Automatic Generation of Robust Granular Plans

Sebastian Ernst and Antoni Ligęza

Institute of Automatics
AGH University of Science and Technology

Explicite, 16 April 2008

Presentation Outline

1. Why?
   - The problem with satellite navigation
   - State of the art
   - Problem examples
   - Motivation summary
2. What?
   - Action plan
   - The ideal satnav
   - Needed improvements over existing methods
3. How?
   - How are we going to do this?
   - Optimal vs. robust planning
   - Linear vs. granular planning
   - Knowledge compilation
   - Robust granular plan execution

Why?

Why do we think there is a problem?

- Satnavs are rather helpful...
- ...right up to the moment when they start to *interfere* rather than *cooperate*.

Planning vs. plan execution

Planning

- Task: search the graph to find shortest path from *Start* to *Goal*
- Performed *a priori*
- Employs classical graph search methods, such as Dijkstra's algorithm, A*, IDA

Plan execution

- Performed by a human executor
- Prone to unforeseen circumstances and human error, but
- Equipped with intelligence and environment observations

Route planners today

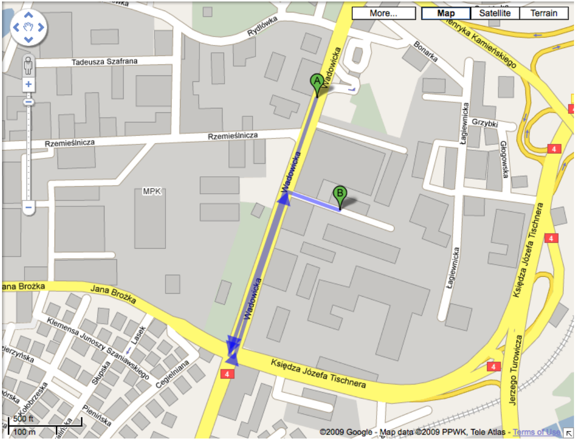
Pros

- Both *a priori* (Google Maps, ViaMichelin, Map24) and real-time (mobile satellite navigation systems)
- Algorithms very effective, maps frequently updated (TeleAtlas, Navteq)
- Commercially feasible and easily available
- Very comfortable to use

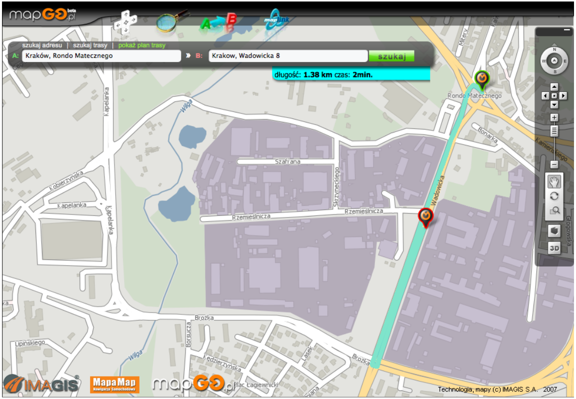
Cons

- Solutions good only as long as no problems occur
- Often too imposing for the driver ("Do as I tell you or get lost")

Example 1 – Google Maps



Example 1 – MapGO



Example 1 – Can't take that turn!

Example 1 – analysis

- The problem here is *map inaccuracy* – an inherent characteristic of the road network.
- Result: directions that cannot be followed.
- Maps *are* and always *will be* inaccurate to some extent.
- Why not show the driver the other possible maneuvers, leading to less optimal but feasible routes?

Example 2 – blocked road

Example 2 – analysis

- The driver may see what's coming up – for instance, that the road is blocked.
- *"Can I avoid the problem by taking this turn?"*
- *"I can't go straight on, should I turn right or left?"*
- The decision has to be made instantaneously, no time for UI interaction!
- A decision made under stress should be as "safe" as possible, but "safe" decisions tend to lead to huge problems.
- The system should *anticipate* sudden changes.

Example 3

Case 1: unexpected obstacle

The driver leaves the original route (or even turns around) to avoid a traffic jam. The satnav, however, stupidly takes every opportunity to put the driver back on the original route, rather than calculate a new one – just because the numbers tell it to do so.

Case 2: recent changes in road network

Some one-way streets have been reversed. The satnav doesn't know, and the driver starts to go around in circles.

Example 3

- Human error can happen, but usually maneuvers are made for *a reason*.
- Problem: The system starts considering alternatives only when something goes wrong. By doing so, it is incapable of recognizing the driver's *intentions* and *motivation*.

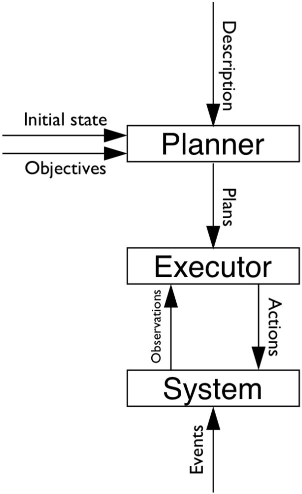
Identify the key problems

- Map inaccuracy
- Alternatives only calculated *on demand* and by *following the events*
- No "big picture" means no *understanding* of *intentions*

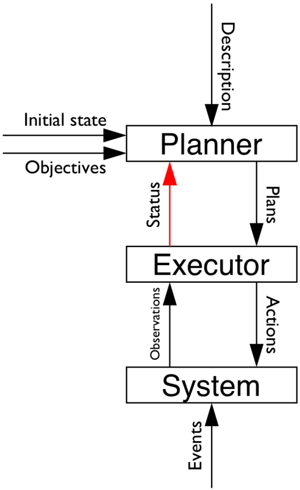
The “missing link”

- The planner *knows* significantly less than the plan executor.
- However, the executor is unable to *communicate* that knowledge to the planner.

Static planning



Dynamic planning



Other problems

- The planner is inaccurately shifted from a *decision support system* to a *decision-making system*.
- No use made of available *human intelligence*.
- Limited computing power leads to "re-planning loops", a condition when the vehicle arrives at the next node before the re-planning is completed.
- Currently used algorithms are not complete. Applications are known to use the strategy of "reach the highway, forget the others".

What?

What are we going to do about it?

- Sketch the ideal satnav solution.
- Determine what needs to be done in relation to current AI planning methods.
- Design the formalisms and methods needed to implement it.
- Ultimately, build a truly AI-based engine for satellite navigation.

Desired features

"The Robust Granular Planning Manifesto"

1. **Provide full information:** The driver should be presented with the "big picture", including the crucial *decision points* at different levels of abstraction.
2. **Do not interfere:** The driver should have the *freedom to choose* the solution; the system is there to merely let them know about the possibilities (and their optimality).
3. **Require no input:** The driver's intentions should be manifested by *actions*, rather than explicit *UI interaction*.
4. **Expect the unexpected:** The system should *anticipate* deviations from the original plan and recognize the driver's decisions.

What it could look like

Current methods and what needs to be changed

- Single-path planning is by far the most commonly used. However, it is not applicable for our needs.
- Multiple paths could be calculated *on demand*. This is time-consuming, and could easily overwhelm mobile devices with limited processing power.
- **Knowledge compilation:** process the input graphs (maps) and generate ready-to-use data (decision tables assigned to branching nodes).
- Possible problem: combinatorial explosion. But (we think) we know what to do about it.
- Advantage: pre-compiled knowledge can be reused for different planning assignments.

How?

The proposed solution

- Introducing the concept of *robust granular planning*.

Robust granular planning

- Robust: multiple-option vs. single-path plans
- Granular: hierarchical structuring of the input data, with knowledge compilation for selected regions

Execution of a robust granular plan

- The executor is given the *freedom* to choose one of the solutions and *knowledge* to aid the selection
- Re-planning time is greatly reduced

Criteria and constraints

Criteria commonly used for route planning

- Route length
- Estimated travel time – calculated using pre-set average speeds for various road types
- Estimated travel cost – by adding road fees and projected fuel consumption

Common route planning constraints

- Preferred intermediate points
- Excluded road segments

Solution robustness

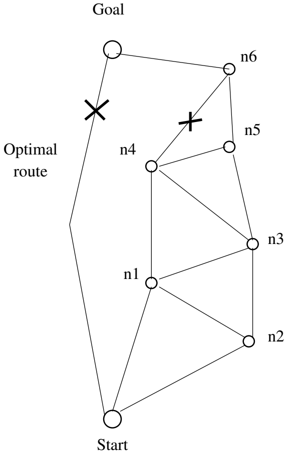
- **Stable quality:** Quality of solutions *optimal* according to given criteria is prone to drastic deterioration if execution of the initial plan fails.
- **Less prone to failure:** A *robust* solution is one that is less likely to fail, even if suboptimal according to classical criteria.
- **An intuitive (and extreme) example:** Using the highway is faster than traveling through the city center, but if anything fails on the highway, the driver may be trapped in a traffic jam, far away from an exit.

Krakï£¡w: a fast solution

Krakï£¡w: a robust solution

Plan robustness: a simple example

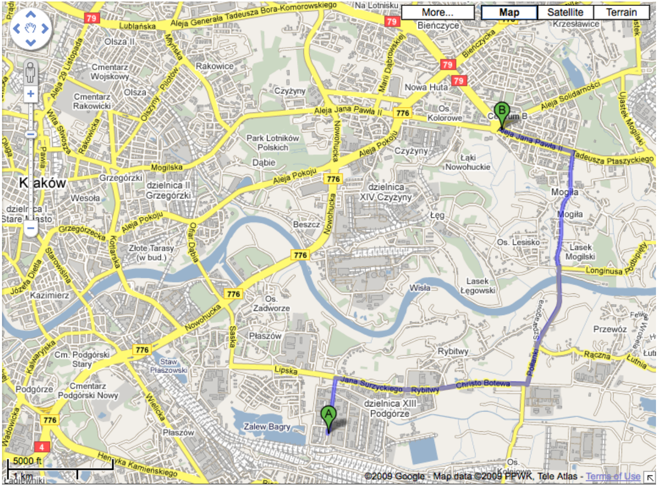
If at point *n*4 the driver should notice that the road is blocked, they might still use a detour (*n*5–*n*6), whereas during execution of the left plan, there only solution is to turn back (if at all possible).



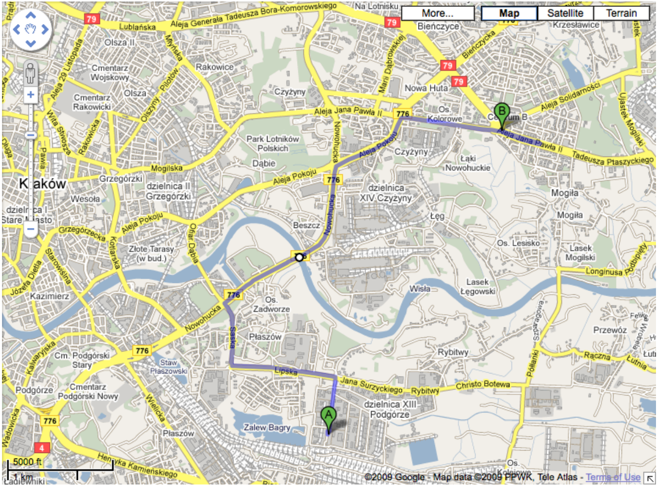
Robustness evaluation

- Robustness of a plan portion (i.e., a decision made at a switching point/junction) can be quantified in a number of ways.
- We shall refer to such value as the *robustness factor*.
- The robustness factor can be defined in a number of ways, taking various features into account.
- Robustness factors can range from simple (branching factor) to very complex (i.e., employment of information theory).

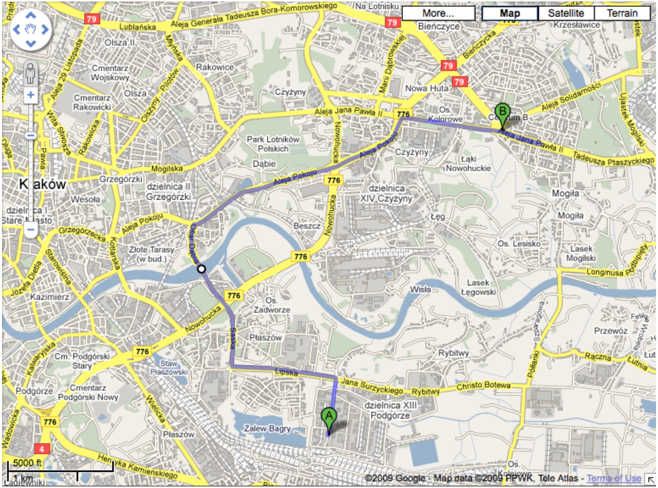
Robustness factor as an optimality criterion



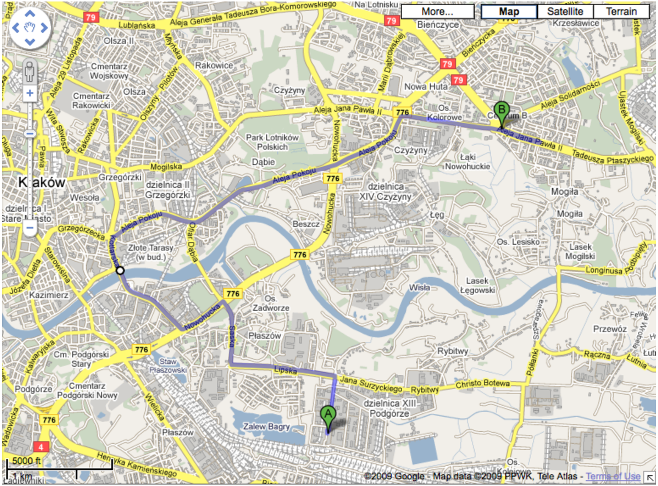
Robustness factor as an optimality criterion



Robustness factor as an optimality criterion



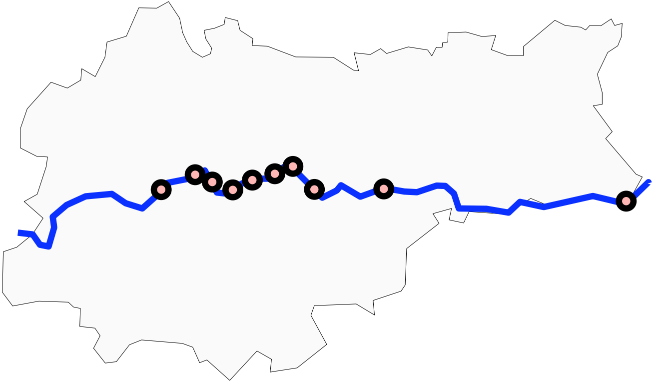
Robustness factor as an optimality criterion



What we mean by *granular*

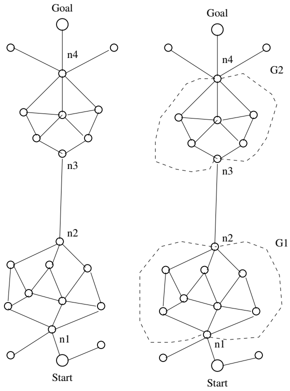
- We use granularity to introduce map abstraction, thus limiting the potential risk of combinatorial explosion.
- The concept of granules is based on the observation that each route in the city (apart from very short ones) leads through a very limited number of *landmarks*, such as bridges.
- The formal notion of granules is introduced in the theory of *granular sets* and *granular relations*.

Krakï£¡w: two natural granules



Granularity: a simple example

In this example, an upper-level plan can be as follows:
Start-n1,n1.G1.n2,
n2-n3,n3.G2.n4,n4-*Goal*



Granule definition

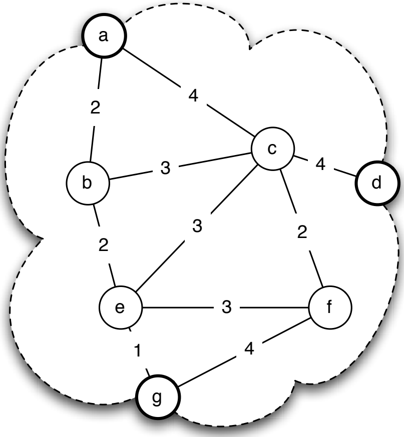
- Granules form a *hierarchical* structure.
- High-level (abstract) granules are easily determined by rivers, railways, highways, etc.
- Currently, granules are introduced manually or semiautomatically. Automatic granulation procedures are being researched.
- One idea consists in determining the junction density on the map plane and using global and local minima for granule boundaries on successive levels.
- Low-level (most detailed) granules are called *elementary granules*.
- Hierarchical granulation can be likened to DFD decomposition.

Example: a real-life elementary granule

Granular planning

- Define reasonably-sized granules with few inputs and outputs, but with numerous alternative inner paths.
- Compile elementary granule data to knowledge (robust plans) to allow future instantaneous granule traversal.
- Using granule inputs, outputs and links at level n, calculate robust plans for level $n+1$.

Another example of a granule



Knowledge compilation: elementary granule

- As elementary granules are reasonably small and compilation takes place off-line, simple graph search algorithms can be used.
- Perform search for all outputs.
- Compile the resulting paths into decision tables, such as the one presented.

n_c	n_{out}	n_{next}	Φ
d	a	c	8
c	a	a	4
c	a	b	5
c	a	e	7
e	a	b	4
e	a	c	7
b	a	a	2
⋮	⋮	⋮	⋮

Knowledge compilation: upper-level granules

- Upper and lower boundaries for input/output combinations are treated as variable-weight edges in the upper-level graph.
- The compilation procedure is analogous to the one used for elementary granules.

n_{in}	n_{out}	Φ_{min}	Φ_{max}	count
a	d	8	13	4
a	g	6	11	3
d	g	9	16	6
⋮	⋮	⋮	⋮	⋮

Information presented to the driver

- Alternative maneuvers
- Their optimality
- Their robustness
- Their extent: does this choice affect the local route, or does it mean choosing another upper-level plan?

Conclusion

- The classical dynamic planning approach is difficult to implement when there is human/machine interaction.
- Granular robust planning defines knowledge *compilation* and *representation* aimed at intelligent and error-prone execution of plans.
- Robust planning is useful when:
  - there is uncertainty whether the initial plan can be executed,
  - there are numerous possibilities of plans with similar costs.
- Granular planning is useful when:
  - the density of the environment varies,
  - there are landmark points which can be used as granule input/outputs.

Bibliography and related work

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
Introduction to Algorithms, Second Edition.
McGraw-Hill Science/Engineering/Math, July 2001.

Stuart J. Russell and Peter Norvig.
Artificial Intelligence: A Modern Approach.
Pearson Education, 2003.

Malik Ghallab, Dana Nau, and Paolo Traverso.
Automated Planning: Theory & Practice.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes.
Fast routing in road networks using transit nodes.
Science, 316(5824):566, April 2007.

Martin Tomko and Stephan Winter.
Considerations for efficient communication of route directions.
2008.

Daniel Delling and Dorothea Wagner.
Landmark-based routing in dynamic graphs.
In *In: 6th Workshop on Experimental Algorithms*. Springer, 2007.

The End

Thank you for your kind attention!

Powered by $\LaTeX$