Collective construction of numerical potential fields for the foraging problem

Olivier Simonin, François Charpillet, Eric Thierry

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

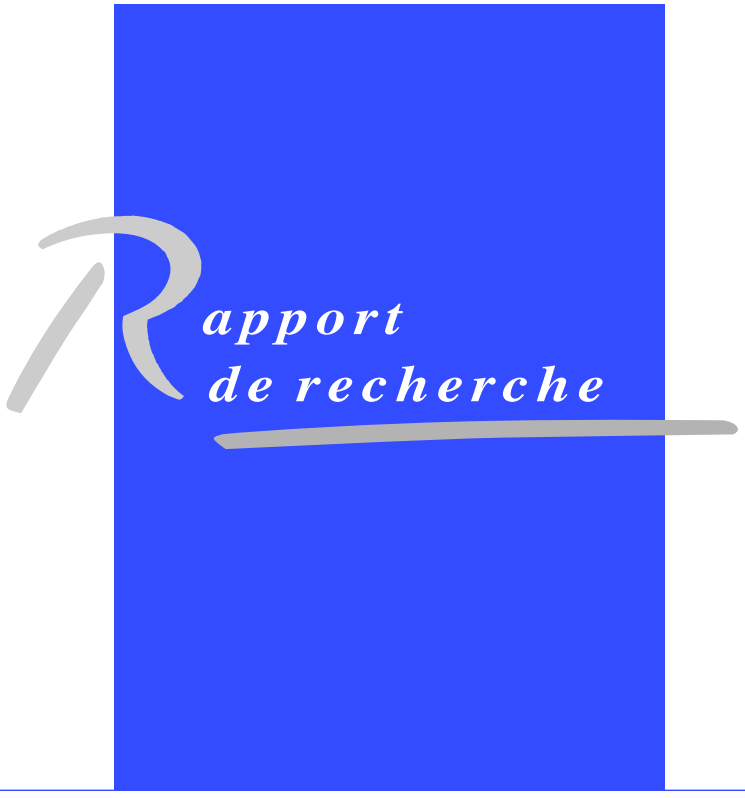
Collective construction of numerical potential fields for the foraging problem

Olivier Simonin — François Charpillet — Eric Thierry

N° ????

Avril 2007

Thème COG



ISSN 0249-6399

Collective construction of numerical potential fields for the foraging problem

Olivier Simonin* , François Charpillet* , Eric Thierry†

Thème COG — Systèmes cognitifs
Projets MAIA

Rapport de recherche n° ???? — Avril 2007 — 20 pages

Abstract: We consider the problem of deploying a team of agents (robots) for the foraging problem. In this problem agents have to collect disseminated resources in an unknown environment. They must therefore be endowed with exploration and path-planning abilities. This paper presents a reactive multiagent system that is able to simultaneously perform the two desired activities - exploration and path-planning - in unknown and complex environments. To develop this multiagent system, we have designed a distributed and asynchronous version of Barraquand's algorithm that builds an optimal Artificial Potential Field (APF). Our algorithm relies on agents with very limited perceptions that only mark their environment with integer values. The algorithm does not require any costly mechanism to be present in the environment to manage dynamic phenomena such as evaporation or propagation. We show that the APF built by our algorithm converges to optimal paths. The model is extended to deal with the multi-sources foraging problem. Simulations show that it is more time-efficient than the standard pheromone-based ant algorithm. Moreover, our approach is also able to address the problem in any kind of environment such as mazes.

Key-words: Artificial Potential Field – collective robotics – foraging task – bio-inspired algorithm –

* MAIA: http://maia.loria.fr
† Lip ENS Lyon

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Construction collective de champs de potentiels numériques pour le problème du foraging

Résumé : Pas de résumé

Mots-clés : Pas de motclef

1 Introduction

Swarm intelligence is now recognized as a robust and flexible approach to deal with distributed problem in complex environments. Synthetic phromones is one of the most popular swarm techniques that provides adaptive solutions to problems such as path-planning [22][25], patrolling [27][26] and foraging [21].

In these days of ubiquitous and cheap computing, the implementation of swarm and collective systems in real world has become more realistic and imaginable. Workplaces, for example, are already equipped with pervasive sensors and ad hoc, self-organized wireless networks (see [17] for details). The RFID technology makes possible to deploy of hundreds of sensors in indoor environments or, to spread them out in the thousands outdoors (as dust sensors, [12]).

In this paper we re-examine such an environment-based approach to study optimal path planning and exploration for the foraging problem in complex environments. The foraging problem is defined as a group of autonomous agents (such as robots) exploring an environment for resources to bring back to a *base* (see [28]). Distributing a foraging activity among a group of agents is particularly challenging when the environment is unknown (no map, no information about obstacles) and contains "awkwardly" shaped obstacles. The lack of environmental knowledge is typical of most realistic situations, such as military campaigns (e.g., the transport of supplies in a hazardous terrain), rescue tasks (e.g., surveillance of catastrophe-hit areas) [2] and planet exploration (e.g., robotic rovers that search for interesting rock samples on a planet [28][6]).

Ants inspired a standard pheromone-based technique to forage with agents/robots [8] [11] [24]. In this model agents drop pheromones in the environment in order to build gradients from resources to their base [5]. This approach, though each agent itself is very simple, requires a large mass of agents and a considerable amount of time for best paths to emerge. Moreover, the pheromone-based technique requires the computation of propagation and evaporation dynamics. Another drawback of this approach is that each agent needs an ad-hoc mechanism for getting back home (for example, compass information). [24] and [21] proposed to use a second pheromone diffusing from the base in order to avoid this problem, but without guaranteeing to build no local minimal in complex environments.

In this paper we explore an original approach that allows to build *optimal paths* for foraging using only reactive (simple) agents, who do not have a map of the environment, and in which the environment is used as a shared memory *but not requires any particular dynamics*. We define agents that do not use pheromones, but instead directly write a gradient as they explore and discover the environment. Such an approach is inspired from centralized planning in which the environment is known. We refer to popular techniques based on Artificial Potential Fields (APFs) approaches [13][15][1].

The APF contruction algorithm proposed in [4] is promising in this regard since it prevents the formation of local minima while computing the shortest paths to the agent destination. In this paper, we define a *distributed* and *asynchronous* version of the [4] algorithm. We prove that our algorithm converges to the optimal APF, i.e., shortest paths are indeed found. Then we show that the model can be efficiently applied to foraging tasks. Performances are analysed and compared to the standard ant-model.

The rest of paper is organized as follows. In section 2, we describe [4] algorithm; specifically, the construction of a numerical potential field and optimal path execution. In section 3, we describe a reactive multiagent system that builds the numerical potential field and prove its convergence to the optimal APF. In section 4, this system is extended to tackle the foraging problem (the corresponding algorithms are given in detail). Then, in section 5, we report statistical measures on performance and comparisons with the ant model. In Section 6, we discuss related work and some clues for a real-world implementation of our approach. Finally, in section 7, we summarize our conclusions.

2 Barraquand's numerical potential field

In order to define agents that can compute optimal paths in a complex environment, we first examine the pre-computed map proposed by Barraquand and colleagues [3] [4]. In the next section, we show how this global pre-computation can be distibuted in a reactive multi-agent system and later, in section 4, applied to the foraging task.

In Barraquand's approach, the environment is represented as a grid. This facilitates the building of potential fields numerically, rather than analytically. Local minima are avoided by computing a "wavefront expansion" from the agent destination, which involves building an ascending APF incrementally.

The environment is represented as a 2D-array of cells and it is denoted by E (see e.g. Figure 1.a). E_{empty} denotes the subset of E containing all positions free of obstacles. Each point $x = (i, j) \in E$ has a maximum of 4 neighbours:
$\{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$.

A potential field is a function $V : x \in E_{empty} \longmapsto V(x) \in \mathbb{R}$

Let x_{goal} be the goal position of the robot in E_{empty}. The Wave-potential V is computed as shown in Algorithm 1 (L_i denotes a list of points in E). The algorithm terminates when E_{empty} has been totally explored. Figure 1.b presents the

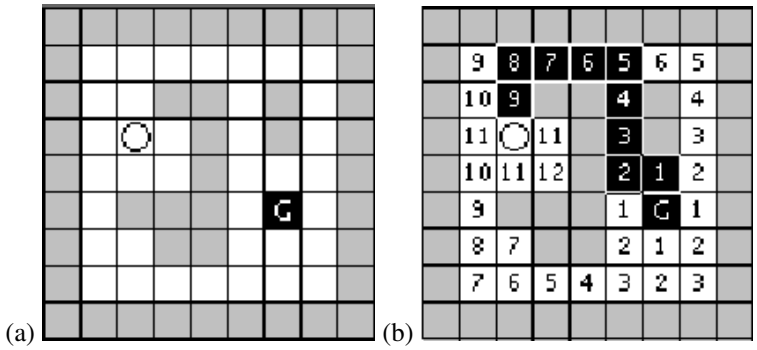


Figure 1: WaveFront algorithm: initial environment state and result of the construction

Algorithm 1 WaveFront expansion Algorithm [4]

For every cell $x \in E_{empty}$, set $V(x)$ to infinity (e.g., a large number M)
Set $V(x_{goal})$ to 0 and the list L_0 to (x_{goal}), $i = 0$
While L_i is not empty Do
 initialize L_{i+1} to the empty list,
 For every point x in L_i,
  For every neighbour y of x in E_{empty},
  if $V(y) = M$ then set $V(y)$ to $i+1$ and insert y at the end of L_{i+1}
 $i = i+1$

result of the algorithm execution on a test environment where the goal position is located at the letter G (each point has a potential value representing the minimum distance to the goal).

We remark that this algorithm is essentially the classic BFS (Breadth First Search) algorithm [7], which computes the path with the least cost between two vertices in a graph, applied to the task of robotic path planning by discretizing the environment.

Using this computation, an agent can now reach the goal by just following the flow of the negative gradient vector field $-\overrightarrow{\bigtriangledown}V$. In a numerical potential field, such a behavior is called a "descent" and is defined as: at every iteration, the agent examines the potential values of the four neighbouring cells and moves to the cell with the smallest value (ties are broken arbitrarily).

In Figure 1.b the agent is represented by a circle and the black cells represent a descent path. The following two properties hold for Algorithm 1 (see details in [4]):

Property 1: In the *Wavefront* numerical potential field produced by Algorithm 1, an agent that always moves to the neighbouring cell with the smallest potential value arrives at the goal location succeeding to avoid all obstacles.

Property 2 : From any location in the *Wavefront* numerical potential field, the path induced by Property 1 is the shortest.

Our objective is to conceive a *multiagent* approach to build a numerical potential field such as the one produced by Algorithm 1. We therefore define a distributed and asynchronous version of Algorithm 1 (i.e., the Barraquand Algorithm).

3 WaveFront computation with reactive agents

As stated before, we are interested to deploy agents that do not have information about their environment. Then the algorithm 1 we just described cannot be directly computed before launching agents for a task such as foraging. We therefore propose to transform the algorithm in a *group* of reactive agents able to build a comparable numerical potential field while exploring the environment. Reactive agents have no individual memory abilities but can cooperate and build patterns through indirect communication. This form of communication is embedded in the environment. It takes its inspiration from ants that drop pheromones as markings to be read at a later time by others (including possibly themselves) [11]. The environment allows to aggregate the local interactions of numerous agents in order to build collective patterns. Such a process is called *swarm intelligence* ([5]). The environment is then treated as a shared memory in which information can be stored and erased.

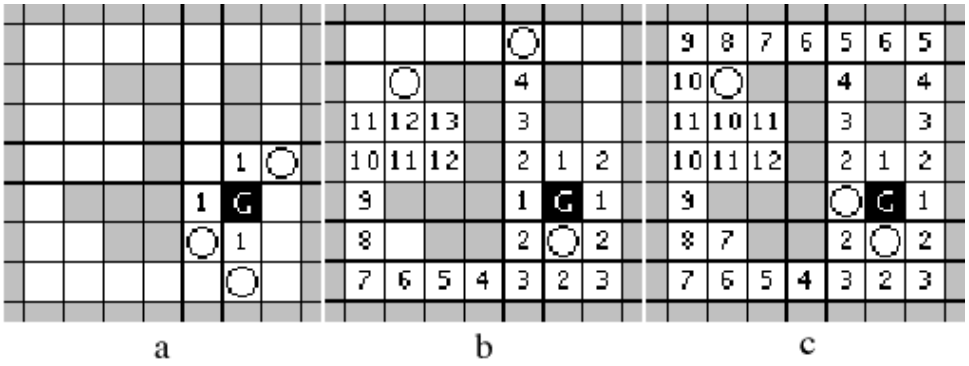


Figure 2: Collective construction of the potential field with 3 agents (steps 2, 17 and 65)

Thus we have the central idea of our model: we treat the numerical values of the APF to build as markings that can be laid down and updated by agents.

3.1 Simple "marking agents"

We now describe the environment and the agents used for the APF construction. The environment is discretized in regular cells. At any time, agents do not know their global location, they just occupy a cell and perceive the 4 neighbour ones. Without loss of generality, in the interest of simplicity, we allow multiple agents to occupy a cell at the same time. Each agent is capable of,

- *reading and writing integers values* on the cell at which he currently is,
- perceiving the 4 neighbouring cells: detecting if any of the cells is an obstacle and reading their values
- moving to a neighbouring cell that is not an obstacle

We call such agents *marking agents*. Initially, the value of the goal cell is 0 and all the agents are placed on it (agents can start from any position but the construction is then slower).

The wavefront potential is collectively and incrementaly built around the initial value through the actions of the agents. Each agent repeatedly follows the steps as represented by Algorithm 2 (EXPLORATION & APF CONSTRUCTION). After choosing a cell and moving, the agent computes the UPDATE-VALUE operation which corresponds to a local wavefront expansion (when moving from a wavefront cell to a cell not yet visited). Then the shortest path from this new cell to the goal cell, in the current wavefront, is equal to
the shortest path to a neighbouring cell + 1 (one move).
The update operation is performed if the new value is lower than the current one.

Algorithm 2 EXPLORATION & APF CONSTRUCTION (Agent behavior)

EXPLORATION
IF there exist neighbouring cells ***without a value*** THEN
  move randomly to one of them and UPDATE-VALUE
ELSE move ***randomly*** to a cell which is not an obstacle and UPDATE-VALUE

UPDATE-VALUE Operation
Compute $val = min(val, 1 + min(\text{4-neighbour values}))$
Write (update) *val* in current cell

As the movements of the agents are based on a random walk, they explore the environment while they mark it. However, it is not just a random exploration because the values of the cells (the markings) influence the choice of each agent in selecting a cell to move to: each agent favors *unmarked* cells over marked ones (see the first line of algorithm 2). As a consequence, their exploration progresses as a partial circular front around the starting cell. Figure 2.a shows such a progression with 3 agents.

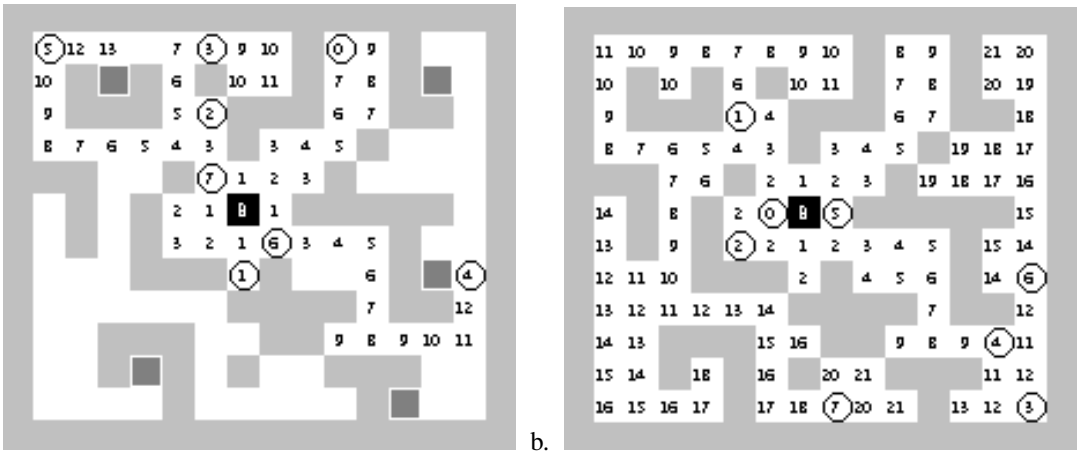


Figure 3: Foraging with eight marking agents in a little maze (a) after 18 iterations (b) final state

The number of agents will not be generally sufficient to sustain an entire circular front progression. So one exploration step, when all the agents move, is not equivalent to one loop of algorithm 1. Figure 2.b shows a situation where the cell with value 13 has clearly not reached the value of the shortest distance to G (which is 11, see Figure 2.c). Then agents may need to visit several times the same cell in order it reachs its optimal value. As the environment is bounded, the time necessary to obtain the complete optimal potential field is also bounded, i.e. the algorithm converges (Figure 2.c), see proof below.

3.2 Convergence

Theorem 3.1 *In a bounded discrete environment containing a destination cell c_0 with potential 0 and some obstacles, a set of agents that follow algorithm 2 build in a finite time the entire wavefront from c_0, i.e. the optimal APF.*

Proof: we define the following subset: let L_i denote the set of cells which attain the value i such that the distance to c_0 is i (i.e. a path with optimal length i). It follows that when all L_i sets are built, the system has converged to its final state (the optimal APF).
In order to proove covergence to the optimal APF, we need to show that each L_i is built in finite time. For this purpose, we consider the succesive constructions, L_0, L_1, L_2, ...
Note that L_0 is equal to the single cell c_0, which is assigned its optimal value 0 before agents begin exploring. The proof then involves showing inductively that if L_n has been built, the system will compute L_{n+1} in a finite time.
Assume L_n is already built. Each cell c_j in L_{n+1} is necessarily a neighbouring cell of some cell in L_n cell. Indeed, the maximum gradient difference between two cells in the optimal APF is one. So, when any c_j cell is visited, its value changes to the final value $n+1$. This value cannot be lower, otherwise the cell would belongs to some set L_k with $k \leq n$, but each such set has been already built, so that cj is not an element of any such set. Moreover, this value cannot be higher due to the definition of the Update operation (algo. 2).
So, to build L_{n+1} from L_n, each cell in L_{n+1} must be visited at least once. Visiting each cell in L_{n+1} takes a finite time because agents do a random walk. Indeed Alg. 2 guarantees that each cell of the environment will be visited once with probability 1, in a finite time.
As each L_i is built in a finite time, the sum of times required to build L_1, L_2,.. and L_{max} is also finite. The assumption that the environment is bounded implies that L_{max} exists. Thus convergence is proved. □

Note that while the participation of multiple agents speeds up convergence (see section 5), this result is true even when only one agent is deployed. The next section shows how this multi-agent construction can be efficiently applied to the problem of collaborative foraging.

4 Application to collaborative foraging

4.1 The multi-source foraging problem

In collaborative foraging, also known as the exploring robots problem [28], mobile agents such as robots *search* for resources in a designated area, and *transport* found resources back to a *base*. Additionally, they may be required to *return* to the places where they found resources if they could not collect all the resources at those places in earlier trips. The robots are assumed to have no knowledge of the environment. The environment may and usually does contain obstacles. The difficulty of foraging depends on the number and shape of obstacles present.
In practice, this last assumption - that the robots forage in an environment unknown to them - is somewhat relaxed by making some other assumption, such as the robots being endowed with a compass or the base being able to communicate with the robots. However, even such facilities cannot help the robots in navigating through obstacles, especially if they are "awkwardly" shaped. In our work, we do not require the robots or agents to have any such extra external facility.

4.2 Foraging using marking agents

We consider now environments that have resources in multiple locations. These locations are of course unknown to the agents. Each location has a given quantity of resource. As illustrated Fig 3.a, a cell in the environment can,

- be an *obstacle* (grey colour),
- contain a *resource* (dark grey colour) whose quantity of resource q_i is bounded,
- be the *base* ('B' letter in black cell), from where all agents start ,
- contain an *agent* (whose ID number is shown in a circle in the cell).

The agents begin all from the base. Note that a cell may contain more than one agent, but sharing a cell has no effect on the actions taken by each agent in it. In addition to the abilities mentioned in the previous section, an agent is able to

- detect the presence of resources in the 4 neighbouring cells,
- load a quantity q_{max} of resource onto itself for transporting back to the base.

Building a wavefront from the base is usefull for any task that requires the agents to traverse the environment many times (e.g. to manage their energy). This is just the case in the foraging problem, where agents may have to make multiple trips from the resource locations to the base to transport back all the resources. Using the wavefront, they just have to descend the numerical gradient to come back without being blocked by obstacles.
Thus, the agent we have just described is essentially a marking agent (cf. section 3.1) with the ability to come back the base as often as it is required to. We define this re-entrant behaviour of the agent as descent motion (presented in section 2). Algorithm 3 represents the complete behaviour of an agent. It is also shown schematically in Figure 4.

Algorithm 3 SEARCH & RETURN TO BASE

SEARCH
IF a ***resource*** *is detected* in a neighbouring cell THEN
  move into this cell, load a quantity q_{max} of resource and execute RETURN to BASE
ELSE execute EXPLORATION & APF CONSTRUCTION (i.e. Algo. 2)

RETURN TO BASE
IF the agent is located at ***base*** THEN unload resource and execute SEARCH
ELSE *move* to a neighbouring cell ***that has the least value*** and UPDATE-VALUE

This is a sort of a rudimentary definition of marking agents who forage. One can note that three tasks can be simultaneously performed by such a multiagent system: (i) *exploration* to discover resources, (ii) *construction* of an optimal APF and (iii) *transportation* of resources to the base. Note that while the third task can be done by one or more agents, the two first tasks are carried out thanks to cooperation between all the agents. Figure 3 illustrates the progression of foraging in a little maze containing 5 location resources (3.a) and the final state attained after 500 iterations (3.b) showing the optimal APF.

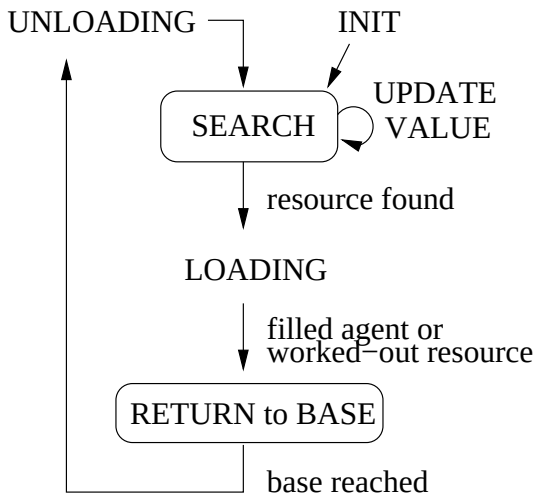


Figure 4: Behaviour of a marking agent in foraging

4.3 An interesting property of the APF contruction

The parallelization of the three tasks is time-efficient (we elaborate in the next section) but has the drawback stated in Section 3.1. As agents explore the environment, they do not immediately build an optimal APF. The value written in a cell the first time it is visited is not necessarily the optimal (i.e., limiting) value. It is a consequence of the random nature of the agents' exploration which implies that the agents do not discover the shortest path to a resource when that resource is detected.
However, our algorithm has one interesting property: when an agent detects a resource *at least one valid obstacle-free path exists* to return back to the base. In fact, one of these paths is a path marked by the discovering agent itself. As stated above, this path is valid but not neccessarily the shortest (only Property 1, section 2 is valid). However, continuous exploration by the agents improves the APF values until there is convergence. If the environment is finite, the values converge (as was proved in the previous section). Thus, Property 2, of optimal path planning, is attainable in a finite time (as is the case in Figure 3.b).
One may wonder if the agents can get stuck in local minima since this construction is asynchonous. The answer to this query is NO. The following lemma ensures this property.

Lemma 4.1 *During the APF construction, for each cell c different from c_0, there exists at least a neighbouring cell with a value lower than cell c.*

Proof: Recall that if an agent moves to a cell, the following update is made: $val = min(val, 1 + min(\text{4-neighbouring values}))$. Consider any cell c_i different from c_0. Every time an agent visits c_i, its value can change to $1 + min(\text{4-neighbouring values})$, which results in at least one neighbouring cell having a value lower than c_i. Now, even if the neighbouring cells of c_i are themselves visited, their values can only change for lower ones (update operation can only decrease the current value *val*). So these visits cannot change the property that c_i has at least a lower neighbouring. □

Lemma 4.1 permits to any agent that evolves in the APF under construction to perform a descent to the base, since from each cell it always exists a lower neighbouring to move. As a consequence, at any point of the exploration, from any cell, our algorithm provides a valid path to any agent to return to the base, irrespective of the number and shapes of the obstacles present.

However, our algorithm in the version presented above does not guide the agent in *returning* to a location that has been discovered to contain resources. In the next section, we show how previously followed paths can be reused and communicated to other agents, in order to improve the foraging.

4.4 Colour-marking (c-marking) agents: cooperation through numeric trails

We now describe how the previous foraging algorithm may be extended so that agents can retrace previously discovered resource locations from the base. Since the locations are not known to the agents *per se* and since agents cannot communicate with one another, the environment itself is turned into a communication tool. In the previous section, we saw that

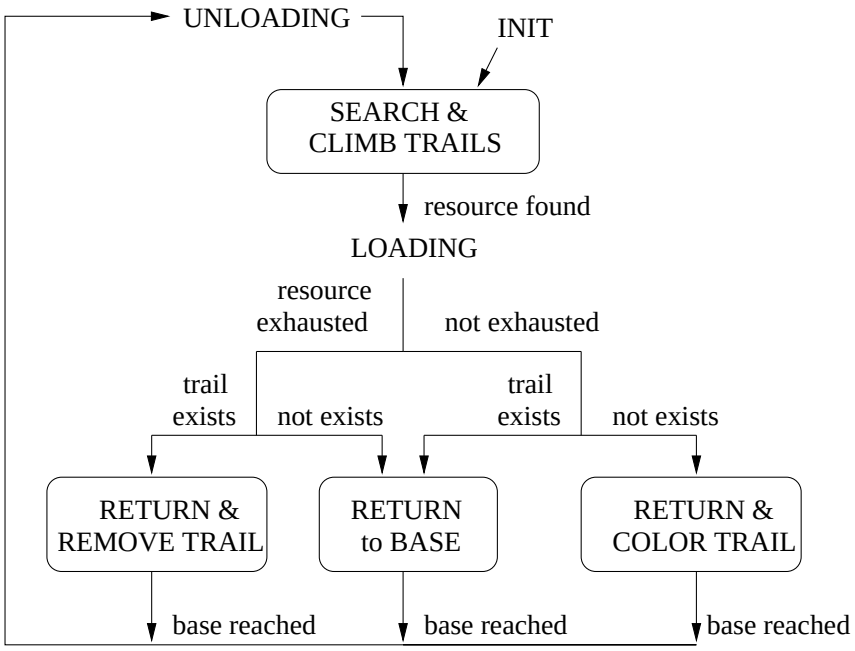


Figure 5: c-marking agents that colour paths between sources and the base

once a location containing a resource is discovered, the task RETURN TO BASE *makes the agent follow a valid path to the base*. The idea is to make the agent marks this path as a trail while returning to the base. Then returning to discovered resource locations will be as simple as following these trails.
In order to allow agents to mark trails in the numerical potential field, we give the agents the ability to write values in a *specific colour*. So when an agent starts its return to the base from a resource location, it rewrites the values in the cells it followed using a *distinctive trail colour* (this task shall be called RETURN & COLOUR TRAIL). Trails marked by an agent can then be used by other agents. Thus the coloured values serve as a form of indirect communication. It is important to note that coloured trails do not change the potential field construction. Besides, as we show now, coloured trails can be easily detected and erased by the agents.
When a location becomes exhausted of its resources, the trail leading upto it must be erased as quickly as possible, since now it is of no use to any agent. In the ant-model, the pheromone evaporation mechanism automatically erases useless trails. In our pheromone-less approach, this task is done by the module RETURN & REMOVE TRAIL, in which the agent does a simple descent of the coloured trail rewriting the values using the *default colour* (signifying absence of a trail). The behaviour of c-marking agents is shown in Figure 5, and the enhanced algorithm corresponding to c-marking agents is as follows:
The RETURN task defined in previous section has now two new versions: RETURN & COLOUR TRAIL (see Algorithm 4) and RETURN & REMOVE TRAIL. The latter task is identical to RETURN & COLOUR TRAIL except in the selection of the next cell and the colour used to write values. In RETURN & REMOVE TRAIL, the ELSE block of RETURN & COLOUR TRAIL is replaced by,

Move to a new neighbouring cell that has ***the same colour as the trail***
Change the cell's colour to ***the default colour***

The LOADING task is now available with a choice of the type of RETURN task (see Algo. 4) which depends on the state of the discovered resource (exhausted or not), The default RETURN to BASE task is the same as in the previous section (i.e., it is a descent of the gradient). Finally, the SEARCH task is extended to allow agents to *climb* trails when they discover one: see SEARCH & CLIMB TRAIL.

4.5 Simulation

We used the TurtleKit toolkit [18] (part of the MadKit Platform [10] [19]) for testing our algorithms. We tested with rectangular environments (grids of a variable size). Obstacles in the environment were defined in two ways: (i) given a desired density, cells were randomly designated as obstacles (ii) obstacles were specifically designed by hand (for example, a maze as in Figure 3). We illustrate the whole behavior of the system by describing an experiment with the following

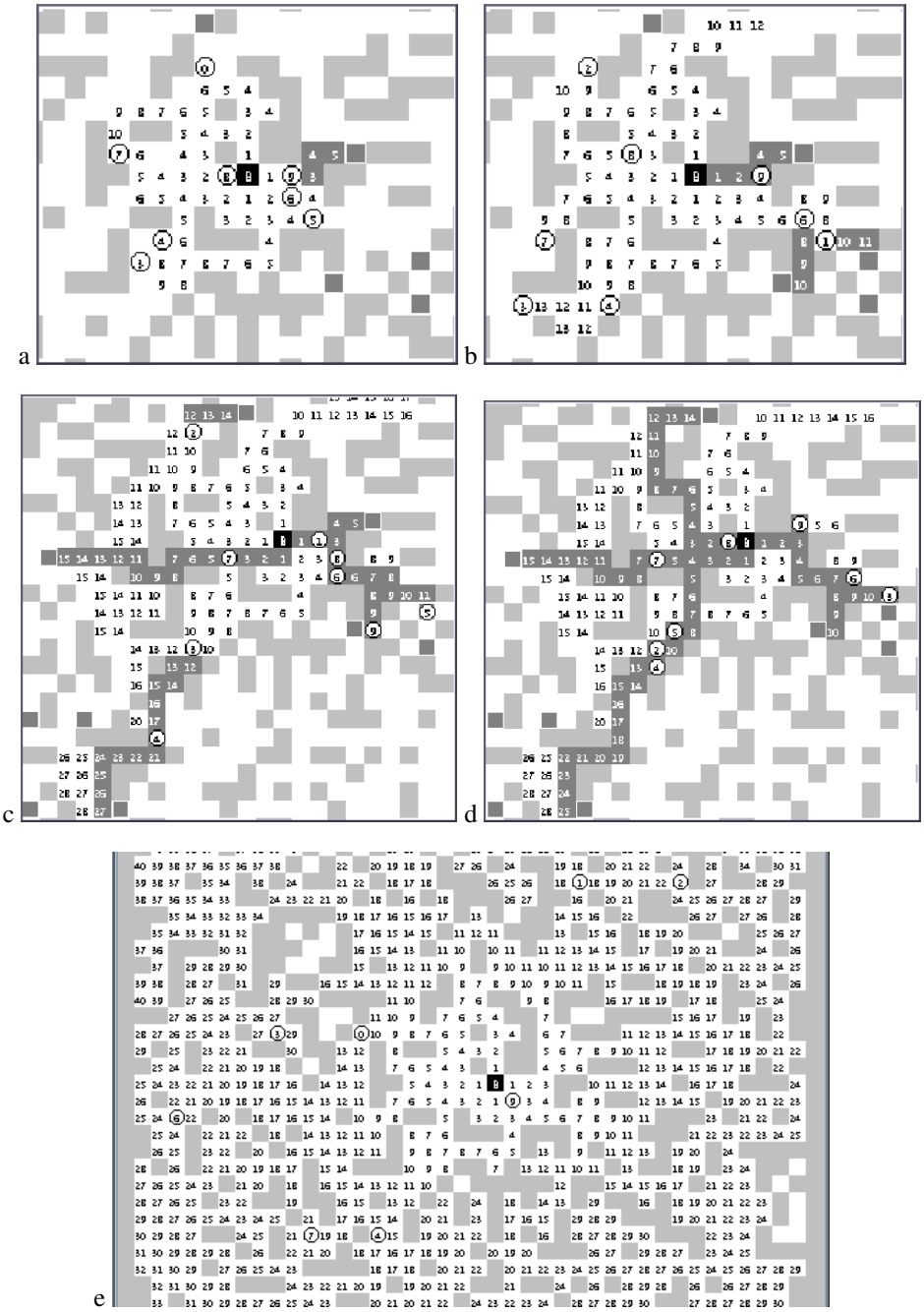


Figure 6: Resolution of the foraging problem with 10 c-marking agents (with **Setup 1**). Trails are represented in dark grey colour. Snapshots at step 11, 21, 53, 76, 1200.

Algorithm 4 colour Marking Agent

SEARCH & CLIMB TRAIL
IF a resource is detected in a neighbouring cell THEN
  move into that cell and run LOADING
ELSE
  IF neighbouring cells are coloured and different from the previous position
  THEN move to highest-valued such cell
  ELSE execute EXPLORATION & APF CONSTRUCTION (i.e. Algo. 2)

LOADING
Pick up a quantity q_{max} of resource
IF the cell is now not exhausted of resources THEN
  IF the cell is coloured **(it is part of a trail)**
  THEN execute an algorithm equivalent to RETURN TO BASE
  ELSE execute RETURN & COLOUR TRAIL (create a trail in a specific colour)
ELSE execute RETURN & REMOVE TRAIL (remove a trail)

RETURN & COLOUR TRAIL
IF the agent is located at ***the base*** THEN
  unload resource and execute SEARCH & CLIMB
ELSE
  Move to a new neighbouring cell with the ***least value***.
  Colour the cell in ***a trail colour***

first setup:

Setup (1):

- A 40 cells x 40 cells environment, 30 % of the cells are obstacles and they are randomly distributed. 20 cells are resource locations, whch are also randomly distributed. Each resource location contains 1000 units of resources.
- Each agent can load a maximum of 100 units onto itself.

We ran 10 c-marking agents who were initially located at the base, which is in the center of the environment. We observe in Figure 6.a that the construction of the numerical potential field starts out as a pseudo-circular front computation (step 11). We can also observe that agent 0 has discovered a resource and started to create a trail as it returns to the base. Figure 6.b shows that at step 21 two other agents have discovered resource locations and started to create trails, while agent 9 performs multiple trips from the base to the resource location he discovered. Later, step 53, Figure 6.c shows that six resource locations were discovered, many agents are involved in transport of resource along trails marked by others (note that an agent may follow a trail even if it does not connect to the base at that point in time, e.g. agent 4). This communcation through trails permits the recovery of all the resources in the environment while the APF contruction proceeds parallely. Figure 6.d shows step 76, where agents 2, 4 and 5 work simultaneously along the same trail (connecting to a resource location at the bottom of the snapshot). On the other hand, the trail connecting the first dicovered resource location is now removed (by agent 9) as it was exhausted.

Note that when agents leave the base, if there is several possible trails they can be selected with equal probability. So the distribution of the agents between several resources is "well-balanced".

At around the 1200^{th} iteration, all resource locations in the envrionment have been discovered and all the resources thereby exhausted, and what is more, the potential field has converged to its optimal value (Figure 6.e).

5 Evaluation of our algorithm and comparison with the ant-model

5.1 Methodology

Criteria We now study the performance of our algorithm (c-marking agents) and compare foraging time with the ant algorithm. We define time to be the number of iterations required by the agents to discover and exhaust all the resources of the environment. One iteration consists for each agent to read or write a value on its current cell and to move to a neighbouring cell. We evaluated the performance of the two algorithms (see 5.5 for evaluation of the ant algorithm) in different configurations (number of agents, size of the environment). To evaluate average performance, we repeated each

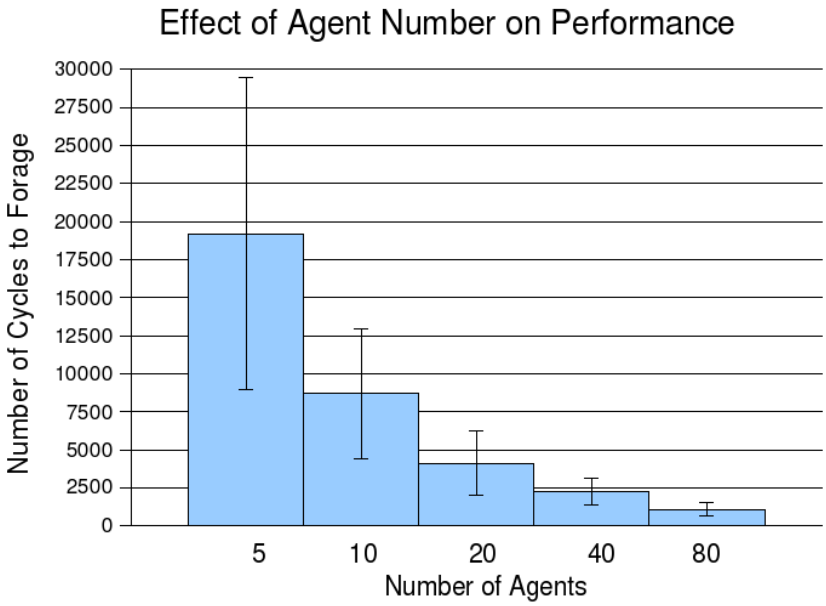


Figure 7: Performance of c-marking agents in **Setup (1)** (Table 1)

#Agents	5	10	20	40	80	160
# Iterations	19200	8697	4114	2263	1070	574
Std. deviation	10071	4282	1890	1134	448	420

Table 1: Influence of increase in the number of agents on the performance of the algoritihm in a 40 x 40 environment with 30% for obstacle density and 20 resource locations each holding a 1000 units of resources (Setup 1).

foraging many times in each configuration (up to 5000).
Basic environment In all experiments, environments are defined as rectangles. The base is always located in the center and all agents start from it. Obstacles and resource locations are spread out randomly in the environment.

5.2 Influence of the number of agents on performance

Experiments of our algorithm show that increasing the number of agents have always the same influence on performances (for every configuration). So, we first describe performances on a representative scenario which is **Setup 1** defined in the previous section.

Table 1 shows the performance of the algorithm in Setup 1 while the number of agents grows. It is represented graphically in Fig. 7. The algorithm was first evaluated for 5 agents and then this number was progressively doubled till 160.
As Table 1 and Figure 7 show, foraging becomes dramatically faster with an increase in the number of agents. With every doubling of agents, the number of iterations required for foraging is halved or reduced even further, which shows the effect of cooperation between agents (we study the super-linearity of their performance in the next subsection). The standard deviation of the number of iterations indicates the impact of the pseudo-random walk in the exploration.
We also evaluated the algorithm by varying other parameters such as obstacle density, resource location density, the quantity of resource at each location, besides the environment size and the number of agents. However, for any profile of parameters, we always obtained the same degree of change in the speed of foraging with a change in the number of agents. At the start of the curve (i.e., for small number of agents), there is a linear or super-linear decrease in the number of iterations. The curve is asymptotic as the number of agents becomes large, representing the fact that for any number of agents, a minimum amount of time is required to (i) reach the resource locations (ii) transport all the resources to the base.

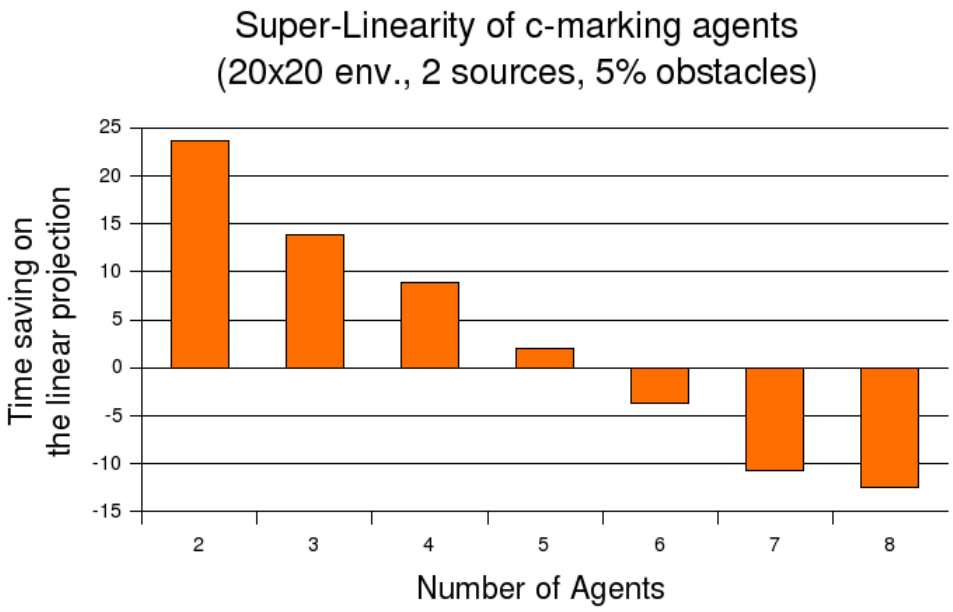


Figure 8: Difference in projected and actual number of iterations required, on average, to achieve foraging in **Setup (2)** - when varying number of agents - (Table (2))

#Agents	1	2	3	4	5	6	7	8
Experim. #iterations	**1790,4**	871,5	582,9	438,7	356,1	302,0	266,5	236,2
Projected #iterations	**1790,4**	895,2	596,8	447,6	358,1	298,4	255,8	223,8
Difference: Proj-Expe		23,7	13,9	8,9	2	-3,6	-10,7	-12,4

Table 2: Time saving on projected performances (in **Setup 2**)

The performance of the algorithm raises a few questions. What are the conditions under which a super-linear performance is achieved? What is the effect of the environment's size on performance? How does our algorithm compare to the pheromone-based ant algorithm ? We now attempt to answer these questions.

5.3 Super-linear performances

A multi-agent system is said to be super-linear if increasing the number of agents increases further than linearly the efficiency with which it performs a task. Such an increase in efficiency is given to represent a measure of cooperation between agents. We measured the performance of c-marking agents in different setups (varying environment size, number of resource locations etc) in order to study super-linearity. We observed that super-linearity does not appear systematicaly but does so for a certain range of the number of agents and for certain problem configurations. We saw in the previous section that increasing the number of agents decreases asymptotically the speed of convergence. So it is meaningful to study super-linearity for a small number of agents.
In order to improve accuracy, we reduced the environment size to 20 x 20 and the obstacle density in it to 5% (**Setup (2)**).
Setup (2): A 20 x 20 cell environment, 5% of the cells are obstacles, 2 randomly located resource locations each containing 1000 units of resources.

We conducted 5000 simulations to obtain the average performance of the algorithm for a given configuration. The number of iterations required by different number of agents (1 to 8) to finish forgaging is shown in Table (2) (top row).

The average number of iterations required by a single agent was found to be 1790.4. We denote this value as V_{ref}. We then computed the projected value of the average number of iterations required for different number of agents as a linear function of V_{ref}. For n agents, the projected value was V_{ref}/n. The difference between the projected and actual values measures super-linearity (time saving). Super-linearity is maximum in passing from one agent to two. Then it decreases. It becomes linear at about 6 agents. For more than 5 agents, the difference converges to a value, which is due to the

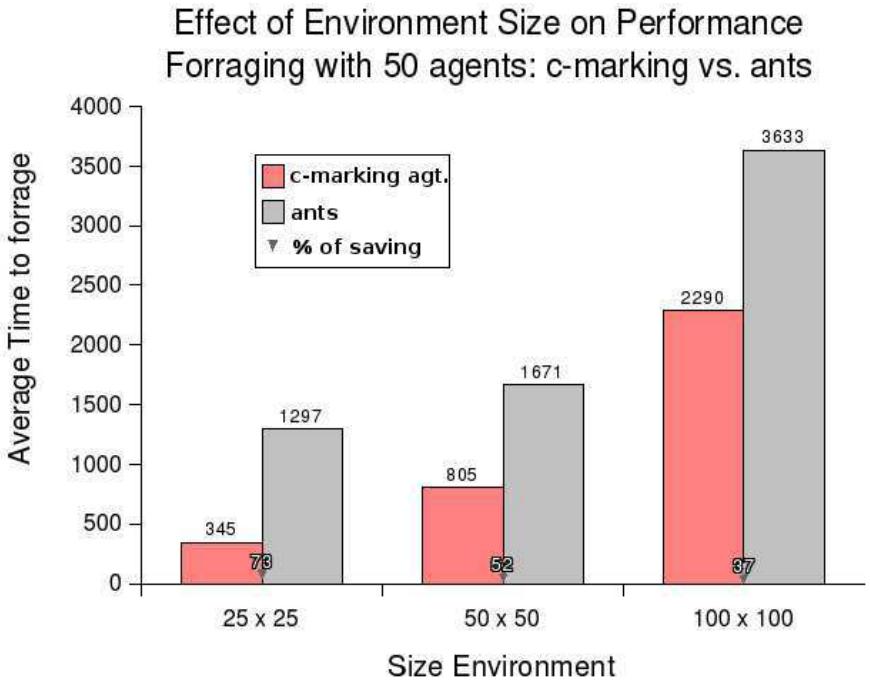


Figure 9: Comparision of the performance of the markings model with the ant-model as the size of environment is varied (**Setup (3)**: 50 agents, 5% of obstacles, 20 resources of 2000 units).

#Env Size$_k$	12x12	25x25	50x50	100x100	200x200
# iterations	155,5	345	805	2290	7844
R		2,22	2,3	2,8	3,43

Table 3: Influence of the size of the environment on the performance of the algorithm, using **Setup (3)**

asymptotic behaviour mentioned before. In our simulations, we could observe that the boundary of super-linear to linear performance can be as high as more than 160 agents (as seen in Table 1).

We conjecture that a super-linear performance is related the ratio (number of agents)/("complexity" of the configuration). By complexity, we mean the size of the environment and the number of resource locations. We intend to study this relationship in our future work. It is motivited by the idea that if the number of agents is small compared to the size of the environment, in one iteration, there is few chance that two agents do identical actions on the environement (due to distances between agents) and each action is useful directly or later to all agents.

5.4 Influence of the size of the environment

We now see how the size of the environment affects the performance of the algorithm when the number of agents is kept fixed. Obviously, a large environment will require more time for exploration and transportation of resources to the base. However, we present some interesting results using **Setup (3)** and by progressively quadrupling the size of the environment.

Setup (3):

- Environments are defined with a 5% obstacle density and 20 resource locations randomly distributed, each with 2000 units of resources.
- The number of agents is 50. Each agent can transport a maximum of 100 units of resources at a time.

The performance of the algorithm on progressively constantly quadrupling the size of the environment, from 12x12 to 100x100 is shown in Table 3 and graphically in Figure 9. Let the ratio $\#Iterations_{size_k}/\#Iterations_{size_{k-1}}$ be denoted by

R. Below 100x100, *R* is close to 2. Over 100x100, *R* quickly exceeds 4. In other words, the time to explore becomes exponential. It is because the pseudo-random walk the agents perform becomes more and more inefficent as the size of the environment increases. In the next section, we compare our results with the ant model in different environments.

5.5 Comparison with the ant model

We now evaluate quantitatively the performance of our algorithm with the classic ant model. In nature, ants deposit a chemical substance called pheromone in the area in which they explore. These deposits are gradually laid in the form of a gradient. This gradient creates paths from food sources to the ant nest [11]. We have implemented a standard algorithm based on the ant model ([24] [23]), shown in Figure 10. We computed several simulations to tune diffusion and evaporation rates in order to obtain the best possible performances. In presented results the diffusion rate is set to 95% and the evaporation rate to 0.5%.
Simulations (e.g. figure 11.b) show that ants climb these gradients to retrieve discovered resources. This form of cooperation is similar to the one used by marking agents. Figure 10.b details the search and climb behavior.

Such a model requires that each ant be able to:

- *drop an amount of pheromone* on the current cell and read the amount currently in the cell,
- perceive if any of the 4 neighbouring cells is an obstacle and read pheromone amount in each of those cells
- move to a neighbouring cell if it is not an obstacle,
- *know the direction of the base from its current cell*

Additionally, to follow the ant model, the following phenomena must also be simulated:

- pheromone evaporation
- pheromone propagation
- addition of an amount of pheromone to the pheromone already present in a cell

One can see that simulating the ant model requires more environment management mechanisms than the marking agents model. Note that the propagation of pheromones may represent a high computational burden.
Another drawback is that each 'ant' must have specific additional information about the location of the nest to be able to return to it. Such information is inadequate when complex obstacles are present in the environment. For example, it can be shown experimentally that ants can get stuck in cavities formed by just a few obstacle-cells (Figure 11.a). Even for a density exceeding just 5 %, such obstacles may form in a random generation.
By contrast, the main advantage of marking agents is their ability to find paths in environments containing obstacles of

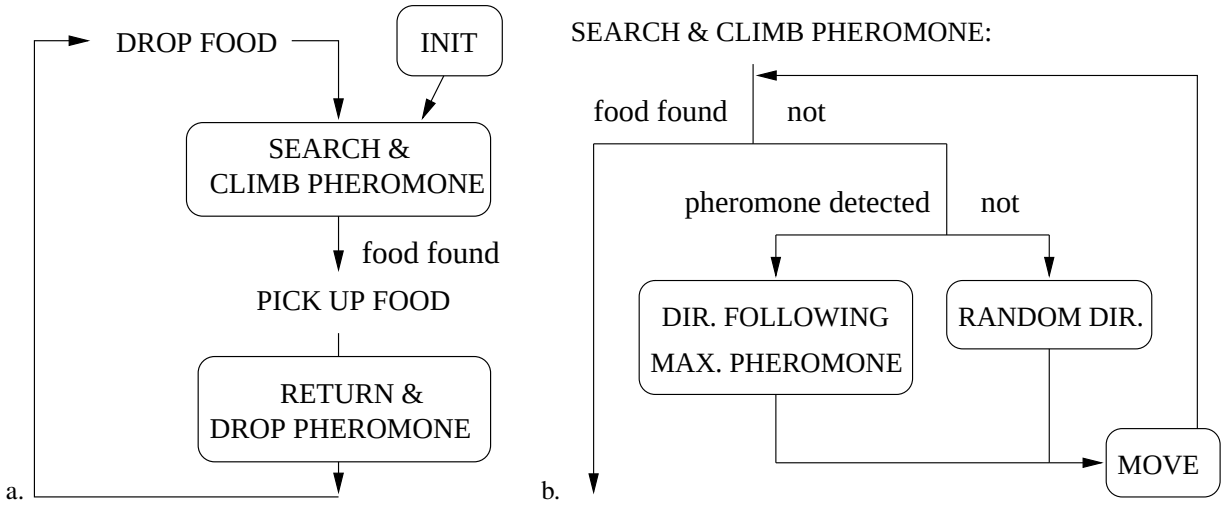


Figure 10: The ant model: (a) global ant behavior (b) details into task Search & Climb pheromone

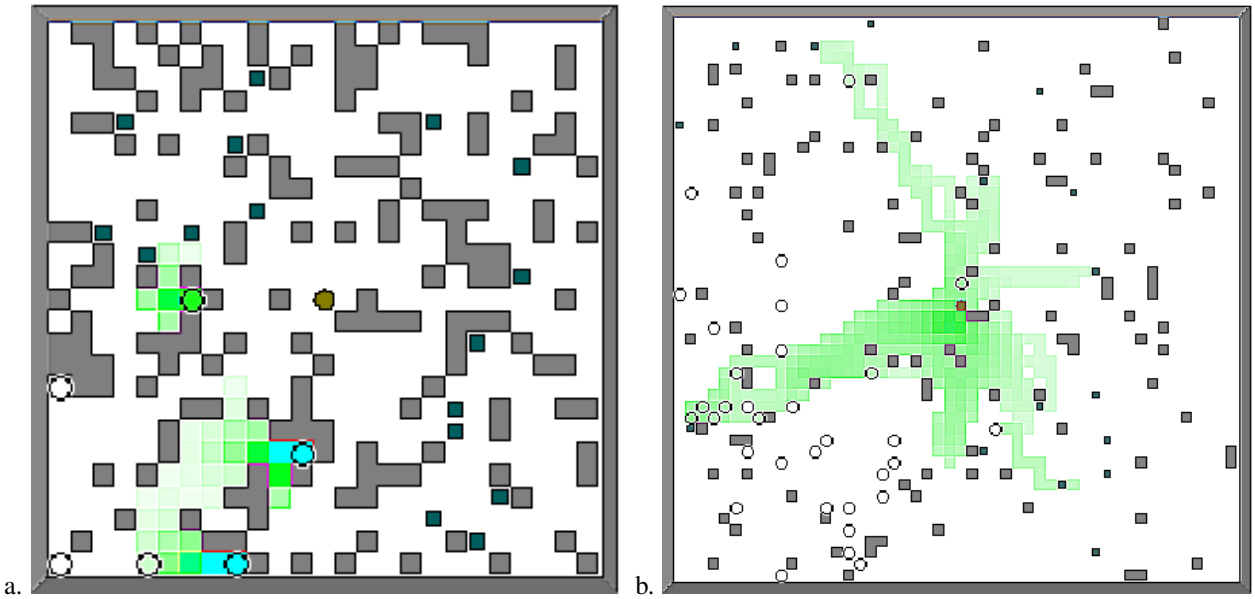


Figure 11: Foraging with ants that drop pheromones and use "compass"-like information to return to their nest situated in the center. a) obstacle density of 30% generates cavities where agents freeze. b) obstacle density of 5% does not interfere with foraging (50 x 50 env., 50 ants, 20 sources)

any shape. As discussed in the next section, the ant model has been extended to address the problem of awkwardly shaped obstacles. The gist of these attempts is that a secondary pheromone is spread from the base in order to build a new gradient that culminates at the base (e.g. [21]). However this pheromone evaporates while this gradient should be maintained in the entire explored environment. The gradient is usually artifically sustained by placing a huge amount of pheromone at the base.

We have compared our algorithm with the ant model in environments containing less than 5% obstacles. In all different configurations, we observed that c-marking agents take less time to finish foraging. Figure 9 shows such a comparison, in which the environment size is varied with Setup 3 (see snapshot Figure 11.b).

Figure 9 shows that as the environment grows the difference in the time required to forage by marking agents and by ants decreases (from 73% to 52% to 37%). It is due to the time needed to explore the environment and to discover the 20 resources.

As shown previously marking agents and ants engage in a pseudo-random walk, implying that they spend a majority of their time in exploration. However, the superior performance of the marking agents as compared to ants on the same foraging task clearly indicates that computing a wave-front while exploring is more efficient: it saves time by providing immediately paths from discovered resources to the base and reduces the amount of time spent in exploration by favouring the not yet explored cells (see section 3.1).

6 Related work

6.1 Pheromone-based techniques for foraging and its variants

As previously outlined, ants inspired pheromone-based models for foraging (see section 5.5). However, it is interesting to consider the impact of the required environmental dynamics to perform the foraging task. The emergence of tidy paths in the environement can be slow as it depends on ierative deposits on the same areas. The persistance of a pheromone-path requires the replenishing of evaporated pheromone at a certain rate, and this in turn requires the participation of a large number of agents (on the other hand, even a single marking agent alone can build a gradient and therefore a path). By the same token, pheromone evaporation is a slow process and therefore it takes a long time for sub-optimal paths to be erased.

For the dynamic version of the foraging problem, i.e. with mobile resources (see e.g. [21]), the pheromone technique has an advantage in that paths are automatically updated. A marking agent, on the other hand, immediately erases paths from which a resource has moved, and if there is no other agent close to it, its new location will be lost. However, we

plan to study a simple extension of c-marking agents to deal with moving resources. It consists to keep the agent at the resource location until another one arrives, in order to extend the trail when the resource moves.

Pheromone-based *learning* algorithms have also been proposed in recent years. They try to take advantage of reinforcement learning or evolutionary algorithms. Agents use pheromone-amount to update the so-called Q-values (state-action utility estimates) in order to improve exploration or the gradient ascent ([20]).
Genetic algorithms are then used to fine tune different parameters and to optimize agent policies (exploration versus exploitation ratio) (e.g. [25]). It is worth pointing out that, besides the time required to compute the propagation and evaporation dynamics, these learning algorithms require additional time to actually converge to the optimal policies.

Pheromone-dependent methods need some ad-hoc mechanisms to be applied when the environment holds obstacles and that it is required for agents to find a path to the base. This could be compass information or some external information (e.g. [29]). This problem can be overcome, as recently proposed by [21], by combining pheromones with reinforcement learning to build numerical gradients. This approach is meant for collaborative forgaging and optionnaly for mobile food resources. In this work 'ants' have abilities quite similar to those of marking agents. They can move, perceive neighbouring cells, and read/write ***real*** values in them. Each agent updates the amount of pheromone in a cell on visiting it using a reinforcement learning update rule (such as TD(λ) learning). Repeating this update operation leads to the diffusion of values (rewards) set at home and at discovered resources. Even if this principle requires numerous agents, it is close to the wavefront expansion exploited in our model. However, the model of [21] does not guarantee to avoid building local minima. For that matter, their paper reports experiments in environments containing only convex obstacles and only one resource.

It is interesting to note while marking agents require only one gradient, reinforcement learning agents are required to construct two gradients to create a path between a resource and the base. Reinforcement learning needs to manage several pheromone-related dynamics in the environment. Therefore the time required for convergence using RL is several-fold than our algorithm which constructs only one gradient.

6.2 Towards a real-world application

We now discuss ways to implement markings and pheromones models in real environments.
[9] proposed that the markings/pheromones be actual physical objects; agents drop landmarks from discovered resources to the base. Even though this approach does not require an evaporation and propagation mechanism, it does demand complex abilities of the robots, to transport and to manipulate the landmarks. Futhermore, the approach does not incorporate the computation of a gradient. This prevents the tackling of complex obstacles such as cavities or mazes. For comparison, our digital markings concept can be considered as an intermediate between physical markings and synthetic pheromones.
The cooperative transport model presented in [29] is an interesting example of a real-world implementation, Each robot is asssumed to know its own location and it can communicate with a centre. The communication facility allows it to leave landmarks in an area that is shared by all the robots. The landmarks are *directional pheromones* (i.e., pheromones that indicate direction). Complex obstacles are not considered because the navigation is based on local (random) information and a compass (external information).
These 'ant-robots' record their movements as a sequence of landmark droppings in the target localization space. Each robot makes use of the landmarks left by other robots. In this sense, landmarks that lead to the discovery of a valid path from a resource to the base, are said to be "shared" by the robots. In this way, a valid path, not necessarily the shortest path, is discovered. This principle, of using a common workspace to share simple information, may be the way to implement our algorithm without the requirement of writing values in the environment, for a real-world application.
As emphazised in the introduction, the implementation of swarm and collective systems has become more realistic with pervasive and ubiquitous computing. Clearly, we can imagine to deploy or to use a wireless network while the robots explore the environment following our algorithm (see for instance [16] for an application of synthetic pheromones spreading). Eventually, simple mobile robots such as required in our model may be produce in quantity as their making becomes cheap (see for instance robots used in [14]).

7 Conclusions and future work

We have defined a distributed and asynchronous version of an algorithm due to [4] that builds an optimal APF for path-planning. Our model relies on a group of reactive agents that build a gradient purely by exploring the environment and that *do not require a map* to do so. The agents are only required to be able to read and write integer values in the discretized environment. We have proved that *this multi-agent algorithm builds a potential field that converges to an optimal one* (in other words, the shortest path from any point in the environment to the base of the agents is determined) (Theorem 3.1). Furthermore, this gradient, even during its construction, allows agents to deal with the multi-source foraging problem.

The computational experience of our algorithm shows that increasing the number of agents decreases dramatically the time required for foraging. In particular, cooperation can provide superlinear performances for small number of agents. We have discussed the differences between our algorithm and the standard ant-model algorithm. Computational experiments show that marking agents are more time efficient than pheromone dependant agents. Moreover, our model, which relies on the construction of only one gradient, whereas several are required by the other approaches, defines a solution for foraging in complex environments (i.e. with obstacles such as cavities or mazes).

The efficiency of our algorithm is on account of two factors:

- When marking agents discover a resource, a valid path from the resource location to the base is *immediately* available to them (since the APF construction starts as soon as agents leave the base).
- In contrast to pheromone based techniques, paths construction and removing do not rely on costly iterative processes such as accumulation and evaporation of pheromones.

Generally, we could observe that to achieve a given computational performance, the marking agents approach requires fewer agents than the ant-model. In complete constrast to the ants-pheronome approach is that, even a very small number of c-marking agents (it can even be a single one) can identify paths for forgaging by building the complete gradient. Such efficiency in the number of agents is afforded because the computation associated with pheromones accumulation is completely avoided.

We intend to elaborate this work in different ways. We plan to study paramaters that induce super-linear performances, to evaluate robustness to noise in navigation, and to adapt the model to deal with moving resources and energy limitation. We also think that the exploration process can be improved by adapting some existing strategies used for covering tasks (e.g. [30]). As we started to study, we intend to show that the proposed multi-agent algorithm can be generalized to the construction of other potential fields and harmonic functions. Eventually we plan to implement the proposed model with small mobile robots.

The author wishes to thank Bruno Scherrer for fertile discussions and reviewing of this work.

References

[1] R.C. Arkin. *Behavior Based Robotics*. The MIT Press, 1998.

[2] M. Asada, P. Stone, H. Kitano, B. Werger, Y. Kuniyoshi, A Drogoul, D. Duhaut, M. Veloso, H. Asama, and S. Suzuki. The robocup physical agent challenge: Phase I. *Applied Artificial Intelligence*, 12, 2-3:251–264, 1998.

[3] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *Robots in Unstructured Environments, Fith International Conference on Advanced robotics*, 2:1012–1017, 1991.

[4] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. on Systems, Man and Cybernetics*, 22(2):224–241, 1992.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. New York, Oxford University Press, 1999.

[6] R.A. Brooks, P. Maes, M.J. Mataric, and G. More. Lunar base construction robots. *IEEE International Workshop on Intelligent Robots and Systems*, pages 389–392, 1990.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[8] J.L. Deneubourg, S. Aron, S. Goss, J. M. Pasteels, and G. Duerinck. Random behaviour, amplification processes and number of participants : How they contribute to the foraging properties of ants. *Physica*, 22(D):176–186, 1986.

[9] A. Drogoul and J. Ferber. From tom thumb to the dockers: Some experiments with foraging robots. In *2nd Int. Conf. On Simulation of Adaptative behavior*, pages 451–459, Honolulu, 1992.

[10] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the 3rd International Conference on Multi-Agent Systems*, pages 128–135, Paris, 1998.

[11] B. Holldobler and E.O. Wilson. *The Ants*. Harvard University, 1990.

[12] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In *International Conference on Mobile Computing and Networking (MOBICOM)*, pages 271–278, 1999.

[13] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 500–505, St. Louis, 1985.

[14] T. H. Labella, M. Dorigo, and J-L. Deneubourg. Division of labor in a group of robots inspired by ant's foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems*, 1:4–25, 2006.

[15] J.C. Latombe. *Robot Motion Planning*. Boston: Kluwer Academic, 1991.

[16] M. Mamei and F. Zambonelli. Spreading pheromones in everyday environments through rfid technology. In *2d IEEE Symposium on Swarm Intelligence*, 2005.

[17] M. Mamei and F. Zambonelli. *Field-Based Coordination for Pervasive Multiagent Systems*. Springer, 2006.

[18] F Michel, G. Beurier, and J. Ferber. The turtlekit simulation platform: application to complex systems. In *Proceedings of the multi-agent systems Workshop of the 1st IEEE International Conference on signal-image technology and internet-based systems*, page 122, 2005.

[19] F. Michel, O. Gutknecht, and J. Ferber. Generic simulation tools based on mas organization. In *Proc. of MAAMAW'2001*, 2001.

[20] N. Monekosso, P. Remagnino, and A. Szarowicz. An improved q-learning algorithm using synthetic pheromones. In *From theory to practice in multi-agent systems, second international workshop of central and eastern europe on multi-agent systems, CEEMAS 2001 Cracow. LNAI 2296*, pages Springer–Verlag, 2002.

[21] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *Proc AAMAS'04*, pages 36–43. ACM, 2004.

[22] H. Van Dyke Parunak, M. Purcell, and R. O'Connell. Digital pheromones for autonomous coordination of swarming UAV's. In *Proc. of AIAA First Technical Conference and Workshop on Unmanned Aerospace Vehicles, Systems, and Operations*, 2002.

[23] H.V.D. Parunak. Go to the ant: Engineering principles from natural agent systems. *Annals of Operations Research*, 1997.

[24] M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. Cambridge, MA: MIT Press, 1994.

[25] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. Brueckner. Evolving adaptive pheromone path planning mechanisms. In *Proc. of AAMAS'02*, pages 434–440, 2002.

[26] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. Brueckner. Performance of digital pheromones for swarming vehicle control. In *Proc. of AAMAS'05*, pages 903–910, 2005.

[27] F. Sempe and A. Drogoul. Adaptive patrol for a group of robots. In *Proc. of IEEE International Conference on Robotics & Systems*, pages 2865–2869, 2003.

[28] L. Steels. Cooperation between distributed agents through self-organization. In *Decentralized AI– Proc. of the First European Workshop on Modelling Autonomous Agents in a Multi-Agelit World (MAAMAW-89)*, pages 175–196. Elsevier Science B.V., Amsterdam, 1989.

[29] R. T. Vaughan, K. Stoy, G. S. Sukhatme, and M. J. Mataric. Whistling in the dark: cooperative trail following in uncertain localization space. In *Proc. of the 4th Int. Conf. on Autonomous Agents*, 2000.

[30] I.A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Mac vs. pc: Determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *the International Journal of robotics Research*, 19(1):12–31, 2000.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399