Chapter 105
Localization on Discrete Grid Graphs

Anna Gorbenko, Vladimir Popov and Andrey Sheka

Abstract Grid graphs are popular testbeds for planning with incomplete information. In particular, it is studied a fundamental planning problem, localization, to investigate whether gridworlds make good testbeds for planning with incomplete information. It is found empirically that greedy planning methods that interleave planning and plan execution can localize robots very quickly on random gridworlds or mazes. Thus, they may not provide adequately challenging testbeds. On the other hand, it is showed that finding localization plans that are within a log factor of optimal is NP-hard. Thus there are instances of gridworlds on which all greedy planning methods perform very poorly. These theoretical results help empirical researchers to select appropriate planning methods for planning with incomplete information as well as testbeds to demonstrate them. However, for practical application of difficult instances we need a method for their fast decision. In this paper we describe an approach to solve localization problem. This approach is based on constructing a logical model for the problem.

Keywords Localization · Grid graph · Genetic algorithm

A. Gorbenko (✉) · V. Popov · A. Sheka
Ural State University, Ekaterinburg, 620083, Russia
e-mail: gorbenko.aa@gmail.com

V. Popov
e-mail: Vladimir.Popov@usu.ru

A. Sheka
e-mail: andrey.sheka@gmail.com

X. He et al. (eds.), *Computer, Informatics, Cybernetics and Applications*,
Lecture Notes in Electrical Engineering 107, DOI: 10.1007/978-94-007-1839-5_105,

105.1 Introduction

A testbed is a platform for experimentation of development projects. Testbeds allow for rigorous, transparent, and replicable testing of scientific theories, computational tools, and new technologies. A typical testbed could include software, hardware, and networking components. Testbeds are widely used for planning. In this context, testbeds are planning domains that allow researchers to evaluate their planning methods, communicate performance results of their methods to others, interpret published performance results of others more easily, and compare their methods against these performance results [1]. Planning researchers have studied in detail the properties of their testbeds for planning with complete information, such as blocksworlds and sliding tile puzzles. Examples of such experimental and theoretical studies include [2–5]. In recent years, planning researchers have become interested in planning with incomplete information (see [6–10]). This is an important research direction because, in the real world, complete information is often not available.

Testbeds should be easy to describe, but they should also provide a wide enough variety to mimic real domains. In particular, testbeds must include cases that are not too easy to solve because otherwise planning methods would appear to be more efficient than they actually are in some of the domains of interest.

Gridworlds are popular testbeds for planning with incomplete information. In [6] studied a fundamental planning problem, localization, to investigate whether gridworlds make good testbeds for planning with incomplete information. In [6] found empirically that greedy planning methods that interleave planning and plan execution can localize robots very quickly on random gridworlds or mazes. Thus, they may not provide adequately challenging testbeds. On the other hand, in [6] showed that finding localization plans that are within a log factor of optimal is NP-hard. Thus there are instances of gridworlds on which all greedy planning methods perform very poorly. In [6] showed how to construct them. These theoretical results help empirical researchers to select appropriate planning methods for planning with incomplete information as well as testbeds to demonstrate them. However, for practical application of difficult instances we need a method for their fast decision. In this paper we describe an approach to solve localization problem. This approach is based on constructing a logical model for the problem.

105.2 Grid Graphs Planning Tasks

We study localization tasks in grid graphs. Localization is a prototypical planning task with incomplete information. Before performing this task the robot knows a map of the gridworld but does not know its start cell. Evidently, the robot may need to localize prior to performing many other tasks.

The sensors onboard the robot tell it in every cell whether the cells immediately adjacent to it in the four compass directions (north, east, south, west) are traversable. The border of the grid graph is untraversable and observed as such. The robot can then move one cell to the north, east, south, or west, unless that cell is outside of the grid graph or untraversable. In the latter case the robot remains in its current cell. We assume a point robot with accurate sensing, perfect actuation, and knowledge of its orientation from an onboard compass.

The robot is localized if it knows its current cell. A deterministic localization plan specifies the movement to execute based on all previous movements and observations. A localization plan is valid if and only if there is no matter which cell the robot is started in; it eventually prints out its current cell or correctly determines that localization is impossible. The objective of planning then is to determine a valid deterministic localization plan that minimizes the number of movements for the worst possible start cell. We first calculate the number of movements for each possible start cell. The worst-case performance is then the maximum of these values.

In the decision version the valid deterministic localization plan problem can be formulated as following.

We can suppose that a grid graph is given by a matrix (g [i, j]) where m and n are dimensions of G, g [i, j] = 1 or g [i, j] = 0, and g [i, j] = 1 if and only if the cell with coordinates i and j belongs to G.

The Valid Deterministic Localization Plan Problem (VDLPP):

Instance: A grid graph G, a natural number K.

Question: Is there a valid deterministic localization plan such that the worst-case performance of this plan does not exceed K?

In [6] showed that VDLPP is NP-complete.

105.3 Logical Model

The propositional satisfiability problem (SAT) is a core problem in mathematical logic and computing theory. Propositional satisfiability is the problem of determining if the variables of a given Boolean function can be assigned in such a way as to make the formula evaluate to true. SAT was the first known NP-complete problem, as proved by Stephen Cook in 1971. Until that time, the concept of an NP-complete problem did not even exist. Considered also different variants of the satisfiability problem.

Encoding problems as Boolean satisfiability and solving them with very efficient satisfiability algorithms has recently caused considerable interest. In particular, local search algorithms have given impressive results on many problems. For example, there are several ways of SAT-encoding constraint satisfaction, clique, planning, maximum cut, Hamiltonian cycle, vertex cover, maximum independent set, and colouring problems. There is a well known site on which posted solvers for SAT [11]. These solvers are divided into two main classes: stochastic local

search algorithms and algorithms improved exhaustive search. All solvers allow the conventional format for recording DIMACS Boolean function in conjunctive normal form and solve the corresponding problem [12]. In addition to the solvers the site also represented a large set of test problems in the format of DIMACS. This set includes a randomly generated problem of SAT. Of course, these algorithms require exponential time at worst. But they can relatively quick receive solutions for many Boolean functions. Therefore, it is natural to use a reduction to different variants of the satisfiability problem to solve computational hard problems.

Note that if we have a valid deterministic localization plan such that the worst-case performance of this plan does not exceed K then we have some sequence of instructions. We can assume that these instructions are defined as follows. "If x [north] = a, x [east] = b, x [south] = c, and x [west] = d, then M", where x [north], x [east], x [south], and x [west] are Boolean variables whose truth means that corresponding cells are vertices of grid graph, a, b, c, d are Boolean constants, and M is the direction. Not very difficult to construct a Boolean function which is true if and only if there is a valid deterministic localization plan consisting of K actions. This function can be constructed so that, using a set of values on which the function is true, we automatically obtain a sequence of instructions. This function gives us not only a SAT-encoding of VDLPP but also a way to obtain a sequence of instructions.

105.4 Robot Experimental Setup

Assumptions under which we consider VDLPP are simplifying but sufficiently close to reality to enable one to use the resulting planning methods on real robots. Greedy localization, for example, has been used on Nomad 150 mobile robots. The success rate of moving was at least 99.57%, and the success rate of sensing was at least 99.38% (see [10, 13]). These large success rates enable one to ignore actuator and sensor noise, especially since the rare failures are usually quickly noticed when the number of possible locations drops to zero, in which case the robot simply reinitializes its belief state to all possible locations and then continues to use the localization algorithm unchanged.

Among practical applications of VDLPP, we can note the air landing of a robot and a variety of tasks for indoor autonomous service robots. Note that the need of localization for indoor service robots occurs either due to temporary equipment failure or because of deliberate refusal to memorize the route. In the first case conditions of VDLPP are not quite consistent with the real situation. In particular, in many cases, we can assume that the robot is close to some known point. In some other cases, we can assume that the robot motion through a fixed direction. These assumptions greatly simplify the solution of the problem of localization.

For our experiments, we use two mobile robots (see Fig. 105.1 and 105.2). Typical failures for our robots are following.

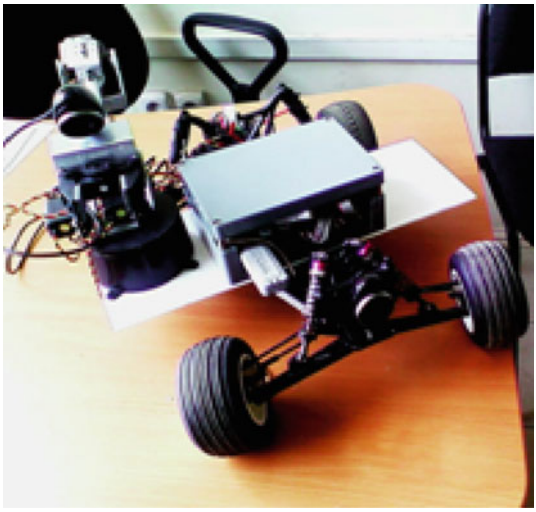

Fig. 105.1 Design of this robot based on the well-known RC cars. From RC-CAR AT-10ES Thunder Tiger [14] we use only the four wheel chassis, the high torque DC-MOTOR and a steering servo. The DC-MOTOR drives the chassis and a steering servo controls the direction. The electronic system based on SSC-32 microcontroller. Onboard computer based on a motherboard with x86 compatible processor AMD Geode LX600 for embedded systems

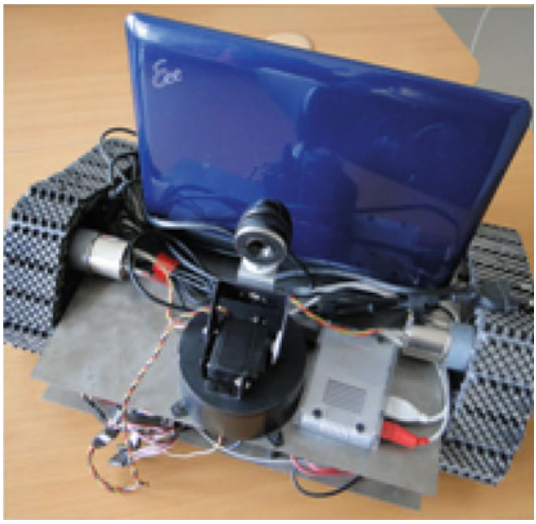

Fig. 105.2 Design of this robot based on the well-known Johnny 5 Robot [15]. By utilizing heavy duty polypropylene and rubber tracks with durable ABS molded sprockets the robot has excellent traction. It includes two 12vdc 50:1 gear head motors and the Sabertooth 2 × 5 R/C motor controller. Onboard computer of this robot is Asus Eee PC 1000HE

A collision with an undetectable object. When there is a collision with an undetectable object robot either stops or changes direction.

A tire started to deflate. In this case we get a predictable distortion of the trajectory of motion.

Rotation of a track is stopped. In this case we get the rotation along a known trajectory.

When the power of a servo motor is turned off, it begins to self-motion. It can give only a single change of the angle.

If you lose the connection with external navigation module motors continue to do the previous commands until the connection is restored. In this case, we know the trajectory of motion. Only the time of movement is unknown.

Loss of the next image in the case of using visual navigation. In this case, it is known that the robot is in a small neighborhood of the known point.

In each of these cases, we can consider instead VDLPP some more simple problem. On the other hand VDLPP of considerable interest in the case when the robot need not memorize the path.

Both of our robots have processors with relatively low performance. The requirement of localization while moving substantially reduces the speed of moving robots. Therefore, in solving many problems, we use visual topological navigation (see [16–18]).

Vision-based navigation systems may use either topological or metrical maps. In topological maps, only places such as rooms and their relations are learned and recognized, whereas in metrical maps, the precise positions of environment features and of the robot are estimated. In realistic scenarios for entertainment robotics, the robot can fall or be blocked in places where sensors will have difficulty to find useful information. In these situations, a metrical approach, that usually requires a continuous tracking of features, will probably fail, whereas a topological approach, able to recognize the rooms and guide the robot between them is more adapted. Moreover, topological approaches may be purely appearance based, thus avoiding the need for camera calibration.

In practice often used simple topological algorithms that allow only follow the target and avoid obstacles. Such algorithms provide very high performance. Note that usually robots do not have enough memory to store full visual series. Therefore, usually after the task the robot needs to solve the problem of localization. For example, the robot is moved directly by the user from one place to another, the robot is moved directly by another robot, robot uses skittles as landmarks and other robot rearranges skittles, etc.

Note that for a relatively small testbeds, we can apply brute force. However, this method is not suitable even for indoor laboratory testing. We create a generator of special hard (see [6]) and natural instances for VDLPP. We use algorithms from [11]. Also we design our own genetic algorithm for SAT which based on algorithms from [11]. We use heterogeneous cluster based on three clusters (Cluster USU, Linux, 8 calculation nodes, Intel Pentium IV 2.40 GHz processors; umt, Linux, 256 calculation nodes, Xeon 3.00 GHz processors; um64, Linux, 124 calculation nodes, AMD Opteron 2.6 GHz bi-processors) [19]. Each test was run on a cluster of at least 100 nodes. The maximum solution time was 14 h. The average time to find a solution was 12.3 min. The best time was 52 s. Note that the calculation exceeds 1 h is quite rare.

It is easy to see that solver for VDLPP give us theoretically a good solver for the valid deterministic localization plan problem whose use gives only a linear slowdown. However, in practice it may be slowing down a hundred times or more. Thus sometimes we can use a supercomputer to find an exact solution. In this case we can apply this solution on the robot. However, in many cases we are forced to

rely on heuristic methods of solution. Nevertheless, using a logical model provides us a good tool for verifying heuristic solutions. In addition, we can expect a significant acceleration of the process by using a more powerful supercomputer. Also we can use a logical model for VDLPP to create a training set for supervised learning of some intelligent algorithm for the valid deterministic localization plan problem.

105.5 Summary

In this paper we have presented an approach to solve localization problem. This approach is based on constructing a logical model for the problem. We also examined practical aspects of using a logical model for the solution of the valid deterministic localization plan problem and for the generation of testbeds for this problem.

References

1. Hanks S, Pollack M, Cohen P (1993) AI Mag 14:17
2. Gupta N, Nau D (1992) Artif Intell 56:223
3. Koenig S, Simmons R (1996) In: Proceedings of the national conference on artificial intelligence, p 279
4. Reinefeld A (1993) In: Proceedings of the international joint conference on artificial intelligence, p 248
5. Slaney J, Thiebaux S (1996) In: Proceedings of the national conference on artificial intelligence planning
6. Tovey C, Koenig S (2000) In: Proceedings of the AAAI conference on artificial intelligence, p 819
7. Koenig S, Likhachev M (2005) Fast replanning for navigation in unknown terrain. IEEE Trans Robot 21:354
8. Koenig S, Smirnov Y, Tovey C (2003) Performance bounds for planning in unknown terrain. J Artif Intell 147:253
9. Mudgal A, Tovey C, Koenig S (2004) In: Proceedings of the international symposium on artificial intelligence and mathematics
10. Tovey C, Koenig S (2010) IEEE Trans Robot 26:320
11. Information on http://people.cs.ubc.ca/~hoos/SATLIB/index-ubc.html
12. Information on http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/satformat.ps
13. Nourbakhsh I (1996) In: Proceedings of the AAAI-96 spring symposium on planning with incomplete information for robot problems, p 86
14. Information on http://www.tiger.com.tw/
15. Information on http://www.lynxmotion.com/c-103-johnny-5.aspx
16. Filliat D (2008) In: IEEE/RSJ international conference on intelligent robots and systems. IEEE computer society press, New York, p 248
17. Park IP, Kender JR (1995) Topological direction-giving and visual navigation in large environments. Artif Intell 78:355

18. Santos-Victor J, Vassallo R, Schneebeli H (1999) In: Christensen HI (ed) Proceedings of the first international conference on computer vision systems. Springer, London, UK, p 21
19. Information on http://parallel.uran.ru/mvc_now/hardware/supercomp.htm