Physical Path Planning Using the GNATs

Keith J. O'Hara, Victor L. Bigio, Eric R. Dodson, Arya J. Irani, Daniel B. Walker, and Tucker R. Balch
The BORG Lab
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: {kjohara, vbigio, edodson, arya, danielbw, tucker}@cc.gatech.edu

***Abstract*— We continue our investigation into the application of pervasive, embedded networks to support multi-robot tasks. In this work we use a new a hardware platform, the GNATs, to aid in path planning. We have implemented a physical path planning algorithm on the GNATs previously studied in simulation. A distributed version of the wavefront path planning algorithm is used to propagate paths throughout the network, thereby planning a path in the real world. This creates a graph of traversable paths that are nearly optimal in a dynamic environment.**

I. Introduction

We continue our investigation into the application of pervasive, embedded networks to support multi-robot tasks. We envision a pervasive, embedded environment to be one in which tens to thousands of inexpensive, low power, resource-constrained communication devices have been introduced. The nodes aid robots in completing their tasks, primarily by providing communication and coordination services. We feel this heterogeneous system of embedded devices and mobile robots puts a natural constraint on the design space of multi-robot systems. The embedded network serves as a pervasive communication, computation, and coordination fabric, while the mobile robots provide sensing and actuation.

O'Hara and Balch investigated in [9] the "use of embedded networks to facilitate mobile robot activities." We have implemented a hardware platform to realize these types of applications. The hardware devices, the GNATs (**G**eorgia Tech **N**etwork for **A**utonomous **T**ask**s**) are low cost and highly configurable. Two are pictured in Figure 1(a). The devices consist of four infrared (IR) emitters, four IR receivers, two visible light LEDs, a button, a Microchip PIC16F87 microcontroller, and a 3V battery. The device costs about $30 to build.

In this paper we show that these highly affordable nodes, in sufficient quantities, can be placed in an environment and are able to display navigation information that can aid robots traversing to a goal location, such as a food cache in a foraging problem. Traditional path planning algorithms for dynamic environments typically require a single mobile robot to build a map, locate itself in that map, update the map as the environment changes, and then finally plan over the map. Instead, we use an embedded network distributed throughout the environment to approximate the path-planning space and use the network to compute the path in a distributed fashion. The network can plan paths for multiple robots going to the same or different destinations without the network nodes having global knowledge or localization capabilities. We show that network can plan paths in a distributed manner and reconfigure (replan) in dynamic environments.

II. Previous Work

In previous work [8] we developed and analyzed two different techniques for distributed path planning when the environment is dynamic and paths are destroyed and created. The algorithm essentially works as a distributed variant of the popular wave-front path planning algorithm, or a breadth-first search from the goal, propagating paths from the goal location. The embedded nodes make up the vertices of the path planning graph, and the network connections between them are the edges of the graph. Mobile robots can then use reactive navigation to traverse the graph by visiting the vertices (i.e. the embedded nodes) to the goal. In order to respond to changes in the environment this graph is maintained as edges are added and removed.

Additionally, in previous simulation studies [9] we showed how the embedded network could support effective cooperative foraging by coordinating coverage patterns and by providing nearly optimal path planning without the network nodes having global knowledge or localization capabilities. Quantitative results illustrated the sensitivity of the approach to different network sizes, environmental complexities, and deployment configuration.

Payton et al. [10] present an approach for large scale multi-robot control referred to as "Pheromone Robotics" inspired by biology. They use a system based on virtual pheromones, by which a homogeneous team of mobile robots use short-range communication to accomplish cooperative sensing and navigation. In Payton's work "virtual pheromones" are communicated over an ad hoc network to neighboring robots. In contrast, in our approach information is distributed by the relatively static, embedded, nodes scattered throughout the environment.

Both Batalin et al. [3] and Li et al. [7] have developed similar approaches using heterogeneous teams composed of mobile nodes and an embedded network. The network of embedded nodes, creates a "Navigation field" [3], which mobile nodes can use to find the their way around. They differ in how they compute this navigation field. Batalin et al. use Distributed Value Iteration [3]. In their approach,

the embedded nodes use estimated transition probabilities between nodes to compute the best direction to suggest to a mobile robot for moving between a start and goal node. These transition probabilities are established during deployment and both the robots and sensor nodes have synchronized direction sensors (e.g. digital compass). Our approach does not require the nodes to store transition probabilities, instead we rely on the communication network to establish the navigation paths. Also, in our approach the mobile robots only need a local sense of direction in order to move toward the correct embedded node. Neither the robots or the embedded nodes need any shared sense of direction.

Li et al. are able to generate an artificial potential field for navigation based on the obstacles and goals sensed by the network. [7] This potential field is guaranteed to deliver the mobile node to the goal location via an danger-free (obstacle-free) path. The field is created by the embedded nodes propagating goal-ness or danger to neighboring nodes. In our approach the embedded nodes do not have sensors, this capability is provided by the mobile nodes, and thus can not sense obstacles directly. We assume the obstacles are sensed indirectly by the resulting communication topology. These three approaches, as well as ours, use distributed dynamic programming [4] to create the navigation field.

Both Batalin et al. and Li et al. used the Motes hardware platform for their physical experimentation. The Motes are a popular platform for sensor networks research. In contrast to the GNATs, Motes have on the order of 32 times more program memory, 10 times more RAM, and for the Berkeley and Harvard implementations, the Atmel ATMEGA128L processor is running at twice the frequency of the GNATs PIC processor. A wide variety of sensor devices may also be connected to the Mote hardware platform giving the Mote the ability to be very aware of its surroundings, whereas our GNATs are sensor-less network devices. The greater processing power and sensing ability gives the Motes a much higher price than the GNATs. The simplicity of the GNATs due to their specialization for mobile robot applications allows us to build them for a price an order of magnitude less than the Motes. This allows us to experiment with very large-scale systems.

III. Approach

Our system is composed of mobile robots with sensors and actuators supported by an embedded immobile network of nodes without environmental sensors. We assume the embedded network nodes (which we have implemented in the GNATs platform) have the following capabilities:

- **Limited computation and memory,** on the order of a PIC microprocessor operating at 4 MHz with a 4096-instruction program eeprom, 256 bytes of data eeprom, and 368 bytes of RAM.
- **Short range communication** with adjacent nodes up to 4 meters away.
- **Communication is blocked by navigation obstacles.**

We assume the robots and embedded nodes communicate using a short-range medium that is occluded by the same objects that occlude navigation (e.g. walls). Line of sight between nodes implies open space for navigation. The mobile robots in our system are somewhat more capable than the embedded nodes. We assume they support:

- Communication with embedded nodes;
- Relative bearing estimation to nearby embedded nodes;
- Local obstacle and attractor sensing;

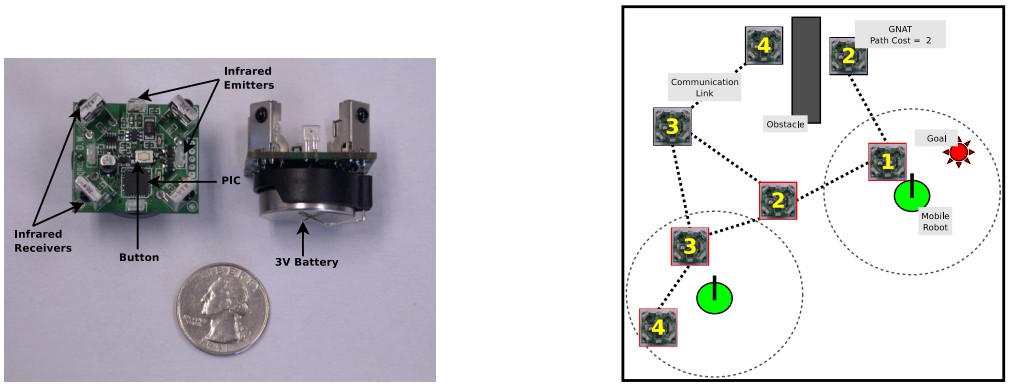
Also, we assume that the mobile robots have at least the same communication range as the embedded nodes. The robots need to roughly estimate bearing to the embedded nodes in order to move toward the desired node, but the robots do not need to estimate range. Significantly, there are a few assumptions we do not make. In particular, **we do not assume localization or mapping capabilities** on the part of the robots or the embedded nodes. No mobile robots or embedded nodes are expected to perform localization or mapping. Furthermore **we do not assume the environment is static**. Obstacles to navigation can appear and disappear since we expect the network to automatically adapt to dynamic conditions.

In this work we do not address the deployment of the embedded nodes. We assume they have already been placed in the environment, and sufficiently cover the space, but their positions are unknown and the uniformity of their placement can vary. Consider the following two scenarios. First, the network is already part of an existing infrastructure–for example, an embedded network in a building, or a sensor network in a forest. Second, the network did not exist before the robots began the mission, but the robots have already installed the network as supporting infrastructure. Motivated by these two scenarios we believe we can separate the deployment and utilization of embedded networks into two different problems. In this work we investigate the latter.

A. The Distributed Path Planning Algorithm

The embedded network creates a navigation network for guiding, or routing, mobile robots in various tasks. We use the network in this work strictly for path planning. In general, several navigation networks can be present in the network simultaneously–for instance, a coverage navigation network may also be present. A mobile robot can then follow the navigation network corresponding to its current goal. An illustration of a simple navigation network is illustrated in Figure 1(b). We are able to use navigation networks to complete distributed path-planning in dynamic environments without mapping or localization.

We follow Payton's virtual pheromone technique [10] in assuming the communication paths are similar to the navigation paths and use this to propagate navigation information. By using a short-range communication medium that is occluded by obstacles to navigation, the communication paths carve out free-space. As pointed out by Payton [10] and Li [7], this results in a kind of distributed physical path-planning. To create a navigation network for a particular goal we use a distributed dynamic programming approach; specifically, we apply the distributed Bellman-Ford algorithm. The Bellman-Ford equation [5] for finding the shortest path from i to j



(a) (b)

Fig. 1. (a) Two GNATs. (b) An illustration showing how the network guides a mobile robot to a goal location.

is:

$$D(i,j) = \min_{k \in neighbors} d(i,k) + D(k,j)$$

Where $D(i,j)$ is the path cost from i to j, and $d(i,k)$ is the distance between i and k. It can be used to find the shortest path to a destination from all nodes. The distributed version of Bellman-Ford was created for network routing protocols [6]. In the distributed network routing version, neighbors share their path costs and the distance between nodes is usually measured in hops. We use distributed Bellman-Ford to effectively create a directed graph of shortest paths from every node to the goal – this directed graph is the navigation network. The embedded network can be thought of as "routing" the mobile robots to their destination. However, note that the embedded nodes do not know the global, or local, position of their neighbors, so they are not directing the robot in any direction. Instead, the mobile robot greedily approaches the lowest-valued node currently in its communication range. As it closes in on the node it will come within communication range of that node's parent. The robot continues this until it comes within sensing distance of the goal.

B. Simulation Results

We implemented this system in the TeamBots multi-robot simulation environment. The control systems were encoded in the Clay behavioral architecture [2] and used motor schemas [1] for the local navigational tasks of moving toward the embedded nodes and avoiding obstacles. In all the experiments we used 2 mobile robots and a group of 16 attractors to represent the goal location. The first robot was placed near the attractors in the center of the environment at the start of the experiment. This robot's purpose was to insert the goal information into the network. This scenario mimics a situation where a robot needs to recruit other robots to a particular location. The second robot was placed in the corner of the environment away from the goal and was responsible for navigating to the goal location. We used 180 embedded nodes. We placed the embedded nodes uniformly across the space, then added error to each node's position by some random amount, the average distance from original position was .5 meters. The robots and embedded nodes had limited sensing and communication ranges of 4 meters that were occluded by obstacles. The environment was $36x36m^2$.

A sequence of screenshots of a simulation using the distributed path planning algorithm is shown in Figure 2. The screenshots show a tree of shortest paths to the goal. The first screenshot shows the initial plan, and the second shows the plan after a door has closed and the network has reconfigured.

C. Hardware Description

We use a group of GNATs to create a navigation network for a particular goal using a distributed dynamic programming approach; specifically, we use them to implement the distributed Bellman-Ford algorithm as described above.

As seen in Figure 1(a), the GNATs are made up of these key components:

- Microchip PIC16F87 microcontroller
- 4 IR emitters
- 4 IR receivers
- 2 visible light LEDs
- reset/input button
- programming port (not shown in figure)
- 3V battery

The PIC microcontroller has a programmable internal oscillator with a frequency range of 125kHz to 8MHz. It has 4K of programmable FLASH program memory and 368 bytes of data RAM. There is also an additional 256 bytes of data EEPROM. The IR emitters have a maximum rating of 100mA and with this we have measured reliable communication over ranges of up to 4m.

D. Physical Path Planning with Real Hardware

In meeting with our design criteria, the GNATs can only communicate with each other via their 4 IR Emitters and 4 IR Receivers. We have developed the capability to send a packet

of data and receive a packet of data, but since we are using a single channel, they cannot transmit and receive at the same time.

In order to implement the functions described above, we established a network of GNATs that can propagate a message containing the distance-to-goal from a goal location to all other GNATs in the network. Each node greedily chooses the lowest distance-to-goal from its neighboring GNATs, and retransmits an incremented value. For our experiments, one gnat is assigned to be the goal (by a push of the button). The goal is a node that transmits a message with a distance-to-goal of 0. So that the network will dynamically react to changes in the environment, the goal periodically transmits a message, and each node then recalculates its distance-to-goal. This is different than the approach in [8] where the nodes monitor their neighbors using heartbeats to maintain the path-planning graph in dynamic environments. The approach presented there limited the amount of replanning needed when the environment changed. Instead, the approach presented here is less efficient and results in globally replanning, but is simpler to implement with the limited communication capability we have on the real hardware.

Given the nature of the communication medium, and the lack of hierarchy and coordination among GNATs, we faced several communication and networking challenges. Since there is only a single communication channel, all transmissions are broadcasts, and any node within transmission range can hear the message. A node can only receive one transmission at a time, and there is no universal timing or coordination to tell GNATs when to send a packet. If two neighboring GNATs transmit simultaneously, then neither will receive the message from the other. Finally, if two or more of a given node's neighbors transmit simultaneously, the node will receive neither message.

As a simple first attempt to implement communication we decided to use a random backoff scheme for delaying data transmission. Each GNAT will listen for a packet for a random time. If there has been no message received during the wait, then the node broadcasts its message. Otherwise the node will choose a new random backoff time and wait again.

Using random backoff we reduced the likelihood of collisions but we also introduced the possibility that an older message from the goal will arrive at the same time that a current message does. In order to avoid the confusion this creates, we have implemented a generation counter. Each time the goal transmits, it increases the generation. The message that is transported thus includes the generation and the hop-count (distance-to-goal).

Even with the generation counter, the possibility still exists that a node will receive a message with a higher hop-count before it receives a message with a lower hop-count. Though the node greedily chooses the lower hop-count, we want to maintain a stable hop-count value unless the environment actually changes. To this end we have programmed the GNATs to (greedily) accept a lower hop-count, but to only accept a higher hop-count if this is the lowest value it receives in two successive transmissions. Therefore, a node is quick to reduce its hop-count, reflecting that it is closer to the goal, but slow to increase its hop-count to avoid flip-flopping.

```
structure PATHMSG
  int generation
  int hopCount
on event MSGRECEIVED(PathMsg recv)
  if recv.generation < curr.generation then
    if recv.hopCount < curr.hopCount or
    curr.hopCount = lastHopCount then
      UPDATEHOPCOUNT(recv)
    end if
    curr.lastHopCount ← recv.hopCount
  else if recv.generation = generation and
  recv.hopCount < curr.hopCount then
    UPDATEHOPCOUNT(recv)
  end if
procedure UPDATEHOPCOUNT(PathMsg recv)
  curr.hopCount ← recv.hopCount + 1
  curr.lastHopCount ← recv.hopCount
  TRANSMITMSG(curr.generation, curr.hopCount)
```

Fig. 3. GNATs path planning algorithm.

The goal transmits its existence, and regularly sends out a packet with a hop-count of 0 and a new generation. The algorithm is shown in Figure 3.

The communication path from the goal to all other reachable GNATs forms a connected graph in which the GNATs correspond to the vertices and the communication paths between adjacent GNATs correspond to the edges. The hop-count represents the minimum number of edges that need to be traversed to reach the goal. A node, while still connected, will dynamically alter its hop-count to reflect changes in the environment. A component of the graph that becomes disconnected will return to the wait state after a predefined timeout period. When connectivity is restored, these nodes will participate in the navigation network again. When goal distances have been propagated to all the nodes, traversal is a simple matter of using a hill-climbing search algorithm: moving to a node, detecting its hop-count, and determining which of its neighbors to travel to next. This step is repeated until the goal is reached.

IV. EXPERIMENTS

We first used the GNATs path planning algorithm to solve the rat maze shown in Figure 4. 16 GNATs are deployed in an artificial maze with one of the GNATs acting as the goal location. The goal location broadcasts the original goal message once every two seconds. The network is able to compute a nearly optimal path and also replan when paths are obstructed. The paths the network computes are similar to navigation paths, this fact agrees with our earlier assumptions.

The next experiment involved a real world environment rather than the rat maze. As in the rat maze the GNATs were able to compute a navigation path and replan when the environment changed. The environment changed when a door closed in this experiment.

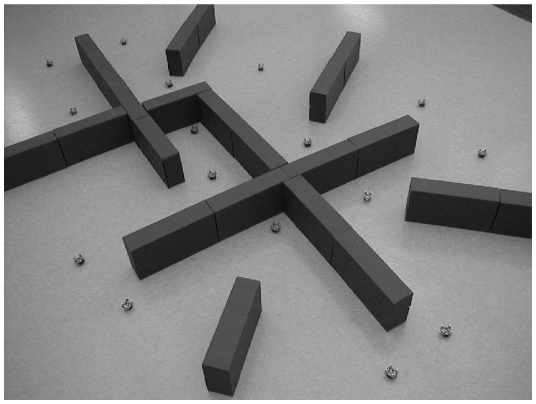


Fig. 4. The GNATs solve a "rat maze".

TABLE I

PROPAGATION TIME FOR PLANNING AND RE-PLANNING IN A DYNAMIC ENVIRONMENT.

Operation	Time
Initial planning for T1	3.5s
Initial planning for T2	5.3s
Re-planning for T1 after closing Door 1	6.8s
Re-planning for T2 after closing Door 1	7.8s
Re-planning for T1 after opening Door 1	4.8s
Re-planning for T2 after opening Door 1	5.6s

In this experiment we deployed 19 GNATs in an office environment as shown in Figure 5. All the GNATs are placed roughly 2-3.5 meters apart. The goal location is labeled with a *G* and it broadcasts the goal message every 2 seconds. The GNATs in their initial wait state are shown as white circles. The nodes where measurement were taken will be referred to as "target" nodes and are labeled as *T1* and *T2*.

Figure 5(b) shows the goal distance propagation in the initial environmental configuration. The optimal path is shown by black GNATs and is directed through Door 1. The GNATs not in the optimal path are represented by white circles. The average time it took for the nodes, *T1* and *T2* to become part of the optimal path took 3.5 seconds, and 5.3 seconds respectively.

To demonstrate the dynamic replanning aspects of the approach we closed door 1, thus invalidating the original path. The time measured from the closing the door to T1 changing its path-cost was on average 6.8 seconds and 7.8 seconds for T2. This is shown in Figure 5(c).

By re-opening Door 1, we force the optimal path back to the configuration in Figure 5(b). The average replanning time in this case is 4.8 seconds to T1 and 5.6 seconds to T2. The replanning times were lower in this case because the algorithm is quicker to take lower hop-counts than higher hop-counts to prevent the flip-flopping described early.

Table I summarizes the the average times taken at T1 and T2 under the previous conditions.

V. CONCLUSIONS AND FUTURE WORK

It has been shown that multiple, computationally limited nodes can compute nearly optimal paths through an environment in a distributed manner. These highly affordable nodes, in sufficient quantities, can be placed in an environment and display navigation information to aid robots traversing to a goal location. We've also shown that the nodes can reconfigure to correct path costs in dynamic environments.

Future work will include using mobile robots to traverse the *navigation network* [9] of GNATs along optimal paths. The GNATs will also be used to coordinate coverage patterns, such as in the foraging problem. The robots will communicate with the GNATs by hosting one on-board as a communication device. Another option is having the GNATs convey information to the robots by simply blinking their visible light or IR LEDs.

In addition, we are investigating ways the network can be used for other types of tasks rather than purely of the navigational variety. For instance, it can be used for complex multi-robot coordination, task allocation, or distributed learning.

The approach also has some notable limitations. For one, the system relies on the assumption that communication paths closely approximate navigation paths. To reduce the error in this approximation we can incorporate the robots' real navigation experiences in computing a path. We believe this approach can be extended to include more capable embedded nodes if the application demands it, but our goal was to show how much can be accomplished with very limited embedded nodes.

Acknowledgment. This work was supported by the National Science Foundation under award #0326396.

REFERENCES

[1] Arkin, R., "Motor schema based mobile robot navigation," International Journal of Robotics Research, vol. 8, pp. 92-112, 1989.

[2] Balch, T., "Clay: Integrating motor schemas and reinforcement learning," Technical Report GITCC-97-11, Georgia Institute of Technology, 1997.

[3] M. Batalin, G. S. Sukhatme, and M. Hattig, "Mobile Robot Navigation using a Sensor Network," In *IEEE International Conference on Robotics and Automation*, pp. 636-642, New Orleans, Louisiana, April 2004.

[4] D. P. Bertsekas, "Distributed dynamic programming," in *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 610-616, 1982.

[5] Bellman, R.E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[6] Ford, L., Fulkerson, D., *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[7] Q. Li, M. DeRosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *9th International Conference on Mobile Computing and Networking*, pp. 313-325, September 2003.

[8] K. O'Hara and T. Balch, "Distributed Path Planning for Robots in Dynamic Environments Using a Pervasive Embedded Network," In *3rd International Conference on Autonomous Agents and Multi-Agent Systems*, July 2004.

[9] K. O'Hara and T. Balch, "Pervasive *Sensor-less* Networks for Cooperative Multi-Robot tasks", in *7th International Symposium on Distributed Autonomous Robotic Systems*, June 2004.

[10] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone Robotics", *Autonomous Robots*, vol. 11, pp. 319-324, 2001.

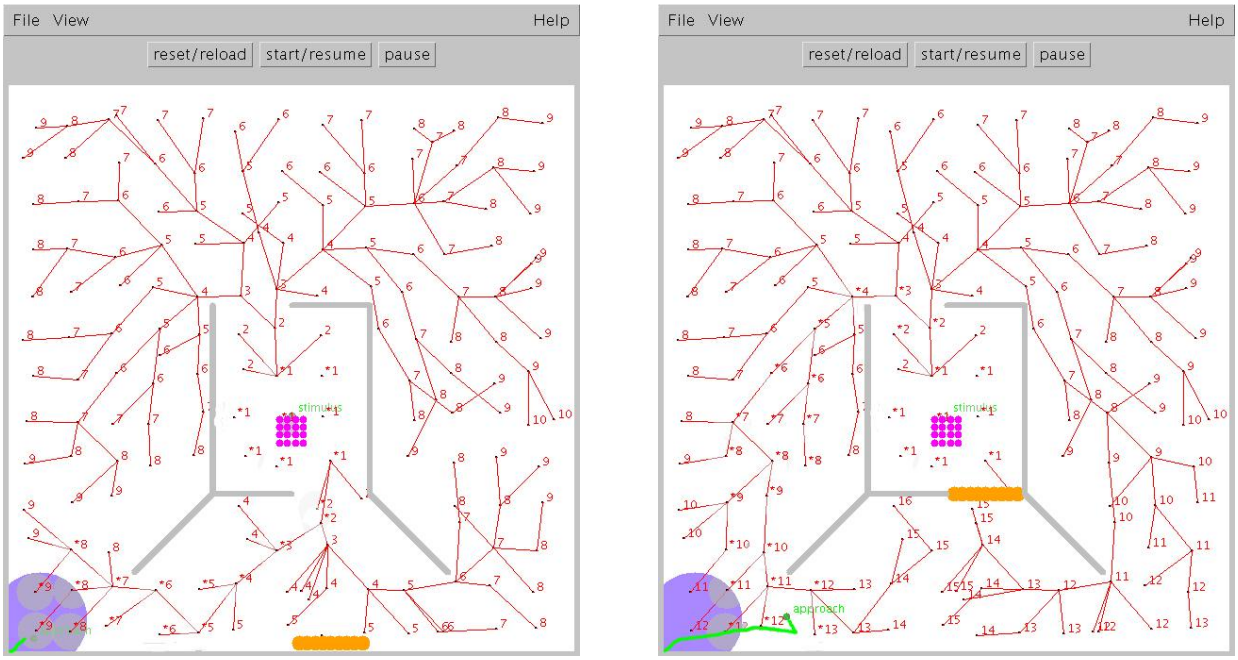
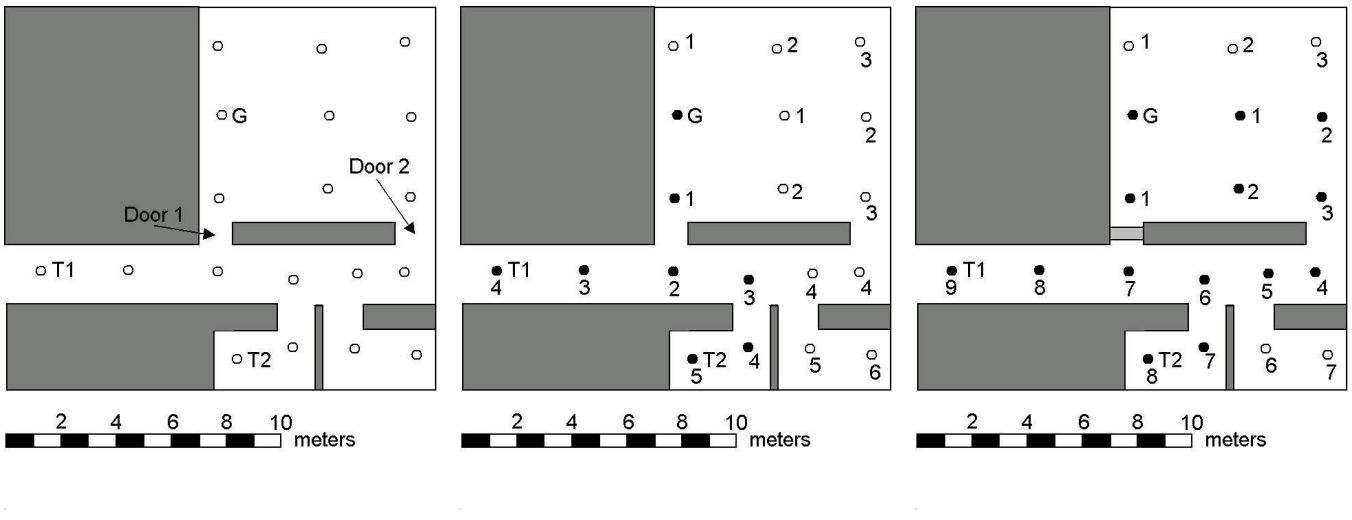


Fig. 2. A sequence of screenshots of a simulation using the distributed path planning algorithm. The screenshots show a tree of shortest paths to the goal. The first screenshot shows the initial plan, and the second shows the plan after a door has closed and the network has reconfigured. The numbers indicate the nodes' distances to the goal, the lines between nodes indicate a parent-child relationship. The goal location is represented by the pink circles in the center, the mobile robots by the green circles, their trails by the green lines, and obstacles by the gray and yellow lines.



(a) Initial planning environment (b) Path planned with no obstacles (c) Dynamically, distributed replanned path

Fig. 5. In (a) all GNATs are shown in the wait state as white circles. With (b) we see the initial goal distance propagation with no obstacles. The numbers next to the GNATs are hop counts. The black circles are GNATs on the optimal path with the other GNATs shown as white circles. (c) shows the dynamically, distributed replanned path after closing Door 1. Again, black circles are GNATs on the optimal path.