Distributed Shortest-Path Finding by a Micro-robot Swarm

Marc Szymanski, Tobias Breitling, Jörg Seyfried, and Heinz Wörn

Institute for Process Control and Robotics (IPR)
Universität Karlsruhe, Karlsruhe, Germany
{szymanski, seyfried, woern}@ira.uka.de

Abstract. This paper describes a distributed algorithm for solving the shortest path problem with a swarm of JASMINE micro-robots. Each robot is only connected via infra-red communication with its neighbours. Based on local information exchange and some simple rules the swarm manages to find the shortest path (shortest path in the number of robots on the path) in a labyrinth with dead-ends and cycles. The full algorithm and simulation results are presented in this paper.

1 Introduction

In swarm robotics an often needed behaviour is to search for interesting spots within the workspace and to form a communication/transportation line between the found spot(s) and another area of interest or another object.

Several researchers proposed algorithms based on signalling wavefronts to solve shortest path problems in sensor and communication networks or the multi robot domain. E.F. Moore described in [1] four wavefront algorithms to find the shortest path in a maze. And also the Bellman-Ford algorithm computes the smallest spanning tree in a maze. O'Hara and Balch described in [2] an algorithm that guides robots with the help of fixed communication nodes exploiting Payton's pheromone algorithm. In Payton's algorithm described in [3] a robot close to the source will broadcast a hop count pheromone message through the swarm. If a robot close to the target gets this message, it will send a second hop count pheromone message in the opposite direction. All robots know the direction vector to the target and the source now. Adding those direction vectors leads to the shortest path. Two problems could occur with this algorithm. Firstly the robots do not know if they are on the shortest path or not. If the robots follow the gradient, they will be guided to the source or the target, but they do not keep up a path between the source and the target. And secondly if two robots in the swarm become source robots at the same time, the algorithm will be confused by different directions. Inspired by those algorithms we tried to overcome this problem. The pheromone used in our algorithm counts the hops up and down. This enables the robots to know wether they are on the shortest path or not despite of cycles in the maze.

We implemented a distributed algorithm, that finds the shortest path between two initiating source robots within a labyrinth and afterwards gathers the other

M. Dorigo et al. (Eds.): ANTS 2006, LNCS 4150, pp. 404–411, 2006.

robots around those initial robots while keeping a communication path between the two sources. The shortest path is found by an unidirectional negotiation algorithm, that sends information waves through the robot swarm.

The algorithm was evaluated in the Breve simulation environment [4] based on a model of the real swarm robot JASMINE[1], see Fig. 1.

2 The JASMINE Swarm Micro-robot

The underlying swarm micro-robot JASMINE was developed especially for swarm robot research and swarm robot games. Despite its small size of about $27 \times 27 \times 35$ mm^3, it has excellent local communication abilities and a far distance scanning and distance measuring sensor. The excellent communication abilities result from six infra-red sensors and emitters arranged around the robot with a displacement of 60°. Those sensors could also be used for short distance measurements. The far distance measuring sensor is hooked to the front of the robot. Two differentially driven wheels give this micro-robot a high manoeuvrability at a high speed. Optical encoders allow odometric measurements in the mm-range. Different from many other swarm robots JASMINE supports only local communication. Long distance communication via radio frequency is not implemented and does not correspond with the views of the construction team about swarm robot capabilities. Figure 1 shows the JASMINE swarm micro-robot and the sensor placement.

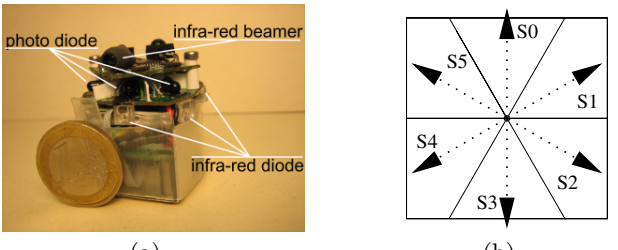


Fig. 1. (a) The micro-robot JASMINE; (b) Numbering and directions of the sensors. Starting with sensor S0 at the robot's front.

3 Distributed Shortest-Path Algorithm

The whole process of searching the shortest path could be separated into three phases:

[1] JASMINE is designed by the *Micromechatronics and Microrobotics Group at the Institute for Process Control and Robotics at the Universität Karlsruhe, Germany,* and the *Collective Micro-Robotics Team at the Institute of Parallel and Distributed Systems at the University of Stuttgart, Germany,* for the I-SWARM project. For more details see *http://www.swarmrobot.org.*

1. uniform distribution phase and search phase,
2. shortest path negotiation phase,
3. and aggregation phase.

Whereas the *uniform distribution phase* ensures that all robots are uniformly distributed through the whole maze and the *aggregation phase* leads to a contraction of the robots at the nearest sources. However, the most interesting phase is the *shortest path negotiation phase*, where the shortest path finding takes place. This paper will focus on the second phase. How a swarm could be dispersed can be found in [5].

4 Shortest Path Negotiation Phase

During the *shortest path negotiation phase* each sensor has a memory for the expected distance *from* and *to* the source $\mathbf{d}_f$ and $\mathbf{d}_t$. It also has to remember from which source it received the pheromone message. This is saved in the *pheromone* vector $\mathbf{p}$.

$$\mathbf{d}_f = (d_f^0, d_f^1, \ldots, d_f^5) \in \mathbb{N}^6 \tag{1}$$

$$\mathbf{d}_t = (d_t^0, d_t^1, \ldots, d_t^5) \in \mathbb{N}^6 \tag{2}$$

$$\mathbf{p} = (p^0, p^1, \ldots, p^5) \in \{0,1\}^6 \tag{3}$$

The upper index always denotes the sensor in respect to Fig. 1(b).

The robots' values are initially set to $\mathbf{p} = -\mathbf{1}$, $\mathbf{d}_t = \infty$ and $\mathbf{d}_f = \infty$. The source robots are initialised with $\mathbf{p} = \mathbf{0}$ for source 0 or $\mathbf{p} = \mathbf{1}$ for source 1, $\mathbf{d}_t = \infty$ and $\mathbf{d}_f = \mathbf{0}$. The basic algorithm is that the sources starts sending the message $\mathbf{m} = (0, \infty, \{0,1\})$ over all six outputs. If a robot receives a message

$$\mathbf{m} = (\bar{d}_f, \bar{d}_t, \bar{p}) \tag{4}$$

it will store those values in the memory of the receiving sensor s

$$d_t^s = \bar{d}_t,\ d_f^s = \bar{d}_f,\ p^s = \bar{p}, \tag{5}$$

and afterwards sends the message

$$\mathbf{m} = (\bar{d}_t - 1,\ \bar{d}_f + 1,\ \bar{p}), \tag{6}$$

over sensors $((s+2) \bmod 6)$, $((s+3) \bmod 6)$ and $((s+4) \bmod 6)$, on the opposite side of the receiving sensor s. This ensures the wave like dispersion of the pheromones. If the source gets a message from the other source it sets $\mathbf{d}_t = \bar{d}_f$ if $(\bar{d}_f < d_t)$ and continues to send the new message. If it gets a message with $\bar{d}_t = 0$ and $\bar{d}_f \cdot \mathbf{1} = \mathbf{d}_t$ the source knows that it got a message over the shortest communication path. It will wait for an arbitrary number of such messages before it sends the shortest path found signal (found_signal) which will start the aggregation phase. This delay ensures that the message was really send via the shortest path and not coincidentally via a second path.

Table 1. Shortest path algorithm for a source robot

```
initialise:
d_f := 0; d_t := ∞; p := {0, 1};
receive_count := 0;

begin:
while (receive_count < receive_threshold) {
   send( −1, d_f + 1, d_t − 1, p);
   if (received message m := (d̄_f, d̄_t, p̄) on sensor s) {
      if (p̄ ≠ p) {
         if (d̄_f < d_t)
            d_t := d̄_f;
         if (d̄_t = 0 ∧ d_t = d̄_f)
            receive_count := receive_count + 1;
      }
   }
}

while (¬stop_condition)
   send( found_signal, 0);
```

```
send(s, d̄_f, d̄_t, p̄) {
   if (s ≠ −1)
      send the message m := (d̄_f, d̄_t, p̄) over sensors
      (s + 2) mod 6, (s + 3) mod 6 and (s + 4) mod 6.
   else
      send the message m := (d̄_f, d̄_t, p̄) over all sensors
}
```

However, this basic algorithm is too simple and would lead to problems with cycles in the connection graph. Table 2 shows an improved algorithm for a robot, that is not a source. As long as it does not get a signal from the sources, that the shortest path was found (found_signal) each robot does the following: After receiving a message on sensor s the robot will test if $\bar{d}_t = 0$ which implies the robot is not on the shortest path and will not commit this message any further, because only a source robot could receive a $\bar{d}_t = 0$ as the distance to itself. If $\bar{d}_t > 0$ the robot will update the memory on the receiving sensor if it has not been updated before or if the received distance

$$\bar{d}_f \leq \min(d_f^i \mid p^i = \bar{p};\ i \in \{0, \ldots, 5\}) \tag{7}$$

is smaller as or equal to all other distances to the goal received from the sending source. If the memory was updated the robot will send the message updated by (6) over the sensors on the opposite of the receiving sensor s otherwise the

Table 2. Shortest path algorithm for a normal (non-source) robot

```
initialise:
d_f := ∞; d_t := ∞; p := −1;
on_shortest_path := false;

begin:
while(received message m ≠ found_signal) {
  if (received message m := (d̄_f, d̄_t, p̄) on sensor s) {
    if (d̄_t ≠ 0) {
      if ( p^s = −1
          ∨ d̄_f ≤ min(d^i_f | p^i = p̄, i ∈ {0,...,5})
          ∨ d^s_f ≥ min(d^i_f | p^i = p^s, i ∈ {0,...,5}\s) ) {
        d^s_t := d̄_t; d^s_f := d̄_f; p^s := p̄;
        send(s, d̄_f + 1, d̄_t − 1, p̄);
      }
    }

    if (∃ i, j ∈ {0,...,5} : d^i_t = d^j_f ∧ d^j_t = d^i_f ∧ i ≠ j)
      on_shortest_path := true;
  }
  wait_for_next_message();
}

goto_source();
```

message will be blocked. A problem that could occur due to communication problems is that a message from one source could be propagated around a loop in the labyrinth and a robot has values from the same source on opposite sensors. If we just use the obvious condition in (7) the robot would not accept any messages from the other source anymore. This would lead to a live-lock and the shortest path would never be found. To solve this, another constraint has to be introduced:

$$d^s_f \geq \min(d^i_f \mid p^i = p^s;\ i \in \{0,\ldots,5\}\backslash s) \tag{8}$$

Equation (8) allows to overwrite a value from one source by the other one in case, that there is a better d^i_f value from the overwritten source on another sensor i.

This behaviour will iteratively lead to the case, that each robot on the shortest path will have at least two sensors $i,\ j,\ i \neq j$, with data from different sources that have cross-over the same values $d^i_t = d^j_f$ and $d^j_t = d^i_f$. The shortest path condition is:

$$\exists\ i, j \in \{0,\ldots,5\} : d^i_t = d^j_f \wedge d^j_t = d^i_f \wedge i \neq j. \tag{9}$$

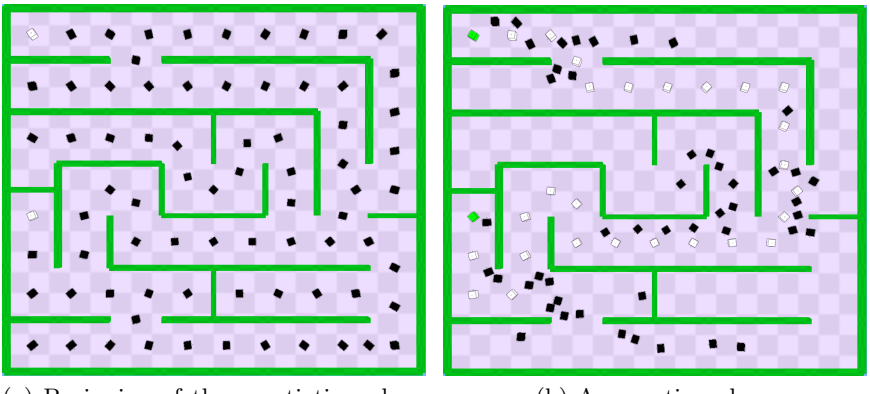
The length of the shortest communication path can be computed as $d_{SP} = d^i_t + d^j_f$.

It is important for the robots to know if they are on the shortest path or not. Because the robots on the shortest path will behave as beacons for the other robots that gather near the closest source or transport objects along this path similar to [6]. The knowledge being or not being on the shortest path triggers different behaviours during the aggregation phase.

5 Experiments

The simulation for evaluating the proposed algorithm is based on a model of JASMINE. The simulation environment Breve described in [4] has been used for the simulation of the JASMINE robots. The distributed shortest-path algorithm was implemented in Steve[2] and MDL2ϵ which is our further development from MDLe by Manikonda et al., see [7].

Many simulation experiments have been performed with several source positions. Figure 2 shows an experiment[3]. Hundred robots have been equally distributed in Fig. 2(a) with a communication range of 20 cm. Figure 2(b) shows the robots in the aggregation phase the white robots stand on the shortest path, the sources are green and the black robots are moving towards the nearest source.



(a) Beginning of the negotiation phase. White robots are the sources.

(b) Aggregation phase.

Fig. 2. Experiments with the simulation environment Breve. Black robots are normal robots, white robots are on the shortest path.

The experiments showed that the algorithm is stable with respect to communication problems that occur randomly distributed over the robots. If a robot cannot transmit or receive any messages it will be detected as an obstacle and the algorithm searches another path around this robot if possible.

[2] Programming language for Breve.

[3] A video showing the whole experiment can be found at `http://wwwipr.ira.uka.de/~szymansk/video/SlimeMould.avi`.

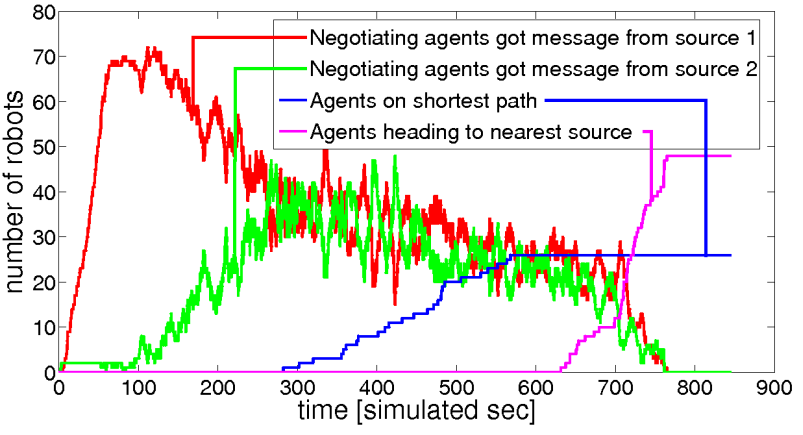


Fig. 3. State distribution during an experiment

Figure 3 shows the state distribution during an experiment. One can see that in the beginning all robots are in the negotiation state and get a message from either source 1 (red) or 2 (green). After 300 seconds the number of robots that got a message from source 1 or source 2 is almost equal. After the swarm reaches this equilibrium the robots on the shortest path become aware, that they are on the shortest path (blue). Some time after the shortest-path has been found the sources start emitting their aggregation message. The number of robots that received this message (magenta) is increasing and the original negotiation messages are suppressed.

6 Conclusion

We described an algorithm for calculating the shortest path in a maze in a distributed manner. The experiments start from the point of equally distributed robots and ends with robts standing on the shortest path.

We showed in several experiments that the algorithm is stable regarding communication errors. This algorithm is currently not scalable in the sense that we add more sources that should be all connected. Experiments showed if a source with the same source identification is added the shortest path between two different sources will also be found. One problem for this algorithm is if an agent is “dead” and does not respond to any input this agent will be treated as a wall. This problem could also be seen as an advantage, because a path around this “dead” robot will be found. This “dead” robot is nothing more than an obstacle.

7 Future Work

In the future we are going to implement the algorithm including the dispersion part on JASMINE robots to evaluate the performance on real robots. It could

also be compared with other algorithms that find the shortest path between two sources in a workspace and build a communication path. And a theoretical proof of the stability of this algorithm could be derived.

References

1. Moore, E.F.: The shortest path through a maze. In: Proc. of the International Symposium on the Theory of Switching, Harvard University Press (1959) 285–292
2. O'Hara, K.J., Balch, T.R.: Distributed path planning for robots in dynamic environments using a pervasive embedded network. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, IEEE Computer Society (2004) 1538–1539
3. Payton, D., Daily, M., Estowski, R., Howard1, M., Lee, C.: Pheromone robotics. In: Special Issue on Biomorphic Robotics. Volume 11 of Autonomous Robots., Springer Netherlands (2001) 319 – 324
4. Klein, J.: breve: a 3D Environment for the Simulation of Decentralized Systems and Artificial Life. In: ICAL 2003: Proceedings of the eighth international conference on Artificial life, Cambridge, MA, USA, MIT Press (2003) 329–334
5. Hsiang, T.R., Arkin, E.M., Bender, M.A., Fekete, S., Mitchell, J.S.B.: Online dispersion algorithms for swarms of robots. In: SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry, New York, NY, USA, ACM Press (2003) 382–383
6. Li, Q., Rosa, M.D., Rus, D.: Distributed algorithms for guiding navigation across a sensor network. In: MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking, New York, NY, USA, ACM Press (2003) 313–325
7. Manikonda, Krishnaprasad, Hendler: Languages, Behaviors, Hybrid Architectures, and Motion Control. Mathematical Control Theory (1998)