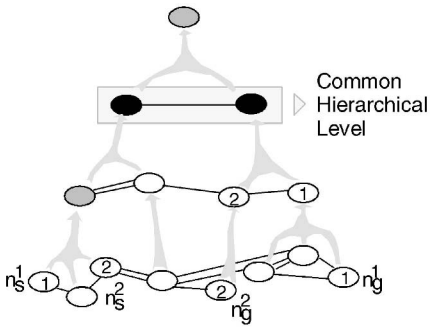


Fig. 8. This figure shows an assignment of pyramids of search to a multiple search problem consisting of two search problems: (n_s^1, n_g^1) and (n_s^2, n_g^2). The pyramid of the first problem consists of nodes either marked with "1" or are gray or black shaded. The pyramid of the second problem consists of nodes either marked with "2" or are gray or black shaded. Gray and black shaded nodes are common to both pyramids. Black shaded nodes belong to a common hierarchical level of this particular assignment.

1. Defining the shared portions of the pyramids of search.
2. Solving each shared portion only once for all the involved search problems and solving each non-shared portion once for each search problem.

Notice that the first stage is needed in order to decide whether the SAC method can be applied. The computational cost of this precalculation can be quite small if the hierarchical levels that are common for several hierarchies are detected when the multiple hierarchy is constructed. In that case, only an exploration of those hierarchical levels for determining whether the search problems are abstracted to the same pair of nodes is required and this is $O(r)$, where r is the number of hierarchical levels of the multiple hierarchy.

4.2.1 Experiments

The multihierarchy being used is the same as in the experiments in Section 4.1. However, the procedure for assigning a hierarchy to each search problem has been designed specifically for the SAC method. In order to describe this special assignment, the concept of *ground cluster* must be introduced (see Fig. 9).

If the *first* hierarchical level of a hierarchy is the one abstracted directly from the ground hierarchical level, then a *ground cluster* is any cluster of nodes (subgraph) that the first hierarchical level of a hierarchy defines on the ground hierarchical level. The importance of the ground clusters comes from the fact that, at the ground level, the search problems are all different and, therefore, to refine the abstract paths from the first hierarchical levels to the ground level of the hierarchy, the hierarchical path search must be performed separately for each problem. In other words, the SAC method cannot reduce the computational cost of refining the problems between the first and the ground hierarchical levels of the hierarchies.

This has an important consequence: If there are ground clusters that are large, the computational cost of refining the problems between the first and the ground hierarchical levels of the hierarchies can be very high in comparison to the reduction obtained by the SAC method at higher levels of the pyramids of search (notice that this effect is even worse if we consider that the ground hierarchical level is the one with more nodes). Experimentally, we have found that this effect can make the reduction of computational cost at higher hierarchical levels insignificant, making the SAC method useless. However, if the size of the ground clusters is kept below a certain small size, the SAC method obtains quite good results.

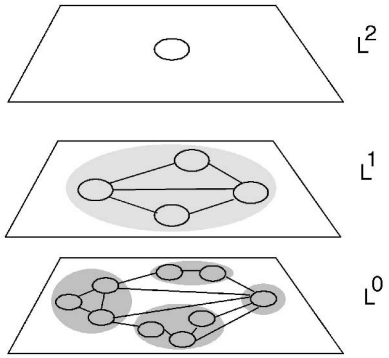


Fig. 9. Ground clusters defined on level L^0 by the nodes of level L^1 (they are shown as dark gray shaded). The clusters of nodes defined by L^2 and L^1 are not ground clusters.

Therefore, the hierarchy that is assigned to a given search problem is the one with its ground clusters smaller than a predetermined (small) size. In our experiments, we have obtained bad results as long as the maximum size of the ground clusters is above 100 nodes.

The number of cases where the SAC method can be used is difficult to determine since they depend closely on the structure of the multiple hierarchy (the number of common hierarchical levels, the maximum size of the ground clusters). Nevertheless, since the algorithm which decides whether the SAC is applicable is not computationally expensive, then this algorithm can be executed whenever a number of search problems have to be solved and, if possible, the SAC method used. In the experiments we have carried out, we have found that, with a maximum size for the ground clusters of four nodes, reductions of up to 25 percent of the computational cost have been achieved with respect to solving the problems sequentially. The computational cost of detecting whether the SAC method is applicable is only about 1 percent of the cost of solving the problems sequentially.

4.3 Multiple Representations in Multihierarchy

Yet another advantage of using a multihierarchy is that the information can be represented by using different systems of concepts (one for each hierarchy). In the following, a robotic example illustrates this use of multiple hierarchies.

The example consists of an environment comprising a small building with five rooms: a laboratory, two offices, a corridor, and a library. An office-delivery robot is in charge of taking small objects from one place to another inside the building. By exploration of the environment, it has been able to identify distinctive places and generate the basic topology ([21], [24]), that is, the plain graph corresponding to the ground hierarchical level of the Multi-AH-graph (see Fig. 10).

Some places are *service points*, that is, locations where the robot can take objects. Other places are *docking stations*: locations where it can connect to the computer network of the building to receive new commands or to connect to a recharging system to recharge its batteries. The robot

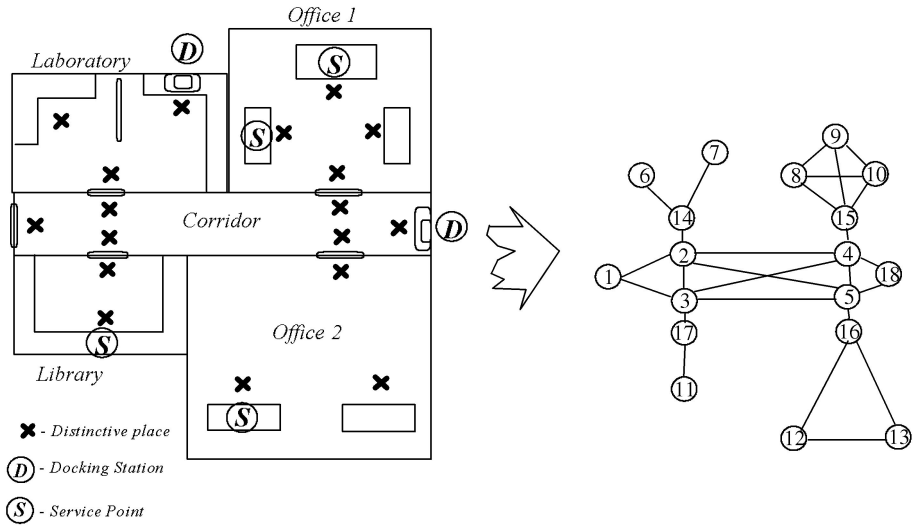


Fig. 10. (a) Environment where an office-delivery robot system performs several operations, such as taking small objects from one service point to another or connecting to a docking station to receive new commands or recharge its battery. (b) Toplogy of distinctive places identified by exploring the enviroment. This is the lowest hierarchical level of the Multi-AH-graph.

maintains a statistic database holding information about the use of each service area and each service point.

A Multi-AH-graph is constructed using the ground topology (automatic construction of multiple hierarchies is addressed in [7]). The following hierarchies are built: a *descriptive* hierarchy, a *docking* hierarchy, and a *service* hierarchy. The *descriptive* hierarchy is useful for describing routes in a user-friendly fashion to human operators, who understand the building as a set of rooms. The *docking* hierarchy is useful when forming docking areas. The *service* hierarchy forms a hierarchy of service areas containing the different service points of the building, useful for maintaining the statistical information for the delivery system. The ground hierarchical level and the universal level are the only shared levels between the three hierarchies. The three hierarchies are shown in Figs. 11a, 11b, and 11c.

Suppose that the robot is located close to the desk of office 1, at distinctive place #9. It receives a command that requires it to go to the library to take a book, that is, to go to distinctive place #11. Thus, the search problem is defined by (n_9, n_{11}). The set of hierarchies available for abstracting both start and goal nodes is the set of all the hierarchies existing in the Multi-AH-graph since both nodes belong to a hierarchical level shared by all the hierarchies. We will use the *service* hierarchy to solve the problem since it seems better than the others for reducing the computational cost (the average size of its clusters is smaller).

After performing hierarchical search, a solution for the search problem at the ground level is $n_9, n_{15}, n_4, n_3, n_{17}, n_{11}$. Its representation at each hierarchy is:[11]

- Solution described in the descriptive hierarchy (useful for communication with human operators):

$$\left\langle \begin{array}{c} Northern\_rooms \cdot Office\_1 \cdot n_9, Northern\_rooms \cdot Office\_1 \cdot n_{15}, \\ Central\_rooms \cdot Corridor \cdot n_4, Central\_rooms \cdot Corridor \cdot n_3, \\ Southern\_rooms \cdot Library \cdot n_{17}, Southern\_rooms \cdot Library \cdot n_{11} \end{array} \right\rangle$$

11. The supernode $n_{Environment}$ is omitted for simplicity.

- Solution described in the docking hierarchy (useful for energy recharging):

$$\left\langle \begin{array}{c} Non\_docking\_north \cdot n_9, Non\_docking\_north \cdot n_{15}, \\ Docking \cdot n_4, Non\_docking\_south \cdot n_3, \\ Non\_docking\_south \cdot n_{17}, Non\_docking\_south \cdot n_{11} \end{array} \right\rangle$$

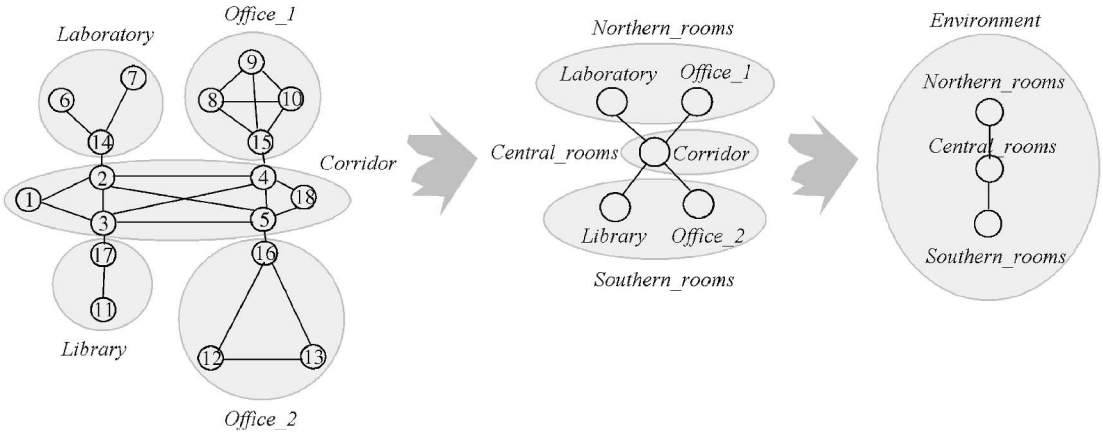
- Solution described in the service hierarchy (useful for statistic purposes):

$$\left\langle \begin{array}{c} Primary\_services \cdot Service\_area\_1 \cdot Service\_north \cdot n_9, \\ Services\_free \cdot Non\_service \cdot Non\_service\_southeast \cdot n_{15}, \\ Services\_free \cdot Non\_service \cdot Non\_service\_southeast \cdot n_4, \\ Services\_free \cdot Non\_service \cdot Non\_service\_west \cdot n_3, \\ Services\_free \cdot Non\_service \cdot Non\_service\_west \cdot n_{17}, \\ Secondary\_services \cdot Service\_area\_0 \cdot Service\_repository \cdot n_{11} \end{array} \right\rangle$$

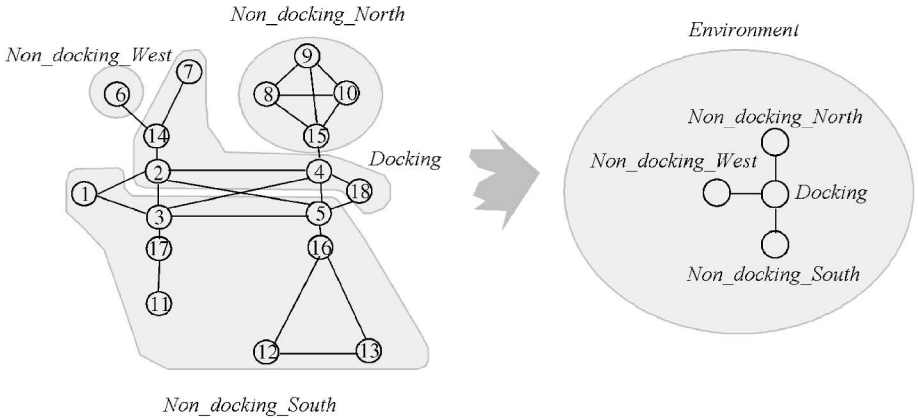
5 Conclusions

In this work, the problem of hierarchical path search has been explored beyond existing approaches, by using multiple hierarchies in the graph model where the search is performed. The paper has presented a formal model of a multihierarchical graph and has shown the advantages of using multihierarchical models rather than single-hierarchy or nonhierarchical ones. These advantages include their better adaptation to a wider range of search problems, their efficiency in solving multiple search problems without using parallelization techniques, and their suitability to express search results on different systems of concepts (different hierarchies of abstraction). Some examples of these features have been presented and the use of multiple hierarchies for solving multiple search problems has been addressed in detail. The results have been described under the Multi-AH-graph model, but their adaptation to different multihierarchical models should not be difficult.

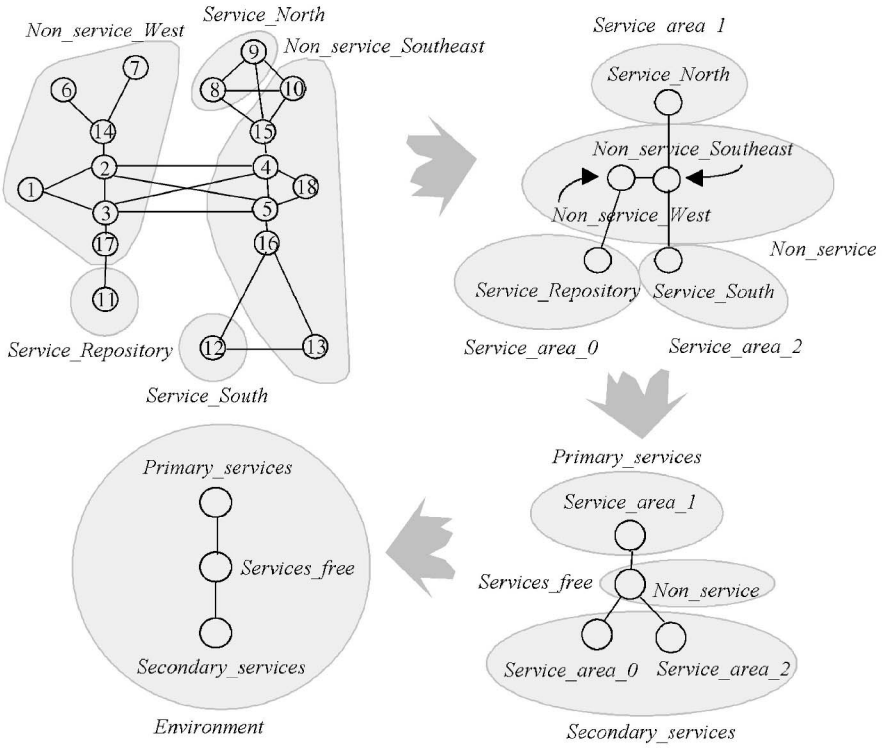
In any situation where searching for paths that connect different elements is required, the adaptive capability



(a)



(b)



(c)

Fig. 11. Multi-AH-graph for the environment of Fig. 10. (a) L(0). Descriptive hierarchy for communications from/to human operators. (b) L(1). Hierarchy of docking points. (c) L(2). Hierarchy of service points. Notice that any subgraph in the figures is connected.

exhibited by multihierarchical models is important. The multihierarchical model presented here and the hierarchical path search algorithm have been completely implemented ([7]).

ACKNOWLEDGMENTS

This work was supported by the Spanish Government under research contract CICYT-TAP99-0948.

REFERENCES

[1] R.A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," *Autonomous Robot Vehicles,* J. Cox and G.T. Wilfong, eds., pp. 290-297, 1990.
[2] A. Bundy, F. Giunchiglia, and T. Walsh, "Building Abstractions," *Proc. AAAI-90 Workshop Automatic Generation of Approximations and Abstraction,* pp. 221-232, 1990.
[3] A. Car and A.U. Frank, "Modeling a Hierarchy of Space Applied to Large Road Networks," Lecture Notes in Computer Science, J. Nievergelt et al., eds., vol. 884, pp. 15-24, 1994.
[4] M. Dario and S. Rizzi, "Dynamic Clustering of Maps in Autonomous Agents," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 11, pp. 1080-1091, 1996.
[5] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.
[6] C. Fennema, A. Hanson, E. Riseman, J.R. Beveridge, and R. Kumar, "Model-Directed Mobile Robot Navigation," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 20, no. 6, 1990.
[7] J.A. Fernández, "Modeling and Generation of Multiple Abstractions for Representing Large-Scale Space: An Application to Mobile Robots," PhD thesis, Dept. of System Eng. and Automation, Univ. of Málaga, Spain, 2000.
[8] J.A. Fernández, J. González, L. Mandow, and J.L. Pérez-de-la-Cruz, "Mobile Robot Path Planning: A Multicriteria Approach," *Eng. Applications of Artificial Intelligence,* vol. 12, no. 4, pp. 543-554, 1999.
[9] J.A. Fernández and J. González, "A General World Representation for Mobile Robot Operations," *Proc. Seventh Conf. Spanish Assoc. Artificial Intelligence (CAEPIA '97),* pp. 35-44, 1997.
[10] J.A. Fernández and J. González, "Hierarchical Path Search for Mobile Robot Path Planning," *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA '98),* 1998.
[11] R.W. Floyd, "Algorithm 97: Shortest Path," *Comm. ACM,* vol. 5, no. 6, pp. 345, 1962.
[12] K. Fujimura, "Time-Minimum Routes in Time-Dependent Networks," *IEEE Trans. Robotics and Automation,* vol. 11, no. 3, pp. 343-351, 1995.
[13] E. Giunchiglia and T. Walsh, "Using Abstraction," Technical Report #9010-08, Instituto per la Ricerca Scientifica e Tecnologica, Italy, 1990.
[14] P.E. Hart, N.J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 4, no. 2, 1968.
[15] R.C. Holte, C. Drummond, M.B. Pérez, R.M. Zimmer, and A.J. MacDonald, "Searching with Abstractions: A Unifying Framework and New High-Performance Algorithm," *Proc. 10th Canadian Conf. Artificial Intelligence (AI '94),* pp. 263-270, 1994.
[16] R.C. Holte, M.B. Pérez, R.M. Zimmer, and A.J. MacDonald, "The Tradeoff Between Speed and Optimality in Hierarchical Search," Technical Report TR-95-19, Univ. of Ottawa, Canada, 1995.
[17] P.D. Holmes and E.R. Jungert, "Symbolic and Geometric Connectivity Graph Methods for Route Planning in Digitized Maps," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 5, pp. 549-565, 1992.
[18] H. Hu and M. Brady, "Dynamic Global Path Planning with Uncertainty for Mobile Robots in Manufacturing," *IEEE Trans. Robotics and Automation,* vol. 13, no. 5, pp. 760-767, 1997.
[19] N. Jing, Y.-W. Huang, and E.A. Rundensteiner, "Hierarchical Optimization of Optimal Path Finding for Transportation Applications," *Proc. Fifth Int'l Conf. Information and Knowledge Management (CIKM '96),* pp. 261-268, 1996.
[20] R.E. Korf, "Planning as Search: A Quantitative Approach," *Artificial Intelligence,* vol. 33, pp. 65-88, 1987.
[21] B.J. Kuipers, "A Hierarchy of Qualitative Representations of Space," *Proc. Working Papers 10th Int'l Workshop Qualitative Reasoning (QR '96),* 1996.
[22] A. Ollero, A. Simon, F. García, and V.E. Torres, "Integrated Mechanical Design of a New Mobile Robot," *Proc. Int'l Federation of Automatic Control Symp.,* 1992.
[23] I.P. Park and J.R. Kender, "Topological Direction-Giving and Visual Navigation in Large Environments," *Artificial Intelligence,* vol. 78, 1995.
[24] E. Remolina, J.A. Fernández, B.J. Kuipers, and J. González, "Formalizing Regions in the Spatial Semantic Hierarchy: An AH-graphs Implementation Approach," *Spatial Information Theory; Cognitive and Computational Foundations of Geographic Information Science,* C. Freksa and D.M. Mark, eds., pp. 109-124, 1999.
[25] E. Rich and K. Knight, *Artificial Intelligence.* McGraw-Hill, 1994.
[26] S. Shekhar, A. Fetterer, and B. Goyal, "A Comparison of Hierarchical Algorithms for Shortest Path Computation in Advanced Travel Information Systems," Technical Report 96-046, Univ. of Minnesota, 1996.
[27] A. Stentz, "Map-Based Strategies for Robot Navigation in Unknown Environments," *Proc. AAAI Symp. Planning with Incomplete Information for Robot Problems,* 1996.
[28] C. Thorpe and J. Gowdy, "Annotated Maps for Autonomous Land Vehicles," *Vision and Navigation: The CMU NavLab,* 1990.
[29] R.J. Trudeau, *Introduction to Graph Theory.* New York: Dover, 1993.
[30] J.A. Tupper, "Graphing Equations with Generalized Interval Arithmetic," PhD thesis, Univ. of Toronto, Canada, 1996.
[31] Q. Yang and J.D. Tenenberg, "Abstraction in Nonlinear Planning," Technical Report CS-91-65, Univ. of Waterloo, Canada, 1994.
[32] J.Y. Yen, "Finding the *k* Shortest Loopless Paths in a Network," *Management Science,* vol. 17, pp. 712-716, 1971.
[33] D. Zhu and J.-C. Latombe, "New Heuristic Algorithms for Efficient Hierarchical Path Planning," *IEEE Trans. Robotics and Automation,* vol. 7, no. 1, pp. 9-20, 1991.

Juan-Antonio Fernández-Madrigal received the MS degree in computer science from the University of Málaga, Spain, in 1994, and the PhD degree in computer science from the same university in 2000. He was granted by the Spanish Government from 1996 to 1999 as a predoctoral researcher. From July 1998 until September 1998 he was with the Computer Science Department of the University of Texas at Austin as a visiting scholar with Professor Benjamin J. Kuipers, working on multihierarchical representations of space. Currently, he is an assistant professor at the System Engineering and Automation Department of the University of Málaga, Spain. His research interests include tools for developing distributed software for autonomous mobile robots, and cognitive robotics.

Javier González received the BS degree in electrical engineering from the University of Seville, Spain, in 1987. He joined the Department of Ingenieria de Sistemas y Automatica and received the PhD degree at the same university in 1988 and 1993, respectively. From 1990 to 1991 he was at the Field Robotics Center, Robotics Institute, Carnegie Mellon University (USA) working on mobile robots. Currently, he is an associate professor at the University of Málaga and has lead several national projects on mobile robots. His research interests focus mobile robot autonomous navigation, world modeling, and computer vision. In these areas, he is author of more than 30 papers in international conferences and journals. Dr. Gonzalez has also written a book about computer vision in Spanish. He is a member of the IEEE.