Being There, Being the RRT: Space-Filling and Searching in Place with Minimalist Robots

Asish Ghoshal and **Dylan A. Shell**
Department of Computer Science & Engineering
Texas A&M University
College Station, Texas, USA
Email: {aghoshal,dshell}@cse.tamu.edu

Abstract

Inspired by the Rapidly Exploring Random Tree data-structure and algorithm for path planning in high-dimensional, continuous spaces, we consider an approach for spanning a space with a group of simple robots. We employ a minimalist approach in which contact sensors form the primary means of communication; the agent's physically embody the elements of the tree through their position and other agents can either follow the tree to useful locations or expand the tree by becoming part of it. Although robots are constrained in some of the operations they may perform in space, we argue that the original space filling aspects of the original data-structure remain in our implementation. We demonstrate that one may perform a planning query from a point to the tree origin directly via message passing where passing involves direct physical motion or simple IR messages. Based on the work done by Werger and Matarić, our implementation proves that it is possible to form and maintain a RRT using simple position unaware robots. The work is important because it demonstrates that decentralized path planning can be performed by simple agents using purely reactive behaviors and at the same time poses significant challenges to keep the shape of the tree intact using position unaware robots.

Introduction

This paper considers the problem of having a group of simple robots equipped with limited, short-range communication, span a physical space. They do this by building an incremental tree in a process analogous to a well-known data-structure, the Rapidly Exploring Random Tree. In fact, "analogous" may too weak a word. The robots actually implement the algorithm, but they do in an unconventional way: they exploit their embodiment so that information usually stored in conventional variables is encoded in the poses of the robots. The work demonstrates what we believe to be a broader idea, namely that several existing spatial algorithms with well-understood properties can be directly implemented on robot hardware so that the resulting properties describe the robots' configurations. The primitives employed by the algorithm point to behaviors that robots need to be able to execute. If an asynchronous implementation of a synchronous data-structure can be made consistent, then the algorithmic analysis can to be carried over to the state of the robots themselves.

Motivation

The motivation for the current work comes from experiments by Werger and Matarić (1996) on encoding information in the environment. In that work the authors demonstrate how a group of simple robots can be used to encode information into the environment that can be used by other robots effectively to achieve a certain goal: they show how a chain of simple robots can be used to guide other robots, responsible for collecting pucks, to and from the designated home region. Furthermore, they argue that the number of taps received on one side of the chain is proportional the relative puck density on that side of the chain; thus, the chain can be gradually moved to the side having greater density of robots.

The second motivation for the work comes from Rapidly-Exploring Random Trees (RRTs) which have been successfully used in various motion planning problems in robotics (LaValle and Kuffner, Jr. 2000). The strength of an RRT lies in its ability to rapidly span a given space (configuration space) and its bias towards unexplored regions. It has been observed that larger Voronoi regions appear on the frontier, thereby biasing the exploration to the unexplored portion of the space.

Contribution

We present an approach of spanning a space using simple robots that physically embody the RRT data-structure. Thus by being in place the robots encode information into the environment which can then be used by other robots to locate the home region and at the same time the tree formed by robots provide sufficient coverage to be categorized as a RRT. Our approach is different from that of Werger and Matarić's in that we extend earlier work by encoding information into the environment so as to perform simple path planning locally. Further, using robots to physically form a RRT had never been studied before.

The rest of the paper discusses the approach and the implementation of the aforementioned algorithm, followed by a brief discussion and conclusion.

Related Work

Donald was an early advocate of minimalist approaches to robotics. He developed the information invariants framework for comparing different sensory-computational systems. In Donald (1995), his most extensive discussion of the framework, he shows that the physical placement of resources in such systems can be particularly important. His formalism enables the effect of co-location of different resources to be quantified by computing the additional information that would need to be transmitted in other configurations. That work was cited as inspiration for Werger and Matarić (1996), and is clearly related to this work in which the physical positions of robots encode state.

Pheromone Robotics: (Payton et al. 2001)
Self-reconfigurable and modular robotics: (Støy, Brandt, and Christensen 2010)
Path planning and morphogenesis: (Nouyan, Campo, and Dorigo 2008) (O'Grady, Christensen, and Dorigo 2009).

Approach

The RRT data-structure

The original algorithm for constructing the RRT for a general configuration space is given below where q_{init} is the initial configuration, q_{rand} is a random configuration and G is the graph representing the RRT. A random configuration, q_{rand}, is chosen in each iteration which determines the new vertex that will be added to the graph. Subsequently the vertex in the tree, q_{near}, which is nearest to q_{rand} is computed. In step 5 of the algorithm a new configuration, q_{new}, is computed by selecting an action that moves q_{near} an incremental distance, Δq, in the direction of q_{rand}. In the final step of each iteration, the vertex q_{new} and the corresponding edge $(q_{\text{new}}, q_{\text{near}})$ is added to the tree.

Algorithm 1 BUILDRRT

```
Require: $q_{\text{init}}$, $K$, $\Delta q$
1: $G.init(q_{\text{init}})$;
2: for $k = 1$ to $K$ do
3:     $q_{\text{rand}} \leftarrow$ RANDCONF();
4:     $q_{\text{near}} \leftarrow$ NEARESTVERTEX($q_{\text{rand}}$, $G$);
5:     $q_{\text{new}} \leftarrow$ NEWCONF($q_{\text{near}}$, $\Delta q$);
6:     $G.add\_vertex(q_{\text{new}})$;
7:     $G.add\_edge(q_{\text{near}}, q_{\text{new}})$;
8: end for
9: return G
```

In our implementation the RRT, G is physically represented by robots. So, robots represent both vertices and edges of the tree where vertex and edge robots are identified based on the operations that they can perform. Two key operations that are central to our implementation are the ADDVERTEX operation and the EXTENDEDGE operation. In the ADDVERTEX operation, a robot is added to the tree as a new vertex. While in the EXTENDEDGE operation, robots are added as edge nodes in order to extend the edge each time until the length of the edge becomes Δq, where Δq is the incremental distance by which the tree gets extended in each iteration. It is ensured that the aforesaid operations do not cancel the effect of each other. The key difference between the above algorithm and ours is that we grow the tree asynchronously. By asynchronous we mean that the steps of the algorithm are not atomic but span multiple iterations and may happen in parallel *e.g.,* while a new vertex is being added on to the tree at one point (ADDVERTEX operation), another robot might be extending an incomplete edge (EXTENDEDGE operation) so long as they do not interfere with each other. The subsequent paragraphs describes our approach in detail.

The configuration space: The configuration space is a two dimensional plane and the position of the robot in the plane represents a particular configuration. The two degrees of freedom of the robot are sufficient to completely represent any configuration within the configuration space. The initial configuration, q_{init}, is chosen by placing a dead robot in the environment. Other robots perform a random walk representing different points within the configuration space at different points of time.

Choosing a random configuration: The random configuration, q_{rand}, is chosen by randomly choosing a robot that represents a point in the configuration space to join the tree as a vertex. As a robot is chosen randomly it starts spiraling out until either it hits another robot or it covers a threshold distance in which case resumes its random walk. In the latter case even though the miss might prove to be costly in terms of time but we can argue that it does not have any effect on the data-structure per se.

Finding the nearest neighbor: The vertex in the tree nearest to q_{rand} is identified by making the randomly chosen robot spiral out until it bumps into a vertex robot which is already part of the tree. But, the spiraling robot may bump into another wandering robot in which case the random point q_{rand} is discarded in the sense that no new vertex is added to the tree and the robot resumes it random walk. Whereas if the spiraling robot bumps into an edge robot then the point q_{rand} is still discarded but the robot begins to trace the tree in an attempt to complete an incomplete edge. Thus, while a spiraling robot may perform ADDVERTEX and EXTENDEDGE operations, a wandering robot can only perform the EXTENDEDGE operation. Figure 1 shows that if the randomly chosen robot lies in the Voronoi region of a robot in the tree then it is nearest to that robot and by spiraling out it would definitely bump into it first unless it hits another moving robot or the radial distance covered by it exceeds the threshold distance.

Computing a new configuration: If a spiraling robot bumps into the tree at the end of a fully completed edge then it is identified as a Vertex robot and is added into the tree in place. By doing so it also initiates an edge which is completed by other wandering robots. So by adding the new robot at the point in which it bumped into the tree we make sure that the edge is extended in the direction of the randomly chosen point q_{rand}. The new robot now has a depth of one more than its parent which is the robot previously at the

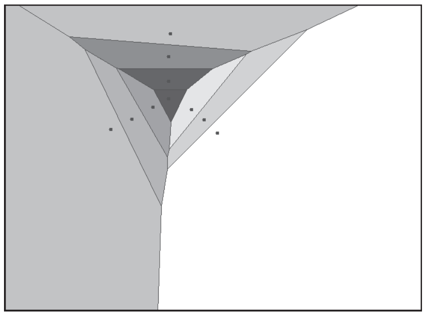


Figure 1: Diagram showing Voronoi regions created by robots in the tree.

end of the edge. The notion of depth here might contradict that of graph theory literature in which only vertices have depths associated with them. But here depths have been assigned to all robots for identifying when an edge is complete and if an incoming robot should be a vertex robot or an edge robot. So, the actual depth of a vertex robot is depth modulo Δq, where Δq is the length of the edge.

ADDVERTEX **operation:** The ADDVERTEX operation is initiated only when a spiraling robot bumps into a fully formed edge in which case it is added in place to the tree thereby increasing the edge by unit length in the direction of the randomly chosen point from where the robot started spiraling. If a spiraling robot bumps into a edge robot then it starts tracing the tree and if it finds a half formed edge then the EXTENDEDGE operation is initiated. This operation is shown visually in Figure 2(a).

EXTENDEDGE **operation:** The EXTENDEDGE operation is initiated only when a wandering or spiraling robot bumps into a half formed edge. In this operation the robot is added to the tree as an edge robot by aligning the robot with the existing edge to form a straight line. an intrepretation of our implmentation is that while the EXTENDEDGE operations are being performed, they make up part of a single "add_edge" operation of the original algorithm See Figure 2(b) for an illustration.

Implementation

The following terminologies are important in understanding the implementation:

1. Region of influence: Region of influence of a robot is the area directly around the robot where its IR messages can be detected by other robots which is called as its field of communication and also the fields of communication of its direct neighbors. See Figure 3.
2. Location estimate: Location estimate of a robot X as perceived by a robot Y and relative to Y is the time elapsed between the time the IR break beam of a robot in the tree breaks upon detecting its parent robot in the tree P and the time when the break-beam breaks upon detecting robot X. So the location estimate of a robot X is an estimate of the robot's position around the robot Y as measured from Y's parent P.

The current implementation uses robots that rely only on contact sensors and InfraRed sensors for communicating with each other. A one byte packet sent using an IR LED is used for communication. The three types of packets that are used are request, response and status packets. The packets also make use of a parity bit for minimal error detection. The different request messages are JOINCHAIN, MOVELEFT, MOVERIGHT, MOVEUP, MOVEDOWN, READYTOJOIN, BUMPNOTIFICATION. The status packet identifies the status of a robot *viz.* part of tree, wandering and spiraling. If the robot is part of the tree then the packet additionally encodes the depth of the robot in the tree. The response packets are either ACK or NACK.

A robot upon bumping into an object sends a bump notification messages while robots part of the tree continuously monitor for IR messages. Whenever a robot in the tree gets bumped it broadcasts status messages after receiving an IR bump notification message. The wandering or spiraling robot upon receiving the status message signals its intention to join the tree by broadcasting "join chain" request packets, which also encode the state of the robot *i.e.,* spiraling or wandering, only after determining that it bumped into a robot in the tree. Now, the robot which is part of the tree determines based on its depth in the tree, if the new robot is supposed to be an edge robot or a vertex robot. The incoming robot is classified as an edge robot if it bumped into a half complete edge while it is classified as a vertex robot if it bumped into a fully complete edge. An edge is deemed complete when its length in terms of number of robots equals Δq. If the new robot is supposed to be a vertex robot but its state was wandering then it NACKs the "Join Chain" request while if its state was spiraling then it ACKs the "join chain" request.

Each robot has the location estimate of its parent, referred hence forth as Ω, which has been determined empirically and corresponds to the time taken for the robot to complete one full rotation. After ACKing the "join chain" requests the robot in the chain starts rotating to find out the location estimate of the incoming robot. The robot in the chain identifies the location, and hence the location estimate Θ, of the incoming robot by ignoring all other robots that are present in its neighbor list.

ADDVERTEX **operation:** If the robot in the chain determines that the incoming robot is a vertex robot then it sends the "ready to join" request to the incoming robot and upon getting the acknowledgement for the request it adds the incoming robot to its neighbor list. The incoming robot in turn on receiving the "ready to join" request which also has the depth of the robot that sent it, increments the depth by one

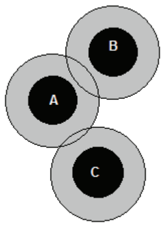
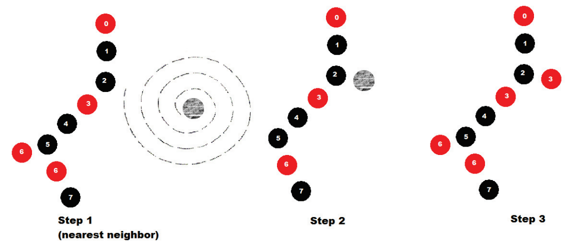
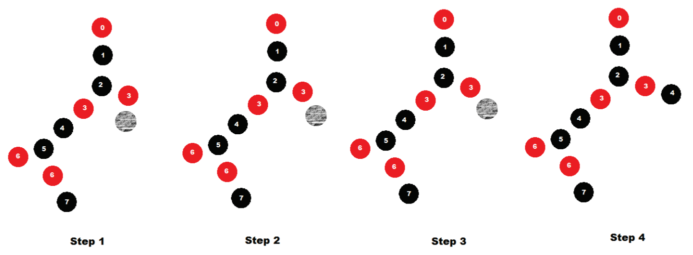


Figure 3: Diagram illustrating the field of communication of each robot shown in gray. The region of influence of Robot A is the combination of field of communication of all three robots.



(a) Robot being added as a vertex robot.



(b) Robot being added as a edge robot.

Figure 2: Diagram illustrating a robot joining the tree either as a vertex (left) or edge (right). Red circles indicate vertex robots while black indicate edge robots.

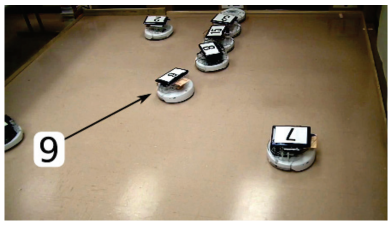


Figure 4: Snapshot of a robot (#9) trying to find the nearest neighbor.

and becomes part of the tree. A snapshot from our implementation is shown in Figure 5.

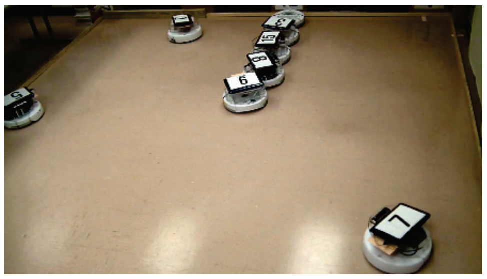


Figure 5: Snapshot of a ADDVERTEX operation. (Figure 4 shows this robot previously having spiralled to locate its nearest neighbor.)

EXTENDEDGE operation: If the robot in the chain determines that the incoming robot is an edge robot then it compares the location estimate of the incoming robot Θ with Ω. If Θ is approximately equal to Ω then it implies the incoming robot is perfectly aligned and hence the robot in the chain sends the "ready to join" request. While, if $\Theta < \frac{\Omega}{2}$ then it means the incoming robot is on the left of the desired position and hence broadcasts the "move to left" command. Similarly, if $\Theta > \frac{\Omega}{2}$ the incoming robot broadcasts the "move to right" command. The incoming robot in turn upon receiving the command moves appropriately and acknowledges the command. After a series of such interaction when the incoming robot has aligned perfectly, the robot in the chain sends "ready to join" request to the incoming robot. The incoming robot in turn on receiving the "ready to join" request which also has the depth of the robot that sent it, increments the depth by one and becomes part of the tree. See Figure 6 for an example.

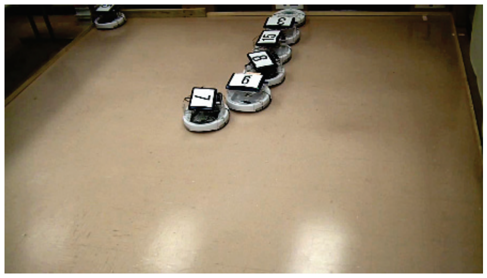


Figure 6: Snapshot of EXTENDEDGE operation.

Tracing the tree: If the robot in the chain determines that the incoming robot is an edge robot then it compares the location estimate of the incoming robot Θ with Ω. If the incoming robot determines that the next position in the edge has already been occupied then based on the value of Θ it sends the "Move down" ($\Theta < \frac{\Omega}{2}$) or "Move up" ($\Theta > \frac{\Omega}{2}$) command to the incoming robot. The incoming robot in turns moves up or down the tree and tries to join the tree back further down the edge or if the edge is complete then it moves away to another branch. Figure 7 provides an example of this operation being performed.

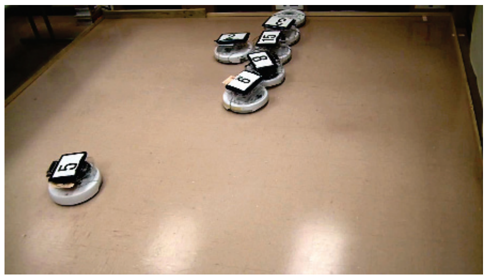


Figure 7: Diagram showing a robot tracing the tree.

Even though two or more robots try to join the tree at different but valid points so that they avoid any physical interaction, the robots are barred from joining the tree if their region of influence overlaps. This strategy prevents robots from getting the wrong message owing to "cross talk".

Experiments

The Robots Our experimental setup comprised of eight iRobot Create robots including a dead robot to serve as the starting point of the tree. The standard Create robots were augmented with an IR LED transmitter capable of transmitting one byte of information. A reflector was used to disperse the IR radiation with the aim of creating an IR field around the robot. See Figure 8.

The setup The experimental arena was an $3.66\text{m} \times 5.05\text{m}$ rectangular area with the starting point located near the center of the top edge of the arena.

The Experiments The first set of experiments aimed at forming a tree with all seven robots starting in wandering state, while the next set of experiments demonstrated four robots joining a partially formed tree of three robots (one vertex robot and two edge robots). Experiments were repeated by varying the parameter Δq from 1 to 3.

While a third set of experiments were also run where taps using bumpers was used as the mode of communication instead of IR messages. A communication method was devised where different taps *viz.* short tap, long tap, very long tap, turbo tap, double tap etc conveyed different information.

Results Using IR messages as the communication mode we were successfully able to form trees of seven robots with Δq set to 3. (Figure 9 is an example from such a trial.) And this combination proved to be the most efficient in terms of the time taken for all robots to join the tree. In most experiments the maximum degree of a vertex in the tree was one while we are confident that we would be able to form trees with maximum degree of a vertex exceeding two by further optimizing the algorithm as stated in the discussion section below. An example of the time spent in the tree is shown at the resolution of individual robots in Figure 10.

With Δq set to 2 and the number of robots at 7 the time taken for all robots to join the chain was higher owing to the fact that more spiraling robots, to be exact 3, were required to join the chain successfully, which is difficult given that every time a spiraling robot bumps into a wandering robot the process of finding the nearest neighbor is aborted and the robot resumes its random walk. With Δq set to 1 and the number of robots at 7 the performance even degraded further.

With physical taps as the communication mechanism we were reliably able to form chains with two robots while

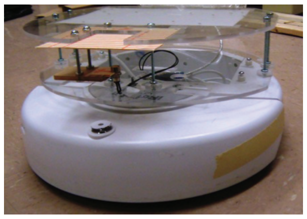


Figure 8: An iRobot Create augmented with an IR LED.

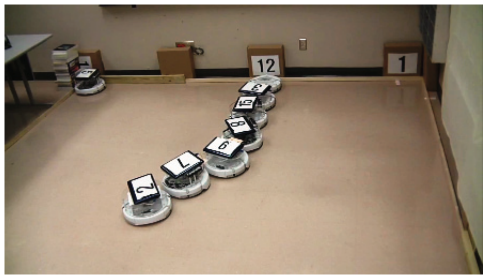


Figure 9: A typical trial that used 7 robots with Δq set to 3.

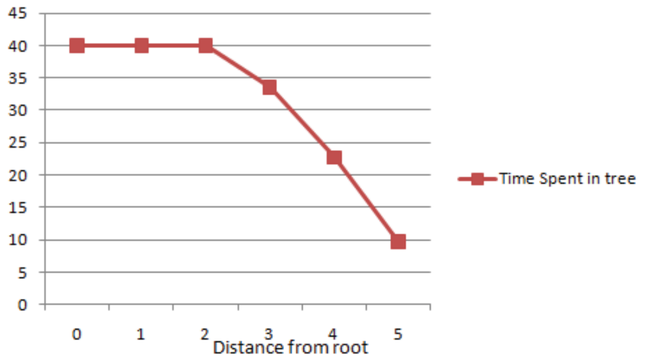


Figure 10: Diagram showing the times at which the robots joined the tree with Δq set to 3.

forming chains of length greater than two posed significant challenges because of the fact that physical bumps slowly pushed robots out of the chain which proved to be very fatal owing to our reliance on dead reckoning to localize neighboring robots and the robot itself. So, a tree of seven robots was almost impossible using taps. But, surprisingly using Δq as 1 the performance improved because all robots joined as vertex robots and so a lot of taps were reduced owing to the fact that none of the robots needed to be aligned.

Discussion

We observed that the frequency with which a robot attempts to spiral out and join the tree as a vertex robot is an important determinant of the overall performance of the system. In our current implementation pseudo random numbers, seeded at the unique robot id for each robot, in the range 1 and 10 determined the minutes a robot spent wandering before attempting to spiral out and join the tree. A longer range implies that a fewer robots would try to join the tree at the same time tending to decrease interaction among robots thereby improving performance but also means that a robot would spend significant time wandering before it reattempts to join the tree back by spiraling out thereby impeding performance.

But using IR as the mode of communication, albeit robots only exchanged a byte at most, improved the robustness of the system dramatically. The implementation ensured that no two robots cancelled the effect of each other *i.e.,* a wandering and a spiraling robot did not initiate the ADDVERTEX and JOINCHAIN operation respectively at the same time in the same region of influence.

Further, we observed that our strategy of aborting the process of finding the nearest neighbor whenever a spiraling robot bumps into a wandering robot proved to be costly and decreased the performance of the system. So, by having a mechanism in which a spiraling robot continues to spiral out in an attempt to find the nearest neighbor in the tree in spite of bumping into another wandering robot and by making sure the original random point generated, q_{rand}, is still maintained we can significantly improve performance.

Our implementation is also significantly optimized in the sense that the tree can be extended along various fronts in parallel as long as region of influences do not overlap. We further optimize the algorithm by allowing a wandering robot to trace the tree in order to complete a half formed edge.

Conclusion and Future Work

From the various experiments that we performed we conclude that minimalistic robots can physically embody a RRT data-structure by using a few simple rules. The trace behavior that we have successfully implemented and demonstrated can be extended to introduce motion planning wherein a different group of robots using the behavior can scuttle between various parts of the tree and then determining optimum routes thereby mimicking a colony of ants. We also plan to extensively run our algorithm in simulation in order to arrive at a mathematical model and gain further insights.

References

Donald, B. R. 1995. On information invariants in robotics. *Artificial Intelligence* 72(1–2):217–304.

LaValle, S. M., and Kuffner, Jr., J. 2000. Rapidly-exploring random trees: Progress and prospects. In *Robotics: The Algorithmic Perspective. 4th Int'l Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Nouyan, S.; Campo, A.; and Dorigo, M. 2008. Path formation in a robot swarm. *Swarm Intelligence* 2(1):1–23.

O'Grady, R.; Christensen, A. L.; and Dorigo, M. 2009. SWARMORPH: Multirobot Morphogenesis Using Directional Self-Assembly. *IEEE Transactions on Robotics* 25(3):738–743.

Payton, D.; Daily, M.; Estowski, R.; Howard, M.; and Lee, C. 2001. Pheromone Robotics. *Autonomous Robots* 11(3):319–324.

Støy, K.; Brandt, D.; and Christensen, D. J. 2010. *Self-Reconfigurable Robots: An Introduction*. Cambridge, MA, U.S.A.: MIT Press.

Werger, B. B., and Matarić, M. J. 1996. Robotic "Food" Chains: Externalization of State and Program for Minimal-Agent Foraging. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats (SAB)*, 625–634.