Engineering Applications of Artificial Intelligence ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A Cooperative Switching Algorithm for Multi-Agent Foraging

Ouarda Zedadra [a,b,*], Hamid Seridi [b], Nicolas Jouandeau [c], Giancarlo Fortino [d]

[a] Department of Computer Science, Badji Mokhtar-Annaba University, P.O. Box 12, 23000 Annaba, Algeria
[b] LabSTIC Laboratory, 8 May 1945 University, P.O. Box 401, 24000 Guelma, Algeria
[c] LIASD, Paris 8 University, France
[d] DIMES, Universita' della Calabria, Via P. Bucci, cubo 41c - 87036 - Rende (CS) - Italy

ARTICLE INFO

Article history:
Received 18 April 2015
Received in revised form
4 January 2016
Accepted 11 January 2016

Keywords:
Multi-agent system
Coordination
Stigmergy
Stigmergic Multi-Ant Search Area (S-MASA) algorithm
Multi-agent Search
Multi-Agent Foraging

ABSTRACT

The foraging task is one of the canonical testbeds for cooperative robotics, in which a collection of robots have to find and transport one or more objects to specific storagepoints. Efficiency in foraging can be improved with coordinated team of robots. Swarm robotics investigates the bio-inspired behaviors of simple species, that provide complex behaviors at the group level. We present a multiagent foraging algorithm named Cooperative Switching Algorithm for Foraging (C-SAF) inspired from the classical ant system. It provides a quick search, optimal homing paths and quick exploitation of food. A qualitative comparison between some foraging related works and the proposed algorithm is given here, as well as a quantitative comparison which shows that our algorithm outperforms the reference c-marking algorithm across a range of different scenarios.

1. Introduction

Multi-Agent Systems are a suitable approach to develop many multi-robot distributed applications such as: mine detecting (Acar et al., 2003; Gage, 1995), search in damaged buildings (Kantor et al., 2006; Jennings et al., 1997), fire fighting (Marjovi et al., 2009), and exploration of spaces (Landis, 2003; Schilling and Jungius, 1996), where neither a map, nor a Global Positioning System (GPS) are available (Batalin and Sukhatme, 2002). The efficiency of the group of robots in this case can be dramatically improved through coordination. Coordination issues (Yan et al., 2013) of multi-robot systems are considering more robots and more complex tasks, sometimes including robots and humans together. Swarm robotics investigates multi-robot coordination in a distributed way. It is interested in the implementation of systems which are composed of thousands of simple robots rather than one single complex robot (Vaughan, 2008). The challenge is to develop a group of robots with limited perception and computing capabilities to resolve complex tasks in a collective and distributed manner. This would allow for reduction of costs with respect to heavyweight approaches based on powerful and expensive cognitive robots/agents. In particular, stigmergic-based coordination mechanisms have been used in many robotics problems (aggregation formation and flocking WeiXing et al., 2006, patrolling Pasqualetti et al., 2010, localization and mapping Stipes et al., 2006, exploration and fire searching Marjovi et al., 2009), where agents adopt an indirect communication by depositing pheromone in their environment. This mechanism takes its inspiration from social insects (ants, bees, termites) which provide collectively intelligent systems (Momen, 2013) that, in spite of the simplicity of their individuals, present a highly structured social organization. As a result of this organization, ant colonies can accomplish complex tasks that in some cases far exceed the individual capacities of a single ant (Dorigo et al., 2000).

Foraging is a complex task that involves the coordination of multiple subtasks each constituting a difficult task (searching, harvesting, homing and unloading). It lends itself to multi-robot systems, even if the task can be achieved by one single robot, it is profitable to use multiple robots with careful design of cooperation and coordination strategies (Winfield, 2009). The sophisticated foraging behavior observed in social insects, provides inspirations to produce simple individuals (like ants) that use simple coordination rules and provide more complex (emergent) behaviors as a whole.

We investigate in this paper the Multi-Agent Foraging problem. Many approaches to such problem have been proposed in the literature so far (Momen, 2013; Hoff et al., 2010; Meng et al., 2012; Hoff

* Corresponding author.
E-mail addresses: zedadra_nawel1@yahoo.fr (O. Zedadra), seridihamid@yahoo.fr (H. Seridi), n@ai.univ-paris8.fr (N. Jouandeau), g.fortino@unical.it (G. Fortino).

http://dx.doi.org/10.1016/j.engappai.2016.01.025
0952-1976/

et al., 2013; Lee and Ahn, 2011; Pitonakova et al., 2014). Here, we propose therefore the Cooperative Switching Algorithm for Foraging (C-SAF), an extended version of the Multi-Agent Foraging algorithms in Zedadra et al. (2015a) and Zedadra et al. (2015b). In this paper, we apply it to large-scale foraging systems with hundreds and thousands of agents, where we consider a wider range of system sizes than the ones in literature works (Hoff et al., 2013; Simonin et al., 2014; Shell et al., 2006) and in our previous work (Zedadra et al., 2015a). We analyze and qualitatively compare our proposed approach to relevant related works through an extension of a reference comparison framework (Winfield, 2009). An in-depth performance evaluation based on simulation is given that allows us to quantitatively compare our approach with cooperative and non cooperative foraging approaches. Results show that C-SAF algorithm takes on average less time to finish the foraging, returns larger food amounts and provides optimal food-to-nest paths without the need to revisit a cell several times. Moreover, it shows to be scalable and presents considerable parallelism with growing agent number. Finally, it outperforms the other approaches in terms of efficiency and scalability.

The remainder of the paper is organized as follows: Section 2 presents the background concepts and definitions, and the related work on foraging tasks. Section 3 describes the C-SAF algorithm. A qualitative comparison between C-SAF and the Non-Cooperative Switching Algorithm for Foraging (NC-SAF), c-marking and Non-Cooperative c-marking (NC-c-marking) algorithms, is presented in Section 4. Section 5 details the different scenarios, obtained results and quantitative comparison between the four analyzed algorithms. In Section 6, we provide a discussion on how to go towards a real robotic implementation of our approach. Finally, we draw conclusions in Section 7 and delineate some future works.

2. Background and related work

2.1. Basic concepts

Here we define and clarify some key terms and concepts that will be used throughout this paper:

- *Agent-Based Modeling* (*ABM*) is a class of computational models for simulating the actions and interactions of autonomous agents (both individual or collective entities such as organizations or groups) with a view to assessing their effects on the system as a whole and analyzing possible emergent behaviors.
- *Swarm Intelligence* is the study of natural and artificial systems of multiple agents that adopt a distributed autonomous control. Instead, global intelligent behaviors emerge from a cooperating collection of simple individual behaviors (Bonabeau et al., 1999).
- *Stigmergy* is a particular form of indirect communication mediated by modifications of the environment used by social insects to coordinate their actions (Grassé, 1959).
- *Artificial Potential Field* (*APF*) is a wavefront of integer values written by agents in the environment, to mark the short distance between any cell and the nest (Simonin et al., 2014).
- *Search* is defined as the act of looking into or over carefully and thoroughly in an effort to find or discover something (Méndez and Bartumeus, 2014).
- *Foraging* is the act of searching for and collecting food at one or more storage points. It is a complex task that involves the coordination of multiple other tasks, such as searching, homing, and grabbing (Winfield, 2009).

2.2. Multi-Agent Foraging problem definition on 2D grids

In the following, we present a formal definition of the Multi-Agent Foraging problem on two dimensional grids with one nest. The environment E is represented as 2D grid of cells of size $N \times N$. $E = E_{Free} \cup E_{Obstacles}$, where E_{Free} denotes the subset of E containing all obstacle-free cells and $E_{Obstacles}$ denotes the subset of E containing obstacles. $E_{Free} = E_{Reachable} \cup E_{Unreachable}$, where $E_{Reachable}$ denotes the subset of E_{Free} containing reachable cells (i.e. all cells that are reachable by agents) and $E_{Unreachable}$ denotes the subset of E_{Free} containing unreachable cells (i.e. cells enclosed in obstacles). $E_{Obstacles}$ is a subset of E defined as a set of obstacle cells where $Obs_i \subseteq E_{Obstacles}$. Each cell $c = (x, y) \in E$ has a maximum of four neighbors $(x-1, y), (x+1, y), (x, y-1), (x, y+1)$. Let c_0 be the origin (or nest) and the starting cell for all the agents positioned at the center of the environment (coordinates (0,0)). Let $E_{Visited}$ be the set of cells already visited where $E_{Visited} \subseteq E_{Reachable}$ and $E_{NotVisited}$ the set of cells not yet visited containing at the starting time all the cells of $E_{Reachable}$ except c_0 where $E_{NotVisited} \subset E_{Reachable}$ and $E_{Reachable} = E_{NotVisited} \cup E_{Visited}$. A is a set of identical agents $a_0 \ldots a_n$, where n is the total number of agents. Each a_i has initial heading (0° = up, 90° = right, 180° = down or 270° = left) and can perceive the four neighbors in up, right, down, and left directions. N food locations (referred to as food density), each with M food items (referred to as concentration) are spread on a set of cells included in the list C_{goal} (list of cells containing a food). Agents do not know the coordinates of the cells in C_{goal}. The goal is therefore to forage all food in C_{goal} by minimizing the $T_{foraging}$ time, i.e. the overall time needed to complete the foraging task. To guarantee completeness of search and reachability of food, we assume that obstacles do not partition E and do not enclose any agent, nest or food cell. Moreover, the problem definition can be easily generalized from one to many nests.

The components of our Multi-Agent Foraging system (*Environment*, *Agent*, *Pheromone*) are modeled as follows:

- *Environment Model*: The environment is organized as a N × N grid with several food locations, one or multiple sinks (or nests), obstacles and pheromone markings. The grid is divided into equal squares in a cartesian coordinate system. Grid maps are thought to an efficient metric for navigation in large-scale (Thrun and Bücken, 1996). Obstacles with rectangular or square shapes take place on some fixed cells, the nest is at the center (in a multi-sinks space, sinks take specific positions). N food with M items take fixed positions on the grid. In the offline version of searching, an agent has already the map of the world and plans its path before starting the search. Whereas, in the online version the agent plans the path directly when searching. The online version of the grid-based exploration has received considerable attention due to its applicability in practice since grids represent a natural discretization of planar environments (Thrun and Bücken, 1996). It is often used for solving tasks like path planning, localization, search, coverage and surveillance (Balch, 1996; Yean and Chetty, 2012; Lau et al., 2013; Panov and Koceska, 2014; Gabriely and Rimon, 2002; Choi et al., 2009). Apart from 2D-grids, that are the most used environments in the Multi-Agent Foraging problem, other approaches do exist. For instance, topological approaches such as those described in Mataric (1994); Gutjahr (2000) and Dorigo et al. (2006), represent robot environments by graphs. Nodes in such graphs correspond to distinct situations, places, or landmarks. They are connected by arcs if there exists a direct path between them.
- *Agent Model*: Agents are modelled as simple, reactive ant-like agents that can move in the four directions up, down, right and left,

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

corresponding to 0°, 90°, 180° and 270° headings, respectively. Agent can detect pheromone at a specific distance (at one cell step). At each move, agents deposit an amount of pheromone to mark the current cell as already visited.

- *Pheromone Model*: As in traditional ant system, agents in our model release pheromone on the ground on their visited cells or even in their four neighboring cells. The pheromone evaporates according to a specific rate. The total amount of pheromone that evaporates at the position i in time t is modeled as in Eq. (1):

$$\Gamma_i(t+1) = \Gamma_i(t) - p*\Gamma_i(t) \quad (1)$$

where p is a coefficient which represents the evaporation rate of trail between time t and $(t+1)$.

2.3. S-MASA algorithm

The S-MASA algorithm (Zedadra et al., 2014), was proposed for multi-target search and foraging tasks. It is designed according to the main ideas of the artificial ant system (Dorigo et al., 2000; Kuyucu et al., 2012; Calvo et al., 2011) and the water vortex dynamics. Basically, the system is a group of homogenous artificial agents (e.g. robots) moving and taking decisions based on the stimuli received from the environment. While agents navigate, they deposit pheromone to mark visited cells and can sense the existence of pheromone in their vicinity. The absence of pheromone in its right cell calls the agent to change its heading and to move to its right cell, else the agent will keep going forward in its current heading. We adopt in this algorithm the opposite logic of ant system theory. So, instead of following the cells that contain pheromone, our agents are guided to not yet visited cells (where no pheromone is present). S-MASA provides: (1) a quick search by the large dispersion of agents when avoiding already visited cells, optimizing by this their chance to find food, (2) optimal paths to return to nest result from the wavefront of pheromone concentration expansion created by agents simultaneously when exploring.

The Detect_Pheromone_Adjust_Heading_And_Move() (detailed in Algorithm 1) is used to extract the stimuli and to decide about the next move to take. The agent changes its heading according to rules in Algorithm 1 if there is no pheromone in its right cell (Fig. 1) else it keeps its heading unchanged. The avoid_obstacle() procedure is used to avoid obstacles. Specifically, agents turn around obstacles in the direction of already visited cells until encountering a not yet visited cell, which means the obstacle is avoided and agents can continue their search.

Algorithm 1. Detect_Pheromone_Adjust_Heading_And_Move.

```
if (∄ pheromone in right cell) then
    if(current_heading = 270) then
        Set heading to 0;
    else
        Set heading to current_heading+90;
    move forward of 1 cell in the heading direction;
else if (∃ pheromone in right cell) and (the front cell is not an
    obstacle) then
    move forward of 1 cell in the heading direction
else if (∃ pheromone in right cell) and (the front cell is an
    obstacle) then
    avoid_obstacle();
```

Statement 2.1. *Considering a set of agents performing Algorithm 1 in a bounded 2D Grid obstacle-free environment (see Section 2.2), each cell will be visited.*

Proof. According to Algorithm 1, an agent a_i moves to $c_x \in E_{NotVisited}$ in its right direction if it exists, else it moves to $c_x \in E_{Reachable}$ in its front direction. Each move of a_i changes the state of c_x to visited, thus reducing $E_{NotVisited}$ and extending $E_{Visited}$. Since $E_{NotVisited}$ is limited, Algorithm 1 is complete because the number of moves is bounded by the number of cells in $E_{NotVisited}$ and, at each step, a cell from $E_{NotVisited}$ is moved to $E_{Visited}$; at a given time t_i, all cells are in $E_{Visited}$ and $E_{NotVisited}$ is empty.

Statement 2.2. *Considering a set of agents performing Algorithm 1 in a bounded 2D Grid obstacle environment (see Section 2.2), each reachable cell will be visited.*

Proof. The strategy used by an agent a_i to avoid an obstacle is to get around the obstacle in the direction of already visited cells until encountering $c_x \in E_{NotVisited}$. According to Algorithm 1, a_i performs the following steps: (1) looks for $c_x \in E_{NotVisited}$ in its right heading, if there exists, it moves to it; (2) else, it looks for $c_x \in E_{Reachable}$ in its front heading, if such a cell exists it moves to it; (3) else, if the cell in front $\in E_{Obstacles}$ then a_i avoids the obstacle as follows: (i) keeps moving to $c_x \in E_{Reachable}$ in its right direction until it locates a $c_x \in E_{Reachable}$ in front direction; (ii) it keeps then moving to $c_x \in E_{Reachable}$ in its front direction until it locates a $c_x \in E_{NotVisited}$ in right, front or left heading respectively. As Obs_i is finite, in a finite time the a_i avoids the obstacle and reaches a cell $c_x \in E_{NotVisited}$, it moves to such a cell and then restart from step (1). At each step, a_i changes the state of c_x to visited, thus reducing $E_{NotVisited}$ and extending $E_{Visited}$. Since $E_{NotVisited}$ is limited, the number of moves is limited too and because each move ends with the addition of a new cell to $E_{Visited}$, at a given time t_i, all cells are in $E_{Visited}$ and $E_{NotVisited}$ is empty, thus the search process ends.

2.4. Related work

The problem of coordination of multiple agents is considered complex (Speranzon, 2006; WeiXing et al., 2006). Pheromone-based coordination, inspired by the way social insects coordinate via leaving marks in their environment, provides fundamentals to design alternative strategies that overcome the difficulties (Barnes et al., 2006) imposed by mathematical formulations, agent and environment models (Jiang, 2006; Freeman et al., 2006). Pheromone-based coordination has been widely applied in exploration and surveillance tasks (Calvo et al., 2011; Kuyucu et al., 2012; Marjovi et al., 2009), distributed search and collective cleanup (Liu et al., 2010; Feinerman et al., 2012), multi-target search (Zedadra et al., 2014) and foraging (Fortino et al., 2014).

As we are interested in Multi-Agent Foraging, below we present relevant literature works on Multi-Agent Foraging. In Hoff et al. (2010), authors propose two decentralized foraging algorithms inspired by ants behavior. Robots are simple in sensing and processing power, they communicate with each other in neighborhood using simple InfraRed ring architecture. However, instead of using chemical pheromone, beacon robots are used to store virtual pheromone. The approach is very costly in number of robots (beacons) required to create the paths. The models proposed in Meng et al. (2012) are two ant colony foraging models adapting Panait's and Wilinskey's models (Panait and Luke, 2004; Wilensky, 1997), by introducing direct interaction (via direct communication) besides indirect interaction (via pheromone). The proposed models improve the ability of the colony to find the shortest paths, however, agents need memory to store additional information about food location. The foraging problem was investigated in Hoff et al. (2013) through three algorithms (gradient, sweeper and adaptive). In the three algorithms, robots need to communicate directly their current position and food position to their neighbors. In the *gradient* algorithm, agents broadcast information about nest and food gradients, they switch from *beacon* to *walker* robots according to some criteria. While in the *sweeper* algorithm, agents use virtual forces to form a line expanding from nest to food

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

that sweeps the search world like the hand of clock; when food is found some robots remain as beacons while others act like walkers. The search becomes slower with long robot lines. The third algorithm (*adaptive*) helps the colony to switch between the two aforementioned algorithms: for close food they use gradient algorithm, for far distance food they use sweeper algorithm and when the number of robots is not sufficient to cover the whole world, robots switch to use random walk. Introducing preferences towards some tasks besides stimulus has some impact on the performance of the colony (Momen, 2013). Authors use a group of heterogeneous agents and a set of local rules to communicate with neighboring agents for each group. Agents switch between three tasks: *foraging, brood caring* and *resting* based on three thresholds for foraging, brood caring and resting. The model brings sufficient improvements but it is still very parameter dependent and several initial parameters need to be adjusted for best results. Authors in Pitonakova et al. (2014) compare simulated robotics swarm of individualist foragers and a swarm of collective foragers. It was shown that recruitment is useful when resources are hard to find, where agents tend to signal to each other the location of deposits and return home on beacons, they use odometry to recognize their location and the food location. Robots need internal memory to store additional information that helps them to react. In Lee et al. (2013) and Lee and Ahn (2011), a honey-bee inspired model for foraging was proposed to improve the energy efficiency by means of search space and labor division. They employ temporary stores at fixed locations, robots need to communicate directly the food location and use GPS to return home. The approach was tested on small scale systems (25 to 40 robots). Two decentralized foraging algorithms that imitate swarm behaviors using a limited number of robots were proposed in Chattunyakit et al. (2013). In both algorithms, robots are equipped with camera, infrared sensors, GPS and communication device. Each robot has a positioning system installed to determine its current location. Random motion is adopted by the two algorithms at first stages of search and robots transmit at each step their current positions between them. In Simonin et al. (2014), authors propose a Multi-Agent Foraging algorithm. It is a parameter free, distributed and asynchronous version of the wavefront algorithm (Barraquand et al., 1992). Agents while exploring the environment build simultaneously paths between food and the nest which results in computing a wavefront from the agent destination which involves building an ascending Artificial Potential Field (APF) incrementally. Due to the pseudo random walk, paths are not optimal and agents need to visit the same cell several times before the APF reaches its optimal value, which results in large foraging times.

C-SAF presents multiple benefits in comparison to the reported foraging works. The random walk used in most of the works as search strategy provides large foraging time and non-optimal paths (Simonin et al., 2014). In our work, it is replaced with a new search algorithm called S-MASA. S-MASA allow agents to: (1) search quickly for food by allowing a large dispersion of agents by avoiding already visited cells as much as possible, (2) the vortex-like movements of agents allow building optimal paths by constructing a wavefront expansion of pheromone. Comparing to the related works, C-SAF is scalable, it could be used with large number of agents and large environment sizes. Agents in C-SAF are provided with very limited sensing (perceive the four neighboring cells) and computational capabilities. They communicate indirectly via pheromone and they need very small memory to store the amount of remaining food, and no specific material is needed (compasses for example). Agents return home by descending the negative gradient of pheromone wavefront created when searching (see Section 2.3). After food is located, agents attract each other to the location using pheromone trails.

Table 1 presents the comparison of the surveyed works with our algorithm C-SAF (Algo2) according to an extended taxonomy based on the one in Winfield (2009). Each one of the four major axis (*environment*, *robot*, *strategy* and *performance*) is characterized by a set of properties in the minor axis which can take specific values denoted in *value* axis in Table 1. Features of the *environment* concern: *search space* which can be bounded or unbounded, *source areas* that represent food locations which can be single or multiple, *object types* of limited or unlimited amount of items which take fixed or random positions (*object placement*) and a storage point where food is returned (*sinks*). The *robot* is defined by: a *number* of agents participating at the mission (single or multiple), their *type* either homogenous which means similar in functions and material or heterogeneous, the *object sensing* which is their range of sensors (limited or unlimited), their ability to know the current position (*localization*), their ability at exchanging information with others (*communication*) and their possession of limited or unlimited stores of energy (*power*).

The *strategy* includes the different sub-strategies used in foraging such as: *search*, *grabbing*, *transport*, and *homing* and if there exist mechanisms of *recruitment* or *coordination* to improve the performances. While in *performance*, we added in bold the different metrics used in the works compared in Table 1, in addition to those that exist in the reference taxonomy (Winfield, 2009).

3. Cooperative Switching Algorithm for Foraging (C-SAF)

The C-SAF Algorithm is a distributed Multi-Agent Foraging algorithm that creates optimal paths from food to nest by only depositing pheromone while searching. By using this algorithm,

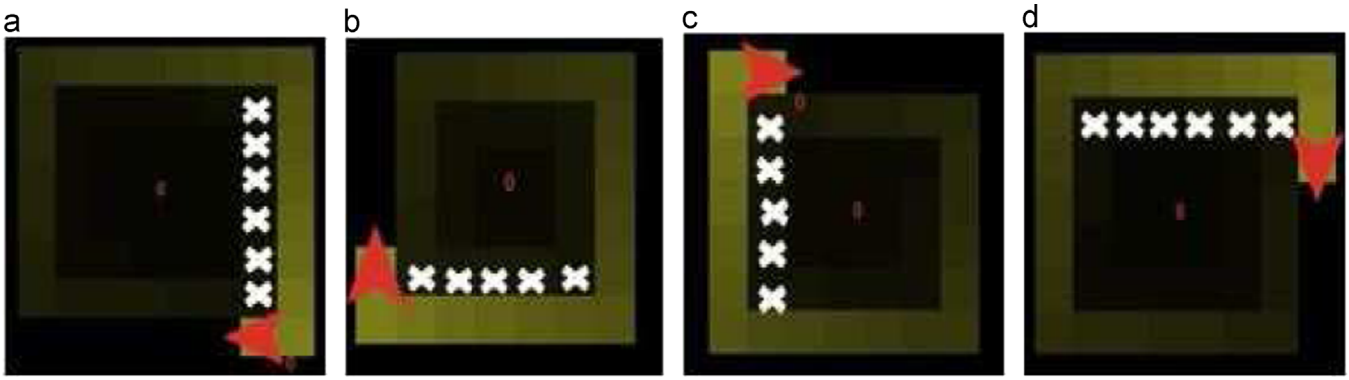

Fig. 1. S-MASA coordination rules which represent the changing of heading from: (a) 180° to 270°, (b) 270° to 0°, (c) 0° to 90°, (d) 90° to 180°, where white crosses represent already visited cells.

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

agents create simultaneously and synchronously a pheromone wavefront expansion from the nest to the food (see Statement 3.1) and can use the negative gradient to go back to nest (see Statements 3.2 and 3.3). It increases the cooperation between agents to transport food by diffusing pheromone when the food is located. Each agent can switch from a state to another according to surrounding events.

In particular the agent switches between several states: *Look-for-Food, Choose-Next-Patch, Pick-Food, Return-and-Color, Return-to-Nest, At-Home, Climb* and *Remove-Trail*. It starts with looking for food (*Look-for-Food* state). If there is no food, it moves in its environment using *Choose-Next-Patch* rules and deposits pheromones that evaporate with time. It changes automatically to *Look-for-Food* state and, if there is food, picks a quantity of it (*Pick-Food* state), and returns home by following the colored trail, if there exists one (*Return-to-Nest*). If there exists no trail, it diffuses the information to its neighbors by depositing recruitment pheromone and creates a trail by depositing return pheromone (*Return-and-Color*) while returning to home. When the home is reached, it unloads the food (*At-Home*) and if the food is exhausted it removes the existing trail (*Remove-Trail*), else it climbs the trail to the food location (*Climb*). Fig. 2 shows the finite state machine of our foraging agent whereas the complete algorithm is given in Algorithm 2.

Statement 3.1. *In a bounded 2D Grid environment (see Section 2.2), a set of agents performing the C-SAF Algorithm builds food paths when searching.*

Proof. Let us define a *food path* as the set of cells free from obstacles relaying nest to food. Let W_i be the set of cells corresponding to the same cost (pheromone value). Several wavefronts exist $W_0 \ldots W_n$, where W_0 is the initial wavefront, which corresponds to the starting cell from where all agents start searching. An agent a_i keeps moving according to the strategy defined in Statement 2.2. At the first step, agents move to reachable non-obstacle cells from W_0 and depose a pheromone creating the wavefront W_1. At the second step, agents move to reachable non-obstacle cells from W_1 and depose a

Table 1
A comparison of Multi-Agent Foraging algorithms based on the enhanced taxonomy. Algo2, Algo3 and Algo5 are respectively C-SAF, NC-SAF, and NC-c-marking (see Sections 3 and 4).

Major axis	Minor axis	Value	Simonin et al. (2014)	Momen (2013)	Meng et al. (2012)	Hoff et al. (2013)	Algo 2	Algo 3	Algo 5
Environment	search space	constrained	X	X	X	X	X	X	X
	source areas	single limited			X				
		single unlimited		X		X			
		multiple limited	X				X	X	X
	sinks	single	X	X	X	X	X	X	X
	object types	multiple static	X	X	X	X	X	X	X
	object placement	fixed location		X	X				
		random	X			X	X	X	X
Robot (s)	number	multiple	X	X	X	X	X	X	X
	type	homogenous	X			X	X	X	X
		heterogeneous		X	X				
	object sensing	limited	X	X	X	X	X	X	X
	localization	non	X		X	X	X	X	X
		relative		X					
	communication	direct		X	X	X			
		indirect	X		X		X	X	X
	power	unlimited	X	X	X	X	X	X	X
Strategy	search	random walk	X	X	X	X			X
		trail following	X	X	X		X	X	X
		follow other robots				X			
		in teams				X			
		S-MASA					X	X	
	grabbing	single	X	X	X	X	X	X	X
	transport	cooperative	X	X	X	X	X		
		single						X	X
	homing	on beacon				X			
		direct trail information			X	X			
		follow trail	X	X	X		X	X	X
	recruitment	direct				X			
		indirect	X	X	X		X	X	X
	coordination	direct		X	X	X			
		indirect	X				X	X	X
Performance	time	**fixed**					X	X	X
		minimum			X				
		varying agent number	X				X	X	X
		varying environment size	X				X	X	X
		first time to find food				X			
	other metrics	**average hunger level**		X					
		food found or not				X			
		rate of returned food					X	X	X

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

pheromone to create W_2. Such process iterates from i to $i+1$ until all reachable non-obstacle cells have been reached. At each step agents relay cells from W_i to cells in W_{i+1}, thus from the starting cell (c_0) there exists a trail that leads to the goal cell (c_{goal}). As agents adopt a vortex-like movements, they need to turn around each other in the direction of non-visited cells in obstacle-free environments (or of the visited cells in obstacle environments). Thus the built food paths are non optimal.

Statement 3.2. *In a bounded 2D Grid environment (see Section 2.2), a set of agents performing the C-SAF Algorithm builds homing paths by following the negative gradient created when searching (see Statement 3.1).*

Proof. Let us define a *homing path* as the set of cells free from obstacles relaying food to nest. Let $\mathbb{P}$ be initial pheromone deposed by an agent at time t_0, and $\mathbb{P}_j$ its value after evaporation where j is the number of evaporations that $\mathbb{P}$ had. At t_1, agents move to all reachable cells in one step from W_0, depose $\mathbb{P}$, thus create the wavefront W_1. At t_2, agents move to non visited neighbors of W_1, depose $\mathbb{P}$ in current cells, thus create the wavefront W_2 at the same time values of W_1 take the first evaporation $\mathbb{P}_1$ ($\mathbb{P}_1 < \mathbb{P}$). At t_3, agents move to non visited neighbors of W_3, they take the value $\mathbb{P}$, while values of W_1 take their second evaporation to get $\mathbb{P}_2$, and pheromones of W_2 take their first evaporation to get $\mathbb{P}_1$ where $\mathbb{P}_2 < \mathbb{P}_1 < \mathbb{P}$ corresponding to W_1, W_2 and W_3 respectively. At each time t_i, if there exists an obstacle then a_i keeps avoiding it in the direction of $c_x \in E_{Reachable}$ until it encounters a $c_x \in E_{NotVisited}$. This process iterates from i to $i+1$ until all reachable cells have been reached, agents can then follow the negative gradient from the greatest $\mathbb{P}$ value to the lowest one (W_0) to get home. The homing paths are shorter than food paths since agents choose at each step cells with the smallest $\mathbb{P}$ value in the four neighboring cells.

Statement 3.3. *The homing paths built according to Statement 3.2 are optimal.*

Proof. Optimality of the paths is considered with respect to the path length or the distance in number of cells between nest and food. We define an optimal path (shortest path) as the path with the smallest number of cells relaying nest to food. The vortex-like movements allow each agent a_i at any time t_j to get four different neighbors: c_{right}, $c_{back} \in E_{Visited}$ and c_{front}, $c_{left} \in E_{NotVisited}$ reachable at one step to the right, backward, forward and left directions respectively. Let $\mathbb{P}_r$ be the pheromone in c_{right}, $\mathbb{P}_b$ the pheromone in c_{back} and $\mathbb{P}_x$ the pheromone in the current cell c_x (c_{front} and c_{left} do not contain pheromones since they are not yet visited), since c_{back} is visited at t_{j-1} and c_{right} is visited at t_{j-n} where n is the number of steps between c_{right} and the current cell (following the principle of turns like vortex in movements). At t_j when c_x is reached, $\mathbb{P}_r < \mathbb{P}_b < \mathbb{P}_x$ because $\mathbb{P}_r$ evaporates $n \times p$, $\mathbb{P}_b$ evaporates $1 \times p$ and $\mathbb{P}_x$ does not evaporate yet (where n is the number of steps between c_{right} and c_x and p is the evaporation rate). According to this principle, when a_i returns home, it chooses $c_x \in E_{visited}$ with the smallest $\mathbb{P}$ which is the c_{right} cell, this process iterates until home is reached, thus from a cell to another the optimal cost is chosen, therefore optimal paths are created.

Algorithm 2. C-SAF Algorithm where pheromone is noted $\mathbb{P}$.

```
if AT-HOME then
    Unload food;
    if(∃ trail and food > 0)then goto CLIMB;
    elseif(∃ trail and food = 0)then goto REMOVE−TRAIL;
    else goto LOOK−FOR−FOOD;
if LOOK-FOR-FOOD then
    if(food > 0)then goto PICK−FOOD;
    else goto CHOOSE−NEXT−PATCH;
if CHOOSE-NEXT-PATCH then
    if (obstacle detected)then Avoid_Obstacle();
    else
        if(brown ℙ here) and (brown ℙ in right cell) then
            Diffuse(ℙ); move to food location using brown cells;
        else
            if(brown ℙ here) and (nobrown ℙ in right cell) then
                Remove brown trail;
            else
                Lay(ℙ);
                Detect_And_Adjust_Heading(ℙ);
                Update(ℙ);
                Move();
if PICK-FOOD then
    Pick up a given amount of food;
    Diffuse(ℙ);
    if(∃ trail)then goto RETURN−TO−NEST;
    else goto RETURN−AND−COLOR;
if REMOVE-TRAIL then
    while ∃ no trail do
        Move to neighboring colored cell with the greater value of ℙ;
        Update its color to the default one (black);
    goto LOOK − FOR − FOOD;
if CLIMB then
    while ∃ no trail do
        Move to neighboring colored cell with the greater value of ℙ;
    goto LOOK−FOR−FOOD;
ifRETURN-TO-NEST then
    while home not reached do
        Move to a colored neighboring cell with the lowest ℙ value;
    goto AT−HOME;
if RETURN-AND-COLOR then
    while home not reached then
        Move to a neighboring cell with the lowest ℙ value;
        Color that cell to a specific trail color (yellow);
    goto AT−HOME;
```

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

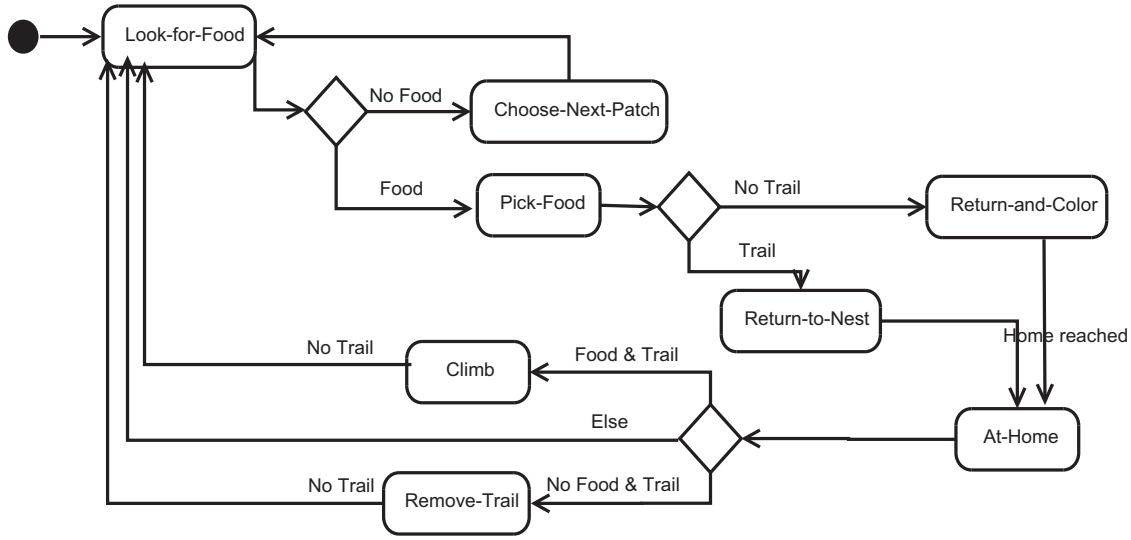

Fig. 2. State diagram showing the possible states of a foraging agent in C-SAF and NC-SAF algorithms. Black circle is the initial state, white diamonds are decision points and rectangles are states.

4. Related algorithms: a comparison

The C-SAF algorithm (Section 3) allow recruiting as much as possible agents in trails to cooperate in exploitation of food. To show that cooperation may improve the performance of the individual agents or the overall behavior of the system, an in-depth quantitative comparison between the C-SAF (Algorithm 2), NC-SAF (Algorithm 3), c-marking (Simonin et al., 2014) (Algorithm 4) and NC-c-marking (Algorithm 5) is given in Section 5.

The NC-SAF algorithm uses the state machine depicted by Fig. 2 and the C-SAF set of behaviors. However, it does not allow cooperation between agents to transport the food when it is found. Instead, each found food is exploited and transported by its finder. The agent finder will not alert the other agents to the food location, hence they proceed to the search process rather than to the exploitation of the food collectively. It uses the same state machine as C-SAF, with changes in the *Choose-Next-Patch* state, where agents in this state do not diffuse brown pheromone to attract other agents. This procedure is used by the C-SAF algorithm to spread information to alert the other agents to the food location in order to exploit it collectively.

Algorithm 3. Variants of the NC-SAF Algorithm with respect to the C-SAF Algorithm (see Algorithm 2), where pheromone is denoted by $\mathbb{P}$.

```
if CHOOSE-NEXT-PATCH then
    if (obstacle detected) then
        Avoid_Obstacle();
    else
        Lay(ℙ);
        Detect_And_Adjust_Heading(ℙ);
        Update(ℙ);
        Move();
if PICK-FOOD then
    Pick up a given amount of food;
    if (∃ trail) then goto RETURN−TO−NEST;
    else goto RETURN−AND−COLOR;
if (LOOK-FOR-FOOD or REMOVE-TRAIL or CLIMB or RETURN-TO-NEST or
RETURN-AND-COLOR or AT-HOME) then
    do the same as in algorithm 2;
```

The c-marking algorithm (Algorithm 4, Simonin et al., 2014) is a parameter free foraging algorithm that is a distributed and asynchronous version of the wavefront algorithm (Barraquand et al., 1992). Agents while exploring the environment build simultaneously paths between food and the nest which results in computing a wavefront from the agent destination which involves building an ascending APF incrementally. Agents need to visit the same cell several times before the APF reaches its optimal value. A c-marking agent is always in one of the states depicted by Fig. 3 showing its finite state machine. It always starts from the state *SEARCH & CLIMB TRAIL* which is the default state for all agents. Agents use random walk for search. When food is located, an agent picks a given quantity and returns to nest by using a trail if it exists or creating one if it does not exist, by following the negative gradient. Other agents can be recruited in existing trails. When the food is exhausted the last agent must remove the trail.

NC-c-marking (Algorithm 5) is an alternative of c-marking algorithm that does not allow cooperation to exploit the food, and so there is no recruitment of agents in existing trails. Each agent which found a food can create a trail and after that climb it to return to the food and remove it at the end. The two algorithms use the same state diagram depicted by Fig. 3, where the difference is in the states *SEARCH & CLIMB* and *RETURN AND COLOR TRAIL*.

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

Algorithm 4. c-marking Algorithm (Simonin et al., 2014).

if *SEARCH* & *CLIMB* (*Repeat*) **then**
  if (food exist in neighboring cell) then move into that cell; goto LOADING;
  if (there exists no colored neighboring cell) then
  |goto EXPLORATION & APF CONSTRUCTION;
  else if (previous position was not *a* trail cell) then
  |Move to the highest valued colored neighboring cell;
  else
  |Move to a new cell with the current trail color;
  goto UPDATE−VALUE;
if EXPLORATION & APF CONSTRUCTION **then**
  if (∃ neighboring cells without *a* value) then move randomly to one of them;
  goto UPDATE−VALUE;
  else move randomly to one of neighbors; goto UPDATE−VALUE;
if UPDATE-VALUE **then**
  Compute $val = 1 + min(4 - neighborvalues)$;
  Write *val* in the current cell if its different from the starting cell;
if LOADING **then**
  Pick up a quantity Q_{max} of food;
  if (food is not exhausted) then if (the cell is colored) then
  |goto RETURN−TO−BASE;
  else
  |goto RETURN − AND − COLOR − TRAIL;
  else goto RETURN−AND−REMOVE−TRAIL;
if *RETURN-AND-COLOR-TRAIL* **then**
  Color the cell in a specific color;
  goto UPDATE−VALUE;
  if (base reached) then Unload food;
  goto SEARCH & CLIMB;
  else move to a neighboring cell with the smallest value;
if RETURN-AND-REMOVE-TRAIL **then**
  if (cell has the trail color)then remove this color and goto UPDATE − VALUE;
  if (base is reached) then Unload food;
  goto SEARCH & CLIMB elseif (there exists *a* neighboring cell with trail color) then
  move to neighboring trail cell
  else move to the smallest neighboring cell;

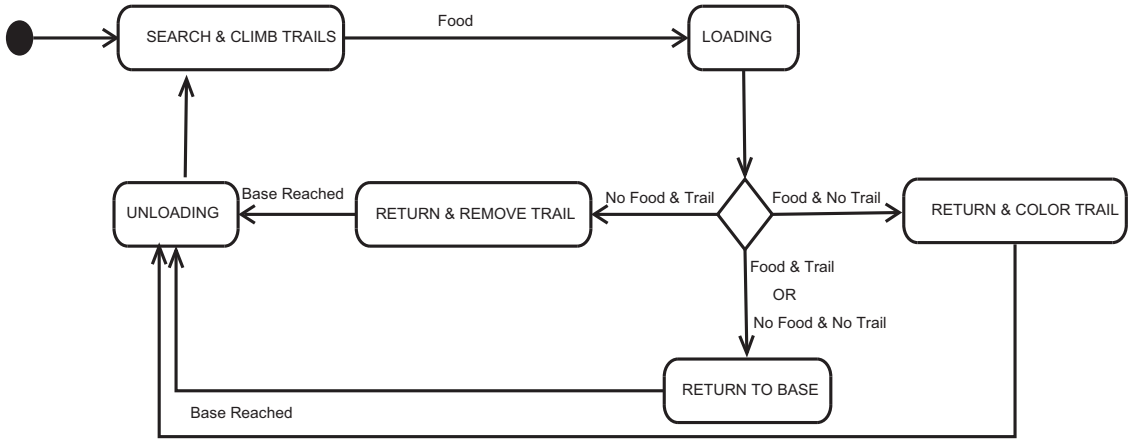

Fig. 3. State diagram showing the possible states of a foraging agent in c-marking (Simonin et al., 2014) and in NC-c-marking algorithms. Black circle is the entry point, white circle is a decision point and rectangles are states.

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

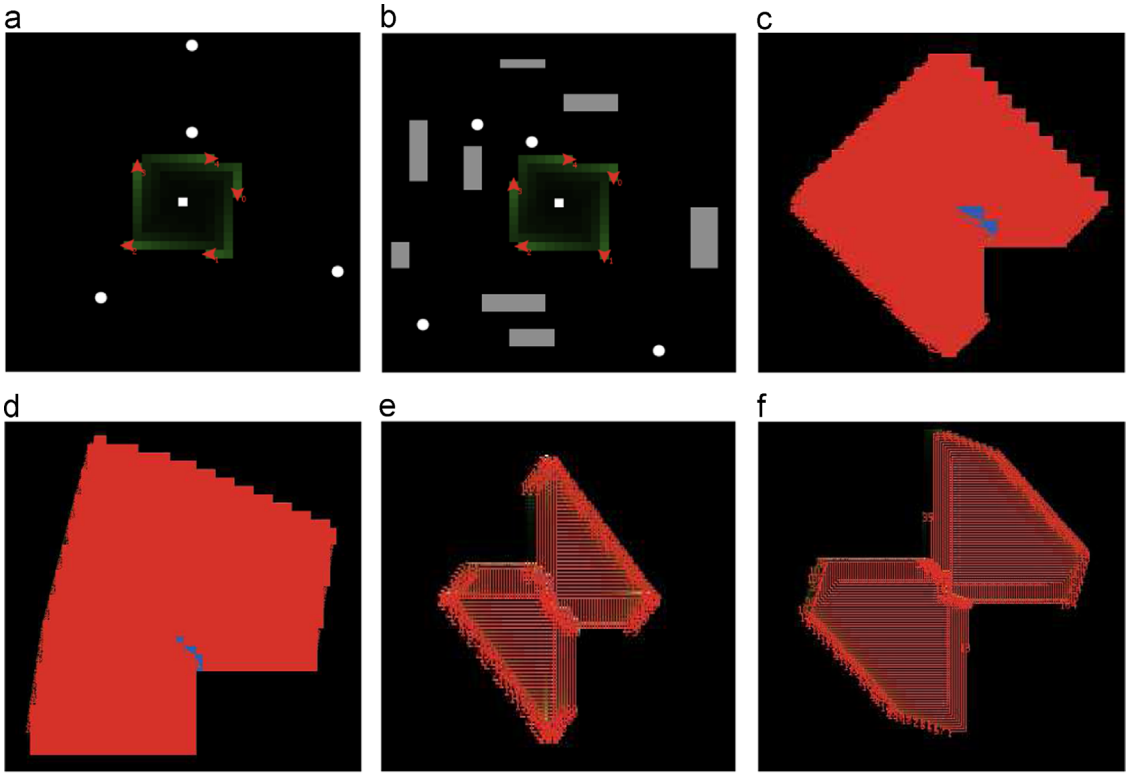

Fig. 4. World setups used in simulations: (a) Obstacle-free environment; (b) Obstacle environment, where arrows are agents, the white cell in the center is the nest, the white circles are food and the gray clusters are obstacles; (c), (d), (e) and (f) present C-SAF execution samples at different stages where Laden agents are in the middle line, and Walker agents spread themselves diagonally on sides, with respectively 1 and 2 food locations.

Algorithm 5. Variants of the NC-c-marking Algorithm with respect to the c-marking Algorithm (see Algorithm 4).

if SEARCH & CLIMB (Repeat) **then**
  if (*food exist in neighboring cell*) **then**
    move into that cell; update marker to true;
    goto *LOADING*
  if *marker* $\bar{t}$*rue* **then**
    if *there exists a colored cell with great value* **then**
      move to it
    else
      goto *EXPLORATION* & *APF CONSTRUCTION*
  else goto *EXPLORATION* & *APF CONSTRUCTION*;
if *RETURN-AND-COLOR-TRAIL* **then**
  color the cell in a specific color; goto UPDATE−VALUE;
  memorize the last step before the base cell;
  if (base reached) then Unload food;
  goto SEARCH & CLIMB;
  else move to a neighboring cell with the smallest value;
if (*EXPLORATION* & *APF CONSTRUCTION* **or** *UPDATE-VALUE* **or** *LOADING* **or** *RETURN-AND-REMOVE-TRAIL*) **then**
  do the same as in algorithm 4;

5. Performance evaluation

In this section, we present a description of the proposed foraging Framework in Section 5.1 and evaluate the performance of the four algorithms (C-SAF, NC-SAF, c-marking and NC-c-marking) in obstacle-free and obstacle environments. We present, in Section 5.2, the simulation parameters and the performance indices used to evaluate the algorithms. In Section 5.3, we describe four fundamental scenarios that are used for simulations, and report the importance and the benefits of such scenarios in understanding and comparing the algorithms. In the final Section 5.4, we present, discuss and compare the obtained results by the different algorithms in the four scenarios.

5.1. A simulation framework for Multi-Agent Foraging

The algorithms C-SAF, NC-SAF, c-marking and NC-c-marking have been implemented through our framework for Multi-Agent Foraging. The framework is composed of several components: the *search world* which is a set of *cells* modeled as stationary agents on which the actors of simulation are deployed, the *food* modeled as stationary agent with a fixed position and limited amount which can be exploited by foragers, *obstacles* and *nest* which take place at fixed positions, *foragers* are modeled as mobile agents that search for food using a searching algorithm (e.g. the S-MASA algorithm Zedadra et al., 2014, see Section 2.3) and transport it, when found, to home while depositing pheromones and *behavior* which includes a specific foraging algorithm (e.g. Algorithms 2–5). The framework is designed to be extensible as the definition of each object is independent from the others, and it can be reused in other foraging algorithms. It is also flexible as new algorithms or even behaviors can be defined and seamlessly used; hence, developers of other foraging algorithms can add new classes, remove classes which are not important in their systems, or modify existing methods (the *move* method of an agent for example), or even add other states (such as *recharging* to take into account energy-aware behaviors), that can be directly defined in the behavior class.

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

The framework is currently implemented in Netlogo (Wilensky, 1999), which is a multi-agent programmable modeling environment that allows us to rapidly prototype systems of situated agents evolving in a two dimensions world. It provides a sequential synchronization of execution where each agent must finish all the commands in the block before the next agent starts and the order in which agents are chosen to run the commands is random. Simulations are performed within a grid environment where cells can be either empty or occupied by food, agent, obstacle or the nest. The position of obstacles is fixed for all simulations in order to exclude its impact on the multi-agent system performance. The *Multi-Agent Foraging framework* which is notably platform-independent, is translated into a Netlogo platform-specific model and implementation. In Netlogo, each *Forager* is modeled as a *Turtle* having the ability to update information of neighboring or current cells when needed (such as depositing pheromones). Foragers communicate indirectly through their world. Communication is modeled directly in the behavior procedure. Agent behavior is modeled as procedures which can be called from any other procedure. *World* is modeled as a set of *Patches* (immobile agents) disposed on grid. Foragers, food, obstacles and nest are placed on the *world patches. Food* is modeled as a set of *Patches* which can take place at some positions on *world patches*.

Even though, in this paper, we implemented the Multi-Agent Foraging simulation framework in Netlogo, the framework can be easily ported onto other agent-based simulator (or general-purpose simulation environment) such as Repast (Collier, 2003), and ELDASim (Fortino and Russo, 2012).

5.2. Simulation parameters and performance indices

The two world setups (obstacle-free and obstacle environments), which are used for simulations, including positions of nest, food, obstacles and agents, are reported in Fig. 4. There are several simulation parameters that must be properly set: (i) the *World size* is the dimension of the search space, it is a grid of $N \times N$ *cells*; (ii) the *Food density* is the number of food locations (sites), each site contains a limited amount of food. These locations take fixed positions in the environment; (iii) the *Food concentration* indicates the amount of food that each site contains (we refer to it as unit in the paper); (iv) the *Agent's capacity* is the amount of food (units) that an agent can transport at each time; (v) the *Agent's number* is the number of agents that participate in each simulation.

To evaluate the performance of the C-SAF algorithm and the other algorithms, three performance indices have been defined:

- *Average Foraging Time* – The amount of time in seconds needed to finish the foraging mission. It is when all the food sites are discovered and exhausted.
- *Total Food Returned* – The total amount of food returned to the nest by all the agents until a given foraging time.
- *Average Path Length* – Represents the path length between food and the nest, measured as number of cells that form the path.

5.3. Simulation scenarios

Simulations are based on the Netlogo-based Multi-Agent Foraging simulation framework (see Section 5.1). Each simulation is repeated 50 times in the four scenarios. The average value is then calculated from the 50 trials for each scenario. Four fundamental scenarios are used to test the performance of the algorithms (see Table 2):

- *Scenario* 1 is set up to test which of the parameters presented in Section 5.2 can affect the average foraging time. It entails four subscenarios. Subscenario 1 is designed to analyze the impact of the *Agent number* on foraging time. Achieving the foraging task can be very time consuming. Using ticks does not reflect the real time needed to accomplish the task, thus it will be difficult to show the gain of the proposed algorithms in according to computation time. To show the main differences when using each of the two units (virtual ticks and real seconds), we analyzed the average foraging time in subscenario 1 through both ticks and seconds. Subscenario 2 is to test how *World size* can affect the average foraging time and if there is any minimal size under which a number of agents can carry efficiently the foraging task. In order to identify whether or not distributing on single or multiple food locations is beneficial, we used subscenario 3 where we varied the *Food density* from 1 to 10. To test the effect of increasing the *capacity of agents*, we defined subscenario 4 in which the capacity is varying from 1 to 100 units.
- *Scenario* 2 is defined to show the evolution pattern of returned food over time for each of the four algorithms in obstacle-free and obstacle environments.
- *Scenario* 3 is designed to study the effect of the agent number on the *Path length*, whether or not using more agents will reduce the path length.

Table 2
Parameters of Scenarios 1, 2, 3, and 4.

Parameter	Value
Scenario 1	*Average time analysis*
World size	100×100 – 1200×1200 cells
Number of agents	1–10000
Food density	1–10 sites
Food concentration	500–1500 units
Agent's capacity	1–100 units
Scenario 2	*Returned food analysis*
World size	1000×1000 cells
Number of agents	800
Food density	1 site
Food concentration	500 units
Agent's capacity	1 unit
Scenario 3	*Path length analysis*
World size	800×800 cells
Number of agents	1–1000
Food density	1 site
Food concentration	500 units
Agent's capacity	1 unit
Scenario 4	*Scalability analysis*
World size	100×100 – 1200×1200 cells
Number of agents	50–10000
Food density	1 site
Food concentration	500–1500 units
Agent's capacity	1–10 units

Table 3
Definition of the terms used in the results tables and figures.

Term	Definition
Obstacle-free world (free)	*Results in obstacle-free environment for the four algorithms*
Obstacle world (obs)	*Results in obstacle environment for the four algorithms*
C-SAF	*Results for Cooperative Switching Algorithm for Foraging*
NC-SAF	*Results for non-cooperative version of C-SAF*
c-marking	*Results for c-marking algorithm*
NC-c-marking	*Results for the non-cooperative version of c-marking algorithm*
STD-DEV	*Standard deviation of obtained results*

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

Table 4
Average foraging time in ticks and seconds when varying *agent number*.

Agent number	1	10	100	300	500	5000	10000
Obstacle-free world- Time in ticks							
C-SAF	*16029*	*1713*	*316*	*226*	*200*	*210*	*219*
STD-DEV	*0,1*	*5,2*	*3*	*0,4*	*0,4*	*0,9*	*0,4*
NC-SAF	16029	12771	12451	12451	12451	12451	12451
STD-DEV	0,1	0,2	3	0	0	0	0
c-marking	4221763	26205	1232	723	953	454	414
STD-DEV	1020	1320	250	71	45	0,8	50
NC-c-marking	3657570	46122	21428	18676	18076	18144	18157
STD-DEV	1520	456	455	81	31	49	44
Obstacle-free world- Time in seconds							
C-SAF	*65210*	*8569*	*1677*	*1206*	*1038*	*1086*	*1482*
STD-DEV	*5,1*	*1,2*	*1,5*	*5*	*2*	*3*	*5*
NC-SAF	65210	62224	54735	40215	40019	40451	42719
STD-DEV	2,3	1,2	1,6	2	2	4	2
c-marking	95320	35799	7672	4309	4385	1373	1509
STD-DEV	1120	920	620	820	670	85	16
NC-c-marking	100345	100251	65300	61764	54517	55322	52906
STD-DEV	620	850	540	702	450	42	23
Obstacle world- Time in ticks							
C-SAF	*16069*	*1753*	*356*	*266*	*240*	*250*	*259*
STD-DEV	*0,1*	*1,3*	*3*	*0,47*	*0,47*	*0,9*	*0,47*
NC-SAF	16069	12811	12491	12491	12491	12491	12491
STD-DEV	0,1	3,2	5	0	0	0	0
c-marking	4221833	26275	1302	793	1023	524	484
STD-DEV	1310	520	321	41	15	20	20
NC-c-marking	3657615	46167	21473	18721	18121	18189	18202
STD-DEV	1120	890	530	61	11	29	24
Obstacle world- Time in seconds							
C-SAF	*65312*	*8671*	*1779*	*1308*	*1140*	*1188*	*1584*
STD-DEV	*2,2*	*1,2*	*1,3*	*5*	*2*	*4*	*4*
NC-SAF	65312	62327	54837	40317	40121	40553	42821
STD-DEV	3,2	1,3	1,2	4	4	8	3
c-marking	95721	36201	8073	4710	4786	1774	1910
STD-DEV	1359	1320	520	620	420	120	105
NC-c-marking	100540	100652	65701	62165	54918	55724	53307
STD-DEV	1020	985	545	423	145	12	16

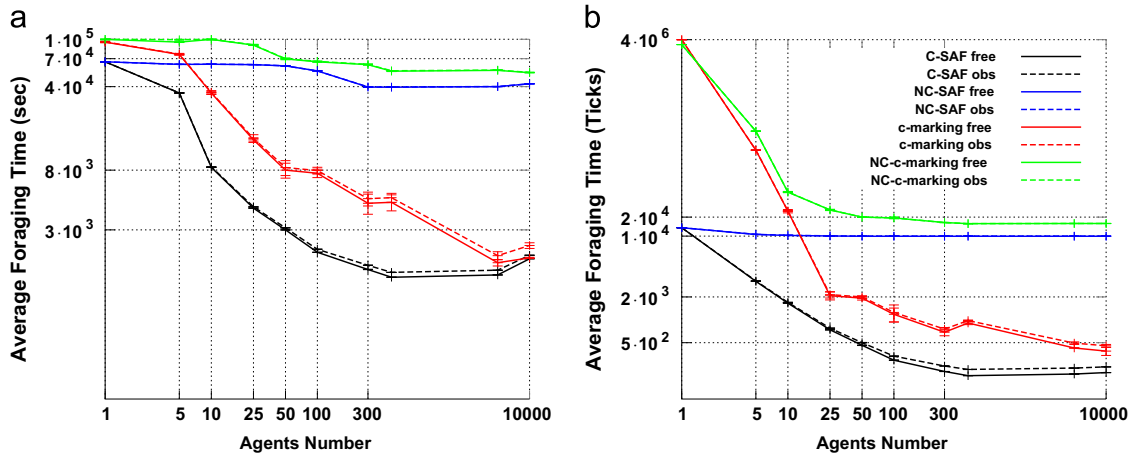

Fig. 5. Average foraging time when varying the agent number: (a) Average foraging time measured in seconds, (b) Average foraging time measured in ticks, where the legend in (b) is the same as in (a).

- *Scenario* 4 is proposed to test the scalability of the algorithms and to analyze how much and in which way agent density variation can affect performances.

5.4. Performance and comparison of cooperative and non-cooperative algorithms

Here we show the results of C-SAF, operating in obstacle-free and obstacle environments, and compare it with NC-SAF, c-marking and NC-c-marking algorithms. We fixed the position of food and obstacles in all simulations to exclude their impact on performances. Table 3 defines terms used in tables and figures about performance evaluation results.

5.4.1. Average time analysis (Scenario 1 results)

We first present in this section the performance of the four algorithms as the number of agents grows where the average foraging time is measured in *seconds* and in *ticks* respectively (see

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

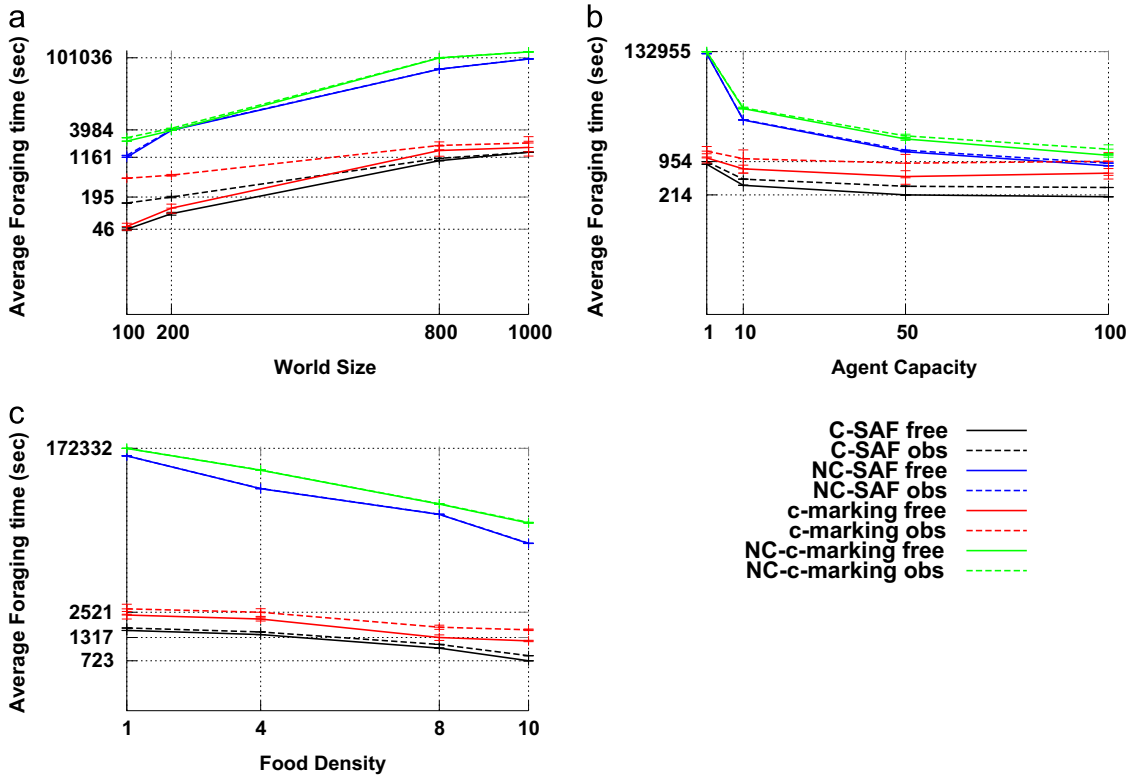

Fig. 6. Results of scenario 1 when varying: (a) world size, (b) agent capacity and (c) food density.

Table 5
Average foraging time (sec) when varying *world size*.

World size	100 × 100	200 × 200	800 × 800	1000 × 1000
Obstacle-free world				
C-SAF	*46*	*92*	*999*	*1355*
STD-DEV	*3*	*6*	*8*	*5*
NC-SAF	1161	3881	60620	95975
STD-DEV	4	5	5	9
c-marking	52	118	1569	1803
STD-DEV	8	24	349	578
NC-c-marking	2389	3860	100634	131428
STD-DEV	2	5	234	845
Obstacle world				
C-SAF	*149*	*195*	*1101*	*1458*
STD-DEV	*2*	*7*	*9*	*7*
NC-SAF	1264	3984	60722	96077
STD-DEV	8	6	9	9
c-marking	454	519	1970	2204
STD-DEV	5	19	344	720
NC-c-marking	2790	4261	101036	131830
STD-DEV	2	9	533	734

Table 6
Average foraging time (sec) when varying *agent capacity*.

Agent capacity	1	10	50	100
Obstacle-free world				
C-SAF	*852*	*330*	*214*	*197*
STD-DEV	*4*	*1*	*4*	*2*
NC-SAF	121941	6165	1490	792
STD-DEV	5	4	3	1
c-marking	1130	692	490	569
STD-DEV	230	123	144	132
NC-c-marking	132554	10313	2639	1272
STD-DEV	430	223	149	102
Obstacle world				
C-SAF	*954*	*433*	*316*	*299*
STD-DEV	*4*	*3*	*1*	*1*
NC-SAF	122043	6267	1592	894
STD-DEV	8	6	9	9
c-marking	1532	1093	892	970
STD-DEV	355	520	431	455
NC-c-marking	132955	10714	3040	1673
STD-DEV	355	320	231	355

Table 4 and Fig. 6, results are obtained by fixing the other simulation parameters as follows: world size = 1200 × 1200 cells, food density = 1, food concentration = 1500 units, agent capacity = 10 units, nest number = 1). The algorithm was first evaluated with small number of agents (1, 5, 10, 25 and 50) and then this number was progressively increased to reach 10000. While the foraging time increases faster as the number of agents increases from 1 to 500. It is higher with 1 agent (65210 s), halved with 5 agents (35796 s) and even reduced more with 10 and 25 agents (8569 s and 3916 s respectively). It is halved from 50 to 300 and reduced considerably from 300 to 500. Above 500 agents, the foraging time decreases slower progressively until it reaches 1482 s, in obstacle-free environment, and 1584 s, in obstacle environment, with 10000 agents. Between 5000 and 10000 agents, in C-SAF there is a slight increase of the foraging time; this happens only when using one food location (as in this case) as it is related to the length of search path that agents need to traverse to get to the food location (Fig. 4(c) and (d) shows an example of execution of such a case). Conversely, when we used multiple food locations (see Fig. 4 (e) and (f)), the foraging time keeps decreasing as also subscenario 4 (see below) will confirm. In c-marking algorithms, foraging time becomes faster as the number of agents increases. However, if the food position is out of the wavefront expansion the pseudo random walk can slow down the search task, thus the foraging time. Foraging time in non-cooperative protocols is very high compared to the cooperative ones, however, NC-SAF give better results than the c-marking one when the number of agents is low (1 and 5 agents) because in c-marking algorithm the wavefront created

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

with a low number of agents is small and the agents take much more time in search. The C-SAF algorithm gives better results than the other three cooperative and the non-cooperative protocols; it is much faster (up to 4 times more) than the c-marking algorithm; in fact, while C-SAF obtains its best performance with 500 agents (1140 s, in obstacle environment), c-marking obtains its best performance with 5000 agents (1744 s). This implies that C-SAF not only outperforms c-marking in terms of foraging time but also it would save a lot of resources so allowing for practical feasibility of the approach. The standard deviation is smaller in C-SAF algorithms and higher in c-marking algorithms where it decreases when increasing the number of agents. Results in obstacle environments are higher than in obstacle-free ones in the four protocols. Curves obtained in the four algorithms show similar trends when using seconds (Fig. 5(a)) or when considering ticks (Fig. 5 (b)). However, to reconcile the results to those that can be obtained in the real world, we use only seconds as measurement unit for the average foraging time in the other scenarios.

In Table 5 and Fig. 6(a), we can observe that foraging time is almost linear with respect to the growth of the world size in the four protocols. Results are obtained by fixing the other simulation parameters as follows: agent number = 800, food density = 1, food concentration = 500 units, agent capacity = 1 unit, nest number = 1.

Table 7
Average foraging time (sec) when varying *food density*.

Food density	1	4	8	10
Obstacle-free world				
C-SAF	*1580*	*1420*	*1002*	*723*
STD-DEV	*2*	*1*	*2*	*1*
NC-SAF	141931	60965	31452	14842
STD-DEV	8	3	3	2
c-marking	2350	2120	1317	1205
STD-DEV	230	103	94	21
NC-c-marking	171931	98210	40895	25122
STD-DEV	630	223	143	102
Obstacle world				
C-SAF	*1682*	*1522*	*1104*	*825*
STD-DEV	*2*	*2*	*3*	*1*
NC-SAF	142033	61068	31554	14944
STD-DEV	10	2	3	3
c-marking	2751	2521	1718	1606
STD-DEV	355	250,1	91,8	30
NC-c-marking	172332	98612	41297	25523
STD-DEV	730	323	103	95

World size can slow down the foraging time in C-SAF algorithm if it cannot contain a large number of agents, so reducing agent cooperation in food transportation. In the c-marking algorithm, when increasing the world size and the number of agents is not sufficient to sustain the wavefront expansion, the paths are not optimal and the foraging time increases. However, in this scenario the number of agents is sufficient to create the wavefront to food so helping at reducing foraging time. The non-cooperative protocols provide very high foraging time in comparison to the cooperative ones and C-SAF provides the smallest foraging time in all world configurations. Standard deviation is smaller in C-SAF protocols and increases in c-marking protocols from smaller to higher values because of the random walk. Results in obstacle environments are higher than in obstacle-free environments.

Increasing the agent capacity helps at reducing the average foraging time in the four algorithms (see Table 6 and Fig. 6(b), results are obtained by fixing the other simulation parameters as follows: world size = 800×800 cells, agent number = 500, food density = 1, food concentration = 500 units, nest number = 1). However, C-SAF algorithm takes much less average time to finish the foraging regarding the other three algorithms. The average time for c-marking agents is better than the one obtainable with the non-cooperative algorithms, NC-SAF and NC-c-marking, but it is much worse compared to C-SAF. The time decreases due to the reduction in the number of turns needed to transport all the food (e.g. agents need 10 turns to transport 10 units when capacity is 1, but they need 1 turn when capacity is 10 units).

Finally, in subscenario 4 (Table 7 and Fig. 6(c), results are obtained by fixing the other simulation parameters as follows: world size = 1000×1000 cells, agent number = 500, food concentration = 500 units, agent capacity = 1 unit, nest number = 1), we varied food density from 1 to 10, which means that we got multiple food locations and the total amount of food is distributed among them. The average foraging time decreases as the food density increases until it is halved in C-SAF and c-marking. However, the average time increases dramatically when the food locations are far away from the nest, especially in c-marking algorithms where the paths are too long and non optimal. Similar performance results are obtained in obstacle environment, with additional time needed to avoid obstacles.

5.4.2. *Returned food analysis (Scenario 2 results)*

C-SAF algorithm has a very fast time to locate and exhaust all the food (1355 s in obstacle-free environment), which is not the

Table 8
Returned food over time until the foraging completion for all the four algorithms.

Time	500	1355	1450	1803	2035	95,975	96,345	131,428	152,412
Obstacle-free world									
C-SAF	*41*	*500*	*500*	*500*	*500*	*500*	*500*	*500*	*500*
STD-DEV	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
NC-SAF	2	5	6	8	9	500	500	500	500
STD-DEV	2,5	1,5	0	0	0	0	0	0	0
c-marking	0	194	251	500	500	500	500	500	500
STD-DEV	0	2	2	35	0	0	0	0	0
NC-c-marking	0	2	2	3	4	335	336	500	500
STD-DEV	0	0,7	0,7	1	2	3	7	1	0
Obstacle world									
C-SAF	*35*	*367*	*500*	*500*	*500*	*500*	*500*	*500*	*500*
STD-DEV	*1*	*2*	*1*	*0*	*0*	*0*	*0*	*0*	*0*
NC-SAF	0	0	1	2	4	466	500	500	500
STD-DEV	0	1	1	2	1	2	1	0	0
c-marking	0	180	241	489	500	500	500	500	500
STD-DEV	0	3	3	23	13	0	0	0	0
NC-c-marking	0	0	0	1	2	331	332	493	500
STD-DEV	0	0	0	0,7	0,7	10	4	16	1

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

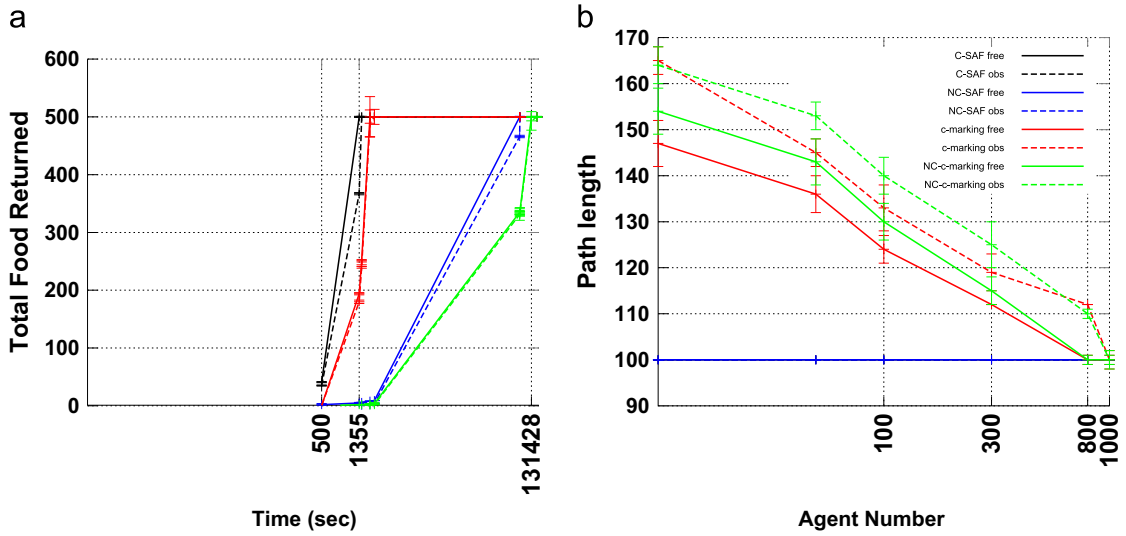

Fig. 7. (a) Returned food analysis; (b) Path length analysis. The legend in (a) is the same as in (b).

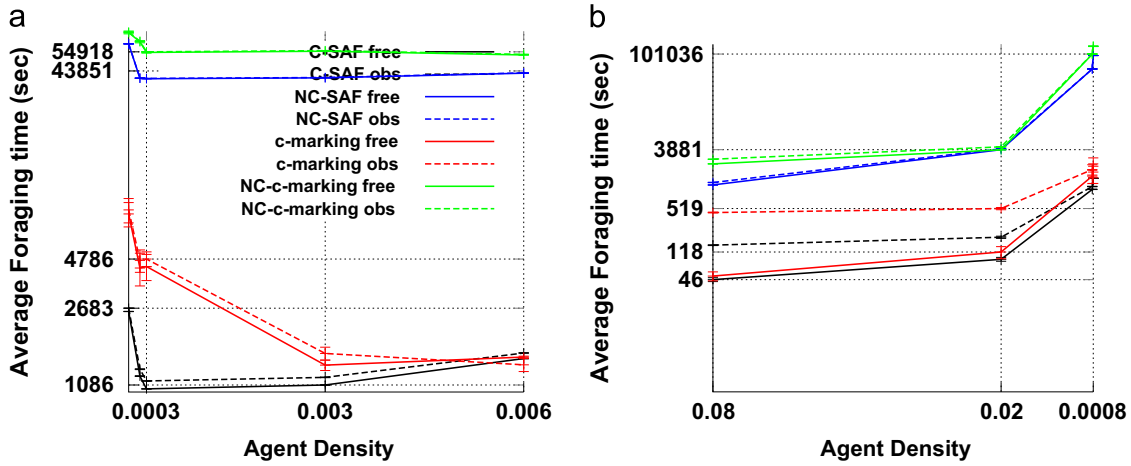

Fig. 8. Scalability analysis: (a) increasing agent density by fixing world size and varying agent number; (b) decreasing agent density by fixing agent number and varying world size. The legend in (b) is the same as in (a).

Table 9
Average path length when varying agent number.

Agent number	10	50	100	300	800	1000
Obstacle-free world						
C-SAF	*100*	*100*	*100*	*100*	*100*	*100*
STD-DEV	*0*	*0*	*0*	*0*	*0*	*0*
NC-SAF	100	100	100	100	100	100
STD-DEV	0	0	0	0	0	0
c-marking	147	136	124	112	100	100
STD-DEV	5	4	3	0,7	1	1
NC-c-marking	154	143	130	115	100	
STD-DEV	5	5	4	3	1	2
Obstacle world						
C-SAF	*100*	*100*	*100*	*100*	*100*	*100*
STD-DEV	*0*	*0*	*0*	*0*	*0*	*0*
NC-SAF	100	100	100	100	100	100
STD-DEV	0	0	0	0	0	0
c-marking	165	145	133	119	112	100
STD-DEV	3	3	5	4	0,7	2
NC-c-marking	164	153	140	125	110	100
STD-DEV	4	3	4	5	1	1

case for the non-cooperative protocols that take about 95975 and 131428 s (NC-SAF and NC-c-marking in obstacle-free environment, respectively). C-SAF retains higher performance than c-marking: about 500 s better. In this scenario, we reported a sample of the total amount of food returned overtime without the analysis of the parameters that can affect its evolution. However, this scenario is very much related to Scenario 1 and, for all the test configurations of Scenario 1, we had the same performance profile for the four protocols. Table 8 and Fig. 7(a), show the results obtained by the four algorithms in obstacle-free and obstacle environments (results are obtained by fixing the other simulation parameters as follows: world size = 1000 × 1000 cells, agent number = 800, agent capacity = 1 unit, food density = 1, food concentration = 500 units, nest number = 1).

5.4.3. *Path length analysis (Scenario 3 results)*

Regardless of the number of agents, C-SAF and NC-SAF protocols produce optimal paths (with 100 cells length) simultaneously when exploring their environment. In c-marking algorithms, the path length is non optimal and decreases when increasing the number of agents until it reaches the optimal value with 800 agents in obstacle-free environment (1000 agents in obstacle environment) (see Table 9 and Fig. 7(b), results are obtained by fixing the other simulation parameters as follows: world size = 800 × 800 cells, agent capacity = 1 unit, food density = 1, food concentration = 500 units, nest number = 1). It is worth noting that agents in the c-marking algorithm need to visit the same

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

Table 10
Scalability analysis: increasing agent density by fixing world size and varying agent number.

Agent density	0.00003	0.0002	0.0003	0.003	0.006
Agent number	50	300	500	5000	10,000
World size	1200 × 1200	1200 × 1200	1200 × 1200	1200 × 1200	1200 × 1200
Obstacle-free world					
C-SAF	*2580*	*1206*	*1038*	*1086*	*1482*
STD-DEV	*4*	*5*	*2*	*3*	*5*
NC-SAF	60302	40215	40019	40451	42719
STD-DEV	4	2	2	4	2
c-marking	8142	4309	4385	1373	1509
STD-DEV	1160	820	670	85	16
NC-c-marking	68900	61764	54517	55322	52906
STD-DEV	1050	702	450	42	23
Obstacle world					
C-SAF	*2683*	*1308*	*1140*	*1188*	*1584*
STD-DEV	*5*	*5*	*2*	*4*	*4*
NC-SAF	60404	40317	40121	40553	42821
STD-DEV	4	4	4	8	3
c-marking	8544	4710	4786	1774	1910
STD-DEV	1220	620	420	120	105
NC-c-marking	69302	62165	54918	55724	53307
STD-DEV	789	423	145	12	16

Table 11
Scalability analysis: decreasing agent density by fixing agent number and varying world size.

Agent density	0.08	0.02	0.001	0.0008
World size	100 × 100	200 × 200	800 × 800	1000 × 1000
Agent number	800	800	800	800
Obstacle-free world				
C-SAF	*46*	*92*	*999*	*1355*
STD-DEV	*3*	*6*	*8*	*5*
NC-SAF	1161	3881	60620	95975
STD-DEV	4	5	5	9
c-marking	52	118	1569	1803
STD-DEV	8	24	349	578
NC-c-marking	2389	3860	100634	131428
STD-DEV	2	5	234	845
Obstacle world				
C-SAF	*149*	*195*	*1101*	*1458*
STD-DEV	*2*	*7*	*9*	*7*
NC-SAF	1264	3984	60722	96077
STD-DEV	8	6	9	9
c-marking	454	519	1970	2204
STD-DEV	5	19	344	720
NC-c-marking	2790	4261	101036	131830
STD-DEV	2	9	533	734

Table 12
Comparison highlights of the most important results between C-SAF and the other algorithms in obstacle environment.

Algorithms	**Performance indices**			
	Foraging time	**Returned food**	**Path length**	**Scalability**
C-SAF	1140 s with 500 agents	500 units in 1450 s	100 cells with 10 agents	2683 s with low density=3E−6; 149 s with high density=8E−2;
c-marking	1774 s with 5000 agents, equals to C-SAF with 200	500 units in 2035 s 241 units in 1450 s	100 cells with 1000 agents; 165 cells with 10 agents	8544 s with low density=3E−6; 454 s with high density=8E−2;
NC-SAF	40,121 s with 500 agents	500 units in 96,345 s 1 unit in 1450 s	100 cells with 10 agents	60,404 s with low density=3E−6; 1264 s with high density=8E−2;
NC-c-marking	53,307 s with 10,000 agents	500 units 152,412 s 0 unit in 1450 s	100 cells 1000 agents; 164 cells with 10 agents	69,302 s with low density=3E−6; 2790 s with high density=8E−2;

cell several times to get to its optimal APF value. The position of the food can affect the path length: it will be shorter if food is close to nest and longer otherwise. However, in c-marking algorithms the number of agents, world size, food position and pseudo random walk can affect the path length. Results obtained with C-SAF and NC-SAF in obstacle-free and obstacle environment are very close and their curves overlap in Fig. 7(b).

5.4.4. Scalability analysis (Scenario 4 results)

In this section, we analyze the scalability of the proposed algorithm through two subscenarios on the basis of the *Agent Density*, defined as *Agent Number* divided by *World Size*: (1) increasing the agent density from 0.00003 to 0.006, by varying the agent number and fixing the world size; (2) decreasing the *agent density* from 0.08 to 0.0008, by fixing the agent number and varying the world size. With reference to subscenario 1, results are shown in Table 10 and Fig. 9(a). In the C-SAF case, the foraging time quickly decreases with the increase of *agent density* until 0.0003 value; from such value, the foraging time starts increasing slightly. A similar trend happens for the c-marking algorithm even though with worse performances. The main reason for this is the long path that agents need to traverse to reach the food location.

In subscenario 4 of Scenario 1 (see Section 5.4.1), we showed that using multiple food locations helps at halving the average foraging time, thus avoiding the limitation in scalability when increasing too much the *agent density*. Regarding subscenario 2, the obtained results are shown in Table 11 and Fig. 8(b). Here we can see that time increases with the decrease of the *agent density*: it is proportional to the growth of the world size. Obviously, increasing the world size requires more time for exploration and transportation of food. In such scenario, the other simulation parameters are set as follows: agent capacity=10 units, food density=1, food concentration=500 units, nest number=1.

5.4.5. Summary of results

In the four different analyzed scenarios (average time, returned food, path length, and scalability analysis), we have shown that C-SAF algorithm outperforms the considered cooperative (c-

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

marking) and non cooperative (NC-SAF and NC-c-marking) algorithms. In Table 12, we report a summary of the comparison between C-SAF and the other algorithms, highlighting the most important performance differences in the four analyzed scenarios. In particular, C-SAF provides on average less time and resources (in terms of number of agents) to finish the foraging task (see first column of Table 12), larger amount of returned food at a given time (see second column of Table 12), and optimal paths (see third column of Table 12). The differences between C-SAF and the other algorithms mainly stems from the more rapid exploration of the environment and the more rapid transportation of food to the nest. Concerning space exploration, our agents use the S-MASA algorithm (see Section 2.3) that, based on a guided search to non-visited cells in vortex-like movements, provides both a large dispersion, therefore a quicker search, and optimal paths to return home. Regarding food transportation, C-SAF agents depose pheromone to attract other agents in vicinity when a food is found and, due to the vortex-like movements, a large number of agents can be recruited and cooperate to transport the food to the nest, thus accelerating the exploitation task. Moreover, the C-SAF algorithm has also shown a good scalability (see forth column of Table 12).

6. Toward a robotic implementation

Stigmergic communications via pheromone trails have shown to efficiently coordinate a team of robots and to allow them to quickly explore a given terrain (Beckers et al., 1994; Svennebring and Koenig, 2004). Most of the works on stigmergic coordination are based on unrealistic assumptions (Kuyucu et al., 2012). This kind of assumptions can help in analyzing and improving pheromone-based algorithms in simulations before their real robotic implementation. One of the key difficulties in the real development of such mechanisms is the implementation of the pheromone itself and how it can interact with agents. To date, several techniques for virtual marking have been proposed.

- Robots can mark physically their trail via physical marks using: *alcohol* (Sharpe and Webb, 1998), *virtual marks* (Svennebring and Koenig, 2004) or *RFID tags* (Mamei and Zambonelli, 2005).
- Robots communicate a pheromone model using wireless network (Vaughan et al., 2000).
- Robots transmit virtual pheromone using infrared communication (Payton et al., 2001).
- Robots can switch to beacons or even use immobile beacons to communicate pheromone like information (Barth, 2003).

Despite the proposed approaches, the implementation of pheromone is still in its early stages and most of the works are available in research laboratories. One of the works that we believe it constitutes an important way to the real implementation of our approach is the one in Ranjbar-Sahraei et al. (2012), where pheromone is defined as an electrical marker that can be placed at given positions in the environment, it can fully evaporate after time. From a theoretical point of view, two changes can take place. First, the robots should take specific initial locations on the four headings (0°, 90°, 180°, 270°) in order to avoid the bottleneck situation around the nest. Second, additional approaches to avoid collisions between robots when using the same homing trail are needed (e.g. leaving the trail for Laden robots). A case study that we plan to consider in the future, is deploying a group of mobile robots, based on a pheromone mechanism similar to the one proposed in Ranjbar-Sahraei et al. (2012), which can perform tasks such as searching for objects. We intend to use the agent programming methodologies and tools from Fortino et al. (2015) and Fortino and Russo (2012), to perform a rapid prototyping of our approach in the PROFETA language (Fortino et al., 2013) in order to investigate which design and implementation choices are required before the real implementation.

7. Conclusion

In this paper, we have presented a distributed foraging algorithm called C-SAF. C-SAF agents are simple, reactive, with small memory and limited perception capabilities. They communicate by depositing pheromone, i.e. stigmergic communications. We have also presented a flexible simulation framework for Multi-Agent Foraging that can be reused and/or extended to address the needs of other foraging algorithms.

Four fundamental scenarios have been modeled and simulated to test the effect of the basic parameters (agent number, world size, food density, food concentration, agent capacity) on the performances of the four cooperative and non-cooperative algorithms (C-SAF, NC-SAF, c-markimg and NC-c-marking) in terms of the defined performance indices, i.e. average foraging time, amount of returned food and average path length. In the first scenario, we have varied the basic parameters to test their effect on the average foraging time measured in real seconds; in the second scenario, we have shown the evolution pattern of returned food over time; in the third scenario, we varied the number of agents to observe its impact on the length of created paths; finally in the fourth scenario, we analyzed the scalability of the algorithm when varying agent density. In all the four scenarios, the C-SAF provided the best results in terms of average foraging time, total amount of returned food and average path length, both in obstacle-free and obstacle environments. The average foraging time was up to 4 times less than the one obtainable by the c-marking algorithm which is a main reference in the literature. Moreover simulation results demonstrate the benefit of cooperation in Multi-Agent Foraging when the team of agents is efficiently coordinated.

The C-SAF outperforms the c-marking one in three fundamental points: (1) the quick search provided by the S-MASA algorithm, (2) the optimal homing paths created simultaneously and synchronously while exploring and, (3) the high level of cooperation to exploit and transport food by diffusing pheromone to neighboring cells. But of course the C-SAF algorithm has some degradation in performances in some cases: (1) the presence of obstacles with complex shape, (2) when food is close to boundaries a small number of agents will contribute to transport because most of them already reach the boundaries and (3) for large-scale worlds, visiting all the environment to reach the food that is very far from the nest, could duplicate the area to search and foraging time increases.

In the near future, we intend to test our approach using real robots, after its implementation and validation supported by the ELDAMeth methodology and the PROFETA language. Moreover, we will provide the implemented Multi-Agent Foraging framework in open-source to support researchers interested in modeling and understanding particular Multi-Agent Foraging systems. In fact, by providing an open-source common framework, we may also facilitate comparison among many different foraging systems.

References

Acar, E., Choset, H., Zhang, Y., Schervish, M., 2003. Path planning for robotic demining: robust sensor-based coverage of unstructured environments and probabilistic methods. Int. J. Robot. Res. 22 (7–8), 441–466.

Balch, T., 1996. Grid-based navigation for mobile robots. Robot. Pract. 2 (1), 6–11.

Barnes, L., Alvis, W., Fields, M., Valavanis, K., Moreno, W., 2006. Swarm formation control with potential fields formed by bivariate normal functions. In: 14th

Mediterranean Conference on Control and Automation, 2006 (MED'06). IEEE, pp. 1–7.
Barraquand, J., Langlois, B., Latombe, J.-C., 1992. Numerical potential field techniques for robot path planning. IEEE Trans. Syst. Man Cybern. 22 (2), 224–241.
Barth, E.J., 2003. A dynamic programming approach to robotic swarm navigation using relay markers. In: Proceedings of the American Control Conference, vol. 6. IEEE, Denver, Colorado, USA, pp. 5264–5269.
Batalin, M.A., Sukhatme, G.S., 2002. Spreading out: a local approach to multi-robot coverage. In: Proceedings of 6th International Symposium on Distributed Autonomous Robotic Systems, pp. 373–382.
Beckers, R., Holland, O., Deneubourg, J.-L., 1994. From local actions to global tasks: stigmergy and collective robotics. In: Artificial life IV, vol. 181. p. 189.
Bonabeau, E., Dorigo, M., Theraulaz, G., 1999. Swarm Intelligence: From Natural to Artificial Systems, vol. 1. Oxford University Press, New York, USA.
Calvo, M.F.R., de Oliveira, J.R., Romero, R.A.F., 2011. Bio-inspired coordination of multiple robots systems and stigmergy mechanims to cooperative exploration and surveillance tasks. In: IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS), pp. 223–228.
Chattunyakit, S., Kondo, T., Nilkhamhang, I., Phatrapornnant, T., Kumazawa, I., 2013. Two foraging algorithms for a limited number of swarm robots. In: Proceedings of SICE Annual Conference (SICE), IEEE, Nagoya, Japan, pp. 1056–1061.
Choi, Y.-H., Lee, T.-K., Baek, S.-H., Oh, S.-Y., 2009. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Saint-Louis, Missouri, USA, pp. 5788–5793.
Collier, N. 2003. Repast: an extensible framework for agent simulation. The University of Chicago's Social Science Research, vol. 36, 2003.
Dorigo, M., Bonabeau, E., Theraulaz, G., 2000. Ant algorithms and stigmergy. Future Gener. Comput. Syst. 16 (8), 851–871.
Dorigo, M., Birattari, M., Stützle, T., 2006. Ant colony optimization. Comput. Intell. Mag. 1 (4), 28–39.
Feinerman, O., Korman, A., Lotker, Z., Sereni, J.S., 2012. Collaborative search on the plane without communication. In: Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing, ACM, New York, USA, pp. 77–86.
Fortino, G., Russo, W., 2012. Eldameth: an agent-oriented methodology for simulation-based prototyping of distributed agent systems. Inf. Softw. Technol. 54 (6), 608–624.
Fortino, G., Russo, W., Santoro, C., 2013. Translating statecharts-based into BDI agents: the dsc/profeta case. In: Multiagent System Technologies, Springer, pp. 264–277.
Fortino, G., Zedadra, O., Jouandeau, N., Seridi, H., 2014. A decentralized ant colony foraging model using only stigmergic communication. In: Proceedings of XV Workshop Dagli Oggetti agli Agenti (WOA 2014), vol. 1260, CEUR.
Fortino, G., Rango, F., Russo, W., Santoro, C., 2015. Translation of statechart agents into a BDI framework for MAS engineering. Eng. Appl. Artif. Intell. 41, 287–297.
Freeman, R.A., Yang, P., Lynch, K.M., et al., 2006. Distributed estimation and control of swarm formation statistics. In: American Control Conference, vol. 7, Citeseer.
Gabriely, Y., Rimon, E., 2002. Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, IEEE, pp. 954–960.
Gage, D., 1995. Many-robot MCM search systems. In: Autonomous Vehicles in Mine Countermeasures Symposium, vol. 9, pp. 56–64.
Grassé, P.-P., 1959. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insectes sociaux 6 (1), 41–80.
Gutjahr, W.J., 2000. A graph-based ant system and its convergence. Future Gener. Comput. Syst. 16 (8), 873–888.
Hoff III, N.R., Sagoff, A., Wood, R.J., Nagpal, R., 2010. Two foraging algorithms for robot swarms using only local communication, in: IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE, 2010, pp. 123–130.
Hoff, N., Wood, R., Nagpal, R., 2013. Distributed colony-level algorithm switching for robot swarm foraging. In: Springer Distributed Autonomous Robotic Systems, pp. 417–430.
Jennings, J., Whelan, G., Evans, W., 1997. Cooperative search and rescue with a team of mobile robots. In: IEEE 8th International Conference on Advanced Robotics (ICAR), pp. 193–200.
Jiang, Q., 2006. An improved algorithm for coordination control of multi-agent system based on r-limited voronoi partitions. In: 2006 IEEE International Conference on Automation Science and Engineering, pp. 667–671.
Kantor, G., Singh, S., Peterson, R., Rus, D., Das, A., Kumar, V., Pereira, G., 2006. Distributed search and rescue with robot and sensor teams, in: Field and Service Robotics, Springer, Berlin Heidelberg, pp. 529–538.
Kuyucu, T., Tanev, I., Shimohara, K., 2012. Evolutionary optimization of pheromone-based stigmergic communication, Applications of Evolutionary Computation. Springer, Málaga, Spain, pp. 63–72.
Landis, A. Geoffrey, 2003. Robots and humans: synergy in planetary exploration. In: Space Technology and Applications Int. Forum-Staif 2003: Conference on Thermophysics in Microgravity; Commercial/Civil Next Generation Space Transportation; Human Space Exploration; Symposium on Space Nuclear Power and Propulsion (20th); Space Colonization (1st), vol. 654, AIP Publishing, Albuquerque, New Mexico, USA, pp. 853–860.
Lau, B., Sprunk, C., Burgard, W., 2013. Efficient grid-based spatial representations for robot navigation in dynamic environments. Robot. Auton. Syst. 61 (10), 1116–1130.
Lee, J.-H., Ahn, C.W., 2011. Improving energy efficiency in cooperative foraging swarm robots using behavioral model. In: 2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), IEEE, Universiti Sains Malaysia, Penang, Malaysia , pp. 39–44.
Lee, J.-H., Ahn, C.W., An, J., 2013. A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: an empirical analysis. In: 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), IEEE, Novotel, Wollongong, Australia, pp. 489–493.
Liu, A.L.D., Zhou, X., Guan, H., 2010. A swarm intelligence based algorithm for distribute search and collective cleanup. In: IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), vol. 2. IEEE, Xiamen University, China, pp. 161–165.
Méndez, D.C.V., Bartumeus, F., 2014. Random search strategies. Stochastic Foundations in Movement Ecology, vol. 23. Springer-Verlag, Berlin Heidelberg, pp. 177–205.
Mamei, M., Zambonelli, F., 2005. Spreading pheromones in everyday environments via RFID technologies. In: Proceedings of the 2nd IEEE Symposium on Swarm Intelligence.
Marjovi, A., Nunes, J.G., Marques, L., de Almeida, A., 2009. Multi-robot exploration and fire searching. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1929–1934.
Marjovi, A., Nunes, J., Marques, L., de Almeida, A., 2009. Multi-robot exploration and fire searching. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1929–1934.
Mataric, M.J., 1994. Interaction and intelligent behavior. Technical Report, DTIC Document.
Meng, Z., Zou, B., Zeng, Y., 2012. Considering direct interaction of artificial ant colony foraging simulation and animation. J. Exp. Theor. Artif. Intell. 24 (1), 95–107.
Momen, S., 2013. Ant-inspired decentralized task allocation strategy in groups of mobile agents. Procedia Comput. Sci. 20, 169–176.
Panait, L., Luke, S., 2004. A pheromone-based utility model for collaborative foraging. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 1. IEEE Computer Society, New York, USA, pp. 36–43.
Panov, S., Koceska, N., 2014. Global path planning in grid-based environments using novel metaheuristic algorithm. In: ICT Innovations 2013, Springer, Ohrid, Macedonia, pp. 121–130.
Pasqualetti, F., Franchi, A., Bullo, F., 2010. On optimal cooperative patrolling. In: Proceedings of the 49th IEEE Conference on Decision and Control (CDC), pp. 7153–7158.
Payton, D.W., Daily, M.J., Hoff, B., Howard, M.D., Lee, C.L., 2001. Pheromone robotics. In: Intelligent Systems and Smart Manufacturing, International Society for Optics and Photonics, pp. 67–75.
Pitonakova, L., Crowder, R., Bullock, S., 2014. Understanding the role of recruitment in collective robot foraging. In: Lipson, H.e.a. (Ed.), Proceedings of The Fourteenth International Conference on the Synthesis and Simulation of Living Systems ALIFE. MIT Press, Javits Center / SUNY Global Center, New York, USA, pp. 1477–1485.
Ranjbar-Sahraei, B., Weiss, G., Nakisaee, A., 2012. A multi-robot coverage approach based on stigmergic communication. In: Multiagent System Technologies, Springer, University of Trier, Germany, pp. 126–138.
Schilling, K., Jungius, C., 1996. Mobile robots for planetary exploration. Control Eng. Pract. 4 (4), 513–524.
Sharpe, T., Webb, B., 1998. Simulated and situated models of chemical trail following in ants. In: Proceedings of the 5th International Conference on Simulation of Adaptive Behavior, pp. 195–204.
Shell, D., Mataric, M.J., et al., 2006. On foraging strategies for large-scale multi-robot systems. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Beijing, China, pp. 2717–2723.
Simonin, O., Charpillet, F., Thierry, E., 2014. Revisiting wavefront construction with collective agents: an approach to foraging. Swarm Intell. 8 (2), 113–138.
Speranzon, A., 2006. Coordination, consensus and communication in multi-robot control systems (Ph.D. thesis), Stockholm, Sweden.
Stipes, J., Hawthorne, R., Scheidt, D., Pacifico, D., 2006. Cooperative localization and mapping. In: Proceedings of the IEEE International Conference on Networking, Sensing and Control (ICNSC), pp. 596–601.
Svennebring, J., Koenig, S., 2004. Building terrain-covering ant robots: a feasibility study. Autonom. Robots 16 (3), 313–332.
Thrun, S., Bücken, A., 1996. Integrating grid-based and topological maps for mobile robot navigation. In: Proceedings of the National Conference on Artificial Intelligence, pp. 944–951.
Vaughan, R.T., Støy, K., Sukhatme, G.S., Matarić, M.J., 2000. Blazing a trail: insect-inspired resource transportation by a robot team. In: Distributed Autonomous Robotic Systems 4, Springer, pp. 111–120.
Vaughan, R., 2008. Massively multi-robot simulation in stage. Swarm Intell. 2 (2–4), 189–208.
WeiXing, Y.X.F., KeJun, W., ShuXiang, G., 2006. Novel algorithms for coordination of underwater swarm robotics. In: IEEE International Conference on Mechatronics and Automation, pp. 654–659.
WeiXing, F., KeJun, W., XiuFen, Y., ShuXiang, G., 2006. Novel algorithms for coordination of underwater swarm robotics. In: Proceedings of the International Conference on Mechatronics and Automation, IEEE, Luoyang, China, pp. 654–659.
Wilensky, U., 1997. Netlogo ants model, Center for connected learning and computer-based modeling, Northwestern University, Evanston, IL. [Online]. Available: ⟨http://ccl.northwestern.edu/netlogo/models/Ants⟩.
Wilensky, U., 1999. Netlogo. ⟨http://ccl.northwestern.edu/netlogo/⟩. In: Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025

Winfield, A.F., 2009. Foraging robots, Encyclopedia of Complexity and Systems Science. Springer, Bristol, UK, pp. 3682–3700.
Yan, Z., Jouandeau, N., Cherif, A.A., 2013. A survey and analysis of multi-robot coordination. Int. J. Adv. Robot. Syst. 10, 399.
Yean, Y.P., Chetty, R.K., 2012. An efficient grid based navigation of wheeled mobile robots based on visual perception, Trends in Intelligent Robotics, Automation, and Manufacturing. Springer, Kuala Lumpur, Malaysia, pp. 128–135.
Zedadra, O., Jouandeau, N., Seridi, H., Fortino, G., 2014. S-MASA, A stigmergy based algorithm for multi-target search. In: Ganzha, M.P.M., Maciaszek, L. (Eds.), Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Annals of Computer Science and Information Systems, vol. 2. IEEE, Warsaw, Poland, pp. 1477–1485.
Zedadra, O., Seridi, H., Jouandeau, N., Fortino, G., 2015. Design and analysis of cooperative and non cooperative stigmergy-based models for foraging. In: IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, Calabria, Italy, pp. 85–90.
Zedadra, O., Seridi, H., Jouandeau, N., Fortino, G., 2015. A distributed foraging algorithm based on artificial potential field. In: 12th International Symposium on Programming and Systems (ISPS). IEEE, Gdansk, Poland, pp. 1–6.

Please cite this article as: Zedadra, O., et al., A Cooperative Switching Algorithm for Multi-Agent Foraging. Eng. Appl. Artif. Intel. (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.025