Large Pheromones: A Case Study with Multi-agent Physical A*

Ariel Felner[1][2], Yaron Shoshani[2], Israel A Wagner[3][4], and Alfred M. Bruckstein[4]

[1] Dpet. of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 84104, Israel
EMAIL: felner@bgumail.bgu.ac.il
[2] Dept. of Computer Science, Bar-Ilan University, Ramat-Gan, 52900, Israel
[3] IBM Haifa Labs, MATAM, Haifa 31905, Israel
[4] Dept. of Computer Science, Technion, Haifa, 32000, Israel

Abstract. Physical A* (PHA*) and its multi-agent version MAPHA* [2, 3] are algorithm that find the shortest path between two points in an unknown real physical environment with one or many mobile agents. Previous work assumed a complete sharing of knowledge between agents. Here we apply this algorithm to a more restricted model of communication which we call *large pheromones*, where agents communicate by writing and reading data at nodes of the graph that constitutes their environment. Unlike small pheromones where only a limited amount of data can be written at each node, the *large pheromones* model assumes no limitation on the size of the pheromones and thus each agent can write its entire knowledge at a node. We show that with this model of communication the behavior of a multi-agent system is almost as good as with complete knowledge sharing.

1 Introduction

This paper introduces the notion of *large pheromones* as a model of communication and global knowledge sharing in multi-agent systems. With *pheromones*, agents communicate by writing and reading data at the nodes of the graph that constitutes their environment (e.g. [7, 8]). Unlike pheromones in previous work where only a limited amount of data can be written in each node, in the *large pheromones* model, there is no restriction on the amount of data that can be written in the the nodes and thus each agent can write its entire knowledge in a node. We apply this model of communication to the multi-agent physical A* algorithm (MAPHA*) which is the multi agent version of Physical-A* (PHA*) [2, 3]. These algorithms modify the A* algorithm to find shortest paths in physical environments with mobile agents that move around the environment and explore unknown territories. These algorithms are designed to minimize the travel effort of the agents. We will show that increasing the amount of data that can be stored at each of the pheromones dramatically reduces the travel effort of the agents. With maximal usage of this model with unlimited size of pheromones the behavior of a multi-agent system is almost as good as with complete knowledge sharing between the agents.

2 Physical A*

The A* algorithm [4] is a common method for finding a shortest path in graphs that have exponential number of nodes (like combinatorial puzzles). A* keeps an *open-list* of generated nodes and expands them in a best-first order according to a cost function of $f(n) = g(n) + h(n)$, where $g(n)$ is the distance traveled from the initial state to n, and $h(n)$ is a heuristic estimate of the cost from node n to the goal. $h(n)$ is *admissible* if it never overestimates the actual cost from node n to the goal. A* was proved to be admissible, complete, and optimally effective [1]. Therefore, any other algorithm claiming to return the optimal path must expand at least all of the nodes that are expanded by A* given the same heuristic h [1]. An A* expansion cycle is usually carried out in constant time as it takes a constant amount of time to retrieve a node from the open-list and to generate all its neighbors by applying domain-specific operators to the expanded node. Thus the time complexity of A* can be measured in terms of the number of generated nodes.

Physical A* (PHA*) modifies A* to find the shortest path in much smaller graphs which correspond to a real physical environment. Consider a mobile agent who needs to find a shortest path between two physical locations and assume that only a very small portion of the environment graph is known to the agent. Since A* is optimally effective, the mobile agent needs to activate the A* algorithm on this physical graph. For this type of graph, however, we cannot assume that expanding a node from the open list takes constant time. Many of the nodes and edges of this graph are not known in advance. Therefore, to expand a node that is not known in advance, a mobile agent must first travel to that node in order to explore it and learn about its neighbors. The cost of the search in this case is the cost of moving an agent in a physical environment, i.e., it is proportional to the distance traveled by the agent. PHA* expands all the mandatory nodes that A* would expand and returns the shortest path between the two points but is designed to minimize the traveling effort of the agent by intelligently choosing the next assignment of the traveling agent. Note that since small graphs are considered here, we can omit the actual computation time and focus only on the travel time of the agent.

Unlike ordinary navigation tasks the purpose of the agent in PHA* is not to reach the goal node as soon as possible, but rather to explore the graph in such a manner that the shortest path will be retrieved for future usage. On the other hand, our problem is not an ordinary exploration problem where the entire graph should be explored in order for it to be mapped out. See [2, 3] for more comparison to other algorithms and problems.

An example for a real application can be the following scenario. A division of troops is ordered to reach a specific location. The coordinates of the location are known. Navigating with the entire division through unknown hostile territory until reaching its destination is unreasonable and inefficient. Instead, one may have a team of scouts search for the shortest path for the division to pass through. The scouts explore the terrain and report shortest path for the division to move along in order to reach its destination. PHA* is an algorithm designed to help these scouts.

PHA* works in two levels. The high level (which invokes the low level as a subroutine), acts like a regular A* search algorithm: at each cycle it chooses the best node from the open-list for expansion. Nodes are evaluated according to a heuristic function $h(n)$, which, in our case, is the Euclidean distance between n and the goal node. If the node chosen by the high level has not been explored by the agent, the low level, which is a navigation algorithm, is activated to bring the agent to that node and explore it. After a node has been explored by the low level it is expandable by the high level. If the chosen node has already been explored, or if its neighbors are already known, then it is readily expandable by the high level without the need to send the agent to visit that node.

In [2, 3] a number of navigation algorithms for the low level are presented. They all attempt to navigate to the target via unexplored nodes. Thus, while navigating through unknown parts of the graph, the agent might visit new nodes that have not been explored yet and explore them on the fly. This may save the need to travel back to those nodes at a later time, should they be selected for expansion by the high-level algorithm. See [2, 3] for a comprehensive description and all technical details of these ideas and algorithms.

2.1 MAPHA*: Multi-agent Physical A*

In [2, 3], PHA* was generalized to the Multi-agent Physical A* (MAPHA*) where a number of agents cooperate in order to find the shortest path. The task is that these agents should explore the necessary portion of the graph, i.e., the A* nodes as fast as possible. The assumption in [2, 3] was that each agent can communicate freely with all the other agents and share data at any time. Thus any information gathered by one agent is available and known to all of the other agents. This framework can be obtained by using a model of a centralized supervisor that moves the agents according to the complete knowledge that was gathered by all of them. Another possible model for complete knowledge-sharing is that each agent broadcasts any new data about the graph to all the other agents.

MAPHA* also uses a two level framework. The high level chooses which nodes to expand, while the low level navigates the agents to these nodes. Since complete knowledge sharing is assumed there is one central high level which activates A* and distributes the agents to different tasks. Suppose that we have p available agents and We would like to distribute these p agents to the nodes from the front of the open lest as efficiently as possible. This is done by distributing more agents to nodes in the front of the window, (i.e. with a relatively small f-value) but on the other hand give high priority to assigning an agent to a relatively close-by node. See [2, 3]

3 Communication models and pheromones

There are many models for communication in multi agent systems. As described above, the most trivial model is complete knowledge sharing where any new discovery of an agent is immediately shared with all the other agents. Other models restrict the level of communication. Some models allow broadcasting or message exchanging between agents but restrict the size or frequency of message. Many times, a penalty cost is associated with each message.

In nature, ants and other insects communicate and coordinate by leaving trails of odor on the ground. The "data" placed on the ground is called *pheromones*. Inspired by nature, a famous model for communicating in a multiagent system is that of ant-robotics, (e.g. [7, 8]). In this model, information is spread to other agents via *"pheromones"*, i.e., small amounts of data that are written by an agent at various places in the environment (e.g. nodes in the graph), and can be later used or modified by other agents visiting that node. In our ant-inspired model we assume that the information network is a graph, and the role of a pheromone is taken by a memory area on each node, that our search a(ge)nts can read and modify. This paradigm suggests a distributed group of one or more lightweight autonomous agents that traverses the environment in a completely autonomous and parallelized way. Data is spread by the agents via these pheromones, which together serve as a distributed shared memory.

Usually, it is assumed that a very small amount of data (no more than a few bytes) can be written in each pheromone. It turns out, however, that despite these severe limitations on pheromone size, such agents are able to cooperate and achieve goals like covering a faulty graph [8], finding an Euler cycle in a graph [9] and solving various combinatorial optimization problems [5]. A small sized pheromone can only include local data and is not very suitable for problems such as finding a shortest path in a graph where global data sharing is needed. In the sequel we consider the effect of using larger pheromones, i.e. storing more data in the nodes, on the efficiency of multiagent search.

4 Large pheromones

We suggest a new model of communication which we call *large pheromones*. Unlike conventional pheromones, we cancel the restriction on the amount of data that can be stored in each node, and consider the effect of this increased storage on the performance of the search algorithm. At the extreme, we assume that an agent can write its entire knowledge base (e.g. a complete list of nodes and edges known to that agent) at each of the nodes. With today's hardware capabilities and computer architecture this is a reasonable assumption. With pheromones, we already assume that each agent has the necessary hardware devices to allow reading and writing data in the environment. We also assume that there is a storage device in each of the nodes. Given that a storage device is installed in each node it is not reasonable to limit the size of the storage device as with current technology memory is very cheap. For example, it is not un realistic to assume, say, one megabyte of memory at a node which can store a graph of tens of thousands of nodes. We can also assume that the time to read and write data from the large pheromones can be omitted when considering the traveling time of the agents. This additional data storage in the *large pheromones* paradigm can help the agent to solve the problem faster and much more efficiently. However, large memory is not always available, e.g. in a system of nanorobots within a hostile environment, where only a very limited use of the environment is possible. Hence, we also consider a more modest memory capacity of the storage devices in the nodes. In that case, given the limited memory capacities and the entire knowledge base, we will address the question of selecting the most relevant portion of the knowledge for storing at the nodes.

4.1 Spreading data with large pheromones

With such large capacities of memory in each node, we present the following communication paradigm between the agents in general and in exploring unknown environments in particular. Each agent maintains a database with a partial graph that is known to it. Similarly, each node holds a database with a partial graph that is 'known' to it, i.e., knowledge that was written to it by the agents. Whenever an agent reaches a node, it merges the data known to it with the data that is written in that node. The agent then writes the combined data in that node and updates its own database according to the new data that was obtained. We call this the *data-merge* operation. In this way data will be spread out very fast as long as agents visit many nodes in many areas of the graph and perform *data-merge* operations at all the nodes that they visit. For example, assume that agent A visits node n and write its knowledge in that node. After a while, agent B visits node n. Agent B will read the information in the node and will merge it with its own knowledge. After merging the knowledge, both agent B and node n hold the information gathered by both agent A and agent B.

If we assume a limited memory capacity of the nodes then the data-merge operation has two stages. First, the agent merges its own data with the data that is written in the node. Then the agent erases the previous pheromone and applies a selection algorithm to determine the most informative portion of the data for writing in the node. This selection algorithm is of course domain dependent.

5 MAPHA* with large pheromones

we no present our new version for MAPHA* where the large pheromones model is employed. When the large pheromones model is employed there is no centralized entity and each agent activates the high-level A* on its own, based on the partial knowledge of the graph that is known to it at any point of time. Thus each agent keeps its own open-list of nodes and chooses to expand the best node from **that** open-list. If, neighbors of that node are not known to the agent, then, as with single agent PHA*, the agent will navigate to that target node with the help of a low-level navigation algorithm. When the large pheromones paradigm is employed then at each node that a navigating agent visits, it performs a *data-merge* operation. Its own knowledge is added to the node and knowledge about nodes that were not known to the agent is now learned by the agent. This might have a positive effect on the open-list and the high-level A* that is activated by this agent. For example, suppose that agent A choose to expand node t from front of the open-list, and this node was not yet explored by agent A. Thus, agent A should now navigate to that node. On its way to t, agent A visits node n. Suppose that another agent, B, already explored node t and later wrote that data in node n while visiting it. When agent A reaches node n and performs a data-merge operation, it learns about node t. Therefore, agent A does not need to continue the navigation to node t and that node can be expanded by agent A immediately.

If we only assume a limited memory capacity of the nodes then after the agent merges its data with data from the pheromone it will have to decide which are the most relevant nodes to write back to the pheromone. We have tried many variants and found out that the best performance was achieved by writing data about nodes that are closest to the current node.

Note that PHA* as well as MAPHA* with large pheromones are deterministic algorithms that are designed to work only in a static environments where the structure of the graph is stable throughout the search. If the graph is dynamic and changes during the search then a probabilistic approach would probably be a better choice.

6 Experiments

We have implemented the algorithms described above and performed experiments on Delaunay graphs [6] which are derived from Delaunay triangulations. Delaunay graphs simulate Roadmap graphs which are real physical graphs.

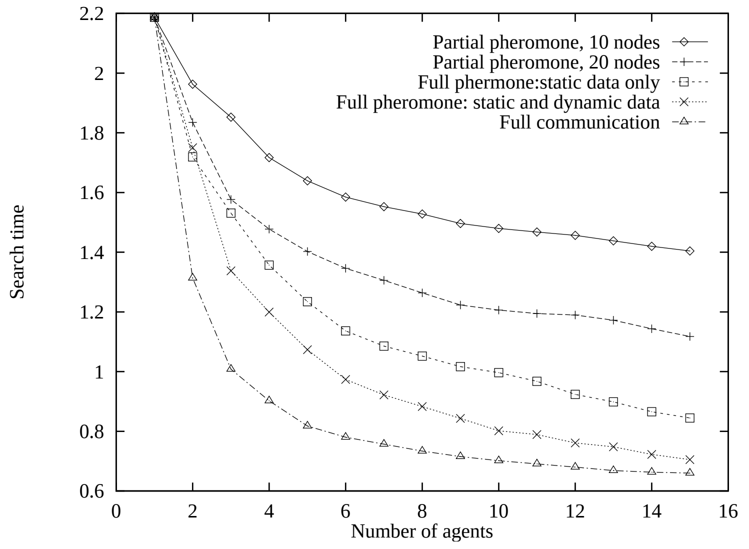

Fig. 1. MAPHA* with large pheromones, 500 nodes

Figure 1 illustrates the time elapsed as a function of the number of agents that were used to find the shortest path with different versions of MAPHA* on Delaunay graph with 500 nodes. Since we assume constant speed, the time is reported as the distance traveled by the agents until the solution was found. Every data point (here and in all the other experiments) corresponds to an average of 250 different pairs of initial and goal nodes, that were picked at random. There are 5 curves in the figure. The bottom curve corresponds to the best version of MAPHA* with full communication from [2, 3] and is used as a benchmark. Other curves show the overall time cost of different versions of MAPHA* with large pheromones. The top two curves show the results where we only assumed a limited memory capacity at the nodes. In particular, in the top curve. 10 nodes were allowed to be written and the second curve allowed 20 nodes to be written at each pheromone. As explained above, the nodes selected to be memorized are those closest to the current node. The rest of the curves assume unlimited data capacity and thus the entire graph can be written at each node. The third curve, "Full pheromones: static data only" shows the case where only static data about the graph was used. Finally, the forth curve, "Full pheromone: static and dynamic data" uses the most powerful pheromone available i.e. unlimited data capacity of the pheromones and both static data about the graph as well as dynamic data about behaviors of other agents.

The results show a clear phenomenon. As the pheromone becomes larger and includes more knowledge a significant improvement in the overall time is obtained. This parametric effect is achieved even though no explicit control is forced by a centralized supervisor.

The figure clearly shows that all the versions that use the large pheromones paradigm with unlimited memory capacity keep most of the potential of the full knowledge sharing paradigm. Their performance is rather close to the performance of the full communication version. This means that with large pheromones, data is spread to other agents rather fast and it is almost as good as full communication and full knowledge sharing. This is true even for the simple version which includes static data only.

Dynamic data, with knowledge about routes, tasks and decisions of other agents further improved the performance. It seems that using it only in the low level did not significantly improve the case were only static data was used. However, adding dynamic data to the high level adds a lot of strength to this algorithm and this last version is almost as good as full communication model.

We have experimented with other sizes of graphs and on sparse and dense graphs and obtained similar results.

7 Conclusions and Future Work

We introduced the notion of *large pheromones* as a communication paradigm in multi-agent systems. We showed that in current technology this paradigm is reasonable and cheap to implement. We used the PHA* algorithm as a test case for this paradigm. Results were very encouraging as data is indeed spread out quite efficiently in all the variations that we checked and the behavior of a multi-agent system is almost as good as with full-communication model.

The question is in what domains small pheromones are not sufficient and large pheromones are needed. We believe that large pheromones are needed in any domain where global data from different areas of the environment is critical to the decision making of all agents at all times. In that case smaller pheromones will not do the job. Our problem of activating A* in a physical environment is an example for this. Since A* expands nodes in a global best-first search order local data is not enough for this as shown in figure 1.

For another example, consider a team of fireman agents that have to extinguish fire. The general geometrical structure of the fire is very important as it might cause agents to move to different locations. A counter-example might be a group of agents who are trying to explore and map unknown territories. Whenever an agent reaches a new node, it learns new valuable information. Thus, when locally realizing that there is a nearby unexplored area, moving to that area is always beneficial. Knowledge of other areas of the environments is not so crucial at every point of time. Similarly, the works in [8, 9, 5] need local data to improve their efficiency and thus very small pheromones were enough.

Future work will proceed by applying this paradigm to other problems. Indeed we are currently implementing these ideas to multi-agent fire detecting. Preliminary results look promising. Also, A mathematical analysis of spreading data should be figured out, providing a better insights and bounds on achievable performance.

References

1. R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the Association for Computing Machinery*, 32(3):505–536, 1985.
2. A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu. Pha*: Finding the shortest path with a* in unknown physical environments. *Journal of Artificial Intelligence Research*, 21:631–679, 2004.
3. A. Felner, R. Stern, and S. Kraus. PHA*: Performing A* in unknown physical environments. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 240–247, Bologna, Italy, 2002.
4. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SCC-4(2):100–107, 1968.
5. V. Maniezzo M. Dorigo and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29–41, 1994.
6. A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations, Concepts, and Applications of Voronoi Diagrams.* Wiley, Chichester, UK, 1992.
7. A. Wagner and A. M. Bruckstein. ANTS: Agents, networks, trees, and subgraphs. *Future Generation Computer Systems Journal*, 16(8):915–926, 2000.
8. V. Yanovski, I. A. Wagner, and A. M. Bruckstein. Vertex-ant-walk: A robust method for efficient exploration of faulty graphs. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):99–112, 2001.

9. V. M. Yanovski, I. A. Wagner, and A. M. Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37:165–186, 2003.