Life-Long Optimization of the Symbolic Model of Indoor Environments for a Mobile Robot

Cipriano Galindo[*], Juan-Antonio Fernández-Madrigal[*], Javier González[*],
Alessandro Saffiotti[†], and Pär Buschka[†],
[*]System Engineering and Automation Department
University of Málaga
Campus de Teatinos, 29071 Málaga, Spain
Email: {cipriano,jafma,jgonzalez}@ctima.uma.es
[†]Center for Applied Autonomous Sensor Systems
Dept. of Technology, Örebro University
S-70182 Örebro, Sweden
Email: {alessandro.saffiotti,par.buschka}@aass.oru.se

(Regular Paper)

This work was supported by the Spanish Government under research contract DPI2005-01391, and by the Swedish KK Foundation.

Abstract

The use of a symbolic model of the spatial environment becomes crucial for a mobile robot which is intended to operate optimally and intelligently in indoor scenarios. Constructing such a model involves important problems not solved completely at present. One is called *anchoring*, which implies to maintain a correct dynamic correspondence between the real world and the symbols in the model. The other problem is *adaptation*: among the numerous possible models that could be constructed for representing a given environment, optimization involves the selection of one that improves as much as possible the operations of the robot. To cope with both problems, in this paper we propose a framework that allows an indoor mobile robot to learn automatically a symbolic model of its environment and to optimize it over time with respect to changes in both the environment and the robot operational needs through an evolutionary algorithm. For coping efficiently with the large amounts of information that the real world provides, we use *abstraction*, which also helps in improving task planning. Our experiments demonstrate that the proposed framework is suitable for providing an indoor mobile robot with a good symbolic model and adaptation capabilities.

Index Terms

Robotics, World Modeling, Evolutionary Algorithms, Anchoring, Symbol Grounding, Situated Agents.

I. Introduction

The use of a symbolic representation of the environment becomes crucial in indoor robotic applications where a situated agent is intended to operate deliberatively and as optimally as possible. In the literature there are mainly three types of environment representations: geometric [2], [24], [46], topological [31], [9], and hybrid [44], [36], [47]. In this paper we are concerned with symbolic representations, since for deliberation the information stored by the robot must represent the world through *qualitative* symbols rather than by quantitative data acquired by sensors. In that sense, we are closer to topological and hybrid approaches, which use explicit qualitative symbols to represent part of physical reality, than to purely geometrical ones, although the qualitative nature of some geometrical representations (mostly those based on feature extraction [2]) makes them also suitable for our approach.

A symbolic representation enables a robot to intelligently deal with the environment, for example for planning tasks (routing, manipulation, automatic recharging, etc.). However, maintaining such a representation involves several important problems, some of them not completely solved or well understood yet. In this paper we are interested in three of them: (i) maintaining the coherence between the real world and the internal symbols, (ii) processing efficiently large amounts of information, which is an important problem for robots that work in real environments (not only large-scaled), and (iii) optimizing the world representation, that is, selecting among all the possible coherent structures of symbols that are suitable for representing a given environment, the most appropriate one for operating as efficiently as possible. Due to the complexity of these problems, the paper is mostly focused on issue (iii). Issues (i) and (ii) are covered through a general framework described below.

Problem (i) is tightly related to the *symbol grounding problem* in psychology [26]. This issue is not completely solved yet, but in the mobile robotic arena it has been addressed recently through the so-called *anchoring* [10], [5]. It has also appeared under other denominations or implicitly in some communities, such as SLAM [45], [25] or visual tracking [28], [40], [1]. Anchoring means to connect symbols of the internal world model of an agent to the sensory information acquired by observing physical objects. These connections can be reified in data structures, called *anchors*, which contain estimations of some attributes of the objects. An anchor is therefore a model of the corresponding physical object, and it can be used by the agent as a substitute for it, for instance for task planning or when the real object is not visible. Notice that anchoring is not a static process, but it must update continuously the symbolic information (deleting/adding symbols, changing their attributes, etc.) by updating the anchors.

Problem (ii) arises when real environments with a potentially large amount of quantitative information are considered, for example when a large number of objects have to be manipulated. In that case, the symbolic model may be large or its processing computationally expensive, and therefore it must be equipped with a suitable structure to enable efficient accesses. A well-known approach to address this problem is the use of *abstraction*, which establishes hierarchically-arranged views of the world. It has been demonstrated that the use of abstraction can reduce the cost of processing information significantly, and even can make some exponential problems tractable. Some examples of efficient processing achieved through the use of abstraction in robotics can be found in [18], [21], [17].

Finally, the need to optimize the model (problem (iii)) arises from the facts that a general-purpose robot does not carry out the same tasks all the time, and that the environment may change.

Changes in the environment may lead to reducing the suitability of the current internal symbolic model for task planning and execution. Also, changes in the operational needs (e.g., some tasks being executed more frequently than others) can make necessary a better arrangement of symbols. Hence, there is a need for some procedure to dynamically optimize the current representation of the world with respect to environmental changes as well as to the characteristics of robot tasks. This optimization procedure can be appropriately addressed by heuristic optimization techniques over the operating life of the agent, e.g., *evolutionary algorithms* [3], as we show in this paper. Notice that we do not try to optimize anchoring, but to adapt the internal model to the dynamics of the already anchored "ground" symbols. Thus, in this paper, we call optimization the adaptation of these higher-levels of the symbolic representation. This simplifies issues (i) and (iii) by decoupling the problem of anchoring from the others.

To the best of our knowledge, the three problems above have not been jointly addressed in any robotic application. In the literature, many works can be found to endow mobile robots with symbolic representations of the environment. Most of these use topological representations [51], [42], [49], [6] and often only for path-planning, (by contrast, our proposal allows us to consider tasks different from navigation/route planning). Many of the existing works cope with the automatic construction of the symbolic representation of the robot space (topology) by anchoring distinctive places from the robot environment to symbols in the topology. Some of them also apply this topological (or in, general hybrid) techniques to large environments by hierarchically arranging the symbolic and also some geometric information [29], [32], [49]. However, not much attention has been paid to the optimization of that internal symbolic structure. As Wah stated in 1989 [50]:

> Despite a great deal of effort devoted to research in knowledge representation, very little scientific theory is available to either guide the selection of an appropriate representation scheme for a given application or transform one representation into a more efficient one.

This statement is still applicable today: only a few works aim to organize symbolic information [41], [43], and they do not pursue the optimality of robot operation under changing conditions. The work presented here focuses on this less explored direction.

In this paper we jointly cope with the commented problems by using a life-long approach that distributes the optimization work along the robot's operational life, considering the main aspects of the dynamics of its environment. This is necessary since a good model constructed at a given time may be not so good in the future. With this idea in mind, we propose a general framework, called ELVIRA, where the three afore-mentioned problems fit well, that is particularly focused on the optimization of the symbolic representation over the robot operational life (problem (iii)) for indoor applications. An evolutionary algorithm has been developed and adapted as an any-time algorithm: the robot has a correct symbolic representation available at any time, but the representation becomes better the longer the algorithm is executed. For dealing with the structuring of the model to manage efficiently large amounts of data (problem (ii)), we have implemented abstraction in a special graph-based structure called *AH-graph*, that has demonstrated its suitability for improving robot task planning within large environments [21], [16]. Finally, to prove the suitability of our approach within a real robotic application, we have integrated into ELVIRA some basic anchoring techniques for extracting topological maps from grid-maps [13], [8], and also image processing techniques for anchoring objects [22] (problem (i)).

The structure of this paper is as follows. The next section describes the general framework ELVIRA. Section III focuses on our implementation of the anchoring part, while section IV gives the details of the model optimization process. In section V some discussion and experimental results are presented. Finally, conclusions and future work are outlined.

II. The ELVIRA Framework

The ELVIRA framework enables a mobile robot to plan and execute its tasks correctly and efficiently within a dynamic and large environment by automatically creating and optimizing a hierarchical symbolic representation of that environment over the robot operational life. ELVIRA has been implemented as a computational system (see figure 1) with two inputs: the task to be performed at each time and the environmental information gathered by the robot sensors, and two outputs: a plan that solves the requested task as best as possible given the knowledge acquired up to the moment, and a hierarchical symbolic structure anchored to the environmental information and optimized for the set of tasks that the robot has carried out up to the moment.

In our implementation, we use sensor data (color images and range-scan data) to anchor symbols which refer to simple objects, rooms, and distinctive places for navigation. We arrange these symbols produced by anchoring into a hierarchical graph-based structure called *AH-graph*

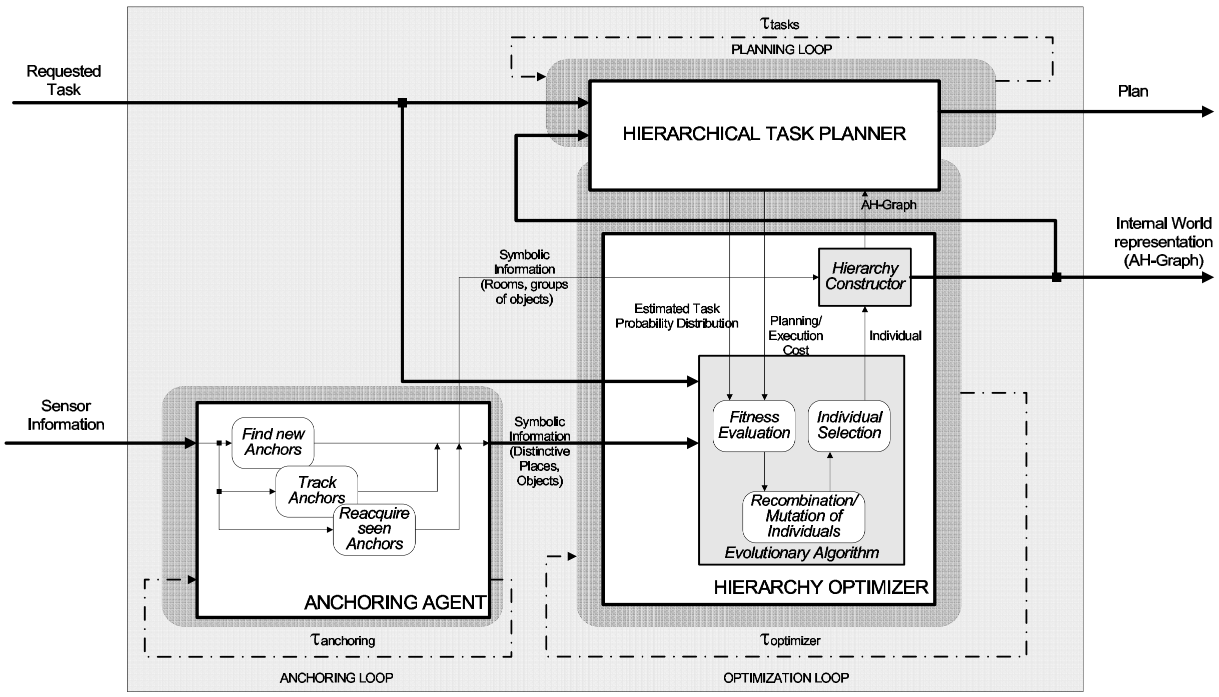


Fig. 1. A general view of our framework ELVIRA. It is fed with both the information gathered by the robot sensors and the requested task. It yields the best known symbolic world representation adapted to the agent tasks and environment, and a resulting plan for the currently requested task. ELVIRA includes three internal loops -anchoring, planning, and optimization- (marked in the figure by dark-gray shaded regions) that run concurrently.

(deeply described in [16], [18]). Briefly, an AH-graph (Annotated, Hierarchical graph) arranges nodes in different layers of detail, called hierarchical levels (see figure 2), ranging from the lowest level, called *ground level*, that contains the maximum detail that is available, to the highest level, called *universal level*, that typically represents the whole environment with a single symbol. Levels in an AH-graph are flat graphs whose nodes are symbols (or group of symbols) and whose arcs represent different relations between them (for example, “navigability”, “relative location”, “proximity”, etc.). Upon the ground level, which varies over time to capture the dynamics of the environment (by means of anchoring techniques), ELVIRA constructs and maintains the upper levels to achieve good symbolic arrangements. We define the goodness of a symbolic structure by its benefits on planning and executing general tasks, as described later on.

As shown in figure 1, our framework carries out the automatic creation and optimization of the robot’s world representation through three processes: the *Anchoring Agent*, the *Hierarchy*

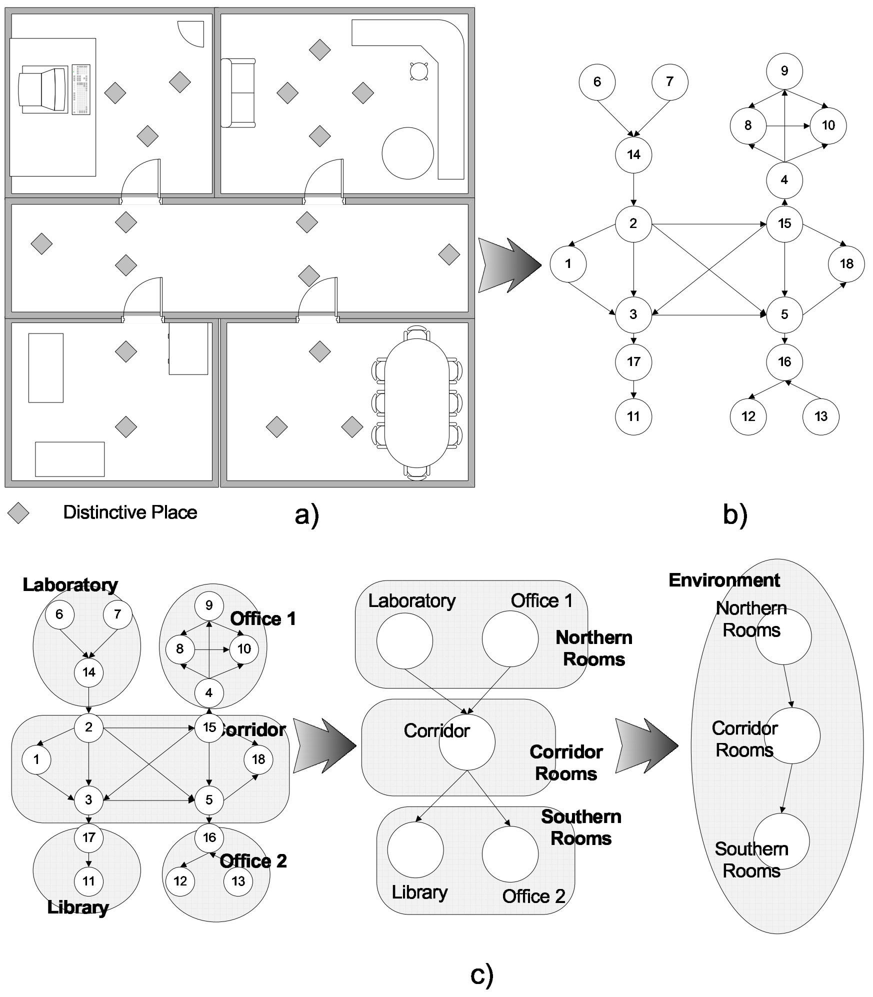


Fig. 2. An example of an AH-graph for modeling space. (a) A schematic map of a real environment. Distinctive places for robot navigation are marked with small diamonds. (b) Ground level of the AH-graph where nodes are symbols that represent those distinctive places. (c) Upper levels of the hierarchy. In these levels, nodes represent more abstract spatial elements, like rooms, areas, buildings, etc.

Optimizer, and the *Hierarchical Task Planner*, which we consider periodically executed with periods denoted in figure 1 as $\tau_{anchoring}$, $\tau_{optimizer}$, and τ_{tasks}. That is, every $\tau_{anchoring}$ time-units the ground level of the symbolic structure is updated with the information collected by the anchoring process; every $\tau_{optimizer}$ the optimization process works to improve the internal hierarchical representation; and every τ_{tasks} a general task (not only navigation) is required to be planned and executed by the robot[1]. Notice that this period-based scheme, which is very useful for evaluation purposes, can be changed into an event-based one with minor modifications.

The next sections describe in more detail the main components of our framework.

III. The Anchoring Process

The *symbol grounding* problem [26] has been present in robotics since its beginnings, although it has not been completely nor satisfactorily solved yet. In a nutshell it consists of how to create and maintain symbols that represent, among others, physical perceived objects. The symbol grounding problem is not the focus of this paper, but an important issue in philosophy and psychology. In this paper, we address a more practically-oriented version of this problem called *perceptual anchoring*.

A. Perceptual Anchoring

Perceptual anchoring or simply *anchoring* is a topic explored explicitly only in the recent robotics literature [10], [5], [19]. It can be thought of as a special case of the symbol grounding problem [26] where the emphasis is put on: (i) symbols that denote only physical entities, and (ii) the maintenance of the connection between these symbols and the corresponding entities via perception while their properties change. Broadly speaking, anchoring consists of connecting internal symbols denoting objects to the dynamic flow of sensor data that refer to the corresponding objects in the external world. Such a connection is achieved by using the so-called *predicate grounding relation*, that embodies the correspondence between predicates in the world model and measurable values from sensor information. For instance, the symbol `room-22` that fits the predicates 'room' and 'large' could be anchored to a local geometric map with certain parameters (shape and dimensions) that corresponds to that room.

The anchoring process, as defined in [10], relies on the notion of *anchors*: internal representations of physical objects that collect all the relevant properties needed for (re-)identifying them and for acting upon them.

The anchoring process comprises three procedures which are concerned with the creation and the maintenance of anchors:

[1]This sampling scheme implies that the internal world representation may be temporarily inconsistent with the environment for short intervals of time. This case is not considered in this paper.

- *Find*. This is the first step, in which an anchor is created for a given object in the environment the first time that it is perceived by the robot, that is, when a certain piece of sensor data fulfills the requirements of a predicate grounding relation, e.g., size and color in an image, which identify a particular object.
- *Track*. During this phase a previously created anchor is continuously updated to account for changes in its properties, using a combination of prediction and new observations.
- *Reacquire*. This phase occurs when an anchored object is re-observed by the robot sensors after it was not seen for some time since. This may happen because the object was occluded, or because it was out of the current field of view of the sensors.

In general, the properties stored in the anchor can be used to estimate the current object's parameters (i.e., its position) for the purpose of controlling action, or to recognize the object when it reappears in the sensor's field of view.

B. The Anchoring Agent

Within ELVIRA, the Anchoring Agent implements a simplified perceptual anchoring process that uses color images and range-scan data to create symbolic representations of objects and of places. The main characteristics of the Anchoring Agent are as follows.

a) The anchors: We only consider three classes of world's elements: *rooms*, *distinctive places* inside them, e.g., the center and the entrance of each room, and *boxes*. Anchors for places and rooms are grounded on sensor data collected by the sonar sensors as explained below; their properties include the position of the place (or center of the room), plus a set of geometric parameters (e.g., room size). Anchors for objects (boxes) are grounded on images collected by a color camera, and include as properties the position of the object, together with its color, shape, and size. All the anchors are linked to ground symbols in the AH-graph world model.

When the anchors are initially created, they are maintained in the robot's "local perceptual space", where their position is stored in robot's coordinates. Anchors in the local perceptual space are continuously updated according to the robot's perceptions and motion. When the robot leaves the current room and enters another one, the anchors in the local perceptual space are discarded, and they are replaced by the ones corresponding to the objects and places in the new room. However, the properties of the discarded anchors are not lost, but they are stored together with the associated symbol in the AH-graph (long-term) world model.

b) Anchor creation: According to the nature of the world entity to be anchored, we distinguish two different mechanisms: (b1) anchoring of simple objects and (b2) anchoring of topological entities.

For (b1) we use the well-known *CamShift* algorithm [1] to detect areas in the camera images with a given distinctive color (red and green in our experiments). When a portion of image meets such a restriction, that is, it can be considered as a box, a new anchor is created for it, storing the position of the box with respect to the current position of the robot as a relevant parameter of the anchor. This information will serve to relate the created symbol that represents the box to the distinctive place where the robot is [22]. Other information included in the anchor is the color and size of the box, in order to distinguish it from other boxes in nearby positions.

Distinctive places as well as open spaces (mechanism (b2)) are anchored using the technique for topology-based mapping presented in [13], although other methods for topological map building could also be used ([32], [44], [29], [9], [30]). Our technique uses the data gathered from ultrasonic sensors to build an occupancy grid-map of the surroundings of the robot (see figure 3a). This grid-map can be seen as a gray-scale image where values of emptiness and occupancy correspond to gray-scale values. We then apply image processing techniques to segment the grid map into *large open spaces*, that are anchored as rooms.

Using the segmented grid map, the Anchoring Agent can also compute some room information like the dimension and shape of regions, as well as distinctive places of the topology such as the center and connection points between rooms. It should be noted that this technique relies on the assumption that the robot operates in indoor environments which have a room/corridor structure. Under this assumption, it has been shown to produce reliable maps [8].

c) Anchor maintenance: Maintenance of anchors, in general, is made through the Track and Reacquire primitives by: (1) predicting how the properties in the anchor will change over time; (2) matching new perceptual information with these predictions; and (3) updating the properties accordingly. We perform in this paper a simple maintenance for anchors: the only time-varying property that we consider is the relative location of objects with respect to the robot. Thus, prediction coincides with self-localization of the robot, which can be solved through well-known methods ([45], [25]).

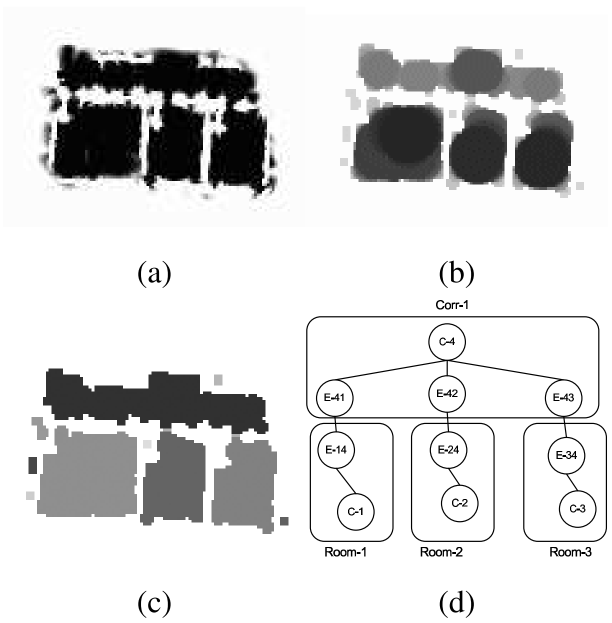


Fig. 3. (a) Original grid-map; (b) Fuzzy morphological opening; (c) Watershed segmentation; (d) Spatial hierarchy involved in this example. Distinctive places are marked with Ci (the center of room i) and Eij (the connection from room i to room j).

The maintenance of anchors stemmed from objects (boxes) implies searching in the model for previously anchored objects with similar parameters, that is, at the same location.[2] If there exists an anchored symbol in the model which is recognized as the one currently detected, its relative position is updated; otherwise, it is considered as a new object, and thus, a new anchor is created. (See [33] for more on perceptual management strategies).

For topological anchors, new perceptual information is fused with the current map [14]. If the fusion is good, the robot updates its self-localization (or, from a dual point of view, it updates the relative position properties of the anchors). If not, the robot may create new anchors corresponding to the newly observed entities (rooms or open spaces).

It should be noted that anchoring is, in general, a very difficult problem, which has to cope with the great uncertainties stemming from the interpretation of perceptual data. A reliable treatment of anchoring would have to deal, among other things, with the problems of uncertainty in the perceived properties, ambiguity in data association, and multi-modal sensor fusion (ee,

[2]Since the errors in estimating robot pose are around 4 cm (following the approach presented in [4]), and the expected errors in computing the relative position of objects with respect to the robot is around 15 cm, we have considered that two objects are the same when they are separated less than 20 cm.

e.g., [19], [7], [33] for some work on these problems in the context of anchoring). Since the focus on this paper is on the optimization of the symbolic model, however, we have preferred to use here a simple version of the anchoring process that solves ambiguous cases by simply selecting the best matching candidate according to simple metrics. As noted above, we have used probabilistic techniques for position update (through SLAM), but here too we only keep the most probable hypothesis. This approach was sufficient in our experiments, since we have used easily distinguishable objects (colored boxes) well spaced between them. In practice, we have also relied on a human supervisor to spot and avoid possible errors in the anchoring process. Clearly, a fully reliable autonomous robot would need to use more sophisticated techniques for vision, localization, and anchoring, but investigating this is out of the scope of this paper.

IV. Optimization of the Internal Symbolic Representation

Apart from the need of being coherent with the real world while planning/executing tasks, a situated autonomous agent should also be able to solve these tasks efficiently, which may become a critical issue when dealing with complex operations and/or with large environments. Hierarchical structures, such as the AH-graph model considered in this work, are helpful in coping with the amount of information that a robot may manage for task planning [21]. Nevertheless, information should still be arranged in the proper hierarchical structure for its efficient treatment, that is, the autonomous agent should not only create a given symbolic structure, but also optimize it over time in order to improve its operation within that environment, that is, to adapt itself to the environment. Two questions arise: how to define the goodness of a hierarchical symbolic structure according to the situatedness requirements of the robot, and how to find the best hierarchy (in a combinatorial space of hierarchies) for the current needs of the robot.

Within ELVIRA, the Hierarchy Optimizer process takes care of the generation and optimization of hierarchical representations with respect to a certain goodness function. This optimizer is implemented through an evolutionary algorithm that explores the combinatorial space of hierarchies over time while taking into account changes in the ground symbols due to the dynamics of the environment. Notice that the combinatorial size of the space of hierarchies makes intractable to generate the optimal one by brute-force methods. We have chosen an evolutionary approach since this paradigm is usually useful in finding good solutions to hard problems for which no polynomial-time algorithms are known to exist [48], [23]. Evolutionary algorithms have been widely applied in the robotic literature, for example for adapting the behavior of mobile robots, for pose estimation, map matching, etc [20], [35], [37]. Some approaches exist for graph clustering, similar to our model optimization process described further on [38], [34]. A general, in-depth classification and explanation of evolutionary algorithms can be found in [3].

In the following subsections we describe our method for optimizing graph-based hierarchies that represent the symbolic knowledge about the environment. We detail the encoding of individuals in the genetic algorithm scheme, the recombination/mutation processes, the evaluation of the fitness (the goodness) of a hierarchy, as well as a study of the complexity and memory usage of our method.

A. Population

The Hierarchy Optimizer starts from a random population of hierarchies that represent different valid possibilities of structuring the symbolic information of the environment. These individuals encode the minimum set of parameters, called *strategy parameters*, to generate the upper levels of AH-graphs upon a ground symbolic level that is maintained by the Anchoring Agent.

An individual of the population (an AH-graph) can be constructed by clustering algorithms that group nodes of a certain hierarchical level to produce the next higher level [16]. In our work we have implemented a clustering algorithm (see the pseudo code in figure 4) that generates clusters given a set of seed nodes: initially, each cluster contains only one seed node, while the rest of non-seed nodes are iteratively added by following a deterministic order in the connectivity of the graph.

Since each hierarchy is a sequence of clusterizations starting at a ground graph (the ground level), we can define it as a set of *chromosomes*, where each one holds information to create a hierarchical level from a lower level through a set of seed nodes. Thus, an individual i_a will represent a hierarchy with k levels constructed upon the common ground level by encoding a vector of *k-1* chromosomes:

$$i_a = \{c_1, c_2, \ldots, c_{k-1}\} = \left\{\left\{n_1^1, \ldots, n_p^1\right\}, \left\{n_1^2, \ldots, n_m^2\right\}, \ldots, \left\{n_1^{k-1}, \ldots, n_r^{k-1}\right\}\right\} \tag{1}$$

where n_i^q is the unique identifier of the *i-th* seed taken from level q that will produce a cluster at level $q+1$. Seeds of the first chromosome refer to nodes of the ground level (that is, symbols acquired directly by the Anchoring Agent), while seeds from the other levels refer to clusters previously generated. Thus, for instance, $n_i^2 = j$ indicates that the *j-th* cluster generated by a seed node of the first chromosome is considered as the *i-th* seed for clustering the second hierarchical level. Figure 5 shows an example of the resultant hierarchy encoded by the 3-chromosome individual *i*=$\{$*(2,8,10), (1,3), (2)*$\}$ upon a given ground graph.

```
PROCEDURE Clustering (Graph g, Seeds s):Clusterization cl
cl={}
FOR i = 1 to size(s)
        cluster = new cluster( s(i) )
        cl=cl + cluster
END
WHILE (UnclusteredNodes()==TRUE)
        FOR j= 1 to size(cl)
            out=ConnectedNodes(cl(j))
            FOR k=1 to size(out)
                IF (Unclustered(out(k))
                    cl(j)=cl(j) + out(k)
                END
            END
        END
END
RETURN cl
```

Fig. 4. Pseudo code of the implemented clustering algorithm. Initially it creates a set of clusters each one containing a single seed (a node from the given graph). These clusters grow up progressively by including connected nodes, in a deterministic order, that do not belong to other clusters. The process finishes when all nodes are clustered. The function *ConnectedNodes* explores the outgoing arcs from a given node to yield the set of its neighbor nodes, while *Unclustered* returns true for a given node when it is not already included in a cluster and *UnclusteredNodes* returns true while nodes from the graph *g* remain not assigned to a cluster. Notice that the algorithm must consider isolated nodes as unitary clusters in order to always end.

B. Individual Recombination

In our hierarchical optimization approach, we combine two individuals (hierarchies) by interchanging part of their genetic information (clustering information), that is, individuals involved in recombination interchange part of their seed nodes. The result is that the original individuals are turned into two new ones (their offspring) containing a mix of the original genetic material.

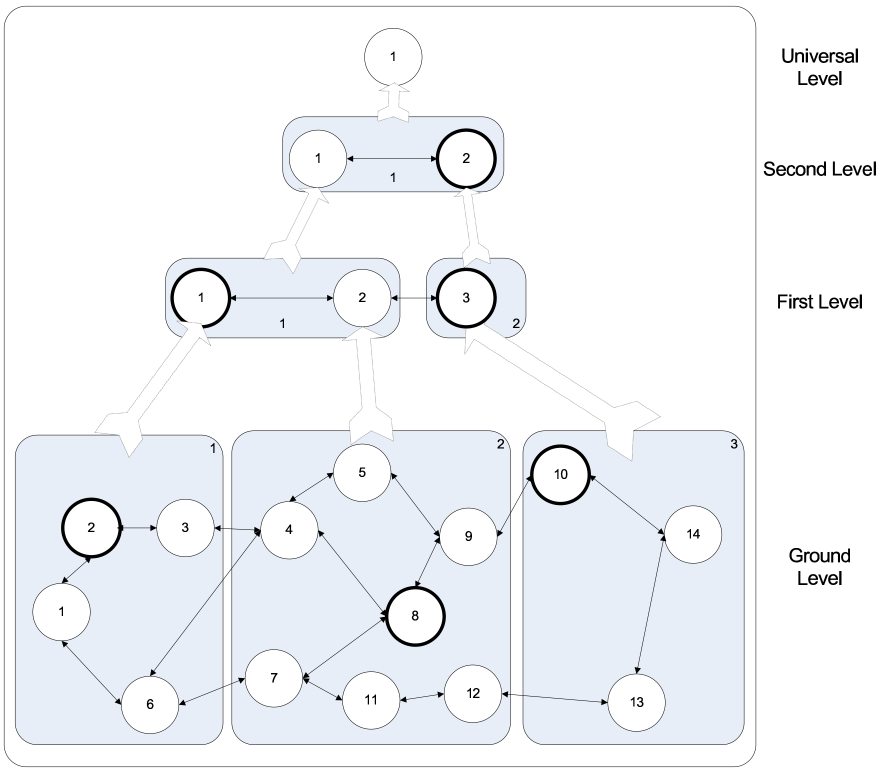


Fig. 5. Example of graph clustering. This is the hierarchy encoded by the individual (*i*={*(2,8,10), (1,3), (2)*}) from a given ground level, after executing successively the clustering algorithm of figure 4. Seed nodes are indicated by thick circles.

Formally, our process for recombination is as follows. Given two individuals ind_1 and ind_2 made up of k and l chromosomes, respectively:

$$\begin{aligned} ind_1 = \{c_1^1, \ldots, c_k^1\} = \Big\{\big\{n_1^1, \ldots, n_a^1\big\}, \ldots, \big\{n_1^i, \ldots, n_w^i\big\}, \ldots, \big\{n_1^k, \ldots, n_c^k\big\}\Big\} \\ ind_2 = \{c_1^2, \ldots, c_l^2\} = \Big\{\big\{m_1^1, \ldots, m_d^1\big\}, \ldots, \big\{m_1^i, \ldots, m_q^i\big\}, \ldots, \big\{m_1^l, \ldots, m_f^l\big\}\Big\} \end{aligned} \tag{2}$$

an *i* chromosome index is selected at random, where $1 \le i \le k, 1 \le i \le l$. Let c_i^1 and c_i^2 be the selected chromosomes of individuals ind_1 and ind_2, that have *w* and *q* seed nodes respectively:

$$\begin{aligned} c_i^1 = \{n_1^i, \ldots, n_w^i\} \\ c_i^2 = \{m_1^i, \ldots, m_q^i\} \end{aligned} \tag{3}$$

A random cross point s ($1 \le s \le w, 1 \le s \le q$) is selected, determining the portion of seed nodes to be interchanged between both chromosomes. The combined chromosomes (denoted with an asterisk) are as follows:

$$\begin{aligned} c_i^{1*} &= \{n_1^i, \dots, n_s^i, m_{s+1}^i, \dots, m_q^i\} \\ c_i^{2*} &= \{m_1^i, \dots, m_s^i, n_{s+1}^i, \dots, n_w^i\} \end{aligned} \tag{4}$$

Thus, the resultant combined individuals are[3]:

$$\begin{aligned} ind_1{}^* &= \left\{\left\{n_1^1, \dots, n_a^1\right\}, \dots \left\{n_1^i, \dots, n_s^i, m_{s+1}^i, \dots, m_q^i\right\}, \dots, \left\{n_1^k, \dots, n_c^k\right\}\right\} \\ ind_2{}^* &= \left\{\left\{m_1^1, \dots, m_d^1\right\}, \dots \left\{m_1^i, \dots, m_s^i, n_{s+1}^i, \dots, n_w^i\right\}, \dots, \left\{m_1^l, \dots, m_f^l\right\}\right\} \end{aligned} \tag{5}$$

It is important to remark that our recombination process does not finish after simply interchanging seed nodes among chromosomes. As commented, seeds of chromosomes that cluster a hierarchical level higher than the ground one represent clusters previously created, and also they can participate in the clusterization of higher levels. That is, a seed node of a chromosome represents a hierarchy which should be interchanged entirely when combining individuals. The following example illustrates the above points.

In the individual *i*=$\{c_1$,c_2,$c_3\}$=$\{$*(2,8,10),(1,3),(2)*$\}$ already represented in figure 5, the seed (3) of c_2 refers to both the cluster generated by the third seed of c_1 (10) and, at the same time, it is referred by the single seed of chromosome c_3 (2). Thus, for instance, moving the seed node (10) of c_1 during recombination should require moving the whole structure generated by (10) to the receptor. For example, using the ground level depicted in figure 5, the result of combining the individual i with another one *h*=$\{c_1$,$c_2\}$=$\{$*(3,13),(1)*$\}$ (see figure 7b), where their first chromosomes are combined and the cross point is set to 1, is performed as follows.

Let the original individuals be:

$$\begin{aligned} h &= \{c_1^h, c_2^h\} = \{(3, 13)\,(1)\} \\ i &= \{c_1^i, c_2^i, c_3^i\} = \{(2, 8, 10), (1, 3), (2)\} \end{aligned} \tag{6}$$

By equation (4), the combination of the selected chromosomes c_1 from both individuals is:

$$\begin{aligned} c_1^{h*} &= (3, 8, 10) \\ c_1^{i\,*} &= (2, 13) \end{aligned} \tag{7}$$

[3]Notice that added seeds may refer to nonexistent clusters within the receptor. In that case, those invalid seeds are not considered in the recombination.

and the resultant crossed individuals (shown in fig. 7) are:

$$\begin{aligned} h^* = \{c_1^{h^*}, c_2^{h^*}, c_3^{h^*}\} = \{(3, 8, 10), (1, 3), (2)\} \\ i^* = \{c_1^{i^*}, c_2^{i^*}\} = \{(2, 13), (1)\} \end{aligned} \tag{8}$$

Notice that after the recombination process, new seeds and an extra chromosome have been added to the individual h, while i has lost one. This is because the interchanged seed (10) from the individual i involves upper hierarchical levels, since it is referred by chromosomes c_2^i, c_3^i. Following the pseudo code of figure 6, function *ContainReferenceToSeed* indicates whether the clusterization of a hierarchical level uses a cluster from lower levels as a seed. Thus, in this example, both functions *ContainReferenceToSeed*$(c_2^i, c_1^i, 10)$ and *ContainReferenceToSeed*$(c_3^i, c_1^i, 10)$ return true since c_2^i and c_3^i both refer to the seed (10)[4] of c_1^i.

After recombination, the individual h^* should inherit the hierarchical structure supported by the seed (10) encoded in the original individual i. Therefore, their chromosomes should contain proper references to the new seeds added (function *AddReferenceToSeed* in the pseudo code of figure 6), and also references to the lost seeds that should be removed (through function *RemoveReferenceToSeed*), as it occurs in the individual i.

C. Individual Mutation

Mutation is a 1-ary operation that imitates a sporadic change in individuals. Although recombination is important to maintain the best characteristics of individuals over generations, mutation is necessary in order to avoid the stagnation of solutions in a local minimum.

In our approach we have followed a simple criterion to perform individual mutation, which consists of eliminating/adding randomly a seed from/to a random chromosome of the individual to be mutated. More formally, the mutation of a particular chromosome c that contains s seeds, defined as $c = (n_1, \ldots, n_s)$, may result in one of the following possibilities:

$$\begin{aligned} c_{add} = (n_1, \ldots, n_s, n_{s+1}) \\ c_{del} = (n_1, \ldots, n_p, n_{p+2}, \ldots, n_s) \end{aligned} \tag{9}$$

[4]This reference can be direct as it occurs in the case of *ContainReferenceToSeed*$(c_2^i, c_1^i, 10)$, or indirect as in *ContainReferenceToSeed*$(c_3^i, c_1^i, 10)$, in which c_3^i contains the seed (2) that refers to the 2-nd seed of c_3^i (3) which finally refers to the 3-rd seed of c_1^i (10).

```
PROCEDURE Recombination (Individual i, Individual h):Individuals
c=SelectChromosomeIndex(i,h)
p=SelectCrossPoint(i(c),h(c))
Seed1=SeedsToBeExchanged(i(c),p)
Seed2=SeedsToBeExchanged(h(c),p)
i(c)=i(c)-Seed1
h(c)=h(c)-Seed2
i(c)=i(c)+Seed2
h(c)=h(c)+Seed1
FOR j= 1 to size(Seed1)
            FOR k=1 to NumberOfChromosomes(i)
                  IF (c!=k && ContainReferenceToSeed(i(k),i(c),Seed1(j)))
                        AddReferenceToSeed(h(k),h(c),Seed1(j))
                        RemoveReferenceToSeed(i(k),i(c),Seed1(j))
                  END
            END
END
FOR j= 1 to size(Seed2)
            FOR k=1 to NumberOfChromosomes(h)
                  IF (c!=k && ContainReferenceToSeed(h(k),h(c),Seed2(j)))
                        AddReferenceToSeed(i(k),i(c),Seed2(j))
                        RemoveReferenceToSeed(h(k),h(c),Seed2(j))
                  END
            END
END
RETURN i,h
```

Fig. 6. Pseudocode for recombination of individuals. After the selection of the chromosomes and the seeds to be exchanged, references to those seeds within the input individuals are detected in order to be updated (in the two ”FOR” loops). The *ContainReferenceToSeed* function checks whether a given chromosome contains a reference to a seed from another chromosome, while *AddReferenceToSeed* and *RemoveReferenceToSeed* are used to insert/erase seeds that refer to clusters previously generated by seeds of other chromosomes.

As noticed when discussing the recombination process, the effect of eliminating/adding seeds may also cause relevant changes in the structure encoded by individuals. For instance, the deletion of a particular seed from a chromosome may modify the number of levels of the hierarchy represented by the individual. This effect can be observed in the following example, where the individual i undergoes the deletion of the first seed of the chromosome c_2, which leads to the elimination of c_3:

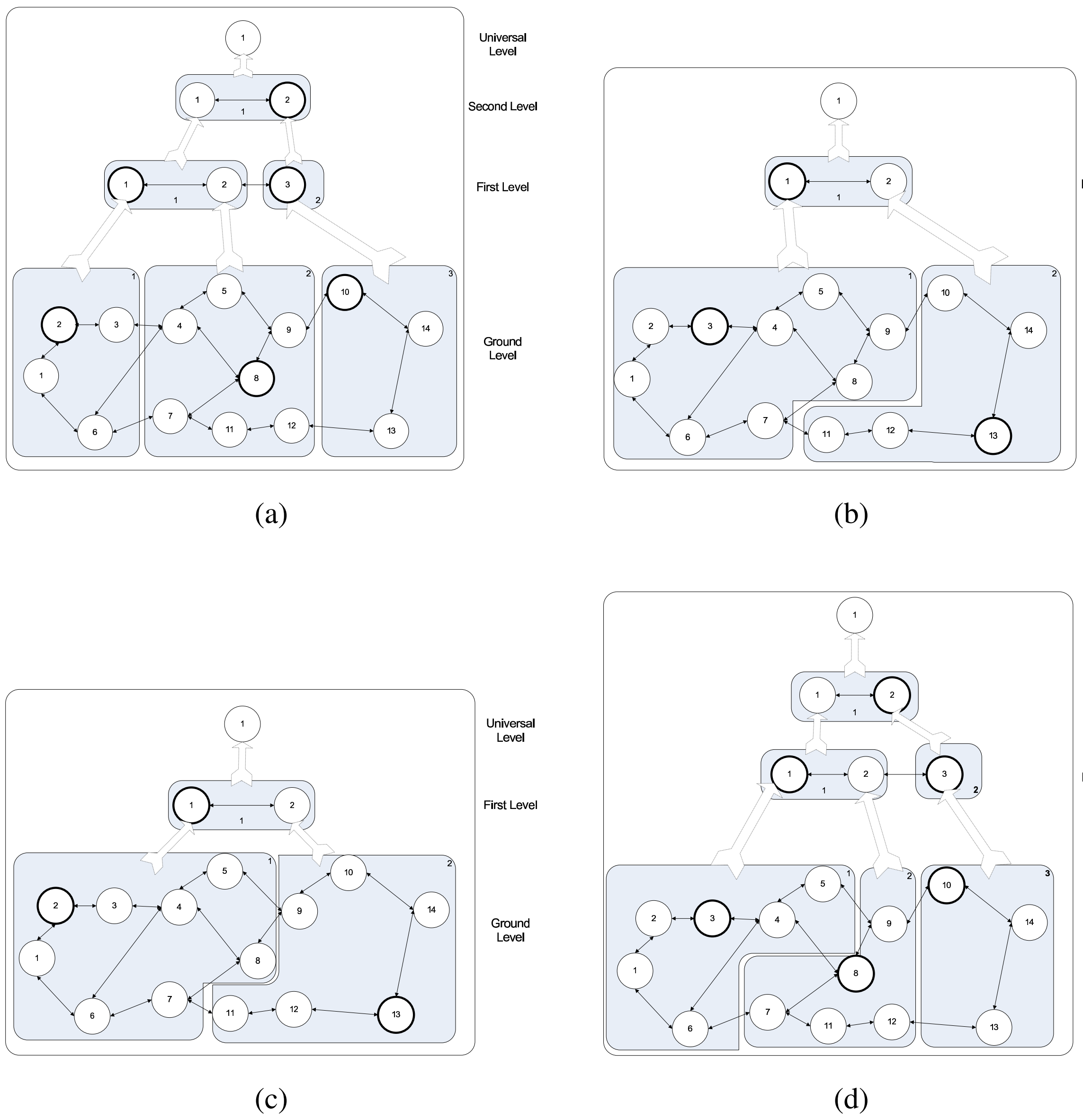


Fig. 7. Recombination of individuals. (a)(b) Hierarchies encoded by the individuals i and h referred in the text. (c)(d) Resultant offspring i^* and h^* after crossing as commented in the text. Notice how recombination combines individual characteristics, such as the number of hierarchical levels.

$$\begin{aligned} i &= \{c_1, c_2, c_3\} = \{(2, 8, 10), (1, 3), (1)\} \\ i^* &= \{c_1, c_2\} = \{(2, 8, 10), (3)\} \end{aligned} \tag{10}$$

Regarding the effect of inserting a random seed into a chromosome, it will only increase the number of clusters at certain levels, but no changes will be produced in the number of levels of the hierarchy.

```
PROCEDURE Mutation (Individual i):Individual
type=RandomSelectTypeofMutation()
c=ChromosometobeMutated(i)
IF (type==Add)
            sn=RandomSeedNode()
            i(c)=i(c)+sn
ELSE
            sn=RandomSeedNode(i(c))
            i(c)=i(c)-sn
            FOR k=c+1 to NumberOfChromosomes(i)
                  IF (ContainReferenceToSeed(i(k),i(c),sn)
                        RemoveReferenceToSeed(i(k),i(c),sn)
                        IF (IsEmpty(i(k))) RemoveChromosome(i(k))
                  END
            END
END
RETURN i
```

Fig. 8. Pseudo code for individual mutation. When a seed is deleted, all references to it are deleted too. When adding/removing seed nodes from a certain chromosome, references from higher ones must be reordered to correctly keep previous references (the two inner 'for' loops).

Notice that the random mutation of individuals (as well as the recombination) may not yield good results in the sense that the offspring is not well adapted to the problem at hand. We rely on the evolutionary process that will eliminate weak individuals through their fitness.

D. Fitness Evaluation

Evolution of the population (AH-graphs) is carried out through the typical genetic operations, recombination and mutation, previously discussed, after a selection process. This selection process, which takes into account the *fitness value* of each individual (that is, the goodness of each AH-graph for improving the situatedness of the robot), has been implemented in our work by using the so-called *roulette wheel* technique[5] [52].

[5]Efficiency has not been our concern here, but simplicity. Instead of this *relative fitness* selection, other types of selection techniques can be used, such as *rank-based* or *tournament* methods [3].

The evaluation of individual fitness involves the construction of the AH-graphs represented by individuals to test their suitability with respect to the current robot tasks. The hierarchy constructor submodule (recall figure 1) uses the information encoded by chromosomes to construct hierarchies based on the clustering algorithm of figure 4. Each constructed AH-graph is then evaluated considering the effort of planning the robot tasks as well as an heuristic estimation of their execution. For this we rely on an efficient task-planning approach called *HPWA* (Hierarchical Planning through World Abstraction) which has been previously presented in [21]. Broadly speaking, this approach takes advantage of the hierarchical structure of the world model by solving the task at a high level of detail, and then uses the resulting abstract plan to ignore irrelevant elements at lower levels with respect to the current goals. In our experiments we have chosen Metric-FF [27] as the embedded planner within the HPWA scheme.

Apart from calculating the individual fitness of each hierarchy, the Hierarchy Optimizer also updates the estimate of the probability distribution (frequency of occurrence) of the tasks over time[6]. With all that information, the final fitness of a given hierarchy H is computed by:

$$\mathit{Fitness}(H) = \frac{1}{\sum_{i=0}^{n-1} \mathit{freq}(t_i) \cdot (p(t_i) + h(t_i))} \tag{11}$$

This function measures the optimality of individuals for planning and executing the current set of robot tasks. In this expression, n is the number of tasks requested to the robot up to the current moment, $\mathit{freq}(t_i)$ is the frequency of occurrence of task t_i, $p(t_i)$ is the planning effort to solve task t_i (planning time) and $h(t_i)$ is an estimate of the cost of executing that task. This execution cost estimate is based on the experiences of the robot in past executions of the task. In our case, we average time consumption, although other estimates can be used, e.g., energy consumption, mechanical stress, or a mixing of several factors, as it is done for route planning in [15]. Notice that including an estimate based on past executions avoids the physical performance of tasks for each individual of the population, which would be unacceptable for on-line optimization.

By considering the cost of planning and execution, the current set of robot tasks, and their frequency of occurrence, the optimization process guides the search to find those symbolic models of the environment better adapted to solve frequent tasks in detriment of other infrequent ones.

[6]Other frequency estimates could be developed, for example, based on Bayes filtering [12].

Notice that the final constructed model can be far from the optimal at first, but our approach guarantees its improvement over time, as it is demonstrated in the experiments of section V.

E. Memory and Time Consumption

In this section, some details regarding memory consumption and computational complexity of our algorithms are given.[7]

1) Memory Consumption: The execution of the genetic algorithm only keeps in memory a set (population) of chromosomes at a given iteration (set in our experiments to *15*, which proved to be sufficient for convergence), and the set of the tasks to be performed. On the one hand, considering the encoding of chromosomes, that is, sequences of seeds (integer numbers), for typical hierarchies with a ground level with n nodes and higher levels that contain half the nodes of the next lower levels, the number of hierarchical levels becomes $log_2 n$ and the number of seeds needed to represent each of these hierarchies is $\sum_{i=1}^{log_2 n} \left(\frac{n}{2^i}\right)$, that is, $O(n)$. In a worse case, in which each hierarchical level reduces only one node, the number of levels is $n-1$ and the number of seeds needed to represent each hierarchy is then $\sum_{i=1}^{n-1} (n-i) = \frac{n(n-1)}{2}$, that is, $O(n^2)$. On the other hand, the information kept in memory for the set of tasks to be performed is negligible: it consists of the definition of the goal to be achieved, i.e. *(at box-1 room-9)*, the frequency of occurrence of each task (a float number), and an averaged value of the execution time of the task (a float number).

2) Computational Complexity: Within the genetic algorithm, chromosomes are constructed at every generation to calculate their fitness. For the clustering algorithm (see figure 4), let r be the number of seeds of a given chromosome that cluster n nodes from the lower hierarchical level, and let c be the average connectivity of a set of nodes. In the worst case, in which $r = n-1$ and $c = n-1$, the complexity of the algorithm is $O(n^2)$ due to the two inner loops. The outer loop (the 'while' loop) ensures that all nodes are clustered, so we can bound the clustering process in the worst case as $O(n^3)$. As this process is repeated per each chromosome of every individual of the population, the computational complexity of calculating the fitness of a population with m individuals becomes, in the worst case, $O(m(n-1)n^3)$, which is $O(n^4)$ when m is a relatively small constant with respect to n.

[7]The complexity study of the anchoring and planning processes is out of the scope of this work.

The recombination process (see figure 6) consists of exchanging seeds between two chromosomes and then updating their references in the upper ones. The complexity of this process is clearly polynomial and it can be bounded as follows. Let s_1 and s_2 be the number of seeds to be exchanged. The first part of the algorithm (select, insert, and remove elements from a list) is polynomial with respect to the number of seeds to be exchanged, that is, $O(max(s_1, s_2))$. The updating of seeds consists of two loops based on s_1, s_2, and on the number of the upper chromosomes of the individuals, which is n in the worst case, as explained in the previous section. Thus, the computational complexity for each part is $O(s_1 n)$, that is $O(n^2)$ in the worst case, being the called functions (*Contain-*, *Add-*, and *Remove-ReferenceToSeed*) linear on the size of the seeds of the chromosome. Thus, the recombination process is $O(\frac{m}{2}n^2)$, considering that each individual is combined with another only once.

Finally, for the study of the computational complexity of the mutation process (see figure 8) only the update of upper chromosomes is considered, and thus, it can be approximated in the worst case as $O(n)$.

The real experiments presented in the next section have been executed on the on-board computer of a mobile robot (Pentium III at 550 Mhz with 256 MB of memory). The convergence of the Hierarchy Optimizer has been reasonable in both configurations, with the times discussed above.

V. Experimental Results

We have evaluated the performance the ELVIRA framework when planning/executing robot tasks. For that, we have conducted a variety of real experiments to test the hierarchy optimization process within a real indoor scenario (subsection A). Subsection B gives a brief discussion about the applicability of our approach to other types of scenarios (outdoor or dynamic).

A. Real Evaluation

For real experiments, ELVIRA has been integrated into a robotic system that runs an implementation of the *Thinking Cap* hybrid architecture, suitable for controlling situated agents [39]. Our experiments have been carried out on a Magellan Pro robot equipped with a ring of 16 ultrasonic sensors and a color camera which provide sensor information to the Anchoring Agent in order to anchor distinctive places for robot navigation and objects (colored boxes of a given size).

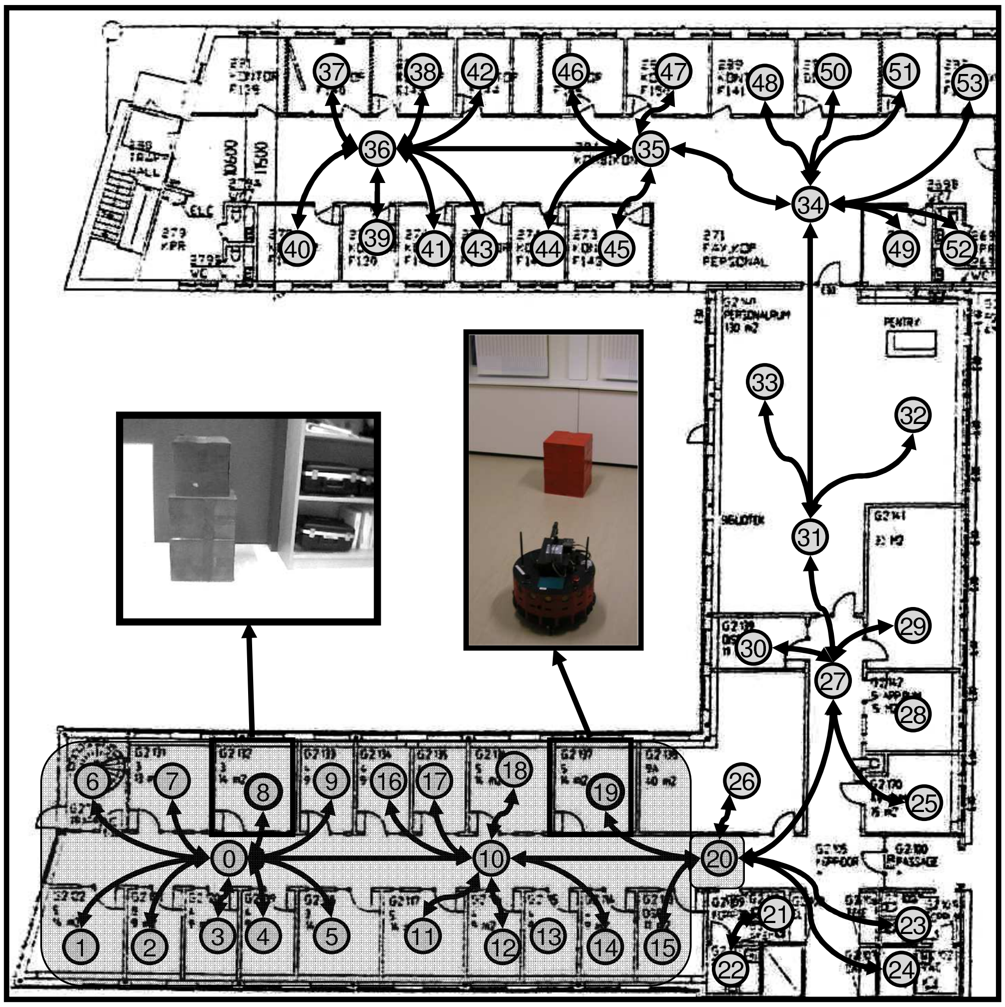


Fig. 9. Real world scenario for our robotic experiment: a part of the Mobile Robotic Lab at the AASS Center in the University of Örebro (Sweden). The shaded region represents the initial set of rooms given to the robot, while the rest has been anchored in successive explorations (for clarity, nodes that represent objects and distinctive places inside rooms are not depicted). In the middle of the figure, we show the robot used in this experiment and the appearance of a typical room.

The test scenario is the office environment depicted in figure 9, made up of several rooms and corridors. It should be noted that although this is not a representative large-scale environment, it is large in fact, since we are considering planning tasks involving not only navigation but also object manipulation, which may easily become an intractable problem [21].

In this experiment, the robot performed within the test scenario during 8 hours of a nonworking day (in three consecutive runs of 2 hours separated by 1 hour for battery recharge). The experiment starts with an initial ground level consisting of a graph of 128 distinctive places (20 rooms/areas), their connections, and 5 boxes placed at certain rooms (see figure 9). Within this initial setup the robot is requested to roam while anchoring continuously new distinctive places and colored boxes found in the environment. In our particular implementation, the Hierarchy Optimizer works to improve the current model with a cycle period of 1 min, which is an upper bound that in our implementation ensures the termination of each optimization step. In parallel, the robot is commanded every 5 min to execute a random task (out of a set of tasks also chosen at random) using its current internal representation of the world. At the end of the experiment, the robot has incorporated into the symbolic model 256 new distinctive places (corresponding to 33 rooms), more than 750 connections between them, and 40 boxes.

Initially we have considered a set of 3 random tasks that involve world elements from the initial ground level given to the robot. Once the robot has explored the major part of the environment (after approximately 40 min), 3 new random tasks are added to the set of robot tasks involving the new anchored information. The type of tasks considered consists of moving a certain box from one location to another, possibly requiring the manipulation of other boxes, e.g., to unstack boxes which are on top of the desired one. More precisely, they include a simple task (Task 3 in figure 10) that only consists of navigating from room 6 to room 14 (see figure 9), and another five for which the robot has to make a plan for manipulating a box (unstack/stack/pick up) from a heap and moving it to another room to placed it on a pile.

Since the chosen robotic platform can not manipulate objects, the robot physically executes only the navigation between the locations involved in the tasks, being the manipulation carried out by a person (although the task-planning process deals with both operations automatically[8].) Both the time required to perform planning and topological navigation are considered to guide the model optimization through the function cost (11).

Figure 10a measures the optimization achieved in the model during the experiment. Observe that after the high raise in the cost of operation at the beginning, which corresponds to the request of a manipulation task (in room 8) and the inclusion in the model of new anchored

[8]Since our concern in this work is the optimization of the internal representation of the robot, possible navigational failures as well as errors in anchoring are not considered in the results presented in this section. In both cases, the robot failure was manually recovered by an operator.

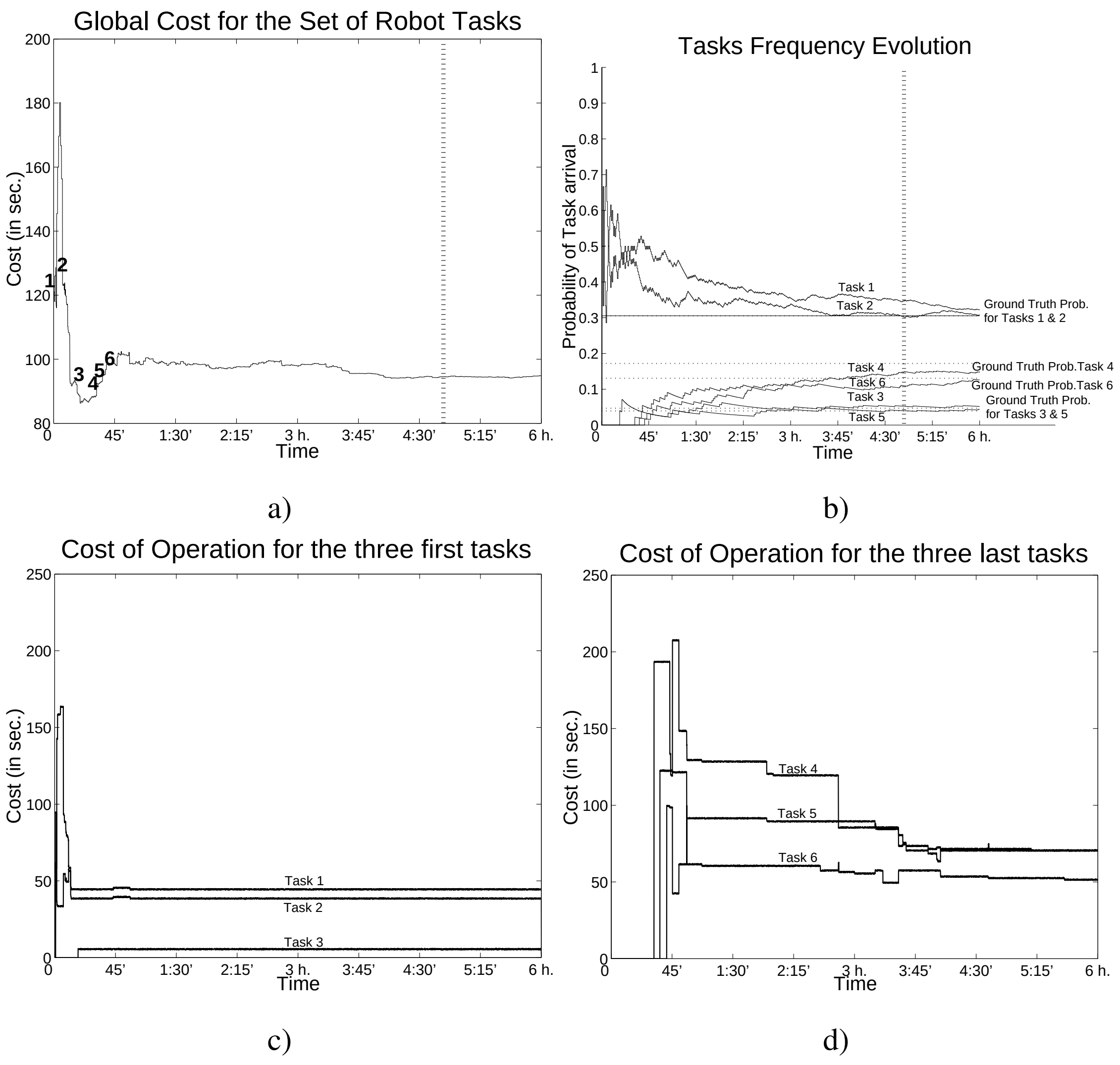


Fig. 10. Model optimization in a real experiment. a) Evolution of the global cost (fitness) for the set of robot tasks, measured according to equation (11). Arrows on the plot indicate the first time that each task is commanded to the robot. Notice how the cost oscillates substantially while the robot has not enough knowledge about its tasks nor the environment. As long as the robot optimizes the symbolic model, oscillation diminishes until a steady-state is reached around the 5th hour of operation. b) Frequency of occurrence of each task perceived by the robot and the ground truth probability of occurrence of tasks (horizontal lines). c-d)Evolution of the cost of solving and executing each robot task along the whole experiment. Notice the overall improvement in planning/execution.

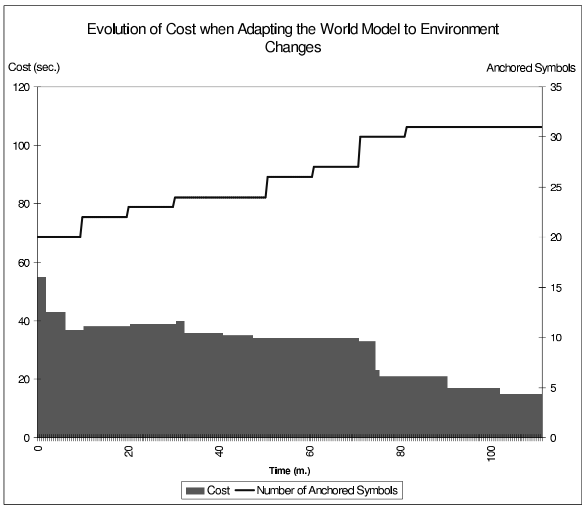
information (a pile found in that room with three new boxes), the optimizer achieves a decrease on the global cost. Thus, after approximately 40 min, the internal symbolic model is optimized to the initial set of tasks and thus the robot has adapted to the environment. After that point, raises in the figures are caused by the inclusion of 3 new manipulation tasks and also by the inclusion in the model of new anchored information. As long as the robot does not perceive changes in the environment, that is, no new anchored information is added to the internal model of the environment (after the 5-th hour), and the estimated frequency of the requested tasks approximates to their actual probability of occurrence (as shown in figure 10b), the overall cost of the robot tasks reaches a steady state. This occurs after around 8 hours from the beginning of the experiment, approximately.

We have also demonstrated that the Hierarchy Optimizer preserves the knowledge acquired by the robot along its operational life. That is, it is far from constructing a new symbolic model from scratch each time world changes. For illustrating that, we consider an experiment similar to the one described before, but the robot is now commanded to solve only one task within a reduced part of our test scenario. Results of this experiment are shown in figure 11. Figure 11a displays the evolution of the cost of operating within the environment, in which the number of distinctive places is progressively incremented. Notice how the increase in the world complexity (the number of distinctive places passes from 20 to more than 30 at the end of the experiment) does not affect the performance of the system. Figure 11b refers to the same results but measuring the cost with respect to the number of places anchored at each moment. This chart shows how the average operation cost per place has been reduced around 75% over time, despite the growth in the complexity of the symbolic representation.

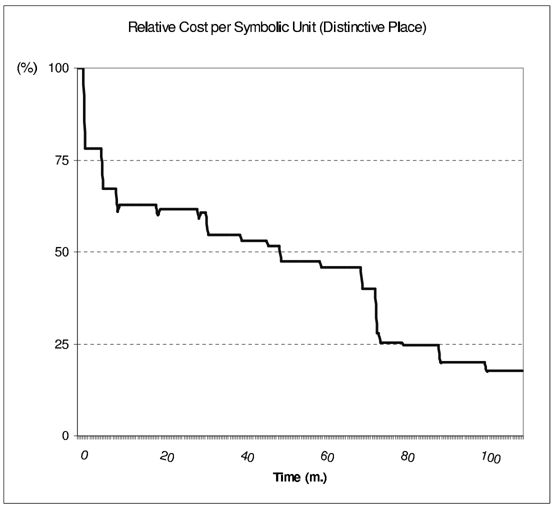
B. Considerations for Outdoor or Dynamic Environments

The approach presented above is mainly intended for, and it has been tested in, indoor environments that have a moderate amount of dynamism. However, we believe that the core ideas of our long-life optimization mechanism could extend beyond these limits, notably to outdoor environments and/or to more dynamic environments, as long as adequate computational power is available.

The application of our approach to outdoor scenarios is possible in principle, provided that a symbolic model of the outdoor environment of interest can be defined. This implies that the environment has enough structure to be amenable to a symbolic representation. However, some thorny problems related to perception should be tackled in this case. Indoor scenarios exhibit some characteristics that provide the symbols that represent the environment with more stability. For instance, light conditions that are of vital importance for visual anchoring of objects are



a)



b)

Fig. 11. Optimization of the AH-graph model under world changes. These charts show the evolution of the cost of operating over time when genetic individuals (AH-graphs) are adapted to changes. a) Observe how the cost only increases slightly when changes are perceived. This is because the robot does not lose completely the experience acquired up to the moment when these changes are detected. Chart b) plots the same results that chart a) but with respect to the number of anchored symbols, normalized by the initial cost, that is, $f(t) = \left(\frac{cost(t)}{places(t)}\right) \cdot \frac{100}{\left(\frac{cost(0)}{places(0)}\right)}$.

more easily controlled in indoor scenarios. Outdoor applications may require more sophisticated anchoring techniques, like the one presented in [11]. This can lead to an important increase in the computational complexity of the approach. Moreover, indoor environments like the one used in our experiments are usually structured in rooms, corridors, etc, which permits the robot to construct maps and extract the type of spatial features (rooms and distinctive places) used in our experiments. For outdoor applications, other features and localization techniques should be considered, as well as different sensors such as infrared cameras or GPS, albeit the symbolic part of our approach should not suffer major changes.

For what concerns highly dynamic scenarios, it should be noted that our approach currently assumes a moderately dynamic environment: objects may be added or removed, but they are otherwise expected to be static. A more dynamic environment would pose two types of problems to the current approach. First, we would need to rely on a more sophisticated anchoring process, able, for instance, to track the position of moving objects, in order to avoid errors in the creation and update of anchors. These errors would produce a great variability in the symbolic model, which might cause the symbolic optimization process lose its convergence. A more sophisticated anchoring process, with advanced tracking capabilities, would also help to cope with cluttered environments. Second, our evolutionary algorithm requires a number of iteration to achieve an acceptable solution. Each time world changes (the ground level of AH-graphs), the algorithm has to face variations on the problem to be solved, and a new period of time is required to attain a solution. Currently, we assume that after a period of time, in which the scenario may gradually change, the environment remain static and unalterable. Further research needs to be conducted to cope with more dynamic situations which would increase the computational cost of our approach. However, neither the hierarchical representation of the space nor its optimization process should undergo fundamental changes.

VI. Conclusions and Future Work

In this paper we have presented a framework to cope with the problem of constructing a near-optimal symbolic model of indoor environments for a situated mobile robot. Our system has been implemented as a software that we call ELVIRA, which has been integrated into a real robot. Our approach considers three issues: (i) the maintenance of the symbolic model anchored and updated with respect to the environmental information, (ii) the structuring of the model for efficient processing in the presence of large amounts of information, and (iii) the optimization of such model with respect to the tasks at hand to improve over time the robot performance.

We rely on abstraction through a mathematical model called AH-graph in order to cope with large environments. The creation and maintenance of the ground dynamic level of the AH-graph is achieved automatically by using an anchoring algorithm that detects rooms or distinctive places for robot navigation, and simple objects. Upon that ground symbolic information, an evolutionary algorithm is responsible for creating a near-optimal hierarchical structure with respect to environmental changes and variations in the robot operational needs. Although evolutionary algorithms are not usually considered in on-line applications, our experiments have shown that when their work is spread over time (considering them as life-long algorithms) they exhibit good results in optimization.

Our short term research is currently aimed to improve the anchoring process in order to make more reliable the robot performance within dynamic scenarios, as well as to integrate ELVIRA in other robotic platforms capable of performing more complex tasks.

REFERENCES

[1] Allen J.G., Xu R., and Jin J. *Object tracking using CamShift algorithm and multiple quantized feature spaces*. Proc. of the Pan-Sydney area workshop on Visual information processing, 2004, Darlinghurst, Australia.

[2] Asada M. *Map Building for a Mobile Robot from Sensory Data*. IEEE Trans on Systems, Man, and Cybernetics, **37**, 6 pp. 1326-1336, 1990

[3] Back T., Hammel U., and Schwefel H. *Evolutionary Computation: Comments on the History and Current State*. IEEE Trans. on Evolutionary Computation, **1**, No. 1 (1997) pp. 3-17.

[4] Blanco J.L., Gonzalez J., and Fernandez-Madrigal J.A., *A Consensus-based Approach for Estimating the Observation Likelihood of Accurate Range Sensors*, IEEE International Conference on Robotics and Automation (ICRA), Rome (Italy), Apr 10-14, 2007.

[5] Bonarini A., Matteucci M., and Restelli M., *Concepts for Anchoring in Robotics. AI*IA: Advances in Artificial Intelligence* (Springer-Verlag, 2001)

[6] Bosse M., Newman P.M., Leonard J.J. and Teller S. *SLAM in Large-scale Cyclic Environments using the Atlas Framework*. Int. Journal on Robotics Research, **23**, 12, pp 1113-1139, 2004.

[7] M. Broxvall, S. Coradeschi, L. Karlsson and A. Saffiotti. *Recovery Planning for Ambiguous Cases in Perceptual Anchoring*. Proc. of the 20th AAAI Conf, pp. 1254-1260. Pittsburgh, PA, 2005.

[8] Buschka P., and Saffiotti A., *A Virtual Sensor for Room Detection*. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, CH, (2002) pp. 637-642.

[9] Choset H., and Nagatani K. *Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization*. IEEE Trans. on Robotics and Automation **17**,2, 2001.

[10] Coradeschi S., and Saffiotti A., *An Introduction to the Anchoring Problem*. Robotics and Autonomous System **43**, No. 2-3 (2003) pp. 85-96.

[11] Doherty P. *The WITAS integrated software system architecture*. Linkoping Electronic Articles in Computer and Information Science, **4**, No. 17, 1999. http://www.ep.liu.se/ea/cis/1999/017.

[12] Doucet A., Godsill S., and Andrieu C.. *On sequential Monte Carlo sampling methods for Bayesian filtering*, Statistics and Computing **10** (2000) pp. 197-208.

[13] Fabrizi E., and Saffiotti A. *Extracting Topology-Based Maps from Gridmaps*. Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), San Francisco, CA, (2000), pp. 2972-2978.

[14] Fabrizi E., Oriolo G., and Ulivi G. *Accurate map building via fusion of laser and ultrasonic range measures*. In D. Driankov and A. Saffiotti editors, *Fuzzy logic techniques for autonomous vehicle navigation*. pp. 257-280. Physica /Springer-Verlag, 2001.

[15] Fernandez J.A., Gonzalez J., Mandow L., and Perez-de-la-Cruz J.L. *Mobile Robot Path Planning: A Multicriteria Approach*. Engineering Applications of Artificial Intelligence, **12**, No. 4 (1991) pp. 543-554.

[16] Fernandez J.A., and Gonzalez J., *Multi-Hierarchical Representation of Large-Scale Space* (Kluwer Academic Publishers, 2001)

[17] Fernandez J.A., Galindo C., and Gonzalez J., *Assistive Navigation of a Robotic Wheelchair using a Multihierarchical Model of the Environment*. Integrated Computer-Aided Engineering **11**, No. 11 (2004) pp. 309-322.

[18] Fernandez J.A., and Gonzalez J., *Multihierarchical Graph Search*. IEEE Trans. on Pattern Analysis and Machine Intelligence **24**, No. 1 (2002) pp. 103-113.

[19] Fritsch J., Kleinehagenbrock M., Lang S., Loemker F., Fink G.A., and Sagerer G., *Multi-model Anchoring for Human-robot-interaction*. Robotics and Autonomous Systems, **43** No. 2-3, (2003) pp. 133-147.

[20] Floreano D., and Mondada F., *Evolution of Homing Navigation in a Real Mobile Robot*. IEEE Trans. on Systems, Man, and Cybernetics **26** (1996) pp. 396-407.

[21] Galindo C., Fernandez J.A., and Gonzalez J., *Hierarchical Task Planning through World Abstraction*. IEEE Trans. on Robotics **20**, No. 4 (2004) pp. 667-690.

[22] Galindo C., Saffiotti A., Coradeschi S., Buschka P. Fernandez-Madrigal J.A., and Gonzalez J. *Multi-Hierarchical Semantic Maps for Mobile Robotics*, IEEE/RSJ IROS 2005, Edmonton, Alberta (Canada), 2005.

[23] Garey M.R., and Johnson D.S., *Computers and Intractability. A Guide to the Theory of NP-Completeness*. (Freeman and Co. (eds.), New York, 1979).

[24] Gonzalez J., Ollero A., and Reina A. *Map Building for a Mobile Robot Equipped with a Laser Range Scanner*. IEEE Int. Conf. on Robotics and Automation (ICRA'94).

[25] Guivant J., Nieto J., Masson F., and Nebot E. *Navigation and Mapping in Large Unstructured Environments*. Int. Journal of Robotics Research **23**, No. 4-5 (2003) pp. 449-472.

[26] Harnand S., *The Symbol Grounding Problem*. Physica D: Nonlinear Phenomena **42** (1990) pp. 335-346.

[27] Hoffmann J. and Nebel B., *The FF Planning System: Fast Plan Generation through Heuristic Search*. Journal of Artificial Intelligence Research, **14**, (2001) pp. 253-302.

[28] Isard M., and Blake A. *Contour tracking by stochastic propagation of conditional density*. Proc. 4th European Conf. On Computer Vision, Cambridge, UK, April 1996.

[29] Kuipers B.J., *The Spatial Semantic Hierarchy*. Artificial Intelligence **119** (2000) pp. 191-233.

[30] Kuipers B.J. and Y.T. Byun, *A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations*. Robot Autonomous Systems **8** (1991) pp. 47-63.

[31] Kuipers B.J. and Byun Y.T. *A Qualitative Approach to Robot Exploration and Map-Learning*. In Proc of the Workshop on Spatial Reasoning and Multi-Sensor Fusion, 1987.

[32] Lisien B., Morales D., Silver D., Kantor G., Rekleitis I., and Choset H. *The Hierarchical Atlas*. IEEE Trans. on Robotics, **21**, No.3 (2005) pp. 473-481.

[33] Loutfi A., Coradeschi S., and Saffiotti A. *Maintaining Coherent Perceptual Information Using Anchoring*. Proc. of the 19th IJCAI Conf. Edinburgh, UK, August 2005.

[34] Maio D., Maltoni D., Rizzi S., *Topological Clustering of Maps using a Genetic Algorithm*. Pattern Recognition Letters **16** (1995) pp. 89-96.

[35] Nordin P., Bazhaf W., and Brameier M., *Evolution of a World Model for a Miniature Robot using Genetic Programming*. Robotics and Autonomous Systems **25**, No. 1-2 (1998) pp.105-116.

[36] Poncela A., Perez E.J., Bandera A., Urdiales C., and Sandoval F.,. *Efficient integration of metric and topological maps for directed exploration of unknown environments*. Robotics and Autonomous Systems, **41**, No.1, (2002) pp. 21-39

[37] Ram A., Arkin R., Boone G., and Pearce M., *Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation*. Adaptive Behavior **2**, No. 3 (1994) pp. 277-304.

[38] Rizzi S., *A Genetic Approach to Hierarchical Clustering of Euclidean Graphs*. Proc. Int. Conf. on Pattern Recognition (ICPR'98). Brisbane, Australia, (1998) pp. 1543-1545.

[39] Saffiotti A., Konolige K., and Ruspini E. H., *A multivalued-logic approach to integrating planning and control*. Artificial Intelligence, **76**, (1995) pp. 481-526.

[40] Shi J., and Tomasi C. *Good features to track*. Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn., (1994) pp. 593-600.

[41] Takeuchi I. and Furuhashi T. *Self-Organization of Grounded Symbols for Fusions of Symbolic Processing and Parallel Distributed Processing*. Proc. of 1998 IEEE Int. Conf. on Fuzzy Systems, (1998) pp.715-720

[42] Tani J., *Model-Based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective*. IEEE Trans.IEEE Trans. Syst. Man and Cybern **26**, No. 3 (1996) p.p 421-436.

[43] Taniguchi T. and Sawaragi T. *Self-organization of inner symbols for chase: symbol organization and embodiment*. 2004 IEEE Int. Conf on Systems, Man and Cybernetics, 2004 pp. 2073-2078.

[44] Thrun S., Bücken, *Integrating grid-based and topological maps for mobile robot navigation*. In Proc. Of the AAAI 13th National Conf on Artificial Intelligence, Portland, Oregon, (1996) pp. 944-951.

[45] Thrun S. *Robotic Mapping: A Survey"*. Exploring Artificial Intelligence in the New Millennium. (Morgan Kaufmann, 2002).

[46] Thrun S., Fox D., and Burgard W. *Probabilistic Mapping of an Environment by a Mobile Robot*. Proc. of the 1998 IEEE Int. Conf. on Robotics & Automation.

[47] Tomatis N., Nourbakhsh I., and Siegwart R., *Hybrid Simultaneous Localization and Map Building: a Natural Integration of Topological and Metric*.Robotics and Autonomous Systems **44**, (2003) pp. 3-14.

[48] Trudeau R.J., *Introduction to Graph Theory*. (Dover Publications, New York, 1993)

[49] Urdiales C., Perez E.J., Sandoval F., and Vazquez-Salceda J. *A hybrid architecture for autonomous navigation in dynamic environments*. In Proc. of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), Halifax, Canada, 13-16 October 2003, pp 225-232.

[50] Wah B., Lowrie M., and Li G. *Computers for Symbolic Processing*. Proc. of the IEEE, **77**, No. 4, 1989 pp. 509-540.

[51] Wallgrün J.O. *Hierarchical Voronoi-based Route Graph Representations for Planning, Spatial Reasoning, and Communication*. In Proc. of the 4th International Cognitive Robotics Workshop, Valencia, Espaa, August 2004.

[52] Whitley, D., *A Genetic Algorithm Tutorial*. Statistics and Computing, **4**, (1994) pp.65-85.