Bee behaviour in multi-agent systems: a bee foraging algorithm

Article · January 2005

CITATIONS
33

READS
296

4 authors:

Nyree Lemmens
12 PUBLICATIONS **173** CITATIONS
SEE PROFILE

Steven de Jong
TNO
29 PUBLICATIONS **397** CITATIONS
SEE PROFILE

Karl Tuyls
DeepMind and University of Liverpool
327 PUBLICATIONS **5,176** CITATIONS
SEE PROFILE

Ann Nowe
Vrije Universiteit Brussel
454 PUBLICATIONS **4,598** CITATIONS
SEE PROFILE

Bee behaviour in multi-agent systems:
A bee foraging algorithm

Nyree Lemmens[1], Steven de Jong[2], Karl Tuyls[2], and Ann Nowé[1]

[1] CoMo, Vrije Universiteit Brussel, Belgium
[2] MICC-IKAT, Universiteit Maastricht, Netherlands
{nlemmens,anowe}@vub.ac.be,
{steven.dejong,k.tuyls}@MICC.unimaas.nl

Abstract. In this paper we present a new, non-pheromone-based algorithm inspired by the behaviour of biological bees. The algorithm combines both recruitment and navigation strategies. We investigate whether this new algorithm outperforms pheromone-based algorithms in the task of foraging.
From our experiments, we conclude that (i) the non-pheromone-based algorithm is significantly more efficient when finding and collecting food, i.e., it uses fewer iterations to complete the task; (ii) the non-pheromone-based algorithm is more scalable, i.e., it requires less computation time to complete the task, even though in small worlds, the pheromone-based algorithm is faster on a time-per-iteration measure; and finally, (iii) our current non-pheromone-based algorithm is less adaptive than pheromone-based algorithms.

1 Introduction

In this paper we introduce a new algorithm inspired by the social behaviour of honeybees. The algorithm consists of a recruitment strategy and a navigation strategy. Recruitment strategies are used to communicate previous search experiences to other members of the colony. Navigation strategies are used to navigate in an unknown world. The algorithm does not use pheromones for either navigation or recruitment, as do ant algorithms. In ant algorithms, paths take a certain amount of time to emerge. Due to the nature of bee behaviour we are able to reduce the time for a path to emerge. We apply the algorithm in the domain of foraging, show its effectiveness, and compare it empirically with Ant Colony Optimization (ACO) [1].

Pheromone-based algorithms are inspired by the behaviour of ants. For an overview, we refer to [1]. In summary, ants deposit pheromone on the path they take during travel. Using this trail, they are able to navigate towards their nest or food. Ants employ an indirect recruitment strategy by accumulating pheromone trails. When a trail is strong enough, other ants are attracted to it and will follow this trail towards a destination. This is known as an autocatalitic process. More precisely, the more ants follow a trail, the more that trail becomes attractive for being followed. Short paths will eventually be preferred.

Non-pheromone-based algorithms are inspired by the behaviour of (mainly) bees and do not use pheromones to navigate through unfamiliar worlds. Instead, for navigation, they use a strategy named Path Integration (PI). Bees are able to compute their

present location from their past trajectory continuously and, as a consequence, can return to their starting point by choosing the direct route rather than retracing their outbound trajectory [2, 3]. For recruitment, bees employ a direct strategy by dancing in the nest. Their dance communicates distance and direction towards a destination [4].

Although ant and bee foraging strategies differ considerably, both species solve the foraging problem efficiently. In the field of Computer Science, researchers have become inspired by the behaviour of social insects, since the problems these insects cope with are similar to optimization problems humans wish to solve efficiently, for instance, the Travelling Salesman Problem. Pheromone-based algorithms are already used to address such problems successfully [1]. Non-pheromone-based algorithms are less extensively studied and research into them only started recently. For instance, [5–8] all present bee-inspired algorithms which pose solutions to different types of problems by employing bee recruitment behaviour. In [2] the navigation behaviour of bees is investigated and applied in a robot. However, these algorithms use only one aspect of bee behaviour, i.e., the recruitment behaviour or navigation behaviour respectively. As such, there are still two important open issues. First, recruitment and navigation algorithms are currently only studied separately; a combined algorithm is undiscovered land. Second, since a combined non-pheromone-based algorithm currently does not exist, comparative studies have not yet been performed. We want to investigate whether our non-pheromone-based algorithm poses a better solution to the foraging problem than a pheromone-based algorithm. More precisely, we want to investigate whether paths emerge faster with our non-pheromone-based algorithm. The comparative studies would need to focus on the efficiency, scalability and adaptability of the algorithms.

Our research addresses both issues. First, a new non-pheromone-based algorithm, which implements both bee recruitment and bee navigational strategies, is presented. Second, we have developed a simulation environment, named BeeHave, in which foraging algorithms can be compared directly. Using BeeHave, we are able to compare the non-pheromone-based algorithm with a pheromone-based algorithm. Extensive experiments have been performed with respect to efficiency and scalability. Moreover, we are able to give an indication of the adaptability of our new algorithm.

In this paper, we present an overview of our research [9]. The remainder of this paper is structured as follows. In Section 2, we describe the biological background of bee behaviour. Section 3 describes how to model bee behaviour. Section 4 describes the simulation environment and the experiments. Finally, in Section 5, we present the conclusion and two options for future research.

2 Biological Background

Honeybee foraging behaviour consists of two types of behaviour, i.e., (i) recruitment behaviour and (ii) navigation behaviour.

In order to recruit members of the colony for food sources, honeybees inform their nest mates of the distance and direction of these food sources by means of a waggling dance performed on the vertical combs in the hive [4]. This dance (i.e., the bee language) consists of a series of alternating left-hand and right-hand loops, interspersed by a segment in which the bee waggles her abdomen from side to side. The duration of the

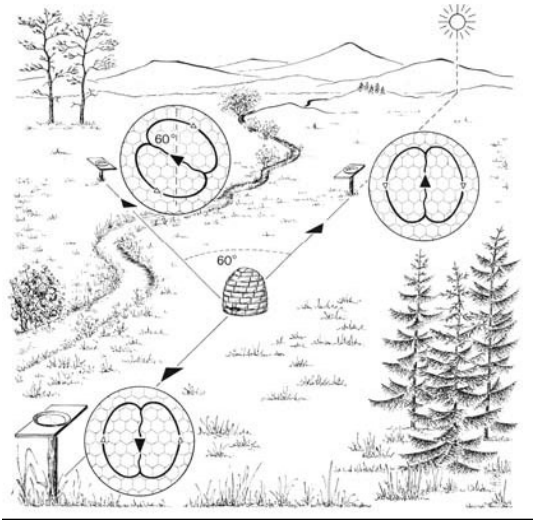


Fig. 1. Distance and direction by waggling dance. Waggling straight up on the vertical comb indicates a food source which is located at an azimuthal angle of 0° while waggling straight down indicates a food source located at an azimuthal angle of 180°. Figure is taken from [13].

waggle phase is a measure of the distance to the food, and the angle between the sun and the axis of this waggle segment on the vertical comb represents the azimuthal angle between the sun and the direction in which the recruit should fly to find the target [4, 10, 11] (see Figure 1). The 'advertisement' for a food source can be adopted by other members of the colony. The decision mechanism for adopting an 'advertised' food-source location by a potential recruit, is not completely known. It is considered that the recruitment amongst bees is always a function of the quality of the food source [12].

Different species of social insects, such as honeybees and desert ants, make use of non-pheromone-based navigation. Non-pheromone-based navigation mainly consists of Path Integration (PI) which is the continuous update of a vector by integrating all angles steered and all distances covered [2]. A PI vector represents the insects knowledge of direction and distance towards its destination. To construct a PI vector, the insect does not use a mathematical vector summation as a human does, but employs a computationally simple approximation [3]. Using this approximation the insect is able to return to its destination directly. More precisely, when the path is unobstructed, the insect solves the problem optimally. However, when the path is obstructed, the insect has to fall back on other strategies such as landmark navigation [14, 15] to solve the problem. Obviously, bees are able to fly and when they encounter an obstacle they can mostly choose to fly over it. However, even if the path is unobstructed, bees tend to use landmark navigation to minimize PI vector errors. The landmarks divide the entire path in segments and each landmark has a PI vector associated with it. In the remainder of this paper, we refer to a home-pointing PI vector as a Home Vector (HV). PI is used in both exploration and exploitation. During exploration insects constantly update their HV. It is however, not

used as an exploration strategy. During exploitation, the insects update both their HV and the PI vector indicating the food source, and use these vectors as a guidance to a destination.

3 Modelling Bee Behaviour

In contrast to existing models [5–8], our new model combines both biological behaviours previously mentioned. First, recruitment behaviour is modelled in analogy with biological bees' dance behavior. Artificial bees share information when they are present in the hive (i.e., the home location of each artificial bee). Whenever an artificial bee is present inside the hive, it looks for any other artificial bee that has previous search experience. When it finds one, it decides whether or not to exploit that previous search experience. Exploiting previous search experience means copying the PI vector (i.e., the directions obtained by the dance). However, the artificial bee can also decide to exploit its own search experience, if available. Biological bees use a (still) unknown decision mechanism to decide whether or not to adopt 'advertised' PI vectors. The decision mechanism is considered to be a function of the quality of the food sources [12]. Our model, however, does not take into account quality assessment, since this was not of direct relevance for our comparison. The decision to adopt a PI vector is based on distance assessment. More precisely, agents prefer to adopt PI vectors that indicate a food-source location on a shorter distance from the hive than their current vector. Second, the navigation behaviour used in the model either exploits previous search experience or lets the artificial bees explore the world randomly. Exploiting previous search experience is guided by the PI vector that artificial bees can obtain by 'following' a dance inside the hive or by using their own previous search experience.

Algorithm 1, which closely resembles the algorithm used for ACO [1], implements both recruitment and navigation behaviour and consists of three functions.[3]

Algorithm 1 Non-pheromone-based algorithm.

1: $ManageBeesActivity()$
2: $CalculateVectors()$
3: $DaemonActions()\{Optional\}$

First, $ManageBeesActivity()$ handles agents' activity based on their internal state. Each agent has six internal states. In each state a specific behaviour is performed. Agent state 'AtHome' indicates that the agent is located at the hive. While in this state, the agent determines to which new state it will go. State changes occur according to Algorithm 2. Agent state 'StayAtHome' also indicates that the agent is located at the hive.

[3] The last function, $DaemonActions()$, can be used to implement centralized actions which cannot be performed by single agents, such as collection of global information which can be used to decide whether it is useful to let an agent dance. In this paper, $DaemonActions()$ is not used.

Algorithm 2 Agent internal-state changes

```
if State is StayAtHome then
    if Vector exists then
        Exploitation
    end if
else if Agent not AtHome then
    if Agent has food then
        CarryingFood
    else if Depending on chance then
        HeadHome, Exploration or Exploitation
    end if
else if Exploit preference AND state is AtHome then
    if Vector exists then
        Exploitation
    else
        Exploration
    end if
else if StayAtHome preference AND state is AtHome then
    if Vector exists then
        Exploitation
    else
        StayAtHome
    end if
else
    Exploration
end if
```

However, while in this state it will remain there unless there is previous search experience available to exploit. In the latter case, the agent will leave the hive to exploit the previous search experience. Agent state 'Exploitation' indicates that the agent is exploiting previous search experience. Previous search experience is represented by a PI vector indicating a food source. The agent determines which cell to move to in order to match the PI vector indicating the food source. Agent state 'Exploration' indicates that the agent is exploring its environment in search for food. Agent state 'HeadHome' indicates that the agent is heading home without carrying any food. The agent reaches home by following its HV. From the moment an agent starts its foraging trip, this HV is continuously calculated for each agent. Agent state 'CarryingFood' indicates that the agent has found food and that it is carrying the food back towards the hive. The agent's return path depends on the same HV as with agent state 'HeadHome'.

Second, $CalculateVectors()$ is used for administrative purposes and calculates the PI vectors for each agent, i.e., the HV and possibly the PI vector indicating the food source. Our model uses a precise PI vector calculation which rules out the directional and distance errors that biological PI is prone to make [3, 15]. It does, however, work in a similar way. The new PI vector is calculated with respect to the old one. Assume that in Figure 2, *A* is the reference point, *B* the starting point of the agent, and *C* the destination of the agent. That would make *a* the travelled distance, *b* the new homing

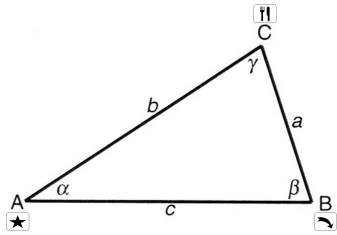


Fig. 2. Triangle ABC. Star icon indicates the hive; Arrow icon indicates the agent's starting point; Food icon indicates the agent's goal.

distance and c the old homing distance. β is the angle turned to get to C and α the angle used for adjusting the old homing angle. In order to calculate the new homing distance, we use the cosine rule and rewrite it to:

$$b = \sqrt{a^2 + c^2 - 2ac \times cos\beta} \tag{1}$$

Using Equation 1 we can now calculate α (the angle used for adjusting the old homing angle), once again by using the cosine rule.

$$\alpha = \arccos\left(\frac{a^2 - b^2 - c^2}{-2bc}\right) \tag{2}$$

Values obtained by Equation 1 and Equation 2 are used to construct the new PI vector. A PI vector thus consists of two values, one indicating the direction and the other indicating the distance. These two values are stored in a variable for each agent. For the sake of clarity, in our model, we calculate exact angles. Biological bees however, approximate these angles.

Bee behaviour's main feature is that it naturally constructs a direct, optimal path between a starting point (i.e., the hive) and a destination (i.e., the food source). One could argue that bee behaviour is a natural way of constructing options in a Markov Decision Process (MDP). Options are courses of action within a MDP whose results are state transitions of extended and variable duration [16]. Such courses of action have proven very useful in speeding up learning and planning, ensuring robustness and allowing the integration of prior knowledge into AI systems [17]. An option is specified by a set of states in which the option can be initiated, an internal policy and a termination condition. If the initiation set and the termination condition are specified, traditional reinforcement learning methods can be used to learn the internal policy of the option. In bee behaviour, the basic actions consist out of moving in different directions over the nodes in a MDP (i.e., the foraging world). The option's policy is represented by the (artificial) bee's PI vector, where the starting state is the hive and the termination state is the food source location.

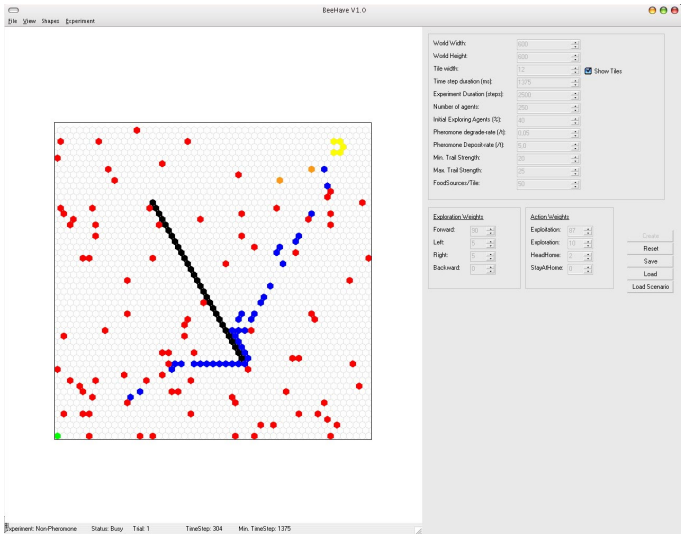


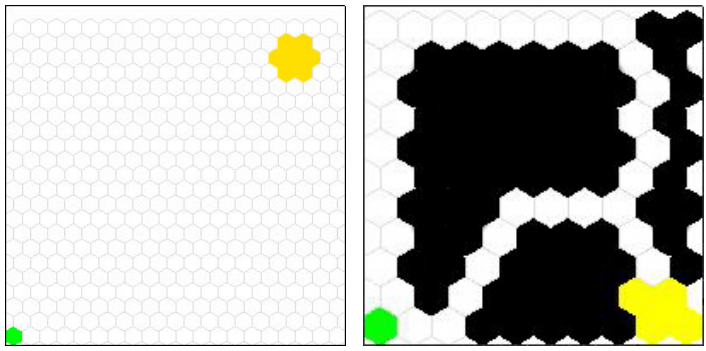
Fig. 3. The BeeHave tool.

4 Simulation Environment and Experiments

To conduct comparative experiments with our bee algorithm and ACO, we created a simulation environment. This environment is called BeeHave and is illustrated in Figure 3.

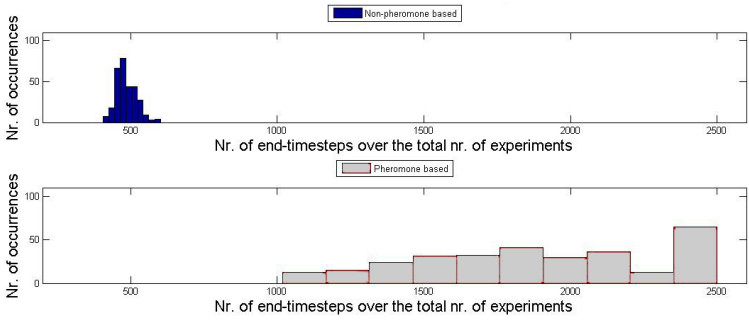
To obtain our data, three experiments have been performed. The experiments were conducted in (i) a small-sized world (i.e., Experiment 1; 110 cells), (ii) a medium-sized world (i.e., Experiment 2; 440 cells), and (iii) a large-sized world (i.e., Experiment 3; 2800 cells). Experiment 1 and 2 each contain five different problem cases (e.g., unobstructed, obstructed, food-source displacement, obstructed with food-source displacement, and multiple foodsources). Experiment 3 only consists of one problem case, i.e., the unobstructed problem case. Each experiment is executed with both the pheromone-based algorithm (i.e., ACO) and the non-pheromone-based algorithm (i.e., our new algorithm). Experiment 1 is executed with 50 and 100 agents, while Experiment 2 is executed with 100 and 250 agents. Chosing higher numbers of agents in either of the two experiments leads to agents flooding in the world, preventing any path from arising. The results of Experiment 1 and 2 are used to obtain our main conclusions. Experiment 3 is used to determine how scalable the algorithms are. The algorithms' scalability is measured with respect to the world size and the number of agents used. In Experiment 3 the number of agents is set to 500.

The comparison is based on efficiency, scalability and adaptability. In Figure 4(a), an example of a medium-sized world is presented. Figure 5 and 6 present the corresponding result figures. The former shows a histogram of the total iterations needed for completing the foraging task at hand. The latter shows a histogram of the average computation time needed per iteration.

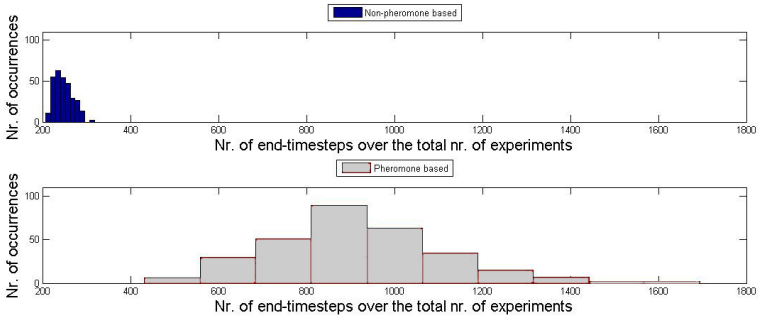


(a) Medium world, basic case (b) Small world, two bridges case

Fig. 4. Simulation worlds.



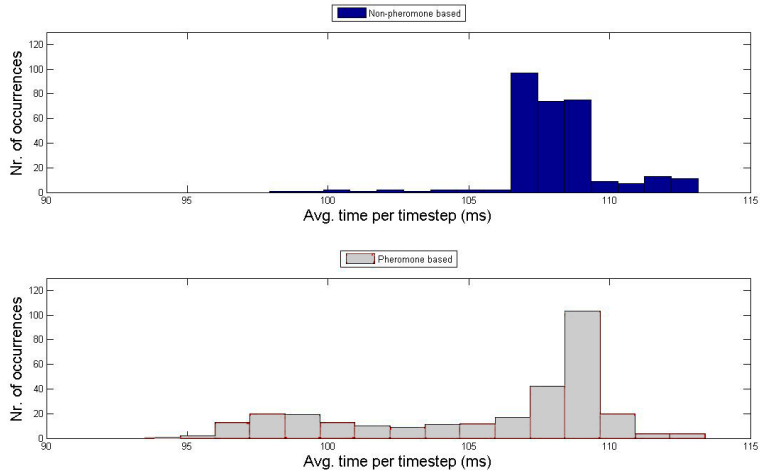
(a) Using 100 agents.



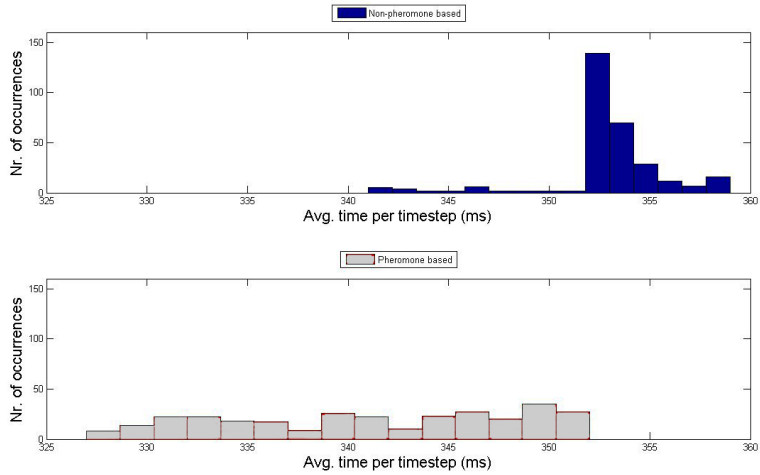
(b) Using 250 agents.

Fig. 5. Histogram of the number of iterations needed in medium-sized, basic-case experiments, with a number of agents as indicated. Black indicates the occurrences for the non-pheromone-based algorithm. Grey indicates the occurrences for the pheromone-based algorithm. Results are obtained after 300 experimental runs.

Considering efficiency, in Figure 5, we can observe that the non-pheromone-based algorithm is more efficient, since it uses significantly fewer iterations to complete the



(a) Using 100 agents.



(b) Using 250 agents.

Fig. 6. Histogram of the average computation time needed per iteration in medium-sized, basic-case experiments, with a number of agents as indicated. Black indicates the occurrences for the non-pheromone-based algorithm. Grey indicates the occurrences for the pheromone-based algorithm. Results are obtained after 300 experimental runs.

task at hand. With an increasing number of agents, the (relative and absolute) efficiency of the algorithm rises. These are typical results found in this research, i.e., they occur in every experiment performed.

Table 1. Performance ratios between pheromone-based algorithm (Pb) and non-pheromone-based algorithm (NPb). I.e., $\frac{Pb}{NPb}$ ratio. Ratios marked with a (*) are influenced by the maximum number of timesteps available. Due to the fact that experiments are terminated after 2500 timesteps, the pheromone-based algorithm was not always able to complete the task set while the non-pheromone-based algorithm always did. The marked ratios values therefore could actually be even higher if we allowed for more timesteps.

World size (number of agents)	Time/Timestep	# Timestep	Total used time
200×200 (50 agents)	1.18	3.09	3.35
200×200 (100 agents)	0.94	3.24	3.04
300×300 (100 agents)	0.98	(*)3.90	(*)3.81
300×300 (250 agents)	0.97	3.69	(3.56
600×600 (500 agents)	0.99	(*)3.31	(*)3.27

In Figure 6(a), we present a histogram of the average computation time needed per iteration in a medium-sized experiment with 100 agents. We observe that the algorithms on average will settle around a computation time of 108ms and 106ms per iteration, respectively. In Figure 6(b) we observe that with 250 agents, the non-pheromone-based algorithm has a mean of 353ms while the pheromone-based algorithm's mean is 341ms and has a wide spread. Even though a statistical test reveals that in both cases, the difference is significant in favour of the pheromone-based algorithm, the total computation time required to complete the task is still much lower for the non-pheromone-based algorithm. Once again, these are typical results; they occur in every small- and medium-sized experiment performed.

Considering scalability, we take into account (i) increasing the number of agents and (ii) increasing the size of the world. With respect to agent scalability, in Table 1, we can observe that when we increase the number of agents and keep the world size constant, ratios decrease (i.e., the pheromone-based algorithm is more scalable with respect to the number of agents). With respect to world scalability, in Table 1, we can observe that when we increase the world size and keep the number of agents constant, ratios increase (i.e., the non-pheromone-based algorithm is more scalable with respect to the size of the world). Overall, the non-pheromone-based algorithm is more scalable than the pheromone-based algorithm since it finishes its tasks much faster.

Considering adaptability, we performed an experiment in which the task set was the Deneubourg Bridge [18]. Figure 4(b) shows the world in which the experiment is performed. In this two-bridges world, the short path is blocked after a certain number of timesteps. The pheromone-based algorithm performs better than the non-pheromone-based algorithm in such a world, see Figure 7. This is because of the fact that by using pheromone trails, the pheromone-based algorithm has more information about the world than the non-pheromone-based algorithm. The latter will try to exploit its most direct path even when this most direct path is blocked. The pheromone-based algorithm however, will eventually move towards the unblocked path due to the accumulated pheromone on this path. To enable the non-pheromone-based algorithm to obtain this environmental information, some extra features have to be added, such as landmark navigation. The results indicate that the pheromone-based algorithm is more adaptive than the current non-pheromone-based algorithm.

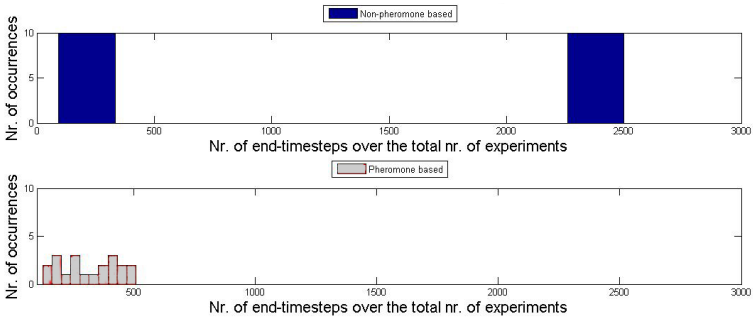


Fig. 7. Histogram of the number of iterations needed in small-sized, two bridges case experiment. Black indicates the occurrences for the non-pheromone-based algorithm. Grey indicates the occurrences for the pheromone-based algorithm. Results are obtained after 20 experiments with 50 agents.

5 Conclusion

Taking into account the results of the experiments in this research, we may conclude that our non-pheromone-based algorithm is significantly more efficient than the pheromone-based algorithm when finding and collecting food.

Concerning scalability, we may conclude that the pheromone-based algorithm is most scalable with respect to the number of agents used, while the non-pheromone-based algorithm is most scalable with respect to the size of the world used. The latter might be a desirable feature; multi-agent systems are mostly applied in large worlds. Furthermore, we may conclude that even in smaller worlds, our non-pheromone-based algorithm requires less total computation time than a pheromone-based algorithm, even if in some cases, the latter requires less computation time per iteration. Besides these benefits, we have to note that our non-pheromone-based algorithm is less adaptive than a pheromone-based algorithm.

Currently, we are extending the recruitment behaviour of our artificial bees. More precisely, we are adding quality assessment for the artificial bee's decision on dance following. Furthermore, we are extending the simulation environment to make it able to construct worlds that are even more dynamic (i.e., moving obstacles and food sources varying in quality). In order to make the algorithm more adaptive, we are investigating whether it is possible to navigate on landmarks. Such landmarks could possibly be created (and decided on) through cooperation between agents or eventually created centrally by the $Deamonactions()$ function (as known in ACO). We are also evaluating to which problems the algorithm could be applied and whether graph-based problems could also make use of path approximation via PI vectors.

We give two more options for future research. First, it might be interesting to construct a hybrid algorithm. By extending the non-pheromone-based algorithm with, for example, pheromone direction markers, we could improve the algorithm's adaptability possibly without decreasing it's efficiency or scalability. Second, by combining PI strategies with potential field searching [19], we could improve local search.

References

1. Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: algorithms, applications, and advances. Technical report, Universit Libre de Bruxelles (2000)
2. Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., Wehner, R.: A mobile robot employing insect strategies for navigation. Robotics and Autonomous Systems **30**(1-2) (2000) 39–64
3. Müller, M., Wehner, R.: Path integration in desert ants, *Cataglyphis Fortis*. Proceedings of the National Academy of Sciences **85**(14) (1988) 5287–5290
4. von Frisch, K.: The dance language and orientation of bees. Harvard University Press, Cambridge, Massachusetts (1967)
5. Lucic, P., Tedorovic, D.: Computing with bees: attacking complex transportation engineering problems. International Journal on Artificial Intelligence Tools **12** (2003) 375–394
6. Nakrani, S., Tovey, C.: On honey bees and dynamic server allocation in internet hosting centers. Adaptive Behaviour **12** (2004) 223–240
7. Chong, C., Low, M.H., Sivakumar, A., Gay, K.: A bee colony optimization algorithm to job shop scheduling. In: In Proceedings of the 2006 Winter Simulation Conference, Monterey, CA USA (2006) 1954–1961
8. Teodorovic, D., Dell' Orco, M.: Bee colony optimization: A cooperative learning approach to complex transportation problems. In: Proceedings of the 16th Mini - EURO Conference and 10th Meeting of EWGT. (2006)
9. Lemmens, N.: To bee or not to bee: A comparative study in swarm intelligence. Master's thesis, Maastricht University, The Netherlands (2006)
10. Michelsen, A., Andersen, B., Storm, J., Kirchner, W., Lindauer, M.: How honeybees perceive communication dances, studied by means of a mechanical model. Behavioral Ecology and Sociobiology **30**(3-4) (1992) 143–150
11. Dyer, F.: When it pays to waggle. Nature **419** (2002) 885–886
12. Camazine, S., Sneyd, J.: A model of collective nectar source by honey bees: selforganization through simple rules. Journal of Theoretical Biology **149** (1991) 547–571
13. Barth, F.: Insects and flowers: The biology of a partnership. Princeton University Press, Princeton, New Jersey (1982)
14. Collett, T.S. Graham, P., Durier, V.: Route learning by insects. Current Opinion in Neurobiology **13**(6) (2003) 718–725
15. Collett, T., Collett, M.: How do insects represent familiar terrain. Journal of Physiology **98** (2004) 259–264
16. Sutton, R., Precup, S., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence **112** (1999) 181–211
17. Iba, G.: A heuristic approach to the discovery of macro-operators. Machine Learning **3** (1989) 285–317
18. Deneubourg, J., Aron, S., Goss, S., Pasteels, J.: The self-organizing exploratory pattern of the argentine ant. Journal of Insect Behaviour **3** (1990) 159–168
19. de Jong, S., Tuyls, K., Sprinkhuizen-Kuyper, I.: Robust and scalable coordination of potential-field driven agents. In: Proceedings of IAWTIC/CIMCA 2006, Sydney. (2006)