Layered Knowledge Architecture For Navigation-Oriented Environment Representation

Dario Maio, Stefano Rizzi
DEIS - CIOC-CNR - Facoltà di Ingegneria, Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna, Italy
e-mail: dariolstefano@deis64.cineca.it

Abstract - This paper describes a layered architecture representing knowledge to be used for navigation in environments where distinctive places are present. Each knowledge layer defines an abstraction of the environment aimed at efficiently supporting the execution of specific navigation tasks. Layers are structured in a taxonomy, each layer being derived from another by applying one of the three abstraction primitives: classification, generalization, aggregation; in particular, aggregation enables partitioning of the environment map into clusters. The validity of the architecture proposed is discussed with reference to different application fields, including autonomous robots and vehicle navigation systems. Some issues related to hierarchical path planning on layered knowledge are then addressed; a general framework for a more flexible formulation of path-planning problems and a technique for decomposing problems on the different layers are proposed. An efficient algorithm to solve a specific navigation problem, named "travelling agent", is presented; the algorithm uses a heuristic criterion to expand, at each step, a path computed within a layer into a path at the next layer down.

Index-terms - abstraction, clustering, hierarchical planning, knowledge representation, path planning

I. INTRODUCTION

Planning a route to navigate an environment is a primary requirement for several applications in different domains, ranging from autonomous robots to vehicle navigation systems. The representation adopted for environmental knowledge has a critical role in enhancing the formulation flexibility of planning problems and in simplifying their resolution. In this work we describe an architecture for knowledge representation, specifically oriented to planning navigational tasks in structured environments. We say an environment is structured if a number of categories of objects and places that can be encountered in it are described a priori. We call *landmarks* the objects and places belonging to a subset of categories which are regarded as distinctive or significant.

According to many cognitive scientists, a cognitive map is organized into successive layers at different abstraction levels [22] [34]. The knowledge architecture we propose is organized in layers determined by the structure of the environment and by the tasks which must be carried out. Each layer can be thought of as a view of the environment at a specific abstraction level; it includes only those details of the environment which are significant for a specific family of tasks or sub-tasks, and represents them in the most suitable formalism [28]. Layers may be abstracted by classification, aggregation or generalization.

A layered representation of the environment is semantically richer than a "flat" representation, thus enabling a more flexible formulation of path-planning problems. Consider, for instance, a

consultant system for planning visits in a city or in a museum. In these applications, the language for user-machine interaction should allow for constraints to be stated as precisely as possible (for the museum consultant: "two hours available time, definitely see all Van Gogh and Monet, a brief stop at Leonardo's Gioconda, overlook English painters"; for the city consultant: "one-hour shopping in downtown (consider the shop hours), walk in the park before sunset, be back to airport by 19.00"). These natural-language requirements are formulated at different levels of abstraction; they can be easily mapped on different knowledge layers to be translated into formal constraints for path planning.

In most route-planning applications, obtaining the solution in real-time has priority over obtaining the optimal one. Our knowledge architecture allows for complex path-planning tasks to be hierarchically decomposed into a number of sequential or parallel sub-tasks, each supported by a specific layer.

Section II surveys the main approaches to navigation-oriented environment representation which can be found in the literature. Section III, after introducing the concept of knowledge layer, defines the operators for layer abstraction. Section IV outlines the functions and structures of the different knowledge layers, used as building blocks in section V to construct specific knowledge architectures for different application domains. Section VI discusses how hierarchical path planning is supported by our architecture by formulating and solving a specific path-planning problem.

II. Related work

In this section we survey the literature related to the two main issues discussed in this paper, namely, representation of environmental knowledge and path planning.

A. Navigation-oriented representation of knowledge

Organizing knowledge of the environment to be used for navigation requires two crucial points to be investigated:

- *what* is the significant information to be extracted from the environment;
- *how* this information is represented, i.e., by means of what formalisms.

The answer to both questions depends essentially on the nature of the tasks which must be performed. A detailed knowledge description might add an unnecessary burden to the execution of high level tasks, whereas a poor description might not be sufficient to establish a connection between the objects in the environment and their internal representation. Furthermore, since each representation formalism is characterized by different individual properties, the choice of a specific formalism could favour certain tasks but penalize some others. For instance, an analogic representation is easy to operate on real sensors, while symbolic knowledge representation allows for a more convenient implementation of reasoning and planning algorithms.

Many approaches to motion planning have been devised in the literature. Symbolic approaches include connectivity graphs [23], occupancy grids [18] and efficient paths [6]; a reinforcement-based analogic approach is proposed in [35]. A mixed approach is introduced in [37], where the navigation problem is generally formulated and a complete algorithm for operating a real robot in a real world is proposed; the algorithm works on an integrated symbolic-analogic representation of the environment. Another mixed approach based on decomposition of motion planning into two levels is described in [19]: at global level a graph whose nodes represent large cells of the agent's configuration space is built; at local level, sub-goals for the planner are generated through a minimum-cost path-finding algorithm.

All the approaches mentioned above differ from ours since they do not provide any abstract representation of the environmental knowledge, so the planning of complex navigational tasks cannot be supported.

Some interesting multi-layered models have been proposed in the literature. In [13], the authors describe a hierarchical model for representing a topographic surface at successively finer levels of detail. In [26], the layered model is called *spatial semantic hierarchy* and consists of a control level, a topological level and a geometric level. In these approaches the environment is represented, at most, at landmark level; in our architecture, higher abstraction levels are defined to be used for formulating and solving path-planning problems.

In [16] a variety of abstractions are used for the purpose of problem reduction at different phases during route planning; the environment representation is solidly grounded to vision data, but the problem of classification of spatial objects is not addressed. In [36], human navigation in the highway network is modelled through a cognitive map including three abstraction levels: the planning level, the instructional level and the driver level. This approach bears some resemblance with ours, since the three levels coarsely correspond to, respectively, the 1-clustered layer, the symbolic layer and the sub-symbolic layer of our architecture.

Our approach differs from all those mentioned above mainly since we propose a general knowledge representation framework which is equally valid in different application fields where navigational tasks must be carried out.

B. Hierarchical planning

The hierarchical approach to the solution of planning problems consists in first constructing an abstract plan which achieves the more general goals, and then refining it into detailed subplans which achieve more concrete goals. The advantage is that the plan is first developed at a level at which the details are not computationally overwhelming. There have been a number of attempts to devise hierarchical planning systems, beginning with Sacerdoti's ABSTRIPS in which a hierarchy of abstract spaces, each dealing with fewer details than the space below it and with more details than that above it, is defined. By considering details only when a successful plan in a higher level space gives strong evidence of their importance, a heuristic search process

will investigate a greatly reduced portion of the search space [33]. In [25], the author shows that the use of abstraction hierarchies in planning can reduce exponential problems to linear complexity if the number of abstraction levels and their sizes are properly determined.

The application of abstraction hierarchies to path planning has been specifically treated in the literature. In [21] the authors propose a cognitive model of path-planning and show how humans often move between different abstraction levels when planning. In [20] a representation for hierarchical plans is devised, and a system for planning the journey of a traveller through a network of rail, sea and air public transport services with time and cost constraints is described. An architecture that supports hierarchical planning involving deadlines, travel time and resource considerations is described in [12]; a partially constructed plan is refined into a plan at a lower level of abstraction by decomposing a complex task into a set of simpler subtasks by applying routines called *task expanders*. In [7], a route-finding algorithm working on a hierarchical representation of the environment based on clustering is outlined.

In all the approaches surveyed, the main issue addressed is the solution of the planning task. Our work, instead, aims primarily to show how the existence of a hierarchy of abstraction levels enables a broader formulation of path-planning problems.

III. KNOWLEDGE LAYERS

A *knowledge layer* is a meaningful abstraction of the environment supporting execution of one or more tasks. Each layer can be thought of as a "window" on the environment which contains only the details significant for these tasks and represents them by the most suitable formalism. Thus, a layer is characterized by knowledge, representation and skill.

The acquisition of the environment description takes place in most cases analogically, for instance from a set of sensor measures or a map. This view of the environment can hardly be exploited directly to carry out complex tasks, hence, knowledge must be reorganized and interpreted by abstracting one or more layers, each suitable for a specific task, from the low-level analogical description. Each of these layers may in turn generate other layers for other tasks, through a procedure of progressive abstraction which creates a taxonomy of layers.

The three operators for abstracting a new layer from an existing one correspond to the three abstraction primitives: the *classification*, *generalization* and *aggregation* operators return layers whose entities are, respectively, classifications, generalizations and aggregations of the entities described in the underlying layer.

Consider the structured environment in Figure 1; the prototypes of several categories of objects that can be found are described a priori, and landmarks are the objects and places which fit one of those categories. Layer A contains the symbolic representations of landmarks, each associated with its position and description. Layer B is obtained by classification of A, and describes landmarks through their categories ("St. Mary" is a church). Layer C is a generalization of B; its entities are categories of categories described in B (the category

"monument" includes the categories "church" and "statue"). Layer D is abstracted from A by aggregation, and describes clusters of landmarks of A ("Broadway" includes "St. Mary" and "Chez Louis"). Layer E is a classification of D and describes clusters through their categories ("Soho" is a quarter).

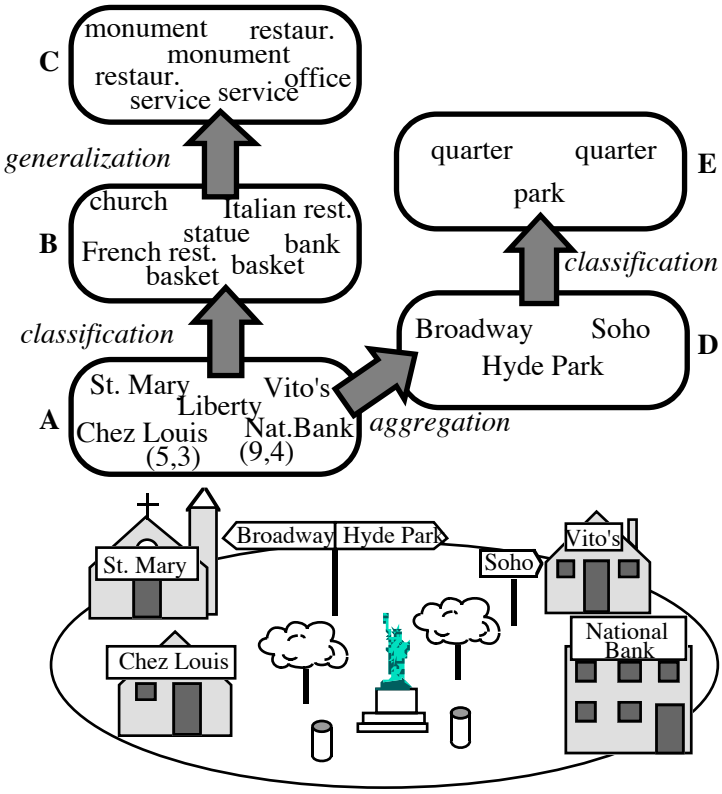

Figure 1. Layers obtained by applying the three abstraction primitives to the set of sensory measures describing an environment. Landmarks which cannot be identified by their description are tagged in layer A with their coordinates.

As confirmed by this example, all three abstraction operators require some meta-knowledge of the environment:

- *Meta-knowledge for classification* describes the categories of objects which may be encountered in the environment; it enables automated recognition of landmarks and allows for sets of similar objects to be addressed ("visit all the churches in Broadway").
- *Meta-knowledge for generalization* describes hierarchical relationships between categories, and adds new semantics to the representation ("go to the nearest restaurant (regardless of whether it is Italian or French)").
- *Meta-knowledge for aggregation* describes significant clusters of objects; it adds new semantic ("go to Hyde Park") and helps in referencing objects ("switch on the computer in room #6").

IV. A LAYERED ARCHITECTURE TO REPRESENT ENVIRONMENTAL KNOWLEDGE

In [9] we have proposed for knowledge representation in autonomous agents a hybrid architecture based on the coupling of a symbolic module and a dynamic module, the first managing explicit knowledge of landmarks and the latter encoding knowledge necessary for movement among landmarks. In this section we generalize this approach by proposing a multi-layered navigation-oriented knowledge architecture to represent structured environments. Four different types of layers are defined, namely sub-symbolic, symbolic, clustered and meta-layers; they are described, respectively, in subsections A, B, C, and D.

A. Sub-symbolic layer

The sub-symbolic layer supports inter-landmark motion. The formalism adopted to represent this layer and the degree to which reactive behaviour is accomplished depend heavily on the application domain. For some applications, reactivity is actually achieved outside the system (for instance, in vehicle navigation systems, by the driver; in systems for planning visits in cities or museums, by the visitor); the role of the sub-symbolic layer thus comes down to the effective presentation of the necessary knowledge (instructions for turning or for keeping the right lane; printed maps showing the detailed itinerary through city streets or museum rooms). A domain where reactive behaviour is totally demanded to the sub-symbolic layer, which is entrusted with direct control of the actuators, is that of autonomous agents; some related issues will be discussed in section V.A.

B. Symbolic layer

The symbolic layer is the foundation for path planning. The environment is described as a map of landmarks and feasible inter-landmark paths (*routes*). A route is an abstraction corresponding to the straight-line connection between two landmarks, and is described by a cost expressing, for instance, the length of the corresponding physical path or the average time spent to cover it. Routes are supposed to be directed, in order to model environments where the cost of a path may depend upon its direction (for instance, a slope in a country environment or a one-way street in an urban environment). We denote with $[l_i \rightarrow l_j]$ the route going from l_i to l_j ($l_i \neq l_j$).

Definition 4.1. Let $L^{(0)}$ and $R^{(0)}$ be the sets of landmarks and routes, respectively, experienced at a given time during exploration[1]. We define as *symbolic layer* the weakly connected[2] directed graph $\mathcal{L}^{(0)} = (L^{(0)},R^{(0)})$, where landmarks are described by a name and a position and routes are associated with a cost.

---

[1] Apex 0 is used to emphasize that, within the multi-level clustering defined below, the abstraction level of landmarks and routes is 0.

[2] A directed graph is called strongly connected if each vertex can be reached from every other vertex through a directed path; weakly connected if each vertex can be reached from every other vertex through a non-directed path (i.e., where the orientations of arcs are ignored).

C. Clustered layers

A clustered layer is abstracted by aggregation from the symbolic layer or from another clustered layer. A hierarchy of clustered layers may thus be abstracted starting from the symbolic layer, to be used for high-level path planning. At the first abstraction level, the environment is described as a map of clusters of landmarks and inter-cluster passageways (*bridges*). At subsequent levels, each cluster includes clusters of the level below. In structured environments, clustering can be based on the available meta-knowledge for aggregation: for instance, in a hospital, clusters corresponding to the progressive abstractions of rooms, wards and departments can be identified. In environments where meta-knowledge for aggregation is not available, clustering must be based on topological and metric criteria. In [27] we describe a technique called *clustering by discovery* for identification of clusters in an unknown environment; clustering is performed during exploration, by minimizing a measure of scattering which takes into account structural and functional specifications.

Definition 4.2. Given a directed graph $\mathcal{L}$ = (L,R), a (*vertex-based*) *clustering* on $\mathcal{L}$ is defined as a partitioning $\xi=\{L_1,...L_m\}$ of L. We call *clusters* the m sub-graphs c_1 = = (L_1,R_1),...c_m = (L_m,R_m), where

$$R_i = \{\ [l \rightarrow l'] \in R:\ l \in L_i \wedge l' \in L_i\ \}\ ,\ i=1,...m$$

For each ordered pair of clusters (c_i,c_j), $i \neq j$, we call *bridge* the (possibly empty) set

$$[c_i \rightarrow c_j] = \{\ [l \rightarrow l'] \in R:\ l \in L_i \wedge l' \in L_j\}\ ,\ i,j=1,...m$$

We denominate with *image of* $\mathcal{L}$ *through* ξ the directed graph $\mathcal{L}^*$ whose vertices and arcs are, respectively, the clusters and the non-empty bridges induced by ξ:

$$\mathcal{L}^* = (\{c_1,...c_m\},\ \{\ [c_i \rightarrow c_j]:\ [c_i \rightarrow c_j] \neq \varnothing,\ i,j=1,...m,\ i \neq j\})$$

Theorem. Let $\mathcal{L}$ = (L, R) be a weakly connected directed graph. The graph $\mathcal{L}^*$, image of $\mathcal{L}$ through a clustering ξ, is weakly connected for each possible ξ.

Proof: We will prove the theorem by contradiction. If $\mathcal{L}^*$ is not weakly connected, then its vertices can be partitioned in (at least) two disjointed sets A and B such that

$$\forall c_i \in A\ (\forall c_j \in B\ (\ [c_i \rightarrow c_j] = \varnothing \wedge [c_j \rightarrow c_i] = \varnothing\ ))$$

If $A'=\{l \in L:\ l \in c_i \wedge c_i \in A\}$ and $B'=\{l \in L:\ l \in c_j \wedge c_j \in B\}$ are the sets of vertices of $\mathcal{L}$ included in the clusters of A and B, respectively, it follows that

$$\nexists\ [l_i \rightarrow l_j] \in R:\ ((l_i \in A' \wedge l_j \in B') \vee (l_i \in B' \wedge l_j \in A'))$$

Hence, $\mathcal{L}$ is not weakly connected.♦

Let a graph $\mathcal{L}$ be given. Several clusterings may be defined on $\mathcal{L}$, generating different image graphs which, in turn, can be clustered. A hierarchical clustering can thus be progressively

defined, producing a hierarchy of graphs whose root is $\mathcal{L}$. For simplicity, we will consider the case in which only one clustering is applied to each graph, so that an n-level hierarchical clustering can be described as a sequence of n progressive clusterings $\xi^{(0)},...\xi^{(n-1)}$.

Definition 4.3. Let the symbolic layer $\mathcal{L}^{(0)} = (L^{(0)},R^{(0)})$ and an n-level hierarchical clustering $\xi^{(0)},...\xi^{(n-1)}$ be given. We name *k-clustered layer* (k=1,...,n) the weakly connected directed graph $\mathcal{L}^{(k)} = (L^{(k)},R^{(k)})$, image of $\mathcal{L}^{(k-1)}$ through $\xi^{(k-1)}$. We name its vertices *k-clusters*, and its arcs *k-bridges*. A k-cluster is denoted with $C_i^{(k)}$ and is described by a name and a position; a k-bridge is denoted with $[C_i^{(k)} \rightarrow C_j^{(k)}]$ and is associated with a cost.

For analogy, in the remainder of the paper the symbolic layer will equivalently be called a 0-clustered layer, and landmarks will be considered as 0-clusters (notations l_i and $C_i^{(0)}$ will be used indifferently).

Definition 4.4. Let α_i be the position of landmark l_i. The position of k-cluster $C_i^{(k)} = (L_i^{(k-1)},R_i^{(k-1)})$ is defined as:

$$pos(C_i^{(k)}) = \begin{cases} \alpha_i , & \text{if } k=0 \\ avg_{\{C_j^{(k-1)} \in L_i^{(k-1)}\}} \left(pos(C_j^{(k-1)}) \right) , & \text{if } 1 \le k \le n \end{cases}$$

where $avg_S(f)$ denotes the average of function f extended to the set S.

Definition 4.5. Let γ_{ij} be the cost of route $[l_i \rightarrow l_j]$, and $ecost(C^{(k)},C^{(w)})$, k,w=0,...n be a function which estimates the cost for reaching $C^{(w)}$ from $C^{(k)}$ on the basis of the straight-line distance between $pos(C^{(k)})$ and $pos(C^{(w)})$ (see Figure 2). We define the *cost* of the k-bridge $[C_i^{(k)} \rightarrow C_j^{(k)}]$ as:

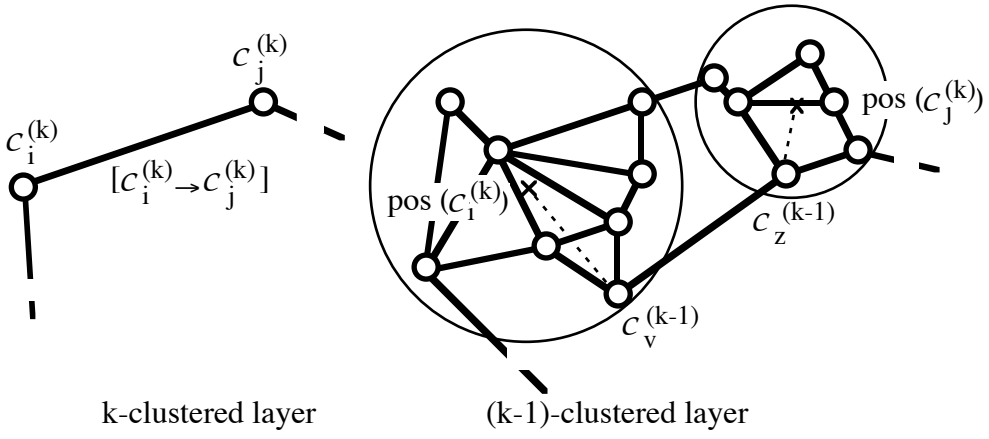

Figure 2. Definition of the cost of a k-bridge. $[C_v^{(k-1)} \rightarrow C_z^{(k-1)}]$ is a generic element of $[C_i^{(k)} \rightarrow C_j^{(k)}]$. Dashed lines represent, respectively, $ecost(C_i^{(k)},C_v^{(k-1)})$ and $ecost(C_z^{(k-1)},C_j^{(k)})$, i.e., the estimates of the costs for reaching (k-1)-clusters from k-clusters.

$$cost([C_i^{(k)} \rightarrow C_j^{(k)}]) = \begin{cases} \gamma_{ij} , & \text{if } k=0 \\ avg_{\{[C_v^{(k-1)} \rightarrow C_z^{(k-1)}] \in [C_i^{(k)} \rightarrow C_j^{(k)}]\}} \left(ec([C_v^{(k-1)} \rightarrow C_z^{(k-1)}]) \right) , & \text{if } 1 \le k \le n \end{cases}$$

where

$$ec([c_v^{(k-1)} \rightarrow c_z^{(k-1)}]) = ecost(c_i^{(k)}, c_v^{(k-1)}) + cost([c_v^{(k-1)} \rightarrow c_z^{(k-1)}]) + ecost(c_z^{(k-1)}, c_j^{(k)})$$

As shown in Figure 3, a k-cluster is contemporarily a vertex of the k-clustered layer and a sub-graph of the (k-1)-clustered layer. Please note that, though $\mathcal{L}^{(k-1)}$ is weakly connected, one or more of the k-clusters induced on $\mathcal{L}^{(k-1)}$ by a given clustering $\xi^{(k-1)}$ may be non-connected graphs. On the other hand, the path-planning algorithms we will consider attempt to build paths within clusters; therefore, we will consider only the clusterings that produce clusters which are at least weakly connected.

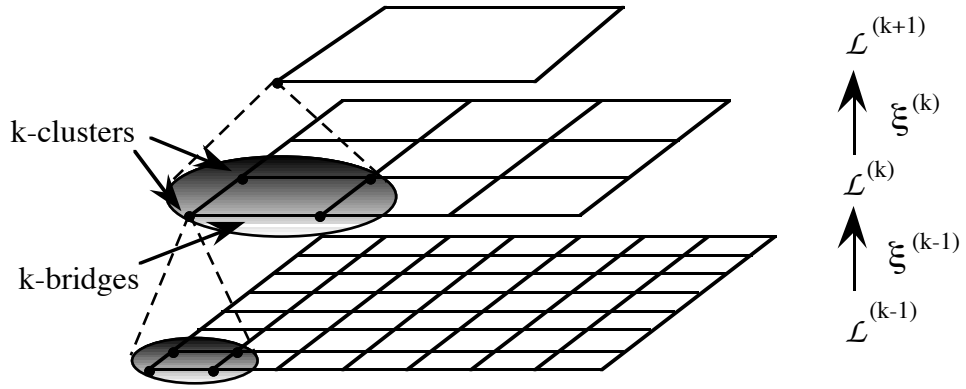

Figure 3. Structural relationship between subsequent levels of clustering.

Given two clusters $c_i^{(k)}$ and $c_j^{(w)}$ with k<w, we say $c_i^{(k)}$ is *included* in $c_j^{(w)}$ ($c_i^{(k)} \subset c_j^{(w)}$) when $c_i^{(k)}$ is a vertex of $c_j^{(w)}$ or is included in a vertex of $c_j^{(w)}$.

D. Meta-layers

Meta-layers allow for objects to be referenced based on their category. A meta-layer may be abstracted by classification from the symbolic layer or from any clustered layer, or by generalization from another meta-layer; it describes entities in the environment through the category they belong to. We denote with $cat(c_i^{(k)})$ the category to which cluster $c_i^{(k)}$ belongs.

Definition 4.6. Let the symbolic layer $\mathcal{L}^{(0)} = (L^{(0)}, R^{(0)})$ be given, and an n-level hierarchical clustering $\xi^{(0)},...\xi^{(n-1)}$ generating the clustered layers $\mathcal{L}^{(k)} = (L^{(k)}, R^{(k)})$, k=1,...n. We name *meta-layers* the sets $M^{(k)}=\{cat(c_i^{(k)}): c_i^{(k)} \in L^{(k)}\}$, k=0,...n.

For simplicity, the formalization of meta-layers abstracted by generalization from other meta-layers is not given.

V. APPLICATION FIELDS

In this section we review some applications, in different fields, sharing motion-planning capability as a necessary requirement. For each of them, we propose a layered architecture and discuss how it addresses the domain-specific issues.

A. Autonomous agents

Autonomous agents are versatile machines capable of interacting coherently with an environment and executing a variety of tasks in unpredictable conditions [11]. Several control architectures for mobile robots have been implemented [1] [5] [18] [38].

Most activities for an autonomous agent involve planning the cheapest path which allows for one or more (possibly inter-related) goals to be achieved while avoiding collisions with obstacles, other agents or people. A description of the topology and metric of the environment must be learned on-line by interpreting sensor data as the agent explores the environment; we assume that the agent's sensors return metric, visual and symbolic information. Metric information enables the agent to compute its current position; it can be obtained by using satellite measurements from the Global Positioning System or by the joint use of a compass and an odometer. Visual information is an "image" of the nearby surroundings obtained through a sonar or a camera. By symbolic information we mean that the agent can tag some landmarks with a name, for instance by "reading" a sign.

As a matter of fact, the problem of constructing an accurate perceptual system for landmarks is very complex. In our approach, recognizing an object as a landmark in a structured environment means ascribing it into one of the categories described by meta-knowledge for classification. An analysis of the pattern recognition techniques for classification of objects is beyond the aim of this paper; see, for instance, [31] where a neurocomputational approach to the extraction of object features from sensory data is presented, and [8] which demonstrates a model-driven approach to landmark extraction.

Within the architecture to be used by an autonomous agent to represent knowledge for navigation, the sub-symbolic layer is responsible for reactive behaviour. To this end, neural models seem to be more appealing than symbolic ones; in fact, distributed knowledge representation leads to interesting properties such as immunity to noise, capability of generalization, etc. which could not be easily obtained with local representation models. We have simulated the behaviour of an agent whose sub-symbolic layer is supported by a neural network. Input to the network consists of metric and sonar measures, which identify the state of the agent in the environment; the output is the action to be executed, i.e., the direction to follow. A reinforcement algorithm is used to learn an optimal mapping from the set of inputs to the set of possible actions, using the current goal (the position of the landmark to be reached) as a contextual input. Specific issues related to the application of reinforcement learning techniques to inter-landmark navigation are discussed in [10] and [35].

The symbolic section of the knowledge architecture depends on how the environment is structured. Consider for instance an autonomous agent in a hospital, where landmarks are associated with serviceable or conceptually relevant entities such as medical equipment, computers, receptions, etc. The agent should be able to execute a variety of tasks, such as visiting the in-patients of a department in order to bring them food and medicines, or delivering

material to wards. These tasks entail path planning, possibly taking into account time, energy or more complex constraints.

A sample layered architecture for the hospital agent is shown in Figure 4.a. Three clustered layers are defined on the symbolic layer: they represent, respectively, rooms as clusters of landmarks, departments as clusters of rooms, hospitals as clusters of departments. A meta-layer describing landmarks through their category (X-ray, TAC, computer, etc.) is abstracted from the symbolic layer; one describing rooms through their type (office, laboratory, ward, etc.) is abstracted from the room layer. Additional meta-layers may be abstracted by generalizing the categories represented (for instance: category of medical equipment, category of radiation-monitored rooms).

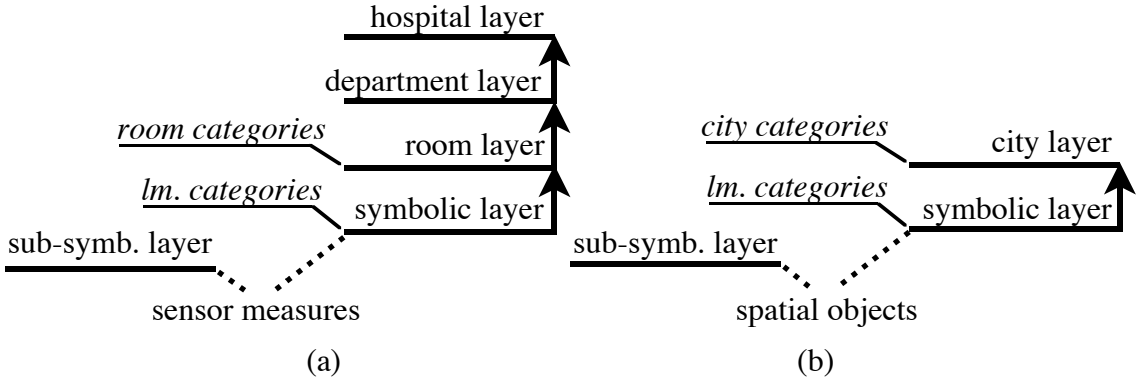

Figure 4. Knowledge architectures for different application domains. Vertical arrows represent aggregation, oblique lines classification/generalization; meta-layers are in italics. (a) hospital autonomous agent; (b) vehicle navigation system.

B. Vehicle navigation systems

Vehicle navigation systems assist drivers in planning trips and selecting routes, by guiding them through geographic space [17]. In commercial systems, cars are equipped with an on-board computer, a dash-mounted graphic screen displaying maps and conveying driving instructions, and sensors that return the car position (for instance, using satellite measurements from the Global Positioning System).

Spatial objects to be modelled include for instance roads, intersections, monuments, shops, and on a different scale cities, highway exits, clover-leaf junctions. Our layered architecture allows for this environmental knowledge to be modelled at multiple resolutions corresponding to different tasks (see Figure 4.b). Landmarks are defined as distinctive points along the roads, relevant buildings, serviceable places. At the lowest level, the sub-symbolic layer represents in detail the roads connecting landmarks; for instance, a crossing between several highways may be described as a complex 3-dimensional structure. The sub-symbolic layer is primarily used to give minute driving instructions. At the higher level, the symbolic layer represents landmarks and their connectivity; queries which can be formulated on this layer include searching for the shortest path among two or more landmarks, where "shortest" may be interpreted in terms of

distance, time, fuel consumption, toll, etc. Abstracting one or more meta-layers from the symbolic layer allows for queries such as "which is the next exit", "which is the closest gas station", "how long to an Italian restaurant" to be answered. At least one clustered layer should be defined, representing cities and the connections between them; it could be used, for instance, to compute the most convenient route between two cities. A meta-layer on the city layer could classify cities in county seats and capital cities.

C. The personal planner

By "personal planner" we mean an on-line system supporting constrained path planning on the city map for scheduling the errands of the day. This application requires a smart user interface allowing insertion of constraints on the path to be planned; in the simplest case, constraints can be declared by writing statements in a formal language. Real-time communication with the outside world is necessary in order to acquire on-line information about the traffic conditions and the social events, the shop and office hours, pictures of places, etc. Output of the system should be a detailed city map where the optimal itinerary is laid out together with the expected times for reaching each place.

The issues in knowledge representation are similar to those discussed for vehicle navigation systems, except that the scale here is mainly sub-urban (see Figure 5.a); strong relevance should be given to the definition of meta-layers of the symbolic layer. An example of a natural language sentence expressing some errands for the day is: "take the dog to Hyde Park (cluster constraint); collect spectacles at the optician (landmark constraint); have lunch at a Chinese restaurant (category constraint); check out some apartments in residential quarters (constraint on cluster category)".

D. Personalized tour planners

The most relevant issues in employing computer technology for the enjoyment of the cultural heritage are: full understanding of the good from the user through a simple, incisive and comprehensive presentation; guided choice of some aspects to be further investigated; planning of the enjoyment consistently with the user's main interests; personalization of the fruition mode according to the user's attitude and to his/her specific needs.

These requirements are equally valid for planning visits to a museum, cultural itineraries within a city and day-trips to Disneyland. CICERO is a consulting station for assisted planning of personalized routes for visiting the Ducal Palace in Urbino, Italy, and is based on a layered knowledge architecture in which landmarks correspond to works of art, information desks, souvenir shops, etc. [30]. The sub-symbolic layer describes the museum topology by means of detailed floor plans. Two meta-layers are abstracted from the symbolic layer: the first classifies landmarks according to their type (painting, sculpture, public service, etc.), the second according to their author (works by Raffaello, works by Signorelli, etc.). A meta-layer which

groups the works of art by their historical period (Renaissance, Fourteenth century, etc.) is abstracted by generalization from the author layer. The complete hierarchy of layers adopted is shown in Figure 5.b.

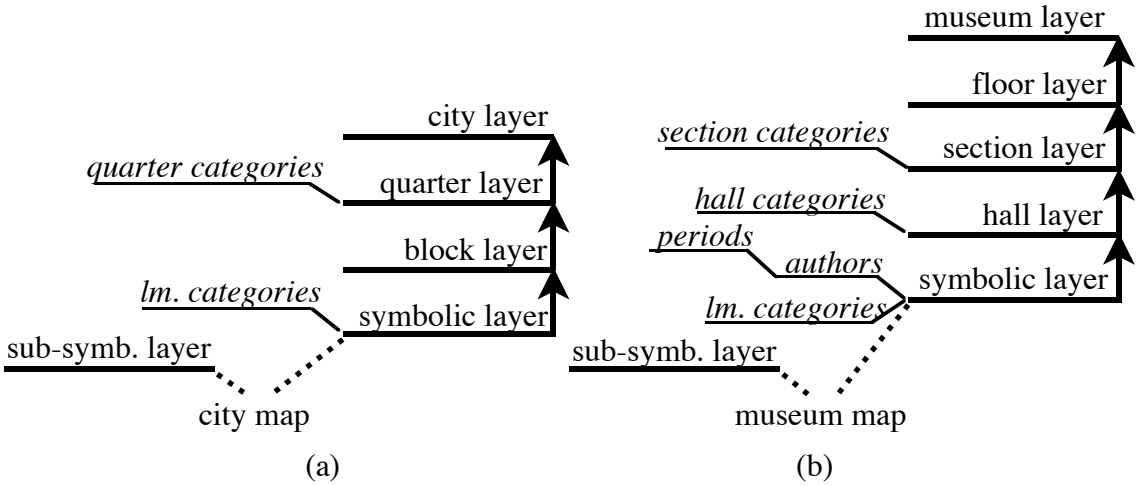

Figure 5. Knowledge architectures for different application domains. (a) personal planner; (b) museum planner.

Within CICERO, meta-layers allow for proposing the works in the museum to the users according to different criteria: by author, by historical period, by type. The user can thus quickly locate any work he/she is interested to see. As to route planning, CICERO allows for expressing constraints in terms of both landmarks ("include in the route the Flagellazione"), categories of landmarks ("include at least one work for each period") and clusters ("visit the Duke's Apartment).

VI. PATH PLANNING ON LAYERED KNOWLEDGE

Knowledge layering impacts on path planning in two ways: by establishing a general framework for a more flexible formulation of path-planning problems, and by reducing the computational effort enabling their solution through hierarchical decomposition techniques [29].

A. Hierarchical formulation of path-planning problems

When enumerating the objects and places to be visited in a path being planned, the new abstraction levels introduced by hierarchical clustering and by meta-layers can be addressed. To this end, the k-clustered layer ($0 \le k \le n-1$), the corresponding meta-layer and the (k+1)-clustered layer can be viewed as three orthogonal axes defining the space of k-clusters. A k-cluster may then be univocally referenced through a triplet of coordinates <*name*, *category*, *(k+1)-cluster*>. As shown in Figure 6, by assigning only one or two of the three coordinates it is possible to reference sets of k-clusters (for instance: "all the parks in the north-district", "all wastebaskets", "all banks named National Bank"). Within these sets, one k-cluster may be addressed by specifying a constraint ("the nearest wastebasket").

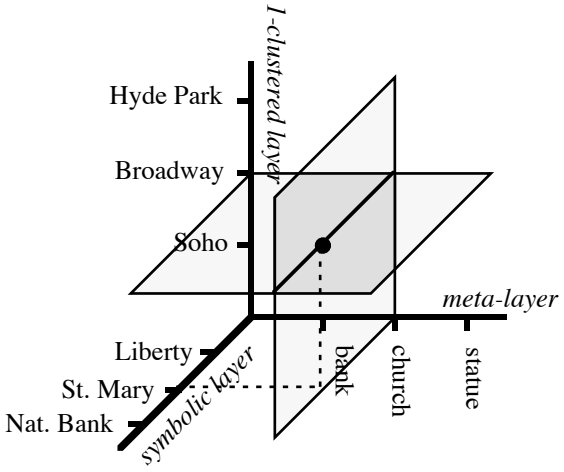

Figure 6. Addressing landmarks. The horizontal plane includes all landmarks in Broadway, the vertical one all churches in the environment. The intersection of the two planes defines the churches in Broadway.

In this subsection we show how a "flat" path-planning problem (i.e., a problem formulated on a graph) can be re-formulated on the hierarchy of layers on which our architecture is founded. Hierarchical formulation is based on the following definition of path:

Definition 6.1. A *k-path* is a sequence $\mathcal{P}^{(k)}=(C_{Z_1}{}^{(k)},...C_{Z_p}{}^{(k)})$ of p adjacent k-clusters (p≥1). The *cost* of $\mathcal{P}^{(k)}$ is defined as:

$$\text{cost}(\mathcal{P}^{(k)}) = \sum_{v=1}^{p-1} \text{cost}([C_{Z_v}{}^{(k)} \rightarrow C_{Z_{v+1}}{}^{(k)}])$$

The elemental instance of path-planning problem consists in finding the cheapest path between any two places. A hierarchical formulation of the *cheapest-path problem* can be found in [28].

A more general instance of this category of problems consists in planning a path which allows for a set of tasks to be carried out in observance of precedence constraints. Each task must be executed either on a specific resource or on any resource of a given category, considering that resources may be accessible only within given time windows. In our approach, each resource S_i to be visited is expressed by specifying one or more coordinates in the space of clusters; depending on which coordinates are given, S_i may correspond to a single cluster (*mono-resource*) or to a set of clusters (*multi-resource*; for instance $S_i=\{C_j^{(w)}: \text{cat}(C_j^{(w)})=c_i\}$).

Definition 6.2. Given the k-path $\mathcal{P}^{(k)}=(C_{Z_1}{}^{(k)},...,C_{Z_p}{}^{(k)})$ and the resource S_i, expressed as a set of w-clusters (0≤k≤n, 0≤w≤n-1), we say that S_i is *visited* in $\mathcal{P}^{(k)}$ in position v (1≤v≤p) if $\exists\, C_j^{(w)}\in S_i$: $((k<w)\wedge(C_{Z_v}{}^{(k)}\subset C_j^{(w)}))\vee((k>w)\wedge(C_{Z_v}{}^{(k)}\supset C_j^{(w)}))\vee((k=w)\wedge(C_{Z_v}{}^{(k)}\equiv C_j^{(w)}))$.

Given $\mathcal{P}^{(k)}=(C_{Z_1}{}^{(k)},...,C_{Z_p}{}^{(k)})$, we will denote with ta_v and td_v, respectively, the planned time for arrival in cluster $C_{Z_v}{}^{(k)}$ (1<v≤p) and the planned time for departure (1≤v<p). If $time([C_{Z_v}{}^{(k)} \rightarrow C_{Z_{v+1}}{}^{(k)}])$ is the time for covering the bridge $[C_{Z_v}{}^{(k)} \rightarrow C_{Z_{v+1}}{}^{(k)}]$, it is ta_{v+1} =

td_v+time($[C_{z_v}^{(k)} \rightarrow C_{z_{v+1}}^{(k)}]$). On these assumptions, we formulate the *travelling agent problem* (TAP) on layered knowledge as follows:

Let a start cluster $C_{start}^{(k)}$, an end cluster $C_{end}^{(k)}$ ($0 \le k \le n-1$), a start time t_{start} and a maximum time $t_{end} > t_{start}$ be given. Let $E \neq \emptyset$ be the set of resources to be visited: $E = Ec \cup Es$, where Ec includes the mono-resources and Es the multi-resources. Each $S_i \in E$ is associated with a *time window* $[w_i, w'_i]$, a *servicing time* $\Delta t_i \le w'_i - w_i$ and a (possibly empty) *precedence set* $\pi_i \subset E$. We assume that the precedence sets are complete, i.e. $\forall S_i \in E\ (\forall S_j \in \pi_i\ (\pi_j \subset \pi_i))$, and that $(\exists S_i \in E : \pi_i = \emptyset)$. Find the cheapest k-path $\mathcal{P}^{(k)} = (C_{z_1}^{(k)}, \ldots C_{z_p}^{(k)})$ such that:

1. $C_{z_1}^{(k)} \equiv C_{start}^{(k)}$
2. $C_{z_p}^{(k)} \equiv C_{end}^{(k)}$
3. $\forall S_i \in E$ (S_i is visited in $\mathcal{P}^{(k)}$)
4. $\forall S_i \in E$ visited in position v ($\forall S_j \in \pi_i$ visited in position z ($z < v$))
5. $td_1 = t_{start}$
6. $\begin{cases} \forall 2 \le v \le p-1: \exists\ S_i \in E \text{ visited in position v } (td_v = \max\{w_i, ta_v\} + \Delta t_i \le w'_i) \\ \forall 2 \le v \le p-1: \nexists\ S_i \in E \text{ visited in position v } (td_v = ta_v) \end{cases}$

   (if the agent reaches S_i before its time window, it waits for service until the time window starts; the costs for traversing k-clusters are already included in the costs of k-bridges)
7. $ta_p \le t_{end}$

B. Hierarchical path planning

In [28] we have introduced a divide-and-conquer algorithm which finds a sub-optimal solution to the hierarchically-formulated cheapest-path problem based on the algorithm which solves the corresponding flat problem. At the w-th step, a w-path is transformed into a (w-1)-path by first selecting the (w-1)-bridges to connect adjacent w-clusters and then determining the cheapest (w-1)-path within each w-cluster involved. Already for n=2 (two levels of clustering), the complexity of our algorithm is about 4% that of the exact algorithm. Obviously, the solution yielded may be sub-optimal, especially if some clusters have concave contours; from tests conducted on a sample of random maps with n ranging between 2 and 5, it appeared that the average shift from optimality is less than 10%.

Our decomposition technique is based on the following definition:

Definition 6.3. A *k-deep path* (shortly, *k-dpath*) corresponding to the k-path $\mathcal{P}^{(k)} = (C_{z_1}^{(k)}, \ldots C_{z_p}^{(k)})$ ($k > 0$) is a sequence

$$\mathcal{D}^{(k)} = (\ \rightarrow C'_{z_1}{}^{(k-1)}],\ [C''_{z_1}{}^{(k-1)} \rightarrow C'_{z_2}{}^{(k-1)}],\ \ldots\ [C''_{z_p}{}^{(k-1)} \rightarrow\ )$$

where $C'_{z_v}{}^{(k-1)} \subset C_{z_v}^{(k)}$ and $C''_{z_v}{}^{(k-1)} \subset C_{z_v}^{(k)}$ for v=1,...p, and

$$[C''_{z_v}{}^{(k-1)} \rightarrow C'_{z_{v+1}}{}^{(k-1)}] \in [C_{z_v}^{(k)} \rightarrow C_{z_{v+1}}^{(k)}] \text{ for } i = v, \ldots p-1$$

The notations $\rightarrow C'_{z_1}{}^{(k-1)}]$ and $[C''_{z_p}{}^{(k-1)}\rightarrow$ are used to specify the entry and exit (k-1)-clusters, respectively.

A k-dpath specifies the (k-1)-clusters for entering and leaving each k-cluster in the corresponding k-path. In general, several k-dpaths correspond to a given k-path; whichever is to be considered the "best" depends on the context. The method we apply to a k-path $\mathcal{P}^{(k)}=(C_{z_1}{}^{(k)},...,C_{z_p}{}^{(k)})$ in order to determine the best corresponding k-dpath is *expand*. Given the start cluster $C'_{z_1}{}^{(k-1)}$ and the end cluster $C''_{z_p}{}^{(k-1)}$, *expand* builds $\mathcal{D}^{(k)}$ incrementally by orderly considering each subsequence $(C_{z_{v-1}}{}^{(k)},C_{z_v}{}^{(k)})$, v=2,...,p. As shown in Figure 7, among the (k-1)-bridges forming the k-bridge $[C_{z_{v-1}}{}^{(k)}\rightarrow C_{z_v}{}^{(k)}]$, *expand* chooses the one $[C''_{z_{v-1}}{}^{(k-1)}\rightarrow C'_{z_v}{}^{(k-1)}]$ which minimizes

$$\text{ecost}(C'_{z_{v-1}}{}^{(k-1)},C''_{z_{v-1}}{}^{(k-1)})+\text{cost}([C''_{z_{v-1}}{}^{(k-1)}\rightarrow C'_{z_v}{}^{(k-1)}])+\text{ecost}(C'_{z_v}{}^{(k-1)},C_{z_{v+1}}{}^{(k)}) \quad \text{for } v<p,$$

and

$$\text{ecost}(C'_{z_{v-1}}{}^{(k-1)},C''_{z_{v-1}}{}^{(k-1)})+\text{cost}([C''_{z_{v-1}}{}^{(k-1)}\rightarrow C'_{z_v}{}^{(k-1)}])+\text{ecost}(C'_{z_v}{}^{(k-1)},C''_{z_p}{}^{(k-1)}) \quad \text{for } v=p.$$

If the start cluster is not specified (i.e, it is NULL), it is implicitly assumed to be $C''_{z_1}{}^{(k-1)}$; similarly, if the end cluster is not specified, it is implicitly assumed to be $C'_{z_p}{}^{(k-1)}$.

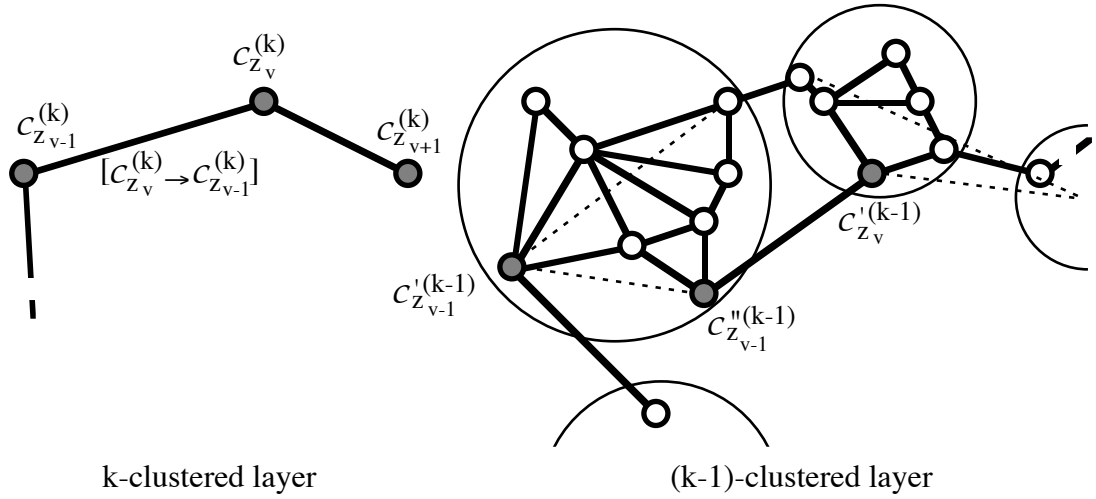

Figure 7. Expansion of a k-path into a k-dpath. The clusters interested by the v-th step of method *expand* are grey. Dashed lines show the estimates used for the costs of traversing $C_{z_{v-1}}{}^{(k)}$ and $C_{z_v}{}^{(k)}$.

A divide-and-conquer algorithm for the TAP can be written if an algorithm solving the corresponding flat problem is devised first. We say a TAP is *flat* when all the resources are expressed as sets of k-clusters, so that it can be solved entirely on the k-clustered layer.

A huge work has been done over the years on the Travelling Salesman Problem (TSP), so that some exact or heuristic algorithms can be found in the literature to solve specific instances of the flat TAP [4] [14] [15] [32]. For example, the algorithm proposed in [3] uses dynamic programming to solve classical TSPs with time windows and precedence constraints, that is, flat TAPs with $Es=\varnothing$ and $\Delta t_i=0$ for every $S_i\in Ec$. Using this algorithm when $Es\neq\varnothing$ would entail solving a different TSP for each cluster belonging to each resource in Es; the associated

computational cost would be very high. The problem of finding the optimal path which, given some sets of vertices, visits at least one vertex for each set, is known in the literature as *clustered TSP*, and can be solved through branch-and-bound techniques provided the branching of states and sets is not too high [2].

The heuristic algorithm we propose for the flat TAP consists of four sequential phases:

1. choice of the initial k-path.
2. progressive insertion in the k-path of the k-clusters in Ec.
3. progressive insertion in the k-path of a k-cluster for each set in Es.
4. completion of the k-path by adding, for each pair of consequent resources visited, the cheapest k-path which connects them.

During the first three phases, the algorithm works on a reduced completely connected graph whose nodes are k-clusters to be visited and whose arcs represent the cheapest k-path between them. In phase 1, the k-path chosen is the one going from the starting k-cluster $c_{start}^{(k)}$ to the "farthest" k-cluster in Ec. In phase 2, the cheapest insertion criterion is adopted. Cheapest insertion is a heuristic used for determining a sub-optimal solution to the TSP on graphs [24], and works as follows. At each step, for each k-cluster not yet in the k-path, the best position for insertion is determined as the position yielding the minimum extra-mileage. The k-cluster having minimum extra-mileage is then inserted in the k-path in its best position.

As to phase 3, each set may include several k-clusters. Strictly applying the cheapest insertion technique would require the extra-mileage to be calculated for each set, for each k-cluster in that set, for each insertion position in the k-path. In order to limit the complexity of the algorithm, for each insertion position we restrict the search space by means of a heuristic criterion based on clustering.

The pseudo-code for the heuristic flat TAP algorithm is listed in the Appendix. The algorithm is implemented in method *flat_TAP* whose signature is defined as follows:

$\mathcal{L}^{(k)}$->flat_TAP($c_{start}^{(k)}$,$c_{end}^{(k)}$,t_{start},t_{end},E,W,T,P,t_e,En) <u>returns</u> $\mathcal{P}^{(k)}=(c_{z_1}^{(k)},...c_{z_p}^{(k)})$:
$\forall v=1,...p$ ($c_{z_v}^{(k)} \in L^{(k)}$) where $\mathcal{L}^{(k)}=(L^{(k)},R^{(k)})$

$c_i^{(k+1)}$->*flat_TAP*($c_{start}^{(k)}$,$c_{end}^{(k)}$,t_{start},t_{end},E,W,T,P,t_e,En) <u>returns</u> $\mathcal{P}^{(k)}=(c_{z_1}^{(k)},...c_{z_p}^{(k)})$:
$\forall v=1,...p$ ($c_{z_v}^{(k)} \in L_i^{(k)}$) where $c_i^{(k+1)}=(L_i^{(k)},R_i^{(k)})$

where W, T and P are respectively the sets of time windows, servicing times and precedences corresponding to the resources in E. Besides the path computed, $\mathcal{P}^{(k)}$, the method returns the arrival time in $c_{end}^{(k)}$ within the path computed, t_e ($t_e \le t_{end}$), and the set En$\subset$E of resources which could not be inserted in $\mathcal{P}^{(k)}$.

Let n be the maximum level of clustering, and λ the number of landmarks in the symbolic layer. On the simplifying hypothesis that all the k-clusters (k=1,...n) contain the same number μ

of (k-1)-clusters, if the n-clustered layer contains exactly one n-cluster[3] it is $\mu=\lambda^{1/n}$. The number of k-clusters in the k-clustered layer may then be assumed to be $\lambda^{1-k/n}$ (k=0,..n). On the assumption that the number of resources which must be visited is independent of λ, the time complexity of the flat TAP algorithm applied to the k-clustered layer is $O(\lambda^{2-2k/n})$, where n is the maximum abstraction level.

In order to estimate how good the solutions yielded by our flat TAP algorithm are, we compared them with those produced by the exact algorithm proposed in [3]. Since that algorithm does not permit specifying that at least one vertex in a given set should be visited, we conducted the tests on a sample of flat TAPs where E≡Ec. Our algorithm succeeded in finding a solution in 90% of cases. The shift from optimality of the solutions obtained is contained within 20% in 80% of the cases; the average shift is about 12%. As a matter of fact, the solutions yielded by our algorithm can be further improved by adopting local optimization techniques such as 3-optimality; a discussion of these techniques is beyond the scope of this paper.

Applying the exact algorithm in [3] to the TAP in its general formulation would lead to prohibitive computational costs. In fact, for each resource expressed as a w-cluster $C^{(w)}$, it would be necessary to solve a different problem for each k-cluster included in $C^{(w)}$. Here we show how the same divide-and-conquer technique used for the cheapest-path problem can be applied to the TAP: on the generic iteration step, a w-path $\mathcal{P}^{(w)}$ is expanded into a (w-1)-path by solving a flat TAP at level w-1 within each w-cluster belonging to $\mathcal{P}^{(w)}$.

Let $\mathcal{P}$ be a path or a deep path; we denote with *length*($\mathcal{P}$) a function which returns the number of elements in $\mathcal{P}$, and with $\mathcal{P}$[v] its v-th element. Given a w-dpath $\mathcal{D}^{(w)} = (\rightarrow C'_{z1}{}^{(w-1)}], [C''_{z1}{}^{(w-1)}\rightarrow \ldots \ldots \rightarrow C'_{zp}{}^{(w-1)}], [C''_{zp}{}^{(w-1)}\rightarrow)$, we denote with $\mathcal{D}^{(w)}$[v]->*in* and $\mathcal{D}^{(w)}$[v]->*out* (v=1,...p), respectively, the clusters $C'_{zv}{}^{(w-1)}$ (entry in $C_{zv}{}^{(w)}$) and $C''_{zv}{}^{(w-1)}$ (exit from $C_{zv}{}^{(w)}$). With $S_i^{(h)}$ we denote a resource expressed as a set of h-clusters. Assuming that the n-clustered layer contains exactly one cluster, $C^{(n)}$, the divide-and-conquer algorithm for the TAP can be sketched as follows:

TAP algorithm

```
{ P(n) = (C(n));
  for w = n downto k+1 do
  { Cs(w-1) = Cstart(k)->ancestor(w-1);      // (w-1)-cluster which includes Cstart(k)
    Cd(w-1) = Cdest(k)->ancestor(w-1);       // (w-1)-cluster which includes Cdest(k)
    D(w) = P(w)->expand(Cs(w-1),Cd(w-1));
    P(w-1) = ∅;
    E' = { Si(hi)∈E : hi<w };                // resources to be visited in P(w-1) (levels 0 to w-1)
    for Si(hi) in E' do                      // for each resource Si(hi) in E, determine a corresponding
                                             // resource Ai(w-1) at level w-1
      Ai(w-1) = { Cj(hi)->ancestor(w-1): Cj(hi)∈Si };
    for v = 1 to length(D(w)) do
```

[3] This assumption can always be satisfied by adding a "dummy" layer on top of the hierarchy.

```
    { if (v=1)
        ts=tstart;
      else
        ts=te+time( [(D(w)[v-1]->out )→(D(w)[v]->in)] );
      Ev = { Ai(w-1) : (Ai(w-1)∩(P(w)[v]->descendants(w-1)) ≠∅ ) };
      // Ev is the set of the resources which have not yet been visited in P(w-1) and are totally
      // or partially included in P(w)[v]; P(w)[v]->descendants(w-1) returns the (w-1)-clusters
      // contained in P(w)[v]
      Pv = { {Aj(w-1) : Sj(hj)∈πi ∧ Aj(w-1)∈Ev } : πi∈P ∧ Ai(w-1)∈Ev };
      // Pv is the set of the precedence sets for the resources in Ev
      append P(w)[v]->flat_TAP(D(w)[v]->in,D(w)[v]->out,ts,tend,Ev,W,T,Pv,te,En) to P(w-1);
      E' = E'- (Ev - En);   // resources not yet visited in P(w-1)
    }
  }
}
```

Please note that, in the implementation of the algorithm, the resources to be visited are stored in sequences rather than in sets, in order to deal properly with the cases in which time windows and/or precedence constraints force the planned path to include the same cluster two or more times in different positions. The set notation is used here in order to simplify formulation.

The TAP can be considered a peculiar version of the TSP, formulated on a hierarchy of graphs and made more complex by the definition of additional constraints strictly related to the nature of the tasks to be executed (time windows, precedence constraints, category constraints, cluster constraints). This makes a significant comparison with other approaches impossible. For this reason, the experimental results we give concern only the decrease in complexity and the shift from optimality yielded by the divide-and-conquer approach.

The time complexity of the divide-and-conquer algorithm can be proved to be $O(\lambda^{(n+3-k)/2n})$; the decrease in time complexity obtained by decomposing the problem is the same as that obtained in [29] for the cheapest-path problem (see Figure 8).

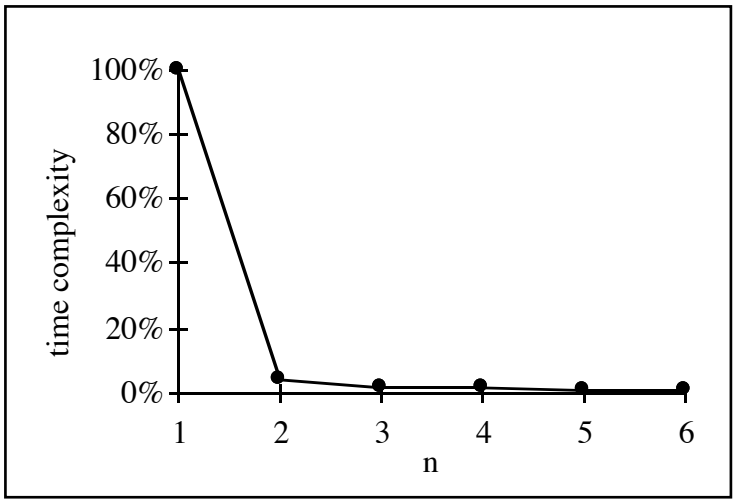

Figure 8. Relative complexity of the divide-and-conquer approach (*TAP* algorithm) as compared to that of the flat approach (*flat_TAP* algorithm); the problems considered are flat TAPs with k=0.

The experimental tests we conducted on this algorithm aim at estimating how much the divide-and-conquer approach affects the performance of the flat algorithm. More specifically, we generated a random set of flat TAPs (k=0) and compared the solutions yielded by the flat TAP algorithm with those yielded by the TAP algorithm. The results are summarized in Figure 9.

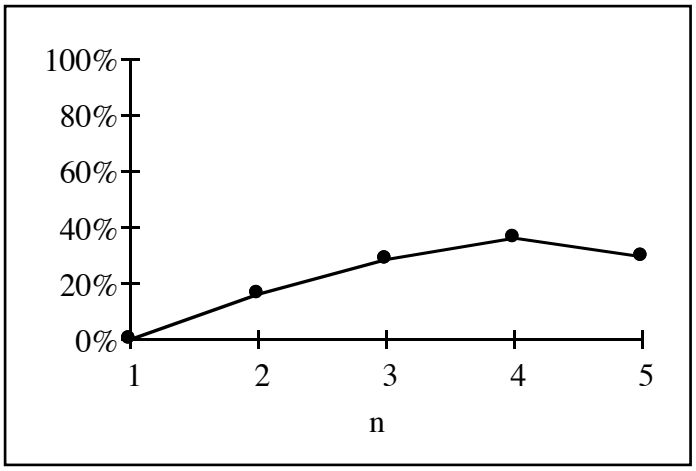

Figure 9. Average shift from optimality introduced by the divide-and-conquer approach. The tests have been conducted on randomly-generated maps whose maximum clustering level, n, ranges from 1 to 5. For each map, a set of flat TAP problems has been generated; the vertical axis measures the relative increase in path length as determined by the comparison between the TAP and the flat TAP algorithms.

VII. CONCLUSION

In this paper a layered architecture to represent knowledge of the environment for navigation has been presented. Each layer is abstracted from another by means of one of three abstraction primitives: classification, generalization, aggregation. The incidence of layering on path planning has been discussed; we have shown that our semantically richer representation of environmental knowledge allows for classical path-planning problems to be re-formulated in a more general fashion. As an alternative to writing *ad-hoc* algorithms to solve the hierarchically-formulated problems, we have demonstrated how a decomposition technique may be adopted to find a sub-optimal solution based on the solution to the corresponding "flat" problem.

Further research topics to be addressed include:

- representation of dynamic environments;
- algorithms for hierarchical topological clustering of unstructured environments;
- algorithms for multi-agent planning;
- definition of *ad hoc* strategies for the exploration of unknown environments.

ACKNOWLEDGMENT

Dr. Marco Patella significantly contributed to the development of the TAP algorithm and to the test of its performance. Dr. Emanuele Tonelli carried out the O_2 implementation of the object-

based schema for layered knowledge. Their work is gratefully acknowledged. We would also like to thank Prof. Aristide Mingozzi for the fruitful discussions on path planning.

REFERENCES

[1] E. Badreddin, "Recursive behavior-based architecture for mobile robots", *Robotics and Autonomous Systems*, vol. 8, pp. 165-176, 1991.

[2] R. Bellman, A.O. Esogbue and I. Nabeshima, *Mathematical aspects of scheduling and applications*, Pergamon Press, 1982.

[3] L. Bianco, A. Mingozzi and S. Ricciardelli, "Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints", Technical Report National Council of Research, Roma, 1992.

[4] L. Bianco, A. Mingozzi, S. Ricciardelli and M. Spadoni, "Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming", *Infor.*, vol. 32, pp. 19-32, 1994.

[5] R.A. Brooks and A.M. Flynn, "A robot being", *NATO Workshop on robotics and biological systems*, Tuscany, Italy, 1989.

[6] O. Causse and J.L. Crowley, "Navigation with constraints for an autonomous mobile robot", *Robotics and Autonomous Systems*, vol. 12, pp. 213-221, 1994.

[7] E. Charniak and D. McDermott, *Introduction to artificial intelligence*, Addison-Wesley, 1985.

[8] H.I. Christensen, N.O. Kirkeby, S. Kristensen, L. Knudsen and E. Granum, "Model-driven vision for in-door navigation", *Robotics and Autonomous Systems*, vol. 12, pp. 199-207, 1994.

[9] P. Ciaccia, D. Maio and S. Rizzi, "Integrating knowledge-based systems and neural networks for navigational tasks", *Proc. IEEE COMPEURO*, Bologna, Italy, pp. 652-656, 1991.

[10] P. Ciaccia, B. Montanari, "Reinforcement-based systems for solving navigational tasks in large domains", *Proc. Int. Workshop On Mechatronical Computer Systems For Perception And Action*, Halmstad, Sweden, pp. 367-374, 1993.

[11] A.A. Covrigaru and R.K. Lindsay, "Deterministic autonomous systems", *AI Magazine*, vol. 12, n. 3, pp. 110-117, 1991.

[12] T. Dean, J. Firby and D. Miller, "Hierarchical planning involving deadlines, travel time and resources", *Computational Intell.*, vol. 4, n. 4, pp. 381-398, 1988.

[13] L. De Floriani and E. Puppo, "A hierarchical triangle-based model for terrain description", in *Theories and methods of spatio-temporal reasoning in geographic space*, A.U. Frank, I. Campari and U. Formentini Eds., Lecture Notes in Computer Science, vol. 639, Springer-Verlag, pp. 236-251, 1992.

[14] M. Desrochers, J. Desrosiers and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows", *Operations Research*, vol. 40, pp. 342-354, 1992.

[15] J. Desrosiers, Y. Dumas, M. Solomon and F. Soumis, "Time costrained routing and scheduling", *Report GERAD*, n. G-92-42, 1993.

[16] R.S. Doshi, R. Lam and J.E. White, "Region based route planning: multi-abstraction route planning based on intermediate level vision processing", in *Autonomous mobile robots*, S.S. Iyengar and A. Elfes Eds., vol. I, IEEE Computer Society Press, pp. 489-504, 1991.

[17] M.J. Egenhofer, "What's special about spatial? Database requirements for vehicle navigation in geographic space", *SIGMOD ACM*, Washington DC, pp. 398-402, 1993.

[18] A. Elfes, "Using occupancy grids for mobile robot perception and navigation", *IEEE Computer*, vol. 22, n. 6, pp. 46-57, 1989.

[19] B. Faverjon and P. Tournassoud, "The mixed approach for motion planning: learning global strategies from a local planner", *Proc. Int. Joint Conf. On Artificial Intell.*, vol. 2, pp. 1131-1137, 1987.

[20] P.J. Hayes, "A representation for robot plans", *Proc. Int. Joint Conf. On Artificial Intell.*, pp.181-188, 1975.

[21] B. Hayes-Roth and F. Hayes-Roth, "A cognitive model of planning", *Cognitive Science*, vol. 3, n. 4, pp. 275-310, 1979.

[22] S.C. Hirtle and J. Jonides, "Evidence of hierarchies in cognitive maps", *Memory and Cognition*, vol. 13, n. 3, pp. 208-217, 1985.

[23] P.D. Holmes and E.R.A. Jungert, "Symbolic and geometric connectivity graph methods for route planning in digitized maps", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 14, n. 5, pp. 549-565, 1992.

[24] D.S. Johnson and C.H. Papadimitriou, "Performance guarantees for heuristics", in *The travelling salesman problem*, E.L. Lawler et al. Eds., John Wiley & Sons, pp. 145-180, 1985.

[25] R.E. Korf, "Planning as search: a quantitative approach", *Artificial Intell.*, vol. 33, 1987.

[26] B.J. Kuipers, R. Froom, W. Lee and D. Pierce, "The semantic hierarchy in robot learning", in *Robot learning*, J.H. Connell and S. Mahadevan Eds., Kluwer Academic Publishers, pp. 141-170, 1993.

[27] D. Maio and S. Rizzi, "Map Learning and Clustering in Autonomous Systems", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 15, n. 12, pp. 1286-1297, 1993.

[28] D. Maio and S. Rizzi, "Knowledge architecture for environment representation in autonomous agents", *Proc. Eighth Int. Symposium on Computer and Information Sciences*, Istanbul, Turkey, pp. 4-11, 1993.

[29] D. Maio and S. Rizzi. "A hybrid approach to path planning in autonomous agents", *Proc. 2nd Int. Conf. on Expert Systems for Development*, Bangkok, Thailand, pp. 222-227, 1994.

[30] D. Maio and S. Rizzi. "Un sistema multimediale per la pianificazione assistita di visite a un museo", *Proc. Convegno Sistemi Multimediali Intelligenti*, Ravello, Italy, pp. 103-111, 1994.

[31] W.A. Phillips, P.J.B. Hancock, N.J. Willson and L.S.Smith, "On the acquisition of object concepts from sensory data", *Neural Computers*, NATO ASI Series, F41, R. Eckmiller and Ch.v.d. Malsburg Ed., Springer-Verlag, pp. 159-168, 1988.

[32] H. Psaraftis, "An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows", Transportation Sciences, vol. 17, pp. 351-357, 1983.

[33] E.D. Sacerdoti, "Planning in a hierarchy of abstraction spaces", *Artificial Intelligence*, vol. 5, n. 2, pp.115-135, 1974.

[34] A.W. Siegel and S.H. White, "The development of spatial representations of large-scale environments", in *Advances in child development and behavior*, H.W. Reese Ed., Academic Press, 1975.

[35] S.P. Singh, "Transfer of learning by composing solutions of elemental sequential tasks", *Machine Learning*, vol. 8, pp. 323-339, 1992.

[36] S. Timpf, G.S. Volta, D. Pollock and M.J. Egenhofer, "A conceptual model of wayfinding using multiple levels of abstraction", in *Theories and methods of spatio-temporal reasoning in geographic space*, A.U. Frank, I. Campari and U. Formentini Eds., Lecture Notes in Computer Science, 639, Springer-Verlag, 1992.

[37] G. Vercelli, R. Zaccaria and P. Morasso, "A theory of sensor-based robot navigation using local information", *Proc. Congresso dell'Associazione Italiana Intelligenza Artificiale*, Italy, pp. 342-351, 1991.

[38] C.R. Weisbin, G. de Saussure, J.R. Einstein, F.G. Pin and E. Heer, "Autonomous mobile robot navigation and learning", *IEEE Computer*, vol. 22, n. 6, pp. 29-35, 1989.

APPENDIX

The pseudo-code for the heuristic flat TAP algorithm is listed below. Apex (k) is dropped from all k-paths and k-clusters in order to make the code more readable. Also, we assume that:

- *length*($\mathcal{P}$) denotes the number of clusters in the path $\mathcal{P}$, and $\mathcal{P}$[v] its v-th cluster;
- $\mathcal{L}$->*flat_CPP*(C_i,C_j) returns the cheapest k-path between C_i and C_j, and d(C_i,C_j) denotes its cost: d(C_i,C_j) = cost($\mathcal{L}$->*flat_CPP*(C_i,C_j)) [4];
- $em_v(C_i)$ denotes the extra-mileage due to insertion of cluster C_i in position v:
  $$em_v(C_i) = \begin{cases} d(\mathcal{P}[v],C_i)+d(C_i,\mathcal{P}[v+1])-d(\mathcal{P}[v],\mathcal{P}[v+1]) & \text{if } v<\text{length}(\mathcal{P}) \\ d(\mathcal{P}[v],C_i) & \text{if } v=\text{length}(\mathcal{P}) \end{cases}$$
- *constraints_ok*(C_i,v*) is a function which re-calculates ta_v and td_v for each position v in $\mathcal{P}$ if C_i is inserted in position v*, and returns *true* if the time window and precedence constraints are not violated, *false* otherwise;
- *candidates*(v,S_i), where $S_i \in Es$, is a function which heuristically selects, from the set of clusters belonging to S_i, those to be considered for insertion in position v.

flat_TAP algorithm

{ En = ∅; // *set of resources which cannot be visited within the path*
$\mathcal{P}$ = (C_{start},C_{end});
if Ec≠∅ **then**
{ find $S_{first} \in Ec$, $S_{first}=\{C_{first}\}$: $\pi_{first} \cap Ec=\emptyset \wedge (d(C_{start},C_{first})+d(C_{first},C_{end})$ is maximum)
insert C_{first} in $\mathcal{P}$; Ec=Ec-{S_{first}};
}
while Ec≠∅ **do** // cheapest insertion for mono-resources
{ **for** S_i in Ec, $S_i=\{C_i\}$: $\forall S_j \in \pi_i \cap Ec$ (S_j is visited in $\mathcal{P}$) **do**
find v^*_i: constraints_ok(C_i,v^*_i) ∧ ($em_{v^*_i}(C_i)$ is minimum);
// v^*_i *is the best position after which* C_i *may be inserted in* $\mathcal{P}$
find $S_{next} \in Ec$, $S_{next}=\{C_{next}\}$: ($em_{v^*_{next}}(C_{next})$ is minimum);
if $em_{v^*_{next}}(C_{next})=\infty$ **then** En=En+{S_{next}}; // *resource cannot be visited*
else insert C_{next} in $\mathcal{P}$ after position v^*_{next};
Ec=Ec-{S_{next}};
}
for S_i in Es **do** // *find the best clusters for each multi-resource*
for v = 1 **to** length($\mathcal{P}$) **do**
{ find $C^*_{i,v} \in$ candidates(v,S_i) : ($em_v(C^*_{i,v})$ is minimum);
// $C^*_{i,v}$ *is the best cluster of* S_i *to be inserted in* $\mathcal{P}$ *after position* v
}
while Es≠∅ **do** // *cheapest insertion for multi-resource*
{ **for** S_i in Es: $\forall S_j \in \pi_i \cap Es$ (S_j is visited in $\mathcal{P}$) **do**
find v^*_i: constraints_ok(C^*_{i,v^*_i},v^*_i) ∧ ($em_{v^*_i}(C^*_{i,v^*_i})$ is minimum);
// v^*_i *is the best position after which a cluster of* S_i *may be inserted in* $\mathcal{P}$

[4] Following the object-oriented notation, writing *o* ->*m*(*p1*,...*pn*) denotes applying method *m* to object *o* with parameters *p1*,...*pn*.

```
        find S_next∈Es : (em_{v*_next}(C*_{next,v*_next}) is minimum);
        if em_{v*_next}(C*_{next,v*_next})=∞ then  En=En+{S_next};   // resource cannot be visited
        else {  insert C*_{next,v*_next} in P after position v*_next;
                for S_i in Es do          // locally update best clusters for each set
                    for v = v*_next to v*_next+1 do
                        find C*_{i,v}∈candidates(v,S_i) : (em_v(C*_{i,v}) is minimum);
             }
        Es=Es-{S_next};
    }
    for v=1 to length(P)-1 do
        insert L->flat_CPP(P[v],P[v+1]) in P after position v;       // L is the k-clustered layer
}
```