Proceedings of the International Multiconference on Computer Science and Information Technology pp. 105–111
ISBN 978-83-60810-14-9
ISSN 1896-7094

A rule-based approach to robust granular planning

Sebastian Ernst
Institute of Automatics
AGH University of Science and Technology
al. Mickiewicza 30
30-059 Kraków
Poland
Email: ernst@agh.edu.pl

Antoni Ligęza
Institute of Automatics
AGH University of Science and Technology
al. Mickiewicza 30
30-059 Kraków
Poland
Email: ligeza@agh.edu.pl

***Abstract*—Real-time execution of planned routes often requires re-planning, especially in dynamic, highly unpredictable environments. Ad-hoc re-planning in case of failure of an optimal (suboptimal) plan leads to deterioration of solution quality. Moreover, it becomes costly and often unmanageable under real time constraints. This paper describes an approach of a partitioning scheme for urban route planning problem domain into a hierarchy of subproblems and an algorithm for rule-based top-down route derivation. A hierarchical, rule-based approach is aimed at granular planning for generating robust, multi-variant plans.**

I. Introduction

ROUTE planning became a classic problem of Artificial Intelligence [1]. Simultaneously, numerous practical planning algorithms were developed and the worked-out techniques found practical applications in complex, realistic domains. There is a very large group of problems which can be solved by an Artificial Intelligence planning techniques and the AI methods are explored in practice. Such problems include mobile robot route planning, production line scheduling, development of automated game playing strategies and, last but not least, vehicle route planning.

There have been developed a number of well-founded plan generation techniques, mostly based on graph-search procedures [2]. A recent handbook on planning provides a comprehensive review of the domain [3]. Practical applications became a part of everyday life [4], [5], [6], [7].

The route planning problem becomes especially nice and interesting results are obtained when the search takes place in regular, sparse graphs, and such a situation takes place during route planning for long distances with use of highways and motor-roads for most of the planned route. On the other hand, efficient planning in a large, congested city, under real time constraints and under highly unpredictable traffic conditions remains quite a challenge even for most sophisticated systems. In fact, even the problem statement should be different here with respect to the classics of planning a route incorporating mostly motorways and inter-city roads. The unpredictability of urban traffic conditions and the relatively dense road network are both factors intrinsic in the urban route planning and making it a difficult task. Modern planning systems often provide unreliable results for such class of hard planning problems, sometimes failing to provide them in time or do not provide them at all.

This paper describes an approach to planning based on combination of plan from components calculated a priori. A set of rules assigned to each such component provides a ready-to-use solution to be selected depending on external conditions and the aimed destination. Instead of a single, optimal plan, a bunch of similar, alternative plans is in use; when execution of one of them fails, another one is activated immediately. A partitioning scheme for urban route planning problem domain into a hierarchy of subproblems and an algorithm for rule-based top-down route derivation are outlined. A hierarchical, rule-based approach is aimed at granular planning for generating robust, multi-variant plans.

In our previous research, we focused on various aspects of urban route planning which can be improved by using methods of artificial intelligence. These include prediction of traffic conditions [8], induction of granularity over plan solutions [9] and adaptation of well-known computer network routing protocols to broadcast partial solutions between plan nodes [10], [11].

The decision to take up the rule-based hierarchical approach to solving urban route planning problems can be best justified by presenting the currently used methods for route planning, their shortcomings and research done to date regarding improvement of those methods.

A. State of the art

Commercially available computer-based route planning systems date back to the 1980s. At the time, products such as AutoRoute had to cope with harsh limitations of computer resources. The vast development of processing power even in portable personal computers remedied that problem, but the search algorithms and optimality criteria remained mostly unchanged since that time.

Classical planning algorithms are based on informed graph-search methods, such as various modifications of the famous A^* algorithm and its descendants (e.g. IDA^*) [1], [3]. The criteria for optimality used by such "classical" route planning methods are, in most cases, limited to:

- finding the shortest route (the so-called *shortest-path algorithms*,

- time-optimal solutions requiring finding the least time-consuming route (where the travel time is calculated using naive criteria based on the type of route, not taking the prevailing traffic conditions into account),
- finding the least expensive route (by adding estimated fuel consumption and road fees/taxes).

The above generic possibilities can be extended by requirement to go through some indicated middle-points, avoid some forbidden points, avoid left-turn, as well as other user preferences.

In general, the worked out planning methods, combined with the processing power of modern computers, are enough for most applications in long-distance route planning, especially when planning is performed *a priori* – before the travel begins. The search graph is usually relatively sparse, and binary congestion warning/road segment exclusion methods provide sufficient means of route customization.

B. Shortcomings of classical planning methods

The planning process becomes more problematic complex task when the following conditions occur:

- the search domain becomes more dense, and hence the effective branching factor is too high for systematic planning; too many alternative plans are to be considered,
- more complex criteria have to be taken into account in the search process, especially ones influenced by unknown, unpredictable conditions,
- it is necessary to perform re-planning, due to a change of traffic conditions or driver disobedience,
- robustness of the plan becomes of primary importance, i.e. the user wants to have a *reliable* plan working under almost any conditions instead of an optimal one, but fallible under expected traffic conditions.

The aforementioned shortcomings become especially apparent in situations where planning has to be performed in real time, namely when the planning methods are used in real-time mobile navigation devices. Such devices often offer limited processing power, making algorithm efficiency even more crucial.

C. Previous research

Efficient plan generation under heavy traffic conditions in complex urban environments has not deserved intensive study in the literature [3]. An approach based on pure graph-search methods seems to be of limited usability. A study of use of methods based on *case-based* approach was presented in [12].

Knowing that pre-computed solutions have to be calculated *a priori*, we focused on several methods providing a compromise between combinatorial explosion and calculation time.

Article [9] introduced a new approach to solving planning problems, based on the concept of maintaining a set of alternative solutions to allow quick plan switching without the need of re-planning. A new concept, called solution robustness, was introduced to allow non-standard optimality criteria. A robust solution is one that is unlikely to fail.

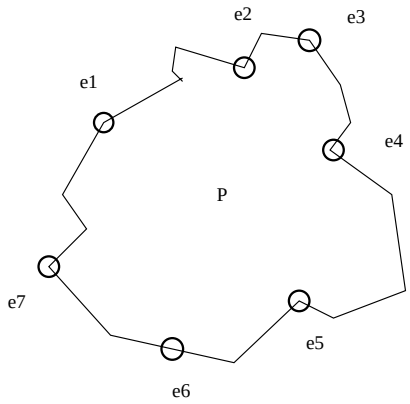

Fig. 1. A schematic presentation of a granule of the planning area with entries.

A different approach was presented in [10]. The concept, based on dynamic computer network routing protocols, is based on partial solution broadcasting between search domain nodes (usually representing junctions). Each node would maintain a "routing table", providing the next node to be visited in order to reach a given destination node.

II. GRANULAR HIERARCHICAL PLANNING: AN INTUITIVE OUTLINE OF THE APPROACH

Since the proposed approach is quite different from the classical ones based on graphs search procedures, below we present an outline of basic ideas of knowledge representation and planning techniques in use.

First, let us point out to the principal assumptions forming foundations of the approach:

- excessive part of planning is performed off-line, a priori, prepared and stored in forms of rules ready-to-use on desire,
- such rules for components of knowledge specific for the area of planning; they can be re-used whenever necessary,
- a plan has hierarchical structure; recursive going up and down is possible,
- instead of a single-thread plan multi-thread (granular ones) are enforced,
- re-planning consist in finding and adapting almost ready plan using prepared a priori knowledge components.

The very basic idea is that a graph modelling the network of routes is divided into separate granules covering it. A granule is and area restricted by some border line where there are some distinguished entry points through which one can enter and leave this area. A good example – for intuition – is a city having several entry roads or a specific district with some entry point. Some illustrative picture of such a granule P with seven entry points is given in Fig. 1. The entry points of granule P are denoted as $P.e1, P.e2, \ldots, P.e7$.

The complete area of planning is divided into such granular components covering it; the principle is to use natural borders and having as small number of entry points as possible. Granules are interconnected by links. In Fig. 2 we have

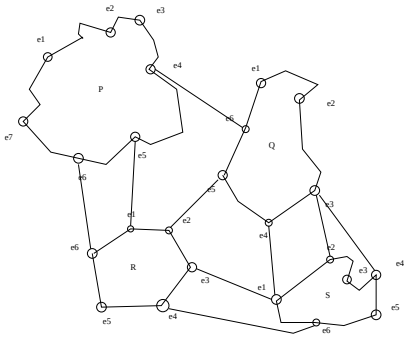

Fig. 2. A schematic presentation of a four interconnected granules within the planning area with entries.

some four interconnected granules, namely P, Q, R and S. For example, granules P and Q are connected by a single link $P.e4 - -Q.e6$, while granules Q and S are connected through three links, namely: $Q.e4 - -S.e1$, $Q.e3 - -S.e2$ and $Q.e3 - -S.e3$. For simplicity we assume that all such interconnections are symmetric.

Now, a simple plan to go from an initial location I (located within some granule, say P) to some destination goal location G (located within some other granule, say S) is a sequence of the form:

$$\pi\colon l_I(I - -P.e4), P.e4 - -Q.e6, e6Qe4, Qe4 - -S.e1, l_G(S.e1 - -G), \quad (1)$$

where

- $l_I(I - -P.e4)$ is a partial plan to move from I to $P.e4$ within P; it may be necessary to employ classical search techniques to generate this part,
- $l_G(S.e1 - -G)$ is a partial plan to move from $S.e1$ to G within S; it may be necessary to employ classical search techniques to generate this part,
- $P.e4 - -Q.e6$, and $Qe4 - -S.e1$ are just trivial transition links,
- $e6Qe4$ is a set of computed a priori plans to move from $Q.e6$ to $Q.e4$ with Q; a plan from the set can be selected according to optimality criteria (a shortest one or a cheapest one) or depending to current time, weather, traffic conditions, etc.

Two further important issues are as follows.

First, any granule can be lowest-level one, where entries are just interconnected through an indivisible network of roads, or a higher level granule covering a set of interconnected granules inside.

Second, for any granule P there is a set of rules, for convenience represented in the tabular XTT form, where each rule is of the form:

$$entry = P.e_i \wedge out = P.e_j \wedge \psi \longrightarrow plan = \pi(P, i, j, \psi).$$

For the granules of the lowest level, a plan $\pi(P, i, j, \psi)$ is just a path through the graph modelling the network of roads. For any higher level granules a plan $\pi(P, i, j, \psi)$ can incorporate higher level notation such as (1); recall that granules of higher levels are recursively divide into lover level ones.

TABLE I
A SCHEME OF A SIMPLE RULE-BASE TABLE FOR STORING PLANS

entry	out	cond	plan
P.e1	P.e2	ψ_1	$\pi(P, 1, 2, \psi_1)$
P.e1	P.e3	ψ_2	$\pi(P, 1, 3, \psi_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
P.e6	P.e7	ψ_{46656}	$\pi(P, 6, 7, \psi_{46656})$

III. IMPORTANT DEFINITIONS

The original form of a route planning problem domain is a graph. Nodes represent individual states (usually junctions), and vertices (edges) represent the connections (roads, streets). Edge weights represent the value of the quality indicator, according to the chosen optimality criteria. As it is possible that there is more than one connection between two nodes, a modified definition for a graph must be used.

Definition 1 (Graph): Let N be a finite set of nodes and let E be a finite set of edges. A graph G is a four-tuple

$$G = (N, E, \alpha, \omega) \quad (2)$$

where α and ω are functions assigning, to every edge $e \in E$, its start node $n_I \in N$ and end node $n_G \in N$, respectively:

$$\alpha : E \to N, \quad \omega : E \to N \quad (3)$$

Please note that every node $n \in N$ will have precise (x, y) coordinates, denoting the geographical location of a given junction.

As the graph represents a fragment of a map – i.e. a city or a part of it, it will most likely be connected with the outside world using roads. We shall therefore create virtual nodes at every exit road, representing entry or exit from the current fragment of the map. That set of nodes shall be called terminators and be defined as $N_T \subset N$. Geographic coordinates for such virtual nodes shall be outside the boundaries of the map region.

Definition 2 (Terminator): A terminator is a virtually-created node $n \in N_T \subset N$, representing entry into or exit out of the domain of the planning process. As junctions outside the domain are not of interest to the planning process, each node $n \in N_T$ may only have at most *one* edge leading to another (non-virtual) node $n' \notin N_T$, $n' \in N$ and, optionally, at most *one* edge in the reverse direction.

Basing upon the graph definition above, we shall now provide the definitions for hierarchical graph partitioning.

Definition 3 (Partition): We shall say that a series of sets $P_1, P_2, \ldots, P_n \subset N$ are partitions and define a partitioning scheme of a graph G if:

- $P_1 \cup P_2 \cup \ldots \cup P_n = E$, i.e. the subgraphs cover all nodes of the graph G being partitioned,
- $P_i \cap P_j = \emptyset$, $i, j = 1, 2, \ldots, n$, $i \neq j$, i.e. any given node can belong to only one of the resulting subgraphs,

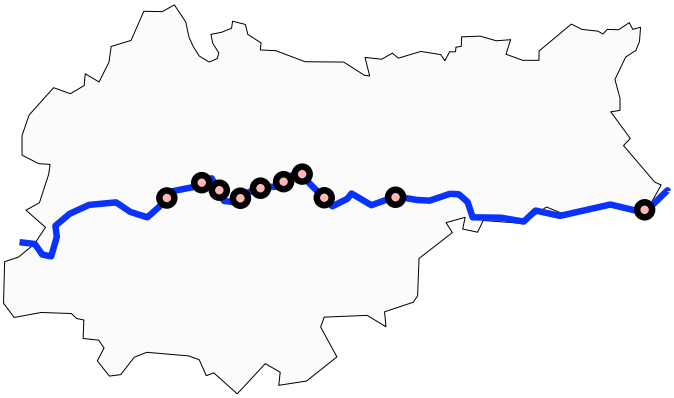

Fig. 3. An outline map of the city of Kraków, showing the boundary between two particles, marked out by the Vistula river, as well as internal links between those partitions.

- every subgraph defined by partitions P_i, $i = 1, 2, \ldots, n$ is connected.[1]

In practice, partition boundaries will often be natural or human-made landmarks which provide a limited number of options of crossing them. Bridges or railways are examples of such landmarks. To clarify the concept of the partition, Fig. 3 shows an outline map of the city of Kraków divided naturally into two partitions by the river Vistula. Circles denote the limited number of ways to get from one partition to another. Definitions of neighboring partitions, partition links and partition boundaries are intuitive. Formal definitions follow.

Definition 4 (Neighboring partitions): We say that partitions P_i and P_j are neighboring if there is at least one pair of nodes connected directly by an edge such that one node belongs to P_i and the other belongs to P_j:

$$\exists n_i, n_j : n_i \in P_i, n_j \in P_j, (n_i, n_j) \in E \tag{4}$$

Definition 5 (Partition link): An internal partition link (partition link) between two partitions P_i and P_j is an edge $l \in E$, $l = (n_i, n_j)$, such that $n_i \in P_i$ and $n_j \in P_j$. We say that link l is associated with partitions P_i and P_j: $l \sim P_i$, $l \sim P_j$.

Intuitively, a partition link provides the means of getting from one partition to another. Coordinates of a link can in fact be defined as an arbitrarily chosen point belonging to the road leading from junction n_i to junction n_j. In practice, this will often be the crossing point between the road and the landmark object used to define the partition in the first place.

On Figure 3, internal links (in this case: bridges) are represented as small circles.

Definition 6 (External partition link): Let G be a graph being the input for the partitioning process at level m, as described by section III-A. An external partition link is an edge $l_x \in E$, $l_x = (n_i, n_j)$ such that:

- $n_i \in N_T$ or $n_j \in N_T$, or:
- for $m = 1, 2, 3, \ldots$ ($m \neq 0$), n_i and n_j belong to two neighboring partitions at level $m - 1$.

[1]A graph is connected if there is a path connecting any two nodes.

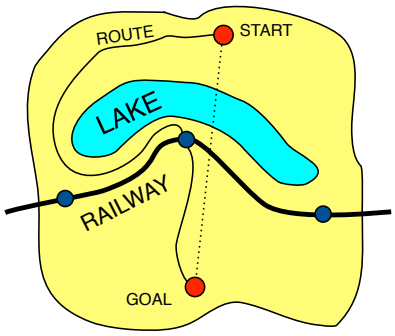

Fig. 4. An example map with a missing partition boundary and the resulting non-optimal solution.

Definition 7 (Partition boundary): The boundary L_i of partition P_i is defined by the set of all its links associated with it:

$$L_i = \{l \in E : l \sim P_i\} \tag{5}$$

Geometric representation of the boundary can be perceived as a series of line segments, joining successive pairs of links, in a way that each linking road is crossed only once.

On Figure 3, the Vistula river is the boundary.

Every partition shall have an array of traversal rules.

Definition 8 (Traversal rule): Let P be a partition as defined by Definition 3, and L be its boundary, as defined by (5). A traversal rule is a five-tuple:

$$R = (P, l_I, l_G, g, \Phi, \Psi) \tag{6}$$

where P is the partition, $l_I \in E$ is the entering link, $l_G \in E$ is the exit link, g is the traversal cost (depending on chosen optimality criteria), Φ is an optional set of optional validity values (e.g. time of day) and Ψ is the traversal plan to be followed.

Definition 9 (Traversal plan): A traversal plan Ψ of length $n - 1$ from link l_I to link l_G through partition P at level m is defined as follows:

$$\Psi = (l_I, P_1, l_1, P_2, l_2, \ldots, l_{n-1}, P_n, l_G) \tag{7}$$

where P_i ($i = 1, 2, \ldots, m$) are partitions at level $m + 1$ (sub-partitions of P), and l_i ($i = 1, 2, \ldots, m$) are links between partitions P_i and P_{i+1}.

A. Partition hierarchy

As mentioned before, the partitioning scheme is hierarchical, e.g. every partition P_i on level n may become the source graph for partitioning on level $n + 1$. On level 0, the source graph is identical to the search domain.

Definition 10 (Elementary partition): A partition P_i at level n which is not divided into other sub-partitions (i.e., no sub-partitions of it exist at level $n + 1$) shall be called an *elementary partition*.

Partitioning should be performed to the point when route planning within a partition can be performed in negligible time using classical shortest path algorithms.

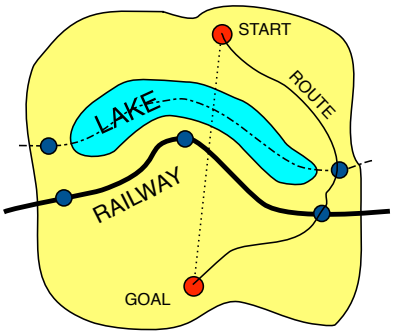

Fig. 5. An example map after adding the missing partition.

In order to maintain consistency throughout the entire hierarchy, partitioning has to be balanced.

Definition 11 (Partition balancing): If L_i is the boundary of partition P_i at level n, and that partition becomes the source for partitioning at level $n+1$, the set of external links L_x for partitions created by dividing partition P_i equals the boundary of that partition (L_i) at level n.

IV. Map partitioning procedures

The process of map partitioning is where intelligent processing actually takes place, thus it requires expert knowledge to be performed properly – to ensure that the entire system yields feasible and optimal results.

The complexity level of the problem makes development of automatic partitioning methods difficult. Therefore, we shall now focus on providing general guidelines for map partitioning, and to discuss the implementation feasibility of an automatic partitioning solution.

A. Guidelines for partitioning strategies

The primary guideline for partitioning the map is appropriate selection of abstraction levels and partition boundaries. For instance, if a significant landmark is likely to be an obstacle, a partition boundary should be placed accordingly to its size.

For example, please compare the partitioning schemes presented on figures 4 and 5. On figure 4, the partition has been split into two sub-partitions (at the next level), using the railway as the boundary. The algorithm presented below uses geometric criteria to order (or choose) the links used (dotted line between the start and the goal). Therefore, it will assume that the middle railway crossing is the one most likely to be used. If the planning strategy is unfavorable (i.e., full review is not done)

B. Traversal plan calculation

For each non-elementary partition, a traversal plan needs to be established. Similarly to the partitioning itself, this article assumes that the traversal plans are prepared by experts. However, rough plans may be pre-computed using simple criteria, such as location of exits and known road network irregularities. Such aid should provide satisfactory results, especially on high abstraction levels. However, it should in all cases be reviewed by an expert.

Automated map partitioning and traversal plan calculation methods are indeed possible, but they are not covered in this article, as they involve preparation of the search domain, and not solving the planning problem itself.

V. Algorithm proposal

Please note that the algorithm is in an early stage of development and therefore some of the assumptions made by it can seem rather naive. Its purpose is to illustrate which stages benefit from using the foundations described in the preceding section.

The algorithm is divided into two main procedures:

- calculate point-to-point route,
- traverse entire partition.

The first procedure (described in section V-A) does not take great benefits from the approach described here, but is needed as an "access road" to true partition traversal. Please note that at any level, the point-to-procedure does not utilize the rule tables associated with partitions.

The *Traverse entire partition* procedure (described in section V-B) is where rule-based processing takes place. No search procedures are utilized, until the elementary partitions are reached and the subproblems become trivial.[2]

A. Calculate point-to-point route

This is the main procedure used for route planning.

1) Input arguments: n_I (start node), n_G (goal node).
2) Recursively locate n_I and n_G to partitions on all applicable levels. (Determine start and goal node addresses).
3) Assume that P_I is the level 0 partition of node n_I, and P_G is the level 0 partition of node n_G.
4) Define point p. Assign the n_I to p.
5) Allocate a node of the search graph used to store search options. Assign n_I to the node.
6) Repeat: (build the search tree)
   a) Draw a line segment from p to n_G and select the internal link $l \sim P_I$.
   b) Add the links between P_I and its neighboring partitions as nodes of the search graph, ordered by the distance from the line drawn in the previous point[3], and add appropriate edges.
7) For all nodes linked with the start node n_I, recursively calculate the point-to-point route from n_I to the geographic location of the link defined by the given node.
8) For all nodes linked with the goal node n_G, recursively calculate the point-to-point route from the geographic location of the link defined by the given node to n_G.

[2]Nevertheless, routes for elementary partitions can also be pre-calculated in order to achieve full independence from search algorithms in the *Traverse entire partition* part of the algorithm.

[3]To reduce complexity, the number of nodes added at each level can be limited, either by setting a limit or by defining the maximum distance from the line drawn in step 6a.

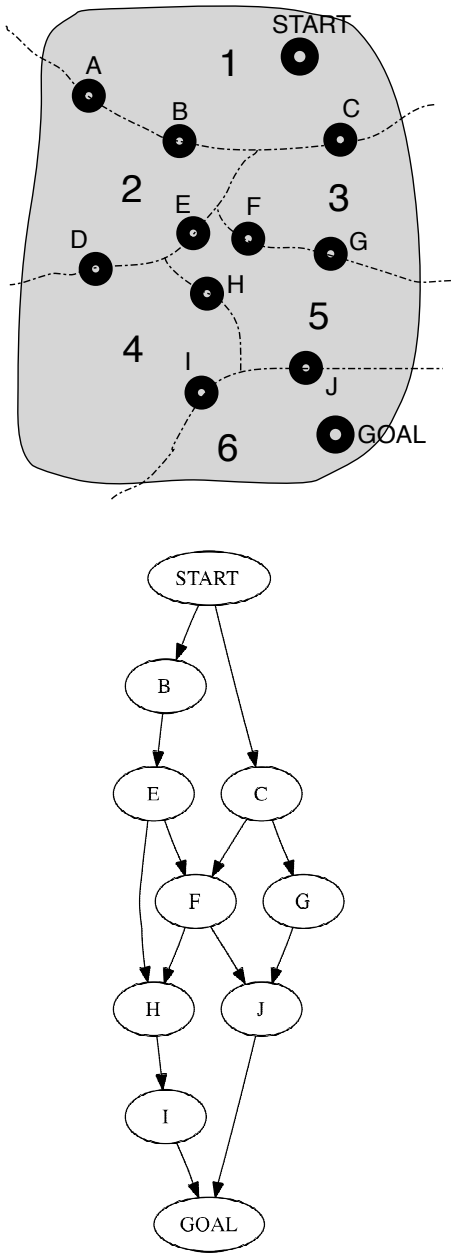

Fig. 6. An example division of a planning problem in to subproblems.

9) For all other nodes, solve the Traverse entire partition problem from their parents in the search tree.
10) For all leaves of the search tree, calculate the point-to-point route from the geographic location of the link defined by the given node to n_G.

B. Traverse entire partition

This procedure is executed if there is an entire partition (at a given level) to be traversed. It only applies to non-elementary partitions (see definition 10).

1) Input arguments: P (partition), l_I (start link), l_G (goal link), Φ (additional criteria; optional).
2) Match rule $R = (P, l_I, l_G, g, \Phi, \Psi)$ with the most favorable value of g.
3) Retrieve traversal plan Ψ ($\Psi = (l_0, P_1, l_1, P_2, l_2, \ldots, l_{n-1}, P_n, l_n)$).
4) Let n be the length of Ψ.
5) For i from 1 to n, recursively traverse entire partition P_i from node l_{i-1} to node l_i.

C. Execution example

As the *Calculate point-to-point route* algorithm may not seem trivial, we provide an example of its operation. Figure 6 shows a rather simple partition, divided into 6 sub-partitions, labelled 1-6. Internal links between those partitions have been labelled as A-J. The start node is located in partition 1, the goal node – in partition 6. Below the drawing is the search graph generated by the procedure described in section V-A.

The *Calculate point-to-point route* algorithm will be used to solve the following subproblems:

- $START \rightarrow B$
- $START \rightarrow C$
- $I \rightarrow GOAL$
- $J \rightarrow GOAL$

The *Traverse entire partition* algorithm will be used to solve the following subproblems:

- $B \rightarrow E$
- $C \rightarrow F$, $C \rightarrow G$
- $E \rightarrow F$
- $F \rightarrow H$, $F \rightarrow J$
- $G \rightarrow J$
- $H \rightarrow I$

VI. Conclusion

The article provides some outline for a new rule-based approach to solving planning problems, with special emphasis on vehicle route planning in dense, urban areas under unpredictable traffic conditions. In such cases re-planning may be of excessive use, and granular planning may proof to constitute a robust approach.

The algorithm provided is at an early stage of development, but is a good illustration of the formal foundations described above.

Future work includes:

- development of automatic map partitioning schemes,
- practical implementation of described methods,
- improvement of the search algorithms.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
[2] N. J. Nilsson, *Principles of Artificial Intelligence*. Tioga Publishing Co., 1980.
[3] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004, iSBN-10:1-55860-856-7.
[4] Wikipedia, "Google maps," http://pl.wikipedia.org/wiki/Google_Maps, 2007.
[5] ——, "Zumi," http://pl.wikipedia.org/wiki/Zumi, 2007.
[6] Map24, "Map24.interia.pl," http://map24.interia.pl/, 2007.
[7] ViaMichelin, "Route planning systems," http://www.viamichelin.com/.

[8] S. Ernst and A. Ligęza, “Analiza mot’liwoąci zastosowania metod int’ynierii wiedzy do budowy inteligentnego systemu planowania trasy w ruchu miejskim,” in *Int’ynieria wiedzy i systemy ekspertowe*, vol. 2, WrocŞaw, Poland, 2006, pp. 25–34.
[9] ——, “Adaptive granular planning for robust plan generation under uncertain traffic conditions,” in *Proceedings of the 16th international conference on Systems Science*, vol. 2, WrocŞaw, Poland, 2007, pp. 388–396.
[10] ——, “Solving planning problems by broadcasting partial solutions,” in *Tools of information technology*, A. Kos, Ed., Rzeszów, Poland, 2007, pp. 79–84.
[11] S. Ernst, “A multi-agent implementation of an automated planner for dynamic environments,” in *Computer Methods and Systems*, M. S. Ryszard Tadeusiewicz, Antoni LigŚza, Ed., Kraków, Poland, 2007, pp. 99–104.
[12] K. Z. Haigh, J. R. Shewchuk, and M. M. Veloso, “Exploiting domain geometry in analogical route planning,” Journal of Experimental and Theoretical Artificial Intelligence, 1997.