

# Goal Sequencing for Construction Agents in a Simulated Environment

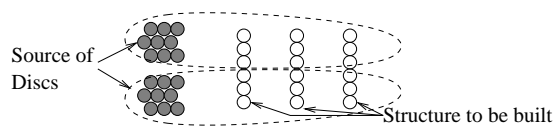
Anand Panangadan and Michael G. Dyer \*

Computer Science Department, University of California at Los Angeles,  
Los Angeles CA 90095, USA  
{anand,dyer}@cs.ucla.edu

**Abstract.** A connectionist architecture enables a society of agents to efficiently construct 2D structures. The agents use internal spatial maps to compute a sequence of construction actions that reduces total distance traveled. All computations are done over grids of neurons interacting locally. Simulation results are presented.

## 1 Introduction

Here, a group of autonomous agents efficiently construct structures in a simulated 2-D continuous environment. Construction involves arranging same-sized discs to form a given 2-D pattern. [2] describes an architecture that performs this task; however, its greedy approach moves a disc to its closest drop-site and this can prove inefficient. For example in figure 1, the fastest way to build the three walls is to build the right-most wall first, then the middle wall and finally the wall closest to the source of discs - if the nearest wall is built first, this would block direct paths from the discs to the other two walls. The agent(s) should also use the upper source of discs to build the upper parts of the walls and the lower source for the lower parts of the walls. This minimizes the distance between the initial and final positions of the discs. Here, we extend the architecture in [2] to enable the agents to place discs in an order that reduces the total distance traveled. Interference between agents is also reduced without communication. The algorithms are computed over grids of neurons using only local interactions.



**Fig. 1.** Arrangement of discs and drop-sites where a greedy approach is inefficient.

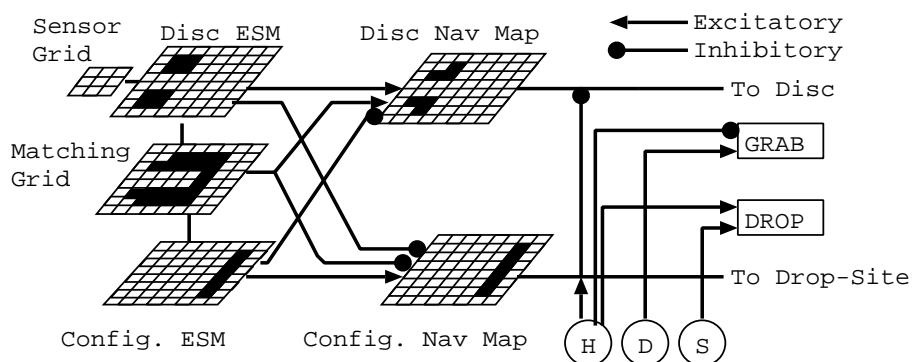
---

\* This work supported in part by an Intel University Research Program grant to the second author.

## 2 Architecture

Agents can sense discs and other agents near them within a limited range. Disc locations are encoded via a grid of neurons (the *Sensor grid*) that divides the sensing area into  $30 \times 30$  unit squares. Sensor neuron activation indicates a disc at the corresponding square. Construction involves moving toward a disc, grabbing it when nearby, moving toward a drop-site and then dropping it. An agent can move forward and turn smoothly in any direction.

The architecture is shown in figure 2. Each agent uses its own Egocentric Spatial Maps (ESMs) [2] to represent the spatial relationship between itself and the discs. An ESM contains neurons arranged in a uniform grid that maintains an egocentric view of the world (the center node always represents the current location of the agent). Each ESM neuron corresponds to a small square area of the world and its activation indicates a disc being present in the corresponding area. Thus each agent maintains a "birds-eye" view of the world around it. As the agent moves, activations are passed to neighboring neurons to maintain egocentricity. New sensor input is integrated into the ESM from the Sensor Grid.



**Fig. 2.** The architecture: Activations (dark cells) on the Matching grid show which discs (marked on the Disc ESM) should be moved to drop-sites (on the CESM). H, D, and S represent the internal state nodes Have-Disc, At-Disc and At-Drop-Site respectively.

The *Disc ESM* (DESM) encodes the locations of discs around the agent. The *Configuration ESM* (CESM) encodes the structure to be built. CESM activations are also shifted as the agent moves, but those activations that encode the structure to be built are set a priori and are not updated by the sensors. Each DESM neuron is connected to a corresponding *Disc Navigation Map* (DNM) neuron. Active DESM neurons initiate spreading activation on corresponding DNM neurons. The CESM active neurons encode drop-off sites and thus discs, once placed at these sites, should not be picked up. Hence these neurons represent obstacles in the path and inhibit their corresponding neurons in the DNM.

Let  $n$  denote an arbitrary neuron in a Navigation map and  $nb(n)$  the set of eight neighboring neurons of  $n$ . Let  $a_n(t)$  be the activation of  $n$  at time  $t$ .

$$a_n(0) = \begin{cases} 1, & \text{if } n \text{ represents a drop-off site} \\ -1, & \text{if } n \text{ represents an obstacle} \\ 0, & \text{otherwise} \end{cases}$$

$$a_n(t+1) = \max_{m \in nb(n)} (a_m(t) - d(n, m)), a_n(t) \geq 0$$

where  $d(m, n)$  is proportional to the distance between the locations represented by nodes  $m$  and  $n$ . This is a parallel implementation of Dijkstra’s shortest path algorithm. The above equations are iterated until the activations stop changing:  $a_n(t+1) = a_n(t) \forall n$ . The gradient created by the spreading activation is the planned path to the nearest disc that is not already at a construction location. Similarly, the Configuration Navigation Map (CNM) is used to compute a path to locations where discs should be dropped. Every node is activated by the corresponding node from the CESM and inhibited by the nodes of the DESM.

There are three *Internal State Nodes* that indicate the current stage of the construction task: *Have-Disc* indicates if the agent is holding a disc; *At-Disc* and *At-Drop-Site* are active if the agent is near a disc or drop-site respectively. If *Have-Disc* is active, the agent moves toward a drop site else it moves toward a disc by following the gradient on the corresponding Navigation map. *At-Disc* and *At-Drop-Site* determine if a disc has to be picked up or dropped.

### 3 Sequence of dropping discs

With this architecture, the agents drop discs at target sites nearest to disc sources. This greedy approach is a distributed neural implementation of the “matrix scan” heuristic for the minimum weighted perfect matching (MWPM) or assignment problem for bipartite graphs [8]. The vertices of one partition are the initial locations of discs and the vertices of the other partition are the drop sites. The edges represent path distances between two vertices. The MWPM problem is to compute a one-to-one matching between disc locations and drop sites such that the sum of the paths is a minimum. Modeling construction as a MWPM problem has the property that in an optimal solution, paths between matched pairs do not cross, reducing interference between agents.

This approach assumes that path length between two locations will not change during construction. However, in the general case, the order in which discs are placed affects the distance traveled by agents to place the remaining discs as illustrated in figure 1. Thus, for efficient construction, the agents have to plan the temporal sequence of goal locations such that the path from every disc to its drop site is a straight line (not blocked by any previously dropped disc). The algorithm described below determines such a sequence in two steps. Each agent first computes a matching between discs and drop sites represented in its ESMs using a greedy heuristic. It then determines which of the drop sites are not on any of the paths (between a disc and its matching drop site) and proceeds to

fill these first. The algorithm is implemented using only a grid of neurons (called the *Matching grid*) that can spread activations among neighboring cells. At the end of the algorithm,  $\text{match}[n] := 1$  for all nodes  $n$  that are on a path between a matched disc and drop-site location. Let  $N$  be the set of all nodes on the Matching grid and  $S, T$  be the set of nodes representing disc locations and drop-sites respectively. If every node  $n$  has an activation  $a_n$ , let  $n^+$  be the neighboring node of  $n$  along the strongest gradient of activation:  $n^+ = \text{argmax}_{n' \in nb(n)}(a_{n'})$ .

```

calculate_matching ( $S, T$ )
  returns  $\text{match}[n] \in \{0, 1\}, \forall n \in N$ 
1:  $\text{match}[n] := 0 \forall n \in N$ 
2: spread activation  $a$  from all  $s \in S$  ( $a_s = 1$ )
3: while there are unmatched nodes in  $T$ 
4:   for all unmatched nodes  $t \in T$ 
5:     {set  $b_n := 1$  for all nodes  $n$  on path from  $t$  to nearest disc}
6:      $b_{t^+} := 1$ ; if ( $b_n = 1$ ) then  $b_{n^+} = 1$ 
7:     if ( $b_s = 1$ )  $\wedge$  ( $a_s = 1$ ),  $s \in S$ 
8:       {match  $s$  to  $t$ , where  $t$  originated  $b$  activation that reached  $s$ }
9:        $b_s := 2$ ; if ( $b_n = 2$ ) then set  $b_{n'} := 2, \forall n' \in nb(n) \wedge b_{n'} = 1$ 
10:      set  $\text{match}[n] := 1$  for all nodes  $n$  on path from  $s$  to  $t$ 
11:       $a_s := 0$ 

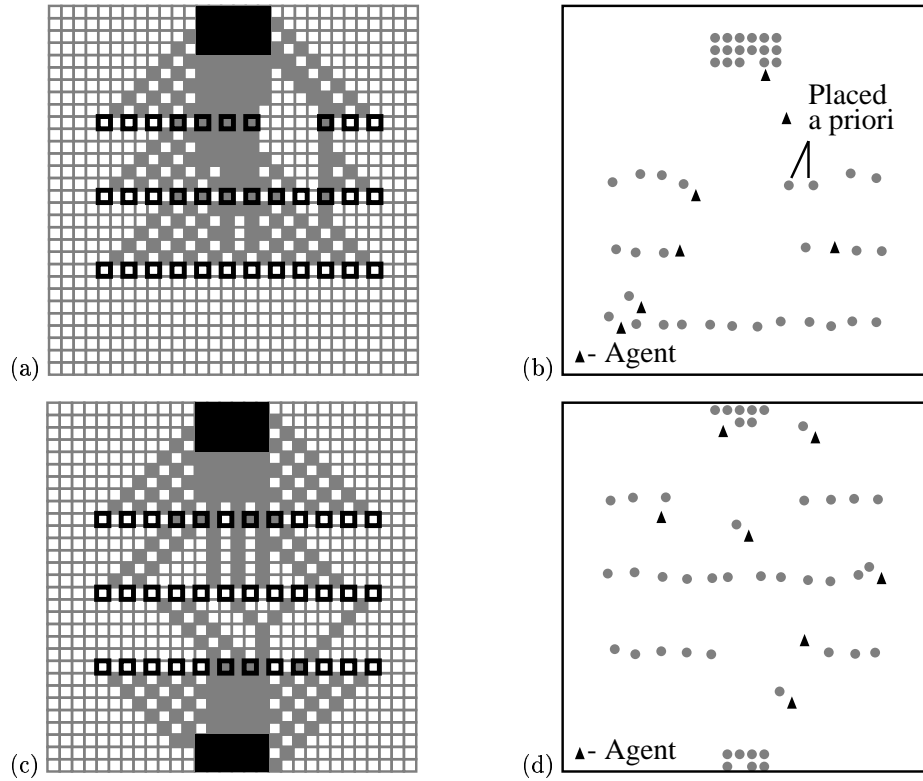
```

In lines 4-6, every drop-site location  $t$  spreads activation  $b$  *against* the gradient of the spreading activation initiated (from discs) in line 2. Thus,  $b_n = 1$  identifies paths between discs and drop sites. When  $b$  activation reaches a disc location  $s$  (line 7),  $s$  and  $t$  are matched by setting  $\text{match}[n] = 1$  for all nodes between them. The nodes on this path are identified by spreading  $b_n = 2$  from the disc location (line 9). Activation from more than one drop site might reach  $s$ , but only the closest drop site (whose activation reaches  $s$  first) is matched since  $a_s$  is reset to 0 in line 10 after  $b$  activation reaches  $s$ .

If the path between a disc and a drop site crosses another drop-site  $t$ , then  $\text{match}[t] = 1$ . Thus, drop-sites whose match value is 0 are chosen first with inhibitory connections from the nodes in the Matching grid to the corresponding neurons on the CNM. To identify the discs that are to be moved to the selected drop-sites, activation is spread along those nodes  $n$  with  $\text{match}[n] = 1$  from the selected drop-sites. If this activation reaches a node representing a disc location, then the corresponding neuron in the DESM is excited.

## 4 Results and Discussion

The performance of the algorithm is studied in both structured (discs and drop-sites grouped into piles/walls) and unstructured (discs and drop-sites randomly placed) environments. The number of agents is varied from 1 to 10 and they are initially distributed randomly within a square of  $100 \times 100$  units.  $\text{match}[n]$  values are shown for structured environments that consists of three parallel walls and one source of discs (figure 3a) and two sources of discs ((figure 3c). The positions of discs while construction is in progress are shown in figures 3b, d. With one source of discs, the sequencing algorithm fills the farthest wall first, then the



**Fig. 3.** match values (gray squares) for 3 walls (dark unfilled squares) and (a) one source (dark filled squares) and (c) two sources of discs. (b, d) Corresponding environment with 7 agents at an intermediate stage of construction.

middle wall and finally the nearest wall (while the greedy algorithm fills them in the reverse order). Note that two discs were already in place and activation flowed around them. With two sources of discs, the agents assign the outer walls to the top and bottom piles. The middle wall is filled from both the piles and is the first to be built. The greedy approach builds the outer walls first and agents have to move around these to build the middle wall. The sequencing algorithm has no advantage in unstructured environments since the probability that three or more discs/sites are collinear is small.

## 5 Conclusions and Related Work

The simulation results show that planning the sequence of disc placement reduces the total distance traveled to complete the construction task. Implementing good heuristic solutions to the MWPM problem directly on the spatial representation is a convenient way of reducing interference between agents without the need for communication. The algorithm also takes into account obstacles on paths

between discs and drop-sites. All computations are performed using only local interactions between neighboring nodes and hence this algorithm can be efficiently implemented in parallel systems.

Hopfield networks have been used for solving combinatorial optimization problems [5] and a formulation to solve the perfect matching problem is given in [7]. However, these methods do not always give good solutions [12] because of the presence of hard constraints [4]. Moreover, the weights in such a network have to be pre-computed by hand. Solving path computation problems using spreading activations is an idea borrowed from the working of ant colonies [3] and has been used for data dispersion in sensor networks [6]. In these works, activation is spread in the forward direction from a single source node and a reverse activation is then directed against the gradient of the forward activation. Spreading activation has been used to control a large number of simulated micro-robots that destroy a brain tumour [9] and to indicate paths that have been discovered by a group of robots to a human situated away from the robots [11]. In our work, the activation is spread on an internal spatial representation to plan a path for a single mobile agent. Goal sequencing strategies that do not explicitly consider the blocking of paths by previously dropped discs are studied in [10]. A comparison of average-case bounds for greedy MWPM heuristics is given in [1].

## References

1. D. Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13:475–493, 1983.
2. G. Chao, A. Panangadan, and M. G. Dyer. Learning to integrate reactive and planning behaviors for construction. In *Proceedings of the 6th International Conference On the Simulation Of Adaptive Behavior*, 2000.
3. G. Dorigo, M. Di Caro. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–265, 1998.
4. A. H. Gee, S. V. B. Aiyer, and R. Prager. An analytical framework for optimizing neural networks. *Neural Networks*, 6:79–97, 1993.
5. J. Hopfield and D. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.
6. C. Intanagonwivat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc of the 6<sup>th</sup> Annual Intl. Conf. on Mobile Computing and Networks (MobiCOM 2000)*, 2000.
7. S. Y. Kung. *Digital Neural Networks*, chapter 9. PTR Prentice Hall, 1993.
8. J. M. Kurtzberg. On approximation methods for the assignment problems. *J. Assoc. Comput. Mach.*, 9:419–439, 1962.
9. M. A. Lewis and G. A. Bekey. The behavioral self-organization of nanorobots using local rules. In *Proc. of the 1992 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 1992.
10. A. Panangadan and M. G. Dyer. Construction by autonomous agents in a simulated environment. In *Proc. of the Intl. Conf. on Artificial Neural Networks*, 2001.
11. D. Payton, M. Daily, R. Estkowski, M. Howard, and C. Lee. Pheromone robotics. *Autonomous Robots*, 11(3), 2001.
12. G. V. Wilson and G. S. Pawley. On the stability of the travelling salesman problem algorithm of hopfield and tank. *Biological Cybernetics*, 58:63–70, 1988.