Energy Expenditure in Multi-Agent Foraging: An Empirical Analysis

Ouarda ZEDADRA and Hamid SERIDI
LabSTIC Laboratory, 8 may 1945 University
P.O.Box 401, 24000 Guelma, Algeria
Department of computer science Badji Mokhtar
Annaba University P.O.Box 12, 23000 Annaba, Algeria
Email: zedadra_nawel1, seridihamid@yahoo.fr

Nicolas JOUANDEAU
Advanced Computing laboratory
of saint-Denis Paris 8 University
Saint Denis 93526, France
Email: n@ai.univ-paris8.fr

Giancarlo FORTINO
DIMES, Universita'
della Calabria
Via P. Bucci, cubo 41c - 87036
- Rende (CS) - Italy
Email:g.fortino@unical.it

Abstract—**A major challenge in swarm robotics is to minimize energy and time costs. We focus in this paper on multi-agent foraging algorithms that uses ant-like agents with limited energy. By considering energy consumption, we propose a new Energy aware Cooperative Switching Algorithm for Foraging (EC-SAF) that optimizes the whole system search and transport operations needed to collect resources over time. Unnecessary moves are avoided according to the following two premises: (1) Quick search and optimal paths provided by Stigmergic Multi-Ant Search Area (S-MASA) algorithm; (2) Quick homing provided by using the optimal paths created while searching. Results indicate that EC-SAF is promising in reducing swarm energy consumption compared to an energy-aware version of the c-marking algorithm (Ec-marking).**

I. Introduction

Swarm robotics is concerned with the design of artificial robot swarms based upon the principles of swarm intelligence [1] [2] [3]. Promising solutions are expected by using numerous simple robots [4] [5] [6]. The collection carries out complex tasks based on simple rules, without spending much computational power and much physical energy [7].

Foraging robots are mobile robots that search for objects, and transport them to one or more storage points. It is a benchmark problem used in swarm robotics for several reasons: (1) It integrates several complex sub-tasks such as exploration, navigation, manipulation and transport; (2) It constitutes a canonical problem for the study of robot-robot cooperation; and (3) Many real-world applications are instances of foraging robots (like cleaning, harvesting, searching and rescuing) [8]. Foraging robots perform tasks that consumes energy, and must have a means of obtaining more energy to complete missions successfully. The most common strategies for powering long-lived autonomous robots are: (1) Capture ambient energy directly from the environment, also known as energy scavenging; or (2) Transfer energy from a recharging station [9]. Several options are possible in the last case: (1) Working robots perform their work until their energy falls below a given threshold. At this time, they return to recharging station, to recharge their energy [10] [11]; (2) Working robots can stay at the working site permanently, while special dock robots visit them periodically to provide them with energy [9] [12]; (3) Robots could transfer the energy also between them by comparing their energy's level [8]; and (4) Robots have to decide between search and transport, where transport can be applied to different resources. Even if resources are unknown at the beginning, as robots can be recruited by others, robots have sometimes to choose between carrying resources to the nest, carrying resources to another location in the field or searching for new resources, according to minimize the global energy consumption and maximize the global resources collected (that are equivalent to the accumulated energy) [13].

In this paper, we study the energy expenditure in the Cooperative Switching Algorithm for Foraging (C-SAF) [14] by addressing and analyzing two points:(1) What is the impact of collective exploitation of food provided by recruitment in C-SAF algorithm on energy consumption ? (2) Does the division of search space (by using multiple sinks) in C-SAF improves energy efficiency ?

C-SAF robots are ant-like agents with limited computing and memory capacities. C-SAF provides quick search by using S-MASA algorithm [15] as search strategy, and quick exploitation by recruiting agents. Results are better than the standard reference algorithm and performances are emphasized with cooperation [16]. In this work, we propose to enhance foraging algorithms efficiency by taking energy into account. Individual energy and overall swarm energy are considered during ressources location and exploitation.

The remainder of the paper is organized such as follows: in Section II we present the EC-SAF algorithm, its Finite State Machine (FSM) and a description of different states and we present the Ec-marking algorithm, an enhanced version of the c-marking algorithm [16], with which we compared the obtained results. In Section III, we present the performance indices used, describe the scenarios used for simulations and compare obtained results of the two algorithms. We finish with a conclusion in Section IV and some future works.

II. An Energy-aware Multi-Agent Foraging Algorithm

In this Section, we present the EC-SAF algorithm, the FSM of our foraging agents and a description of different states. We present the Ec-marking algorithm, with which we compare the obtained results.

A. EC-SAF Algorithm

Our foraging agents use a four layered subsumption architecture [17] where each layer implements a particular behavior: *Environment exploration* is the lowest priority layer in this architecture. It consists in exploring the environment, therefore, it includes the states *Choose-Next-Patch* and *Look-for-Food*. *Food exploitation* consists in exploiting food when it is found, it envelops the states *Pick-Food*, *Return-to-Nest*, *Return-and-Color*, *At-Home*, *Climb* and *Remove-Trail*. *Recharging energy* consists of the set of states that allow agents to return home to recharge when their energy falls below a threshold, it includes the states *Return-to-Nest*, *Return-and-Color*, *Remove-Trail* and *At-Home*. *Obstacle avoidance* is the higher priority layer, it implements the obstacle avoidance behavior. Higher priority layers are able to subsume lower levels in order to create viable behavior (see Figure 1 for an illustration of the architecture). The behavior of our foraging agents is enhanced from [14] to deal with energy limitation. It is shown by the state transition diagram in Figure 2 where dotted arrows represent the new transitions used when the current energy of an agent (E_c) falls below the fixed threshold (E_{min}). States are described below and the enhanced algorithm is given by Algorithm 1:

Look-for-Food: If $E_c > E_{min}$ and there exists a food here, agent executes *Pick-Food* state, while if there exists no food it executes *Choose-Next-Patch* state. If its $E_c <= E_{min}$, it turns to *Return-to-Nest* state if there exists a trail, or to *Return-and-Color* state, if there exists no trail.

Choose-Next-Patch: If an obstacle is detected, the agent calls the procedure Avoid_Obstacle(). If no obstacle is there, the agent climbs the brown trail to reach the food location if there exists one, it spreads then the information to its left cell. It lays a limited amount of pheromone $\mathbb{P}$ in current cell, adjusts its heading by executing S-MASA Algorithm [15] and moves one step forward. It turns automatically when finished to *Look-for-Food* state.

Pick-Food: If $E_c > E_{min}$, agent picks a given amount of food and spreads the information to its left cell. However, If $E_c <= E_{min}$ it does not pick food. It executes in the two cases *Return-to-Nest* state, if there exists a trail or *Return-and-Color* state if there exists no trail.

Return-to-Nest: The agent moves to one of colored neighboring cells with the lowest $\mathbb{P}$ value. It remains in this state until home is reached, it turns then to *At-Home* state.

Return-and-Color: The agent moves to one of the four neighboring cells with the lowest $\mathbb{P}$ value and marks its trail with yellow and remains in this state until it reaches the home; it turns then to the *At-Home* state.

At-Home: The agent unloads food if it carries one. If its current energy *(E_c)* is below *(E_{min})*, it recharges its energy to the maximum amount E_{max}. It goes to *Climb* state if there exists one and the amount of food is > 0. If amount of food is $= 0$, it executes *Remove-Trail* state, else it turns to *Look-for-Food* state.

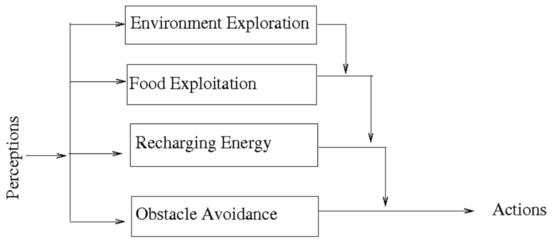


Fig. 1: Subsumption architecture of our foraging agents

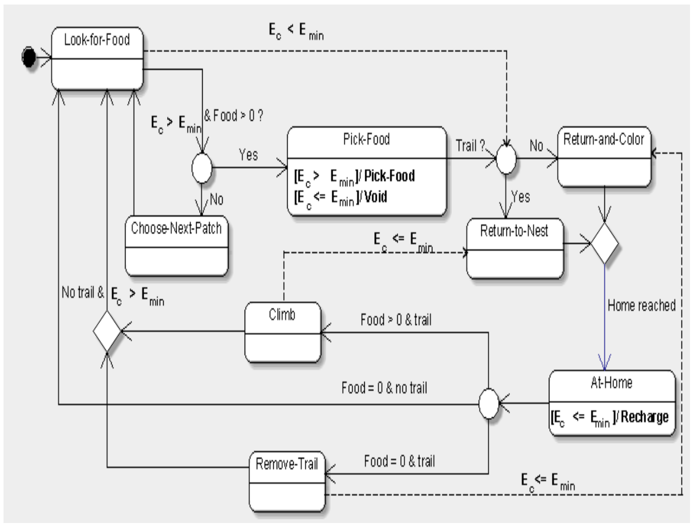


Fig. 2: State transition diagram of a foraging agent. Dotted arrows are the added transitions related to the recharging behavior of agents

Climb: Agent moves to one of its four colored neighbors with a $\mathbb{P}$ value greater than the $\mathbb{P}$ value of the current cell. It remains in this state until no colored cell (yellow trail) exists and its $E_c > E_{min}$, it turns then to the *Look-for-Food* state. If its $E_c <= E_{min}$ and since there exists a trail, it executes the *Return-to-Nest* state in order to return home for recharging its energy.

Remove-Trail: The agent moves to a colored cell with the greatest $\mathbb{P}$ value and resets its color to the default color (black). It remains in this state until no colored cell is found and $E_c > E_{min}$, it turns then to the *Look-for-Food* state. If $E_c <= E_{min}$, the agent returns to home to recharge and executes the *Return-and-Color* state since it already removed the existing trail and to keep track of the last position from where he will continue removing after it recharges its energy.

The modeling of the components of our multi-agent system are detailed below:

- *Environment Model:* an N X N grid world. It contains a set of agents, food, nest and obstacles.
- *Agent Model:* agents are reactive. They have limited processing power and memory, simple sensors (perceive the four neighboring cells), do not know the position of food nor the map of environment and use different kinds of pheromone to communicate.

```
LOOK-FOR-FOOD
if ($E_c <= E_{min}$) then
    if ($\exists$ trail) then goto RETURN-TO-NEST;
    else goto RETURN-AND-COLOR;
else
    if (food $> 0$) then Diffuse($\mathbb{P}$); goto PICK-FOOD;
    else goto CHOOSE-NEXT-PATCH

CHOOSE-NEXT-PATCH
if (obstacle detected) then Avoid_Obstacle();
else
    if (brown $\mathbb{P}$ here) and (brown $\mathbb{P}$ in right cell) then
        Diffuse($\mathbb{P}$); move to food location using brown cells;
    else
        if (brown $\mathbb{P}$ here) and (no brown $\mathbb{P}$ in right cell) then
            Remove brown trail;
        else
            Lay($\mathbb{P}$);Detect_And_Adjust_Heading();
            Update($\mathbb{P}$); Move();
goto LOOK-FOR-FOOD

PICK-FOOD
if ($E_c > E_{min}$) then Pick up a given amount of food;
Diffuse($\mathbb{P}$);
if ($\exists$ trail) then goto RETURN-TO-NEST;
else goto RETURN-AND-COLOR;

RETURN-TO-NEST
while home not reached do
    move to a colored neighboring cell with the lowest $\mathbb{P}$;
goto AT-HOME;

RETURN-AND-COLOR
while home not reached do
    move to a neighboring cell with the lowest $\mathbb{P}$;
    Color that cell to a specific trail color (yellow);
goto AT-HOME;

AT-HOME
unload food;
if ($E_c <= E_{min}$) then Recharge $E_c$ to $E_{max}$;
if ($\exists$ trail and food $> 0$) then goto CLIMB;
else if ( $\exists$ trail and food $= 0$) then goto REMOVE-TRAIL;
else goto LOOK-FOR-FOOD;

CLIMB
while $\exists$ trail do
    if ($E_c <= E_{min}$) then goto RETURN-TO-NEST;
    else Move to neighboring colored cell with the greater
    value of $\mathbb{P}$;
goto LOOK-FOR-FOOD;

REMOVE-TRAIL
while $\exists$ trail do
    if ($E_c <= E_{min}$) then goto RETURN-AND-COLOR;
    else Move to neighboring colored cell with the greater
    value of $\mathbb{P}$;
    Update its color to the default one (black);
goto LOOK-FOR-FOOD;
```

Algorithm 1: EC-SAF Algorithm to deal with energy limitation, where pheromone is noted $\mathbb{P}$.

- *Pheromone Releasing and Evaporation:* three kinds of pheromone are used, Two of them have no diffusion and evaporation properties and they are used either to mark a transport or recharging trails (yellow color) or a recruitment trail (brown color). The third one is used to mark already visited cells. As in the classic ant system, the pheromone evaporates according to a specific rate.

B. Ec-marking Algorithm

The Ec-marking algorithm, which is an enhanced version of the c-marking algorithm [16] to deal with energy limitation, is given by Figure 3. Agents while exploring the environment build simultaneously paths between food and nest which results in building an ascending Artificial Potential Field (APF) incrementally. An Ec-marking agent is always in one of the states depicted by Figure 3, where the enhancements made represented by filled states, dotted transitions and bold guards. This set of states is described below:

SEARCH & CLIMB TRAIL: If food >0 and $E_c > E_{min}$, the agent executes the *LOADING* state. If $E_c <= E_{min}$, it looks for trail, if there exists one it executes *RETURN TO BASE*, else if there exists no trail it executes *RETURN & COLOR TRAIL*, else it moves to food if it is found and if not it moves to a neighboring cell not marked yet.

LOADING: The agent picks a given amount of food and food here is exhausted and there exists a trail, it executes *RETURN & REMOVE TRAIL*, if food is not exhausted and there exists no trail, it executes *RETURN & COLOR TRAIL*, else it executes *RETURN TO BASE*.

RETURN & COLOR TRAIL: The agent moves to one of the four neighboring cells with the lowest APF value and changes its color to a trail color, it remains in this state until it reaches the home; it turns then to the *UNLOADING* state.

RETURN & REMOVE TRAIL: The agent moves to one of the four neighboring cells with the lowest APF value and changes its color to the default one (black), it remains in this state until it reaches the home; it turns then to the *SEARCH & CLIMB TRAIL* state.

RETURN TO BASE: The agent moves to a colored neighboring cell with value minimal to its current value, until it reaches the home; it turns then to the *SEARCH & CLIMB TRAIL* state.

UNLOADING: The agent unloads the food at home. If $E_c <= E_{min}$, it recharges its energy to E_{max} and if its statue is recharging it changes state to the *REMOVE RECHARGING TRAIL*, else it changes to the *SEARCH & CLIMB TRAIL.*

REMOVE RECHARGING TRAIL: The agent moves to a colored neighboring cell with higher APF value and resets its color to the default one, until no colored cell is found it changes then to the *SEARCH & CLIMB TRAIL.*

III. Simulation Results

Three performance indices are used to compare the algorithms (*Total food returned*, *Energy efficiency* and *Total energy*). Through simulations, we compared the four algorithms

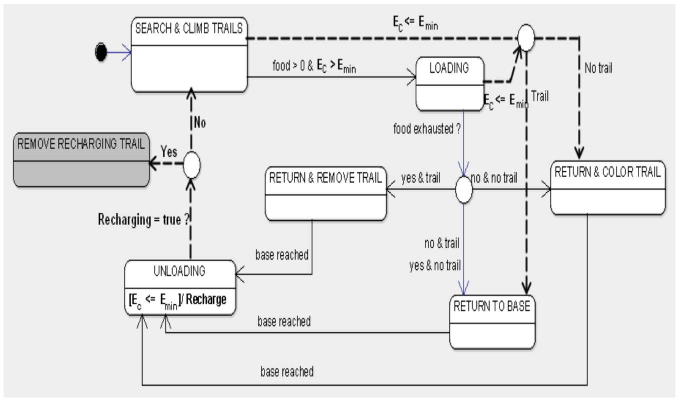


Fig. 3: State transition diagram of an Ec-marking agent, where colored state, dotted transitions and bold guards are related to energy-aware behavior of agents

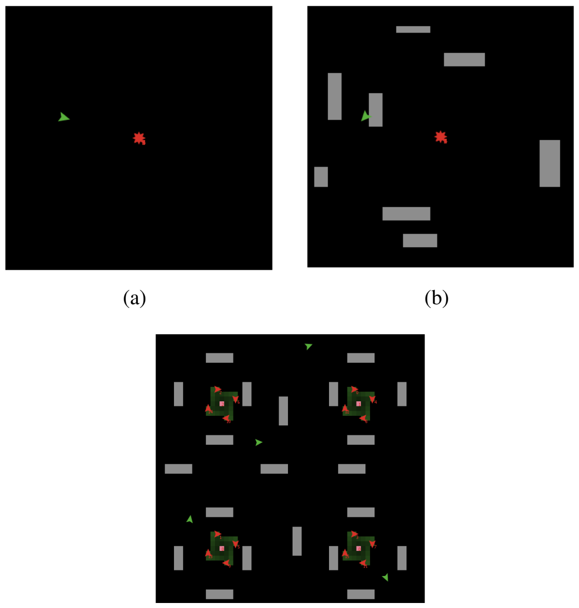
(EC-SAF [Algorithm 1], C-SAF [14], c-marking [16] and Ec-marking [Figure 3]) on the basis of *Total food returned*, to verify if an energy-aware management can improve performances. After that, the two energy-aware algorithms proposed in this paper (EC-SAF and Ec-marking) are compared between each other on the basis of *Energy efficiency* (E_{eff}) and *Total energy* performance indices.

- **Total food returned**: is the total amount of food (in units) returned over some elapsed time.
- **Energy efficiency**: it is the energy spent while foraging one food location. It is calculated according to equation 1.

$$E_{eff} = \frac{TotalEnergyOfConsumedFood}{NumberOfReturnedFood} \tag{1}$$

  Where *Total Energy Of Consumed Food*, is the sum of each agent energy spent in exhausting one food location starting from finding the food until it is exhausted. *Number Of Returned Food* is the quantity of units of food returned;
- **Total energy**: is the total energy spent by all agents to search and exhaust all the food locations.

The energy consumption of an agent at each state is defined on the basis of the power of real equipment (such as motor, sensor and processor) required to achieve that state. It is inspired by the B-swarm model [10]. Agent consumes 1 unit of energy per simulation update for the states that do not need hard work (such as *Climb* and *Return-to-Nest*), while in states that need hard manipulation such as: depositing pheromone (in *Choose-Next-Patch* state), loading food (in *Pick-Food* state), unloading food (in *at-Home* state), pick-up pheromone (in *Remove-Trail* state), and deposit pheromone (in *Return-and-Color* state), the agent consumes 5 units of energy per simulation update. However, for the *Avoid-Obstacle ()*, agent changes its direction only and consumes 3 units of energy per simulation update. For the Ec-marking, the energy consumed is: 5 units of energy per simulation update for states *SEARCH*, *LOADING*, *RETURN & COLOR TRAIL*, *RETURN & REMOVE TRAIL*, *UNLOADING*, *REMOVE RECHARGING TRAIL* and 1 unit of energy per simulation update for states *CLIMB TRAIL*, *RETURN TO BASE* and 3 units of energy per simulation update for the *avoid obstacle()*.

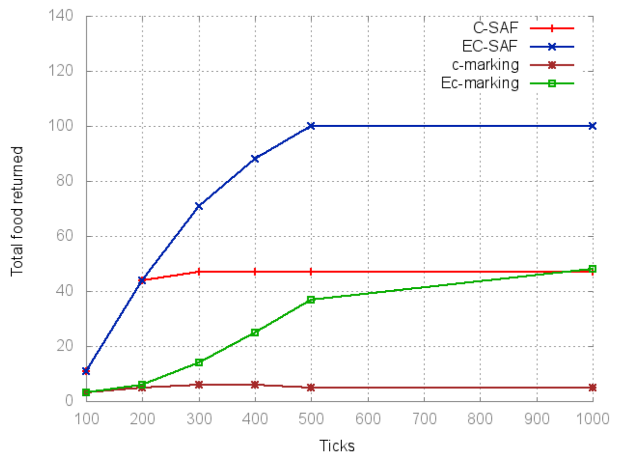


(a) (b)

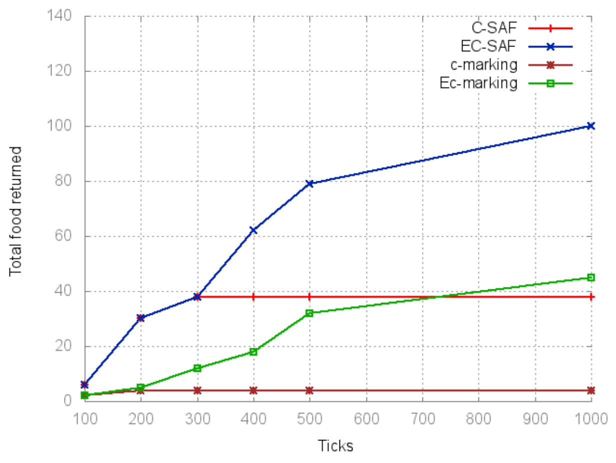
(c)

Fig. 4: World setups used in simulations (a) obstacle-free environment (b) obstacle environment (c) environment with 4 sinks, where red arrows are agents, green arrows are food locations, gray blocks are obstacles and pink squares are sinks

Simulation is based on Netlogo [18]. The simulation results are the average of ten simulations. Four kinds of simulations are reported in this paper. In each simulation several related parameters are to be fixed: agent parameters (number and capacity), world parameters (size, complexity and sinks number) and food parameters (density and concentration) where: *Agent's number* is the number of agents that participate at each simulation, *Agent's capacity* is the amount of food (units) that an agent can transport at each time. *World size* is the dimension of the search space, it is a grid of *N X N cells*, the world is obstacle-free or obstacle represent the *world complexity*, *sinks number* is the number of the home or base station to where agents store food and recharge energy. *Food density* is the number of food locations (sites), distributed randomly in the environment. *Food concentration*, indicates the amount of food that each site contains (we refer to it as unit in the paper). At first stage, we wanted to test if energy-aware can improve efficiency of our C-SAF algorithm or not, therefore, we proposed scenario 1 (see TABLE I), where we calculate the total amount of food returned over some elapsed time. The obtained results with the four algorithms (C-SAF, EC-SAF, c-marking and Ec-marking) are depicted by Figure 5.



(a)



(b)

Fig. 5: Simulation results of scenario 1 in : (a) obstacle-free environment, (b) obstacle environment.

TABLE I: Parameters of scenario 1, scenario 2, scenario 3 and scenario 4

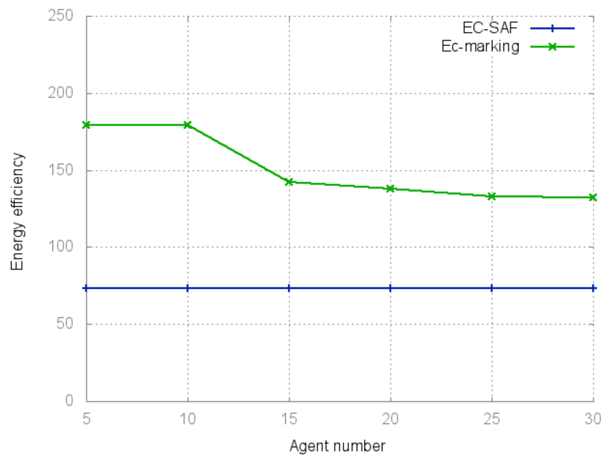
	Parameter	Value
Scenario 1	*Total food returned Analysis*	
	World size	40 X 40 cells
	Number of agents	8
	Food density	2 sites
	Food concentration	50 units
	Agent's capacity	1 unit
	Sinks number	1
Scenario 2	*Energy Efficiency Analysis*	
	World size	40 X 40 cells
	Number of agents	5 – 30
	Food density	1 site
	Food concentration	20 units
	Agent's capacity	1 unit
	Sinks number	1
Scenario 3	*Energy Efficiency Analysis*	
	World size	80 X 80 cells
	Number of agents	48
	Food density	1 site
	Food concentration	20 units
	Agent's capacity	1 unit
	Sinks number	1 – 16
Scenario 4	*Total Energy Analysis*	
	Number of ticks	50 – 300
	World size	40 X 40 cells
	Number of agents	15
	Food density	1 site
	Food concentration	20 units
	Agent's capacity	1 unit
	Sinks number	1 sink

From Figure 5, we observe that in C-SAF and c-marking algorithms, the total food returned increases with the increase in ticks (below 300 ticks) because agents still have energy, when their energy is exhausted, agents die and the total food returned does not increase (over 300 ticks). While the total food returned keep increasing in the energy-aware versions of the algorithms (EC-SAF and Ec-marking), because agents return to recharge when their energy falls below the fixed threshold and resumes their tasks. From this experimental results, we conclude that an energy-aware version are needed to improve performances.

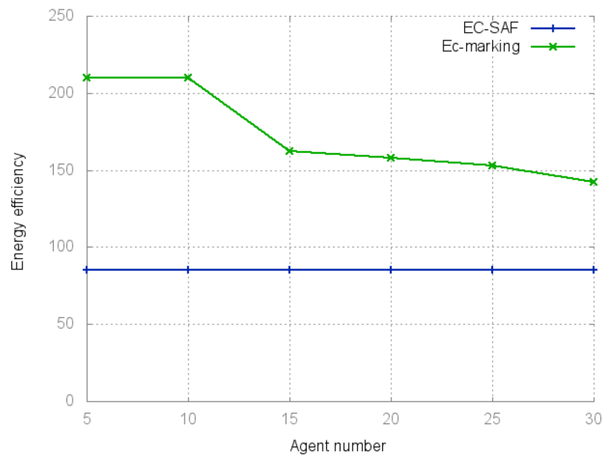
The EC-SAF algorithm is also compared with the Ec-marking one, in order to test if it can improve energy consumption or not. We proposed therefore, Scenario 2, scenario 3 and scenario 4, where the two first ones are used to test the impact of varying agent's number and sink's number (search space division) on energy efficiency. While in scenario 4, we observe the whole energy consumed over some elapsed time, to test whether the algorithm consumes much or less energy when operating (see TABLE I for the description of the three scenarios). The three world setups that are used for simulations including positions of nest, food, obstacles and agents, are reported in Figure 4.

A. Results in Scenario 2

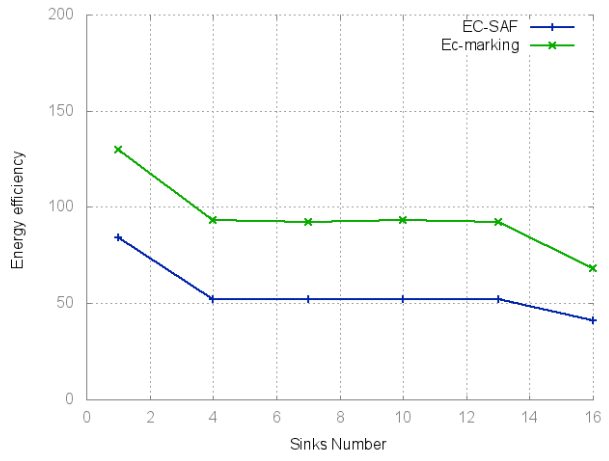
The energy efficiency in EC-SAF does not change when changing the number of agents. In opposite to the number of ticks that is reduced with the increase of agents number, the energy consumed by one agent is the same consumed by multiple agents that participate at food exploitation. However, the energy efficiency in c-marking decreases with each increase in agents number. The energy consumed depends on the length of path that relays the food and the home. In Ec-marking algorithm, the paths are not optimal when number of agents is small thus energy consumption is great, with the increase of agent's number the length of paths will be reduced and the energy consumption decreases. Because of the optimal paths provided by S-MASA algorithm [15], EC-SAF gives better results than the Ec-marking one (see Figure 6(a)). Results in obstacle environment are similar to the ones in obstacle-free environment configuration, with additional steps needed to avoid obstacles (see Figure 6(b)).



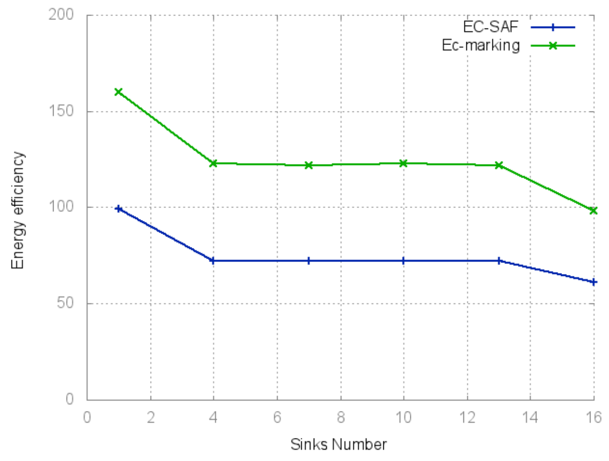
(a)



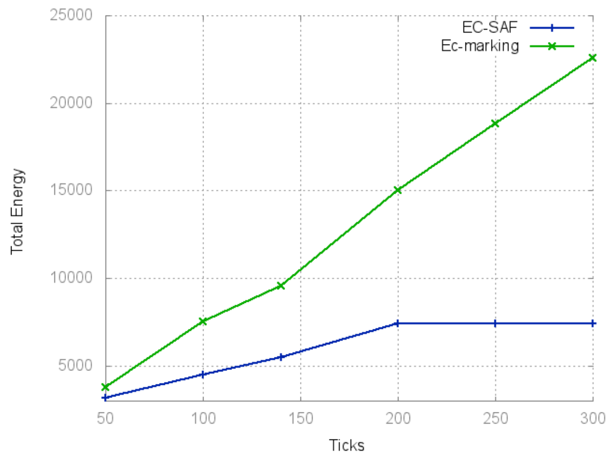
(b)



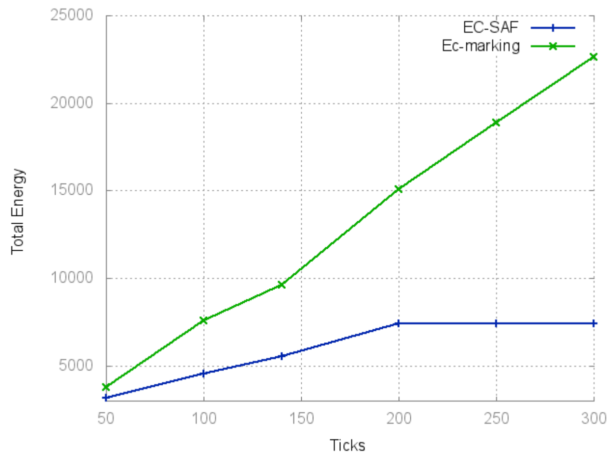
(c)



(d)



(e)



(f)

Fig. 6: Simulation results of: (a), (b) scenario 2 in obstacle-free and obstacle environment. (c), (d) scenario 3 in obstacle-free and obstacle environment. (e), (f) scenario 4 in obstacle-free and obstacle environment.

B. Results in Scenario 3

Using multiple sinks divide the whole search-space into sub-search spaces with smaller size. When increasing the the number of sinks the number of sub-spaces is increased too and the size is reduced more. The path length to food is reduced each time we increase the number of sinks (food takes fixed position in all simulations). In EC-SAF less consumption of energy is reached with 16 sinks, where the size of sub-spaces is the smallest and the path length to the food is the smallest (6 cells). The energy consumption is greater with 4, 7, 10 and 13 sinks (the path length to food is 8 cells). However, it is the greatest with 1 sink, because the search-space size is the greatest and the path length to food is the longest (14 cells). In Ec-marking, the energy consumption is great with 1 sink because the search-space size is the greatest, the path length to food is the longest and paths are not optimal. It is reduced with 16 sinks, since the path length to food is reduced. ES-CAF provide less energy consumption rather than Ec-marking because of the optimal paths induced by the S-MASA algorithm [15]. Figure 6(c) shows the results comparison between the two algorithms. Results in obstacle environment are same as in obstacle-free environment configuration, with additional steps needed to avoid obstacles (see Figure 6(d)).

C. Results in Scenario 4

The total energy increased with increasing the number of ticks as shown in Figure 6(e). It is stable in EC-SAF above 200 tick because the agents reach the boundaries of the search world (the finish time of the foraging is 140 ticks). However, the finish time of foraging in Ec-marking is 300 ticks and until this time the total energy continue to increase. Also in this scenario EC-SAF provides a less consumption of energy in comparison to Ec-marking one. Also in obstacle environment, the total energy in the two algorithms increased with increasing the number of ticks but it is more in Ec-marking algorithm than in EC-SAF algorithm (see Figure 6(f)).

IV. Conclusion

We investigated in this paper the energy efficiency and the total energy consumed of the EC-SAF algorithm as changing the number of agents (to test the benefit of collective foraging), changing the sinks number (to test the benefit of dividing search space) and calculating total energy consumed over ticks (to test the impact of the search strategy used). Simulation results show that energy efficiency in EC-SAF, does not change when changing agents number because the set of agents execute the same states as one agent and thus consume the same energy consumed by one agent. It can be reduced when using multiple sinks, and it depends on the path length between food and home, if the path is reduced with search space division the energy efficiency is even reduced (and vice versa). While the total energy increased with the increase in number of ticks, it stops changing and becomes stable when all food is foraged and the search space boundaries are reached. EC-SAF gives better results than the enhanced c-marking one, because of the optimal paths and the quick search provided by S-MASA algorithm [15].

In the future, we intend to explore other environment configurations and examine other possibilities to reduce the energy consumption in EC-SAF.

References

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[2] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[3] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," *Robotica*, vol. 31, no. 03, pp. 345–359, 2013.

[4] S. Konur, C. Dixon, and M. Fisher, "Analysing robot swarm behaviour via probabilistic model checking," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 199–213, 2012.

[5] A. Saxena, C. Satsangi, and A. Saxena, "Collective collaboration for optimal path formation and goal hunting through swarm robot," in *5th International Conference on Confluence The Next Generation Information Technology Summit (Confluence)*. IEEE, 2014, pp. 309–312.

[6] G. Pini, A. Brutschy, M. Frison, A. Roli, M. Dorigo, and M. Birattari, "Task partitioning in swarms of robots: An adaptive method for strategy selection," *Swarm Intelligence*, vol. 5, no. 3-4, pp. 283–304, 2011.

[7] D. H. Kim, "Self-organization for multi-agent groups," *International Journal of Control Automation and Systems*, vol. 2, pp. 333–342, 2004.

[8] A. F. Winfield, S. Kernbach, and T. Schmickl, "Collective foraging: cleaning, energy harvesting and trophallaxis," *Handbook of Collective Robotics: Fundamentals and Challenges, Pan Stanford Publishing, Singapore*, pp. 257–300, 2011.

[9] A. Couture-Beil and R. T. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 1363–1368.

[10] L. Pitonakova, R. Crowder, and S. Bullock, "Understanding the role of recruitment in collective robot foraging," *MIT Press*, 2014.

[11] J.-H. Lee, C. W. Ahn, and J. An, "A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2013, pp. 489–493.

[12] S. Kernbach and O. Kernbach, "Collective energy homeostasis in a large-scale microrobotic swarm," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1090–1101, 2011.

[13] A. Campo and M. Dorigo, "Efficient multi-foraging in swarm robotics," in *Advances in Artificial Life*. Springer, 2007, pp. 696–705.

[14] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino, "Design and analysis of cooperative and non-cooperative stigmergy-based models for foraging," in *Proceedings of the 19th IEEE International Conference on Computer Supported Cooperative Work in Design*, 2015 (to appear).

[15] O.Zedadra, N. Jouandeau, H. Seridi, and G. Fortino, "S-MASA: A stigmergy based algorithm for multi-target search," in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 2. IEEE, 2014, pp. 1477–1485.

[16] O. Simonin, F. Charpillet, and E. Thierry, "Revisiting wavefront construction with collective agents: an approach to foraging," *Swarm Intelligence*, pp. 113–138, 2014.

[17] R. A. Brooks, "A robust layered control system for a mobile robot," *Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[18] U. Wilensky, "Netlogo. http://ccl.northwestern.edu/netlogo/,," in *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL*, 1999.