routes with nodes at a higher level. We have not yet implemented hierarchies in our direction network.

Several extensions will have to be made for this network to scale-up. As routes get stronger, activity may get out of control if inhibitory links are not added. For example, as one direction out of a node gets activated, this could inhibit activation of other directions. Also, we would like to add a compensatory learning rule so that the network can adapt to large-scale changes in the environment. In compensatory learning, links are weakened as well as strengthened. For example, if the robot has frequently traversed a route from A to B to C and then is forced to begin traversing a route from A to B to D, the link from B to C would weaken as the link from B to D strengthens. Adding a compensatory learning rule to our system would be a minor next step. Finally, we are interested in applying directional spreading activation networks to non-spatial domains. We hope that the focus of activity and the directionality of sequences that this network representation gives us will be helpful in solving other, non-spatial problems.

In summary, we have extended the traditional spreading activation concept as it applies to navigation such that more spatial information can be encoded while still using the original tools. This approach enhances traditional networks by reducing their reliance on external cues for orientation and by further focusing the search for routes. In addition the routes that are returned are automatically encoded with spatial information. Such capabilities are especially critical for mobile robots giving their limited perceptual abilities to date. By making the robot's representations better reflect its experiences and usage of those representations we can shift some of the burden of navigation from the robot's perceptual system to the representations themselves hence bestowing the ability to take advantage of the robot's strengths while minimizing its weaknesses.

References

[Brooks, 1985] Rodney A. Brooks. Visual map making for a mobile robot. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.

[Byrne, 1979] R. W. Byrne. Memory for urban geography. *Quarterly Journal of Experimental Psychology*, 31(1), 1979.

[Chown *et al.*, 1992] Eric Chown, Stephen Kaplan, and David Kortenkamp. Prototypes, location and associative networks (PLAN): Towards a unified theory of cognitive mapping. *The Journal of Cognitive Science*, 1992.

[Kortenkamp *et al.*, 1992a] David Kortenkamp, L. Douglas Baker, and Terry Weymouth. Using gateways to build a route map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1992.

[Kortenkamp *et al.*, 1992b] David Kortenkamp, Terry Weymouth, Eric Chown, and Stephen Kaplan. A scene-based, multi-level representation for mobile robot spatial mapping and navigation. Technical Report CSE-TR-119-92, The University of Michigan, 1992.

[Kuipers and Byun, 1987] Benjamin J. Kuipers and Y.T. Byun. A qualitative approach to robot exploration and map-learning. In *Proceedings IEEE Workshop on Spatial Reasoning and Multi-Sensor Fusion*, 1987.

[Levenick, 1991] James R. Levenick. NAPS: A connectionist implementation of cognitive maps. *Connection Science*, 3(2), 1991.

[Mataric, 1990] Maja K. Mataric. A distributed model for mobile robot environment-learning and navigation. Technical Report AI-TR 1228, MIT Artificial Intelligence Laboratory, 1990.

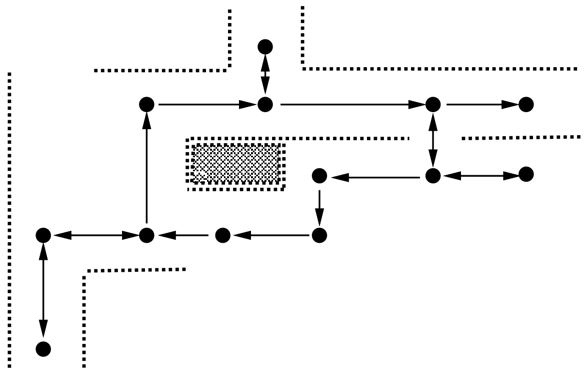

Figure 6: The network of Figure 4 is superimposed on the hallways and rooms of the basement of our laboratory, showing the detection of gateways. The distance from D to H is about 12m.

coalesces at a central sub-goal. Also important is that many of the nodes in the network had zero activity during the search (only active nodes are represented in the figure). The directionality of the search causes whole areas of the network to be excluded from the search space. This reduces confusion in the system.

The three failures in the test cases were caused by a common problem in spreading activation networks. When links between nodes are very strong, a subgoal can be found when activation comes from only a single direction. For example, the start node can activate a tightly connected neighboring node above threshold without any activation from the goal node. A number of these errors can be solved by tediously tweaking the network's learning rate and its subgoal threshold. We didn't have enough experience with the parameters of this network to successfully eliminate some errors. However, even the best networks will fall prey to this problem.

5 Mobile robot issues

We have begun implementing a mobile robot navigation system that will have the directional spreading activation network outlined in this paper at its heart. However, before we could get to the stage of traversing and remembering routes using an actual robot, several issues needed to be addressed. While these issues are not the thrust of this paper, a brief overview of how we are approaching them will allow for the directional spreading activation network to be put into context (see [Kortenkamp *et al.*, 1992b] for more details on the mobile robot implementation).

5.1 Detecting nodes

The first issue that needed to be addressed is where the robot detects nodes. We have defined certain places in the environment as gateways. Gateways mark the transition from one space to another. In an indoor environment, gateways can be entrances to rooms, intersections of hallways or turns. Gateways are detected by using sonar sensors on the robot. The sonar sensors search for openings that are large enough to pass through. Once the robot passes through the opening it calls it a gateway and creates a node. The direction through which the robot passed is the forward direction of the node. Our gateway detection algorithm has been shown to be reliable and repeatable on an actual mobile robot in an actual indoor environment [Kortenkamp *et al.*, 1992a]. Figure 6 superimposes the hallways and rooms in the basement of our laboratory on the network constructed in Figure 4 and shows the gateways between them. All of the gateways shown can be detected by our sonar algorithm.

5.2 Recognizing nodes

While we have a reliable algorithm for detecting gateways, this algorithm, using only sonar sensors, cannot detect a specific gateway. Gateways, in and of themselves are not very distinguishable. To help us distinguish individual gateways we do two things. First, we classify each gateway as a certain type (e.g., intersection, entrance to room, etc.). Second, we use a camera to store visual cues that are available at the gateway. Currently, only vertical edges are used as visual cues. By matching the type of gateway, the vertical edges and by knowing the expected gateway given the robot's previous location, we have shown that the robot will be able to distinguish where it is (see [Kortenkamp *et al.*, 1992b]).

One significant problem that using a real robot presents over the simulation used in this paper is recognizing the same place when it is approached from different directions. If this is not done, there will be no network, only a collection of routes and, thus, no way of combining pieces of several routes to create new ones. We are still examining how to use various geometric and environmental constraints to overcome this problem.

6 Conclusions

The major limitation of the network discussed in this paper is that it is far more complex than an ordinary network, because it is encoding much more information. This limits the lengths of routes that can be extracted. However, as is demonstrated in NAPS, hierarchy can be used to overcome this limitation by representing long

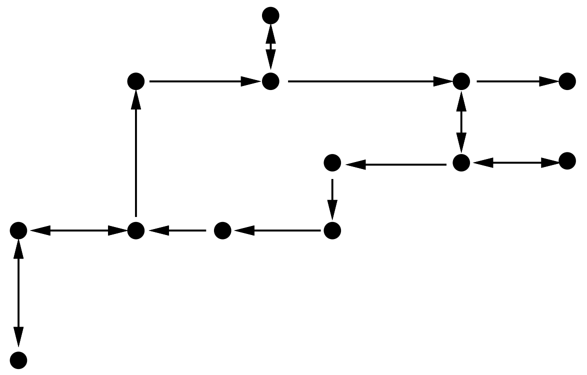

Figure 4: An experimental network with 13 nodes, 32 in-nodes, 32 out-nodes and 33 links.

rather then spreading the activation through the entire network. This can result in less confusion for the robot and more clarity in its route extraction.

A final advantage of directional spreading activation networks is that they can serve as a stepping stone to larger, more spatial representations that represent the overall spatial structure of the environment. Directional networks still only represent directions between neighboring places; directions between distant places would require an additional level of representation. We are actively working on mechanisms and representations for this level, both in a complete theory of human cognitive mapping and a related project on cognitive mapping for mobile robots [Chown *et al.*, 1992].

4 Preliminary results

We have constructed (in simulation) a large, directional network of places and tested its ability to extract routes. The network is pictured in Figure 4. It corresponds to a series of hallways and rooms in the basement of our laboratory. The nodes correspond to places that have been experimentally determined to match those places that our actual robot recognizes as distinctive with its sonar sensors (see the next section for details on this process). Thus, while all results are in simulation since it would have taken too long to conduct extensive experiments with a real robot, the network corresponds to a network that an actual robot would construct.

The network was formed by traversing four routes: A-B-C-D-E-F, F-E-G-H, F-E-G-I-J, and J-I-K-L-M-C-B-A. The first and last routes where traversed four times and the middle two routes where traversed three times. After the network was formed, it was tested using 26 pairs of start and goal nodes and seeing if a correct subgoal node was found. A subgoal is considered correct if it lies on

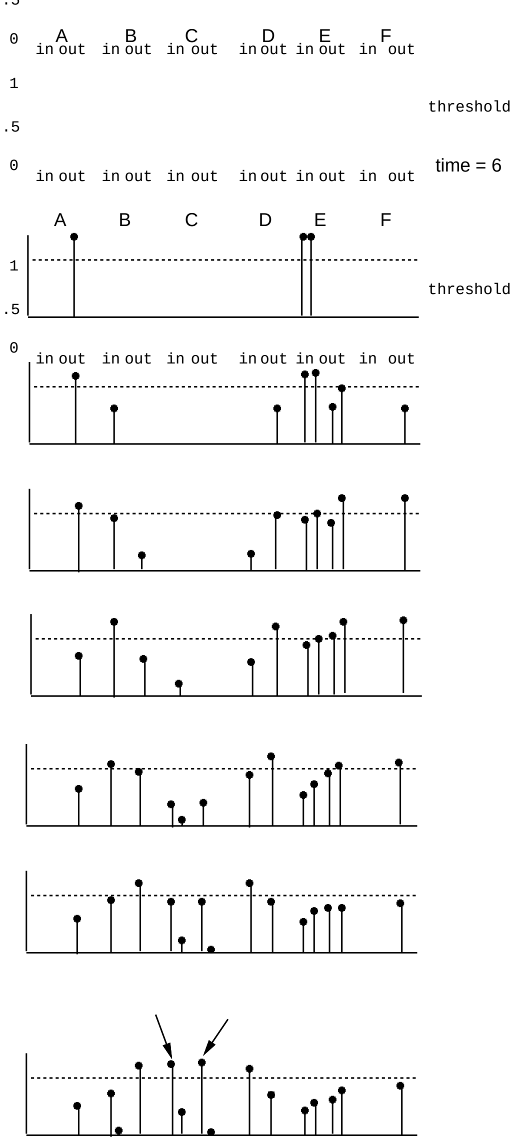

Figure 5: The spread of activation from start node A to goal node E of the network in Figure 4 is shown. Each node has in-nodes and out-nodes, some have several of both as multiple routes pass through. Each node that was activated above 0.0 is shown. The threshold is at 0.75. A sub-goal node is decided when both an in-node and an out-node of a node are above threshold, as is shown with arrows for node C.

a path between the start and the goal and the correct in-nodes and out-nodes corresponding to the start and goal direction are activated. Of the 26 pairs of start and goal locations, 23 were processed correctly. Considering the small amount of time we have been experimenting with this network, the results are heartening. Even well-tuned systems, such as NAPS, only get success rates in the low 90 percent range.

Figure 5 shows how activity spreads through the network during a typical search. Notice how fatigue causes activity to slowly move through the network, almost as two waves from the start and the goal, until the activity route will be preferred. There is significant adaptive advantage in biasing toward the familiar, as the familiar route will usually be the safer route since it has been traversed repeatedly in the past and, therefore it is more likely that future traversals will be successful.

Route are made stronger by incrementing the strengths on the connections between nodes with each traversal of a route. The equation for incrementing is taken from NAPS and is:

$$S(l) = S(l) + (1 - S(l))^2 * LR,$$

where S is the strength of connection l, and LR was the learning rate (set to 0.20 for most of our experiments).

In addition, when a node is repeatedly visited, it gets more resources, making it more resistant to fatigue (fatigue is discussed in the next section). Resources are increased by the following equation:

$$R(n) = R(n) + (1 - R(n))^2 * LR,$$

where R is the resources of node n and LR is again the learning rate.

3 Searching the network

Searching a directional network is similar to that of many spreading activation networks, such as NAPS, except it is much more directed. First, all of the out-nodes of the start node are activated (in this implementation their activation is set to the maximum level of 1.0). Similarly, all of the in-nodes of the goal node are activated. The simulation then iteratively propagates activation across each connection. The input activity of a node is determined by the following equation (similar to the equation used in NAPS):

$$I_i(n) = \sum_{l_{m,n}} A_i(m) * S(l_{m,n}),$$

where A_i is the activation level of a node at time i, $l_{m,n}$ is a connection between node n and node m and S is the strength of the connection.

After activity is propagated through one generation of connections (i.e., activation is spread to the immediate neighbors of each node), each node in the network is fatigued. Fatigue is a control mechanism that is used in NAPS to prevent the network from becoming too active. As a node is activated it consumes resources; the more highly activated it is, the more resources it consumes. As a node consumes more resources it fatigues and its activation level is reduced until its resources can be replenished. Thus, fatigue is dependent upon the activation level of the node and the resources that it has. These combine to produce a fatigue level, which is a value between 0 and 1 with 0 meaning no fatigue and 1 meaning completely fatigued. Thus, activation of a node n is updated accordingly by the following equation (similar to the equation used in NAPS):

$$A_{i+1}(n) = (1 - F(n)) * (A_i(n) + (I_i(n) * (1 - A_i(n)))),$$

where F is the fatigue level of a node and I_i is the net input activation to the node at time i (calculated above).

After each generation of activation, the network is searched for a node that has one, and only one, in-node above a threshold (in the experiments, the threshold was an activation level above 0.75) *and* one, and only one, out-node above a threshold. Such a node is called a sub-goal. The sub-goal becomes the new goal and the process is repeated, with one significant difference, since the active in-node of the subgoal represents the direction of the previous node, activation only needs to be spread in that direction. So only the in-node in that direction is activated for the new search. However, all out-nodes of the start place have to be re-activated since no information about direction from the start is yet available. The network is not cleared of activity between each search as the previous search should have primed the activation of the route between the start and the goal allowing for quicker activation of other subgoals.

At the end of the search, the next place in the route will be known, as will its direction from the current place. The robot can move to this place and searching for the next place along the route can begin, On this next search, the direction to the goal is known (it is the active out-node of the subgoal) so activity can be concentrated in this direction. This procedure continues until an entire route is traversed.

3.1 Advantages of direction

A directional network has several advantages over a network that does not code direction, such as the network used by Toto. The first advantage is that the robot now has a simple, spreading activation means for representing the direction in which it should turn at each node. At each node the robot knows where to turn to move to the next node. In many other spreading activation networks, such as NAPS or Toto, the robot would have to rely on external cues, such as landmarks, in order to determine where to turn to move to the next node if there were several options.

A second advantage of a directional spreading activation network is that activation can be more focused. If the robot knows what direction (in the network) the goal is, then it can focus its activation in that direction,

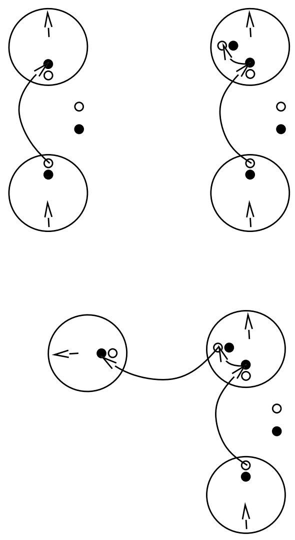
Figure 2: (a) Connections for the first move. (b) Connections for the turn. (c) Connections for the second move. The fixed, ego-centric orientation of each node is represented by an arrow.

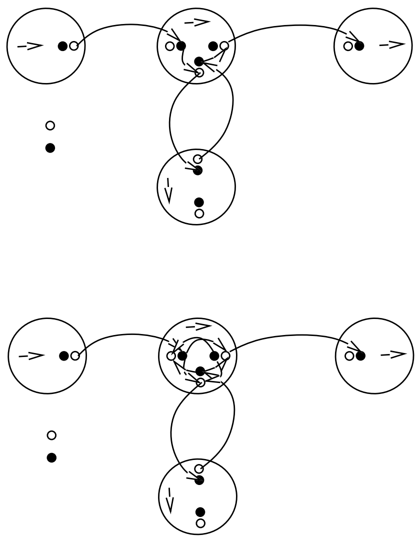
Figure 3: (a) Two routes from A to B to C and from C to B to D. (b) The same representation with internal connections to allow for discovery of novel routes. Arrows show the ego-centric, forward direction of each node.

node of A is connected to the in-node of B, the path between A and B is uni-directional (if the path where from B to A then the out-node of B would be connected to the in-node of A). This represents the fact that routes are not bi-directional.

Now the robot turns 90 degrees left at B. A new pair of in-nodes and out-nodes is created and placed at the 90 degree left direction of the node. Now the backwards in-node of B is connected to the 90 degree left out-node of B (see Figure 2b). Finally, the robot proceeds to place C and the backwards in-node of place C is connected to the 90 degree left out-node of place B, creating an entire route from A to B to C (see Figure 2c).

2.1 Multiple routes

The previous section showed how a single route through a series of nodes is connected. If two different routes share a common node the same procedure can be repeated to represent the second route, as is shown in Figure 3a for a route from A to B to C and from C to B to D. However, this structure lacks one of the biggest features of a spreading activation network – the ability to combine pieces of different route to create a new route that has never been traversed in its entirety. For example, given the network in Figure 3a, if we ask the robot to find a route from place A to place D it will fail as the connections do not meet at any single node. The solution is to add internal connections between all of the in-nodes and all of the out-nodes within a single node (Figure 3b). This will allow for retrieval of new routes that are constructed out of pieces of previously traversed routes. This is also the reason that in-nodes and out-nodes are constructed in pairs.

With these new internal connections, the spreading activation network will act like a true network and not like a collection of single routes. There is considerable adaptive advantages in having a true network, the most important advantage is that the robot is not limited to traversing routes that it has already experiences, but can find and traverse novel routes.

2.2 Adaptation

As routes are traversed, the network adapts so that more frequently traversed routes will be stronger than less frequently traversed routes. This has two consequences. First, frequently traversed routes will be extracted from the network more quickly, since they are more easily activated. Since routes that were traversed frequently in the past will likely be traversed often in the future, extracting them more quickly is an advantage. Second, if the robot has traversed two different routes from one place to another, the one that has been traversed more frequently will be preferred. This is not the case in many systems, such as Toto, where the shorter

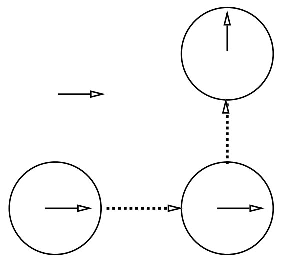

Figure 1: A route containing three nodes, each with a fixed direction. The route had the robot start at A proceed to B, turn left and move to C.

2 Constructing the network

The critical difference between our perspective and that of most researchers in mobile robotics is that we emphasize experience whereas most research emphasizes the finished representation. For example, in Toto, all routes are considered equal in desirability and bi-directional (i.e., if the robot can travel one way down it route in can travel the other way). However, not all routes have these properties. Some routes, although less optimal, may be desirable because they pass by landmarks that the robot can used to orient itself. Other routes can only be traversed in a single direction, maybe because the robot can only push a door open and it cannot pull a door open. Our representation is fundamentally driven by the robot's experiences in the environment.

Another critical difference between our representation and that of other researchers using topological networks is that our network explicitly encodes direction. Thus, constructing our network is slightly more difficult than constructing a traditional topological map. First, each node in the network has a fixed forward direction and any other directions from that node are referenced to the fixed forward. Forward for each node is defined as the direction *away from* the previous node (or, in the case of the first node, the direction in which the robot was started). For example, Figure 1 shows a route with three places, A, B and C. Let's assume that a robot starts at A and is facing B (i.e., it doesn't need to turn in order to reach place B). When the robot reaches B, the previous place (A) is directly behind it, so the direction of B is the same as A. Then the robot turns 90 degrees to the left and approaches place C. The forward of place C is directly away from the previous place (B). The fixed forward direction is shown as an arrow in Figure 1. In this representation the robot only needs to keep track of its orientation with respect to the place it just left, not with respect to any other place or to a world coordinate system. This means that dead reckoning errors will not accumulate over the entire representation, but will be confined to travel between places.

A problem with fixing the orientation of nodes is that the same node can be approached from several different directions. In Figure 1 for example, if the robot goes from A to B to C then the nodes are oriented one way, but if the robot goes from C to B to A the nodes are oriented a different way. We solve this problem by fixing the orientation of a node during the first traversal of a route containing that node. Any other traversal of routes containing that node are referenced to this fixed orientation, which never changes. This is not ideal, as one can imagine a node having several different "forwards" depending on the route that is currently being traversed. Easing this restriction is a current research topic.

Once the orientation of a node is fixed, it is relatively easy to represent directions to and from neighboring nodes. We let each node contain a collection of sub-nodes, each of which correspond to a direction to or from that node. There are two kinds of sub-nodes: out-nodes and in-nodes. Out-nodes represent the direction in which the next place along a route lies. In-nodes represent the direction in which the previous place along a route lies. Appropriate connections are formed between in-nodes and out-nodes to create routes. In simple terms a node represents "I am here", an out-node represents "I am on my way to there" and the in-node represents "I have just left from there" along with directions to the latter two places. This is a significant increase of information in the network, but, of course, the network is also more complex than a traditional spreading activation network.

Figure 2 shows how the in-nodes and out-nodes along a route are connected. In-nodes are filled circles and out-nodes are open circles. Connections between nodes are represented by the arrows. However, the arrows are only for clarity of presenting the construction of the network; during activity passing, activation flows in both directions along a connection. This does not mean that routes are bi-directional as the in-nodes and out-nodes explicitly represent direction as will be shown later.

In this example, the robot starts at place A and moves forward to place B. The forward directions of node A and node B are fixed, as is shown by the arrows in the nodes. Then, an out-node and an in-node are created in the forward direction of A (in-nodes and out-nodes are always created in pairs, the reason for this will be stated later) and an out-node and an in-node are created in the backward direction of B. Then the out-node of A and the in-node of B are connected, representing the fact that the robot moved from A to B (see Figure 2a). Since the out-

A directional spreading activation network for mobile robot navigation.*

David Kortenkamp and Eric Chown
The University of Michigan
Artificial Intelligence Laboratory
1101 Beal Avenue
Ann Arbor, MI 48109
korten@engin.umich.edu

Abstract

Spreading activation networks are attractive for compactly representing and searching a topological map of places. However, a traditional spreading activation network that links distinctive places lacks potentially useful information, such as directions, one-way routes and safe routes that are not optimal. This paper describes a spreading activation network that can represent such information without departing from the simple mechanisms that underlie spreading activation networks. The starting point of this research is the NAPS connectionist cognitive mapping model. This research is part of a larger framework for a complete theory of cognitive mapping in humans and a related project in developing cognitive maps for mobile robots.

1 Introduction

Topological or route maps have become increasingly popular in mobile robotics research [Brooks, 1985; Kuipers and Byun, 1987]. Such maps are attractive because they encode an environment very compactly by only storing distinctive places much as humans do [Byrne, 1979]. A common representation for such maps is as a spreading activation network, where each node in the network is a distinctive place and each connection is a path between two places. An example of such an implementation is the robot Toto [Mataric, 1990]. Aside from the compactness issue, spreading activation is attractive because it allows for efficient search, potentially in parallel, in a fashion that happens to be analogous to human processing. However, a simple network linking distinctive places lacks some potentially useful information that we shall show can be encoded simply and efficiently. First, differentiating connections according to experience can be helpful because not all paths are created equal; some are safer, more familiar, etc. Second, spreading activation relies on passing activity in both directions along a path which intrinsically assumes that all paths are bi-directional. Finally, a connection does not capture spatial relationships between nodes, only the fact that they are connected. Networks that do not incorporate such information essentially perform breadth-first searches and return the shortest sequence possible. Networks which do incorporate this information can focus search in terms of preferred routes, exchanging the optimality of minimum distance for a smaller chance of getting lost or getting into dangerous areas.

While the aim of the system presented in this paper is for use on mobile robots, many of the ideas are inspired by human and animal cognitive mapping. The optimality we strive for is not necessarily mathematical, but instead is more practical, recognizing that the real world is too complex to be perfectly encoded. Our implementation recognizes that errors can occur, but works to minimize their impact. In turn we argue that these goals can be accomplished within the spreading activation framework and don't require an arsenal of additions such as marker passing or labeled connections. Our implementation, an extension of the NAPS (Network Activity Passing Simulation) system [Levenick, 1991], extends the node and path concept to better reflect experience, but does so without fundamentally changing their character. In this paper we show how to construct a directional spreading activation network of routes and how to search such a network. We also present some preliminary results using a simulated network and look at the issues involved in moving the implementation to an actual mobile robot.

*This work is partially sponsored by the Department of Energy Grant No. DE-FG02-86NE37969