Convergence and rate of convergence of a simple ant model

Amine Boumaza, Bruno Scherrer

Convergence and rate of convergence of a simple ant model

Amine Boumaza, Bruno Scherrer

ABSTRACT

We present a simple ant model that solves a discrete foraging problem. We describe simulations and provide a complete convergence analysis: we show that the ant population computes the solution of some optimal control problem and converges in some well defined sense. We discuss the rate of convergence with respect to the number of ants: we give experimental and theoretical arguments that suggest that this convergence rate is superlinear with respect to the number of agents. Furthermore, we explain how this model can be extended in order to solve optimal control problems in general and argue that such an approach can be applied to any problem that involves the computation of the fixed point of a contraction mapping. This allows to design a large class of formally well understood ant like algorithms.

Keywords

Multiagent Systems::Emergent behavior, Multiagent Systems:: Multiagent planning, Agents::Formal models of agency

1. INTRODUCTION

Swarm intelligence [2, 5] was introduced in the 1990s as a novel nature-inspired approach for problem solving. The inspiring source is the behavior of real insects: a population of simple interacting agents, communicating indirectly through an environment, constitutes a *massively distributed* algorithm for solving a given task (e.g. foraging, flocking, labour division, prey capture, ...). About neural networks, another nature-inspired approach for *massively distributed* computing, Kohonen wrote in 1988 [9]:

> *"One of the fundamental tasks of this new "neural networks" science is to demonstrate by mathematical analyses, computer simulations, and even working artificial sensory and control systems that it is possible to implement massively parallel information-processing functions using components the principles of which are not mysterious but already familiar from computer technology, communication science, and control engineering. There is nothing in the "neural network" area which were not known, in principle at least, from constructs already in use or earlier suggested."*

The motivation of this paper is to argue for a similar statement about swarm intelligence.

Classical engineering problem solving and swarm intelligence can be viewed as alternative approaches of the same problem. Then what makes these approaches seem so different ? As Sutton pointed it [14] (when discussing the relations between "modern" Machine Learning and "classical" Intelligent Control), we could

> *"characterize the split as having to do with the familiar dilemma of choosing between obtaining clear, rigorous results on the one hand, and exploring the most interesting, powerful systems on the other."*

Roughly speaking, research on swarm intelligence is often focused on impressive proof-of-concept applications through extensive experimental simulations while classical engineering problem solving relies on theoretical convergence proofs for "toy" problems. The point here is not to judge these approaches (they are clearly complementary) but rather to express our belief that filling the gap between both is a great research challenge.

There are few works in the literature that actually try to fill this gap, in particular about ant-like algorithms. An interesting exception concerns one of the probably best known instance of swarm intelligence: the Ant-colony optimization (ACO) metaheuristic for combinatorial problems. The theoretical works on ACO have recently been reviewed in [7]. The authors present theorems for "convergence in value" (ACO is guaranteed to find an optimal solution in finite time with a probability that can be made arbitrarily close to 1) and "convergence in solution" (provided a properly designed decreasing exploration parameter, the ACO asymptotically converges to an optimal solution almost surely). They also relate ACO to "more standard" optimization techniques such as stochastic gradient descent and cross entropy.

The interested reader should go through this article for further references.

In this article, we come back to the original inspiration of ant algorithms, where a population of simple agents (that may be viewed as mimicking the behavior of real ants) efficiently solve a foraging problem. We will describe some massively distributed ant models that are guaranteed to asymptotically solve the problem of foraging in a certain sense. Moreover, we will provide a rate of convergence analysis with respect to the number of agents.

The remaining of the paper is organized as follows. Section 2 provides a precise description of our ant model and discuss some simulations that were made to measure the convergence and the rate of convergence. Section 3 gives a formal proof that there is convergence in some sense and the theoretical analysis is used to discuss the influence of the model parameters. Section 4 analytically discusses the rate of convergence: it gives formal arguments that explain why one observes a superlinear rate of convergence. Finally section 5 provides a discussion about the scope of our model and its relations with some works in the literature.

2. A SIMPLE ANT MODEL

In this section, we describe a simple ant model, which is aimed at solving a foraging task. We provide simulations that show that the ants end up foraging the food from the food source to the nest and we provide experimental data that show that the efficiency of this distributed model is superlinear in the number of ants.

2.1 Description of the ant model

Consider a set of artifical ants that move on a 2D-grid[1] (the environment) on which they update artificial "pheronomone traces". Each cell s of this grid stores two pheromone traces as two real numbers: $V_1(s)$ and $V_2(s)$. We consider that there are exactly four types of grid cells: one cell is the *nest*, one cell is a *food source*, there are several *bad* cells and all the others are *free* cells.

Each ant can be in two possible states: either it is carrying food (state "has food") or it is carrying nothing (state "has nothing"). Ants move around and their state change according to the following natural rules: when an ant gets to the food source, its state is set to "has food"; When an ant gets to the nest, its state is set to "has nothing" ; if before being set to "has nothing", the ant that got to the nest had food (i.e. it has just brought some unit of food to the nest) then we increment a food counter. This counter will serve as a global measure of performance of the ant population for the foraging task.

We now describe the dynamics of our model. At the beginning:

- The food counter is set to 0.
- The positions of the ants are initialized arbitrarily (e.g.: all ants are initialized at the nest or ants are initialized uniformly at random on the grid, etc...). All ants are set to be in the "carry nothing" state.
- The pheromone values are initialized arbitrarily (e.g.: 0, random, etc...)

[1]As the reader will understand in the following, more complicated graph structures could be used, but we here restrict to the 2D grid case for the sake of simplicity.

At each time step, each ant does two things:

- It updates the two local pheromone traces $V_1(s)$ and $V_2(s)$ of its current cell s using the pheromone values of its four neighbours (we write the set of neighbours $\mathcal{N}(s)$), therefore using only local information. In fact, the update only requires the knowledge of the maximum and average of both pheromone values over the neighbors: $\max^i(\mathcal{N}(s)) \triangleq \max_{s' \in \mathcal{N}(s)} V_i(s')$ and $\text{avg}^i(\mathcal{N}(s)) \triangleq \frac{1}{4}\sum_{s' \in \mathcal{N}(s)} V_i(s')$ with $i \in \{1, 2\}$.

$$V_1(s) \leftarrow \begin{cases} -1 \text{ if } s \text{ is a } \textit{bad} \text{ cell} \\ 1 \text{ if } s \text{ is the } \textit{food source} \text{ cell} \\ \beta\left(\alpha \max^1(\mathcal{N}(s)) + (1-\alpha)\text{avg}^1(\mathcal{N}(s))\right) \\ \qquad\qquad \text{otherwise} \end{cases}$$

$$V_2(s) \leftarrow \begin{cases} -1 \text{ if } s \text{ is a } \textit{bad} \text{ cell} \\ 1 \text{ if } s \text{ is the } \textit{nest} \text{ cell} \\ \beta\left(\alpha \max^2(\mathcal{N}(s)) + (1-\alpha)\text{avg}^2(\mathcal{N}(s))\right) \\ \qquad\qquad \text{otherwise} \end{cases}$$

  where $0 \leq \alpha \leq 1$ and $0 \leq \beta < 1$ with the inspired hindsight condition (see the foregoing analysis) that $\beta < 1$ if $\alpha = 1$.
- It moves to one of its neighboring cell: with probability ϵ ($0 \leq \epsilon \leq 1$) (that we shall call the *exploration* rate) it moves uniformly at random to one of its neighbors. With probability $1 - \epsilon$, it moves to the neighbors that has the highest "pheromone" value V_1 or V_2 depending on its state: the ant uses the pheromone values of V_1 if it is in state "carry nothing" and the values of V_2 if it is in state "carry food".

The β parameter can be seen as some sort of "evaporation parameter" and is typically to be set close to 1. The α parameter, which we shall call the noise parameter for reasons that will be explained later, should typically be set either close to 0 or close to 1. Two simple peculiar instances of our model correspond to the parameter choices ($\alpha = 0$, $\beta = 1$) and ($\alpha = 1$, $0 < \beta < 1$). In the former choice, the general pheromone equation (the "otherwise" case above) reduces to:

$$V_i(s) \leftarrow \text{avg}^i(\mathcal{N}(s)) \tag{1}$$

and in the latter choice to:

$$V_i(s) \leftarrow \beta \max^i(\mathcal{N}(s)) \tag{2}$$

which is a simple linear update. As it will appear clearly in the analysis, the question whether the ant activities (pheromone values updates and moves) are done synchronously or in some sort of asynchronous way is unimportant.

To sum up, to identify precisely an instance of this model (and for the reader to be able to reproduce the experiments we will soon describe), the following data needs to be specified:

- the environment: the set of cells, and their types (nest, food, free, bad),
- the way we initialize the position of the ants and the pheromone values V_1 and V_2 over the environment,
- the noise parameter α, the evaporation parameter β and the exploration parameter ϵ.

The model we have just described is made of very simple reactive agents that communicate indirectly through the environment. In some way, our ant model is simpler than "usual" ant models like the one of the ACO heuristic: in our case, pheromones need not to be evaporated in *every* cell in the environment (this is a heavy computation when running a typical ACO-like algorithm); in our model, some sort of evaporation is done through the β parameter of the local pheromone update. However, our ants use *two* pheromone potentials whereas "usual" ants only use one; but as should become clear to the reader soon, this is to compensate the fact that our ants, as completely *reactive* agents, don't memorize the path to come back to the nest (as is done in ACO).

2.2 Simulations

In this section we illustrate the behavior of our ant model with simulations. As we will see, we will observe some form of convergence. We will begin by describing what form of convergence we have. We will afterwards explain an experimental setting that will allow us to measure the rate of convergence. Finally we will briefly comment the results, as a formal more in-depth analysis follows in the sections 3 and 4.

Let us consider a typical run of the algorithm during which we fixed the parameters as follow: $\epsilon = 0.8$, $\alpha = 0.7$, $\beta = 0.9999$ and m the population size is set to 150. Figure 1 shows snapshots of this run at different time steps. On the figure can be seen the evolution of the ants starting at the nest and evolving through the environment searching for food. Once the food source is found by an ant, more ants move toward the source and a trail starts to form between the nest and the source. As time goes, a good proportion of the ants follow the trail and the dynamics seems to stabilize. Depending on the value of the noise parameter α, we can observe different forms of paths, or even sometimes no path. Figure 2 illustrates the asymptotic distribution of ants for different values of α: there are paths except for the case $\alpha = 0.6$.

Suppose a path emerges between the nest and the food source. We do observe such a path but it would be interesting to measure something objective that reveals this path. This can be done through the counter of food we introduced in the previous subsection. We can draw the curve showing the increase of the accumulated quantity of food brought back to the nest over time. A typical such curve is shown in figure 3. After some time, we observe that the accumulated quantity of food brought back to the nest increases linearly. This means that the foraging behavior of the ants has somehow converged. Section 3 will provide theoretical arguments that characterize precisely this convergence, and we will in particular explain why we don't see a path for certain values of α.

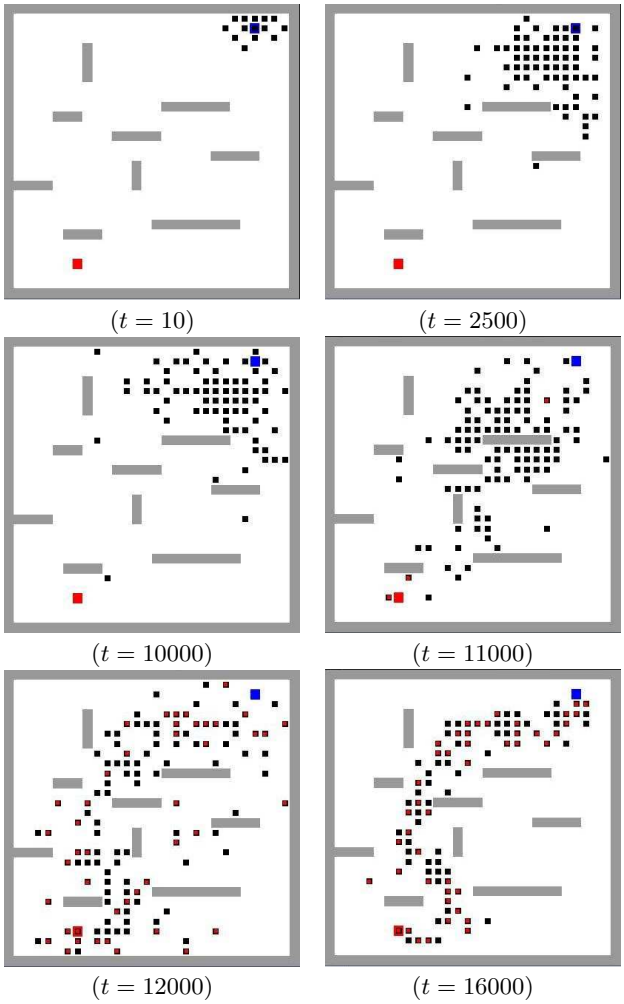

Figure 1: Snapshots of the ant model: red dots represent ants carrying food from the food source (bottom left) back to the nest (top right), and black dots are ants not carrying food.

Such a food quantity curve experimentally shows that there is convergence. It can also be exploited to measure some sort of convergence rate. To do so at certain intervals a line is fitted on the accumulated food quantity data using a linear regression. At some point of the simulation, the parameters of the fitted line stabilize within a certain relative confidence interval. Experiments showed that using a portion of the data (at time t use the data registered from t to $0.5t$) gives robust estimations. In a way that resembles the time constant calculus for electrical components, we define the time of convergence as the intersection of the fitted line with the the x-axis once the linear regression has stabilized (see figure 3). A rate of convergence is then computed as the inverse of the time of convergence.

Using this, one can compute some statistics on the rate of convergence to evaluate its dependence with respect to the number of ants. For a given population size, an experiment consists in running the algorithm for a number of times on a certain environment. In each experiment, twenty runs were

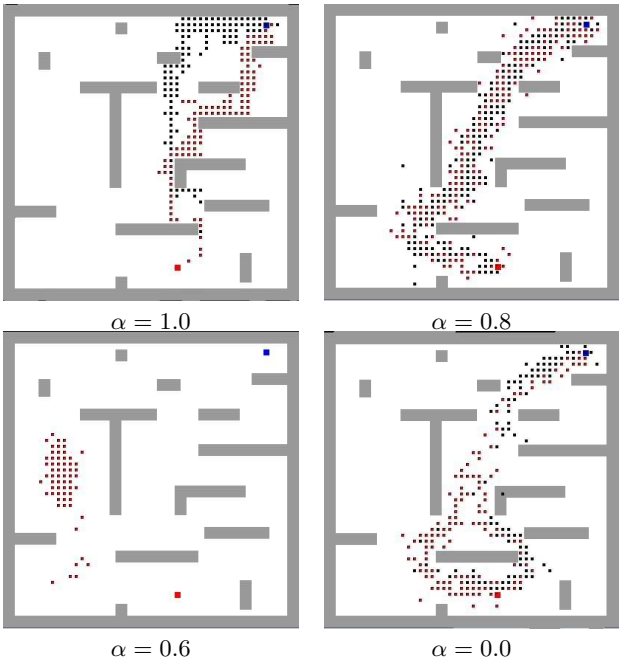

Figure 2: Limit distribution of the ants for various values of α.

carried out with the same parameter values (α, β, ϵ and n). At the beginning, the ants can either be initialized at the nest or uniformly over the environment. If for a given experiment the ants were initialized at random, the same starting positions were used for each run. At each run, the rate of convergence is measured, and at the end of the experiment we compute the mean rate and its standard deviation. On figure 4, we present a typical curve showing the convergence rate with respect to the population size: for these experiments, we took $\alpha = 0.7$, $\beta = 0.9999$, $\epsilon = 0.8$, the relative confidence interval of the linear regression was set to $\pm 10^{-4}$ and the ants were initialized at the nest. Such a curve shows experimentally that when the number of ants increase linearly, the increase in rate of convergence is superlinear. In other words, if the number of ants is doubled the rate increases by more than a factor two. This apparently surprising result will be discussed in section 4: we will provide formal arguments that explain why the rate of convergence can be superlinear.

3. CONVERGENCE ANALYSIS

In this part, we are going to give an analysis of our ant model. We will prove that it converges in some sense. To understand what our ant model does and why we see a path emerge, we will first need to make a detour into the theory of discrete-time stochastic optimal control, and particularly into the Markov Decision Process (MDP) framework. A MDP is a controlled stochastic process satisfying the Markov property with rewards (numerical values) assigned to states. Formally, a MDP is a four-tuple $\langle S, A, T, R \rangle$ where S is the *state space*, A is the *action space*, T is the *transition function* and R is the *reward function*. T is the state-transition probability distribution conditioned by the action. For all

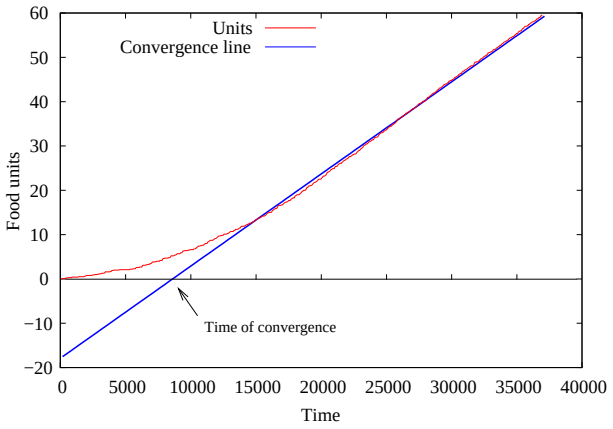

Figure 3: Measuring the rate of convergence: food units correspond to the accumulated amount of food brought back to the nest normalized by the number of ants, and the convergence line corresponds to the linear regression over the data.

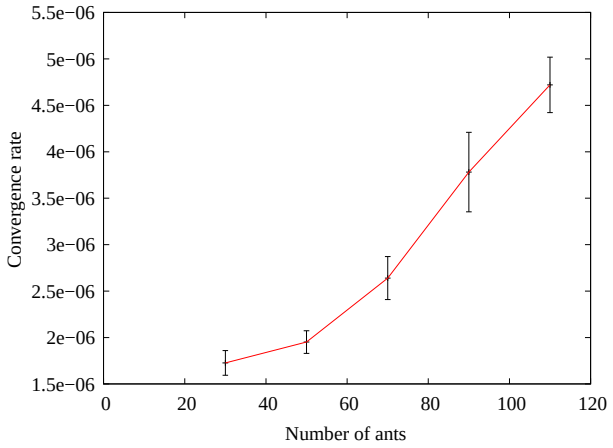

Figure 4: Convergence rate with respect to the number of ants.

states s and s' and actions a,

$$T(s, a, s') \stackrel{def}{=} \Pr(s_{t+1} = s' | s_t = s, a_t = a).$$

$R(s) \in \mathbb{R}$ is the instantaneous reward for being in state S. In the MDP framework, the *optimal control problem* consists in finding a *policy*, that is a mapping $\pi : S \to A$ from states to actions, that maximises the expected long-term amount of rewards, also called value function of policy π:

$$V^{\pi}(s) = \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t . R(s_t) | s_0 = s\right]. \quad (3)$$

We here consider an infinite time horizon; also, future rewards are discounted exponentially with a discount factor $\gamma \in (0, 1)$ (setting $\gamma < 1$ can be seen as a mathematical trick so that the above performance criterion is bounded). Given an MDP model, it is shown [12] that there exists a unique optimal value function V^* which is the fixed point of the following mapping on functions, also called Bellman

operator:

$$\forall W \in \mathbb{R}^S, [B.W](s) = \max_a \left(R(s,a) + \gamma . \sum_{s'} T(s,a,s').W(s') \right) \quad (4)$$

Once an optimal value function V^* is computed, an optimal policy π^* can immediately be derived as follows:

$$\pi^*(s) \in \arg\max_a \left(R(s,a) + \gamma . \sum_{s'} T(s,a,s').V(s') \right). \quad (5)$$

The fundamental reason why the above Bellman operator B has a unique fixed point is related to the fact it is a *contraction mapping* with contraction factor at least γ: i.e. for all pairs of real functions U, U' on S,

$$\|BU - BU'\| \leq \gamma \|U - U'\|$$

where $\|.\|$ is the "max-norm" on S: $\|U\| = \max_s |U(s)|$. A standard approach for solving the optimal control problem is to use a sequential iterative procedure, known as *Value Iteration* [12], which consists in initializing a value function V^0 arbitrarily and iterating the following process:

$$\text{For all states } s \in S, \text{ do: } V^{t+1}(s) \leftarrow [BV^t](s).$$

Because of the contraction mapping property, this sequence is guaranteed to asymptotically converge to the optimal value function V^*, from which we can deduce the optimal controller π^* (cf equation 5). Furthermore, such an iterative sequence, has at least a linear rate of convergence γ:

$$\|V^{t+1} - V^*\| \leq \gamma . \|V^t - V^*\| \leq \gamma^{t+1} . \|V^0 - V^*\| \quad (6)$$

In [3], the authors argue that an asynchronous version of Value Iteration:

$$\text{Pick (randomly) a state } s \in S \text{ and do: } V(s) \leftarrow [BV](s) \quad (7)$$

will also converge to the fixed point V^*, *as long as all states keep on being picked*. In the asynchronous case, one can rewrite a variant of equation 6 as:

$$\|V^{k_t} - V^*\| \leq \gamma \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\|, \quad (8)$$

where $k_0, k_1 \ldots$ is an increasing series such that $k_0 = 0$ and all component of s (all states) are updated at least once between instant k_t and $k_{t+1} - 1$ for all t (see [4] p. 27). Each time all the states have been updated, we know that V approaches V^* at least at a linear rate γ. In fact, once again, the proof of such a result relies on the contraction mapping property. While the aim in [3] was to come up with efficient parallel implementations on real parallel computers, ours is a bit different: we are going to show that the optimal control computation fits in the (virtual) ant paradigm.

Coming back to the ant model we described earlier, we are going to make a clear link between what the ant population computes and some optimal control problems. Indeed, we will show that the pheromone values V_1 and V_2 each corresponds to the value function of a control problem. This is our main first result:

PROPOSITION 1. *Consider the ant model described in section 2.1. If the exploration rate $\epsilon > 0$, then the pheromone value V_1 (resp. V_2) asymptotically converges to the optimal value function of the MDP $M_1 = \langle S, A, T_1, R_1 \rangle$ (resp. $M_2 = \langle S, A, T_1, R_1 \rangle$) with discount factor β where:*

- *S is the set of grid cell plus an extra "end state".*
- *A is the set of the four cardinal moves (north, east, south, west)*
- *The transition T_1 (resp. T_2) is characterized as follows: 1) when, in a state s corresponding to a free cell or the nest cell (resp. to a free cell or the food cell), one chooses one of the four directions $a \in A$, the probability of actually making the move in the direction a is $\alpha + \frac{1-\alpha}{4} = \frac{3\alpha+1}{4}$ while the probability of getting to each of the 3 other neighbors is $\frac{1-\alpha}{4}$. 2) From all the other states, that is from bad cells, the food and the "end state" (resp. from bad cells, the nest and the "end state"), there is, for every action, a probability 1 to get to the "end state", which is an absorbing state.*
- *The reward R_1 (resp. R_2) is defined as follows: For all states s corresponding to a free cell or a nest cell (resp. a free cell or a food cell), the reward is 0. For the state corresponding to the food (resp. the nest), the reward is 1. The reward is -1 for all states corresponding to bad cells and 0 for the "end state".*

The proof of the above result simply consists in checking that the ant model is an asynchronous version of the value iteration algorithm for the corresponding problems: concretely, we need to check that, for all cells, the updates for V_1 and V_2 are identical to the one that would be done by the Bellman operator (equation 7). This is a simple question of rewriting. For instance, for the update of $V_i(s)$ when s is a free cell, we have:

$$V_i(s) \leftarrow \beta \left(\alpha \max_{s' \in \mathcal{N}(s)} V_i(s') + \frac{1-\alpha}{4} \sum_{s' \in \mathcal{N}(s)} V_i(s') \right) \quad (9)$$

$$\Leftrightarrow \quad V_i(s) \leftarrow \beta \max_{s' \in \mathcal{N}(s)} \left(\alpha V_i(s') + \frac{1-\alpha}{4} \sum_{s'' \in \mathcal{N}(s)} V_i(s'') \right)$$

$$\Leftrightarrow \quad V_i(s) \leftarrow \beta \max_{s' \in \mathcal{N}(s)} \left(\left(\alpha + \frac{1-\alpha}{4} \right) V_i(s') + \frac{1-\alpha}{4} \sum_{s'' \in \mathcal{N}(s) \setminus \{s'\}} V_i(s'') \right)$$

$$\Leftrightarrow \quad V_i(s) \leftarrow \beta \max_{a \in A} \left(\sum_{s' \in S} T_i(s,a,s') V_i(s) \right) \quad (10)$$

We let the interested reader check that this equation also holds in the other cases. Finally, the condition $\epsilon > 0$ ensures that all states will keep on being visited and updated, which in turn ensures the convergence of this asynchronous version to the optimal value functions.

Now let us interpret what this means and especially why we observed the emergence of paths between the nest and the food source in the experiments. Solving M_1 (resp. M_2) means finding a policy that will generate trajectories that avoids (on average) the bad cells for which the reward is -1 and try to reach the food cell (resp. the nest cell) for which the reward is 1; because of the discount factor $\beta < 1$, the optimization also tries to minimize the time to reach the

food cell (resp. the nest cell). In other words, M_1 (resp. M_2) are natural formulations of the control problem: "go to the food cell (resp. the nest cell) by avoiding the bad states" assuming that there is some noise in the transition models T_1 and T_2. The level of noise is related to α: calling α a "noise parameter" was thus fully justified.

In each state, the optimal action is the one for which the max is reached in equation 10 above, and it is easy to see that this optimal action is the one that points to the cell for which the max is reached in equation 9: that is the cell with the highest pheromone value. The pheromone values asymptotically converge to the corresponding optimal value functions V_1 and V_2. Therefore, the moves of the ants, which are (recall the ant move description in section 2.1) a mixture of random uniform moves (with the weight ϵ) and the action that climbs up the pheromone value (with a weight $1-\epsilon$), converge to a mixture of random uniform moves and the optimal corresponding moves. The smaller ϵ, the clearer paths can be observed when the pheromones have converged (at the limit $\epsilon = 0$ all the ants follow the optimal policy). However, the smaller ϵ, the longer it will take for the ants to repeatedly visit all the states and make the pheromone values converge. This trade-off is typical of the optimal control theory and is known as the exploration-exploitation dilemma (see for instance [15]).

In the description of the algorithm in section 2.1, we wrote about the parameters α and β that we needed $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ with the condition that $\beta < 1$ if $\alpha = 1$. We can now explain this condition from the optimal control viewpoint: setting the discount factor of a MDP to a value stricly lower than 1 ensures that the infinite time horizon performance (eq. 3) remains bounded and that the Bellman operator is a contraction mapping. As soon as, in the transition models of our specific MDP M_1 and M_2, there is some amount of noise (i.e. $\alpha < 1$), then, for whatever action, there is a probability 1 of ending up into one of the absorbing "end state" (with zero reward) and this suffices to ensure that the performance criterion remains bounded and the Bellman operator is contracting. However, in the purely deterministic case ($\alpha = 1$), the discound factor β needs to be set to a value strictly lower than 1.

We can explain further the influence of the noise parameter α. When α is equal to 1 (and the update is eq. 2), there is no noise in the transition model and the optimal policies exactly match the shortest path (with the respect to the Hamming distance, see figure 2, case $\alpha = 1$) between the nest and the food source. When we decrease α, the level of noise increases and the optimal policies get smoother (they try to stray away from the bad states). At some point, when α goes on decreasing, there is so much noise that the probability of first reaching a bad state when trying to reach the food source (or the nest) is too big for any policy. Consequently the optimal behavior consists in simply staying away from the bad cells and not trying to reach the food source (or the nest): in this case it is better to have a 0 reward than a -1 (which happens when one hits a bad state). This explains why there was no path in figure 2 for $\alpha = 0.6$. When α goes on decreasing and gets close to 0, something apparently strange happens: paths appear again. We can give two explanations of this: 1) when the noise gets so big that the influence of the actions nearly vanish, the optimal controller cannot even prevent from hitting a bad state and, as a kamikaze that would know he's going to die anyway, it again becomes interesting to try to reach the food source (or the nest). 2) At the limit when $\alpha = 0$ (and the update is eq. 1), the pheromone potentials, which satisfy an equation of the type $V_i(s) = \text{avg}^i(\mathcal{N}(s))$ is equal to a discrete harmonic function and it is known that climbing a harmonic function can be used for navigation (see [6] for further details).

4. RATE OF CONVERGENCE ANALYSIS

We saw that our ant model converges by arguing that it is an asynchronous computation of two contraction mapping fixed points: pheromone potentials that results from the ants local updates eventually stabilize towards the optimal value functions of some control problems, which guide the ants between the nest and the food source. The aim of this section is to study the *rate of convergence* of this process with respect to the number of ants. We shall describe some objects and results related to Markov chains on graphs that highlight the experimental observation (made in section 2.2) that this rate of convergence is superlinear: doubling the number of ants may accelerate the process by more than a factor two.

To study the rate of convergence, we go back to equation 8 that describes, in general, the convergence rate of the asynchronous computation of a contraction mapping fixed point. We rewrite it here for the sake of clarity:

$$\|V^{k_t} - V^*\| \leq \gamma \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\|, \qquad (11)$$

Recall that $k_0, k_1 \ldots$ is an increasing series such that $k_0 = 0$ and all component of s (all states) are updated at least once between instant k_t and $k_{t+1} - 1$ for all t. The rate of convergence is thus clearly related to this random variable k_t: the slower k_t grows the faster the process converges. Since we are interested in the rate with respect to the number m of ants, we can make this dependence explicit by writing k_t^m. What we need to study is thus $k_{t+1}^m - k_t^m$.

The good news about $k_{t+1}^m - k_t^m$ is that it can be related to a known object of the probabilty literature. Consider its expectation $E[k_{t+1}^1 - k_t^1]$: it is the average time for one ant to visit all the cells of the environment. More formally, if we see the environment as a graph G (there is one node for each cell and a connection when two cells are neighbors) it is the average time for a Markov chain (that describes the positions of the ant) to visit all the nodes of the graph G, and such a quantity is usually called the *cover time of the Markov chain on the graph* G [1]. Similarly, for any m, $E[k_{t+1}^m - k_t^m]$ is the average time for several parallel Markov chains to visit all the nodes of the graph G and it is known as the *cover time of the parallel Markov chains on the graph* G.

Now comes the bad news about $k_{t+1}^m - k_t^m$: it is *in general* very hard to compute the cover time of a graph for a given Markov chain and a given initial distribution: they are computable in exponential time and it is not known whether it is possible to approximate them in deterministic polynomial time [8]. It is obviously harder to compute the cover time of a graph for several Markov chain. For our specific ant model, it is even harder to compute the cover times

since the Markov chains and the distribution of ants vary over time in a non-trivial way: the transition probabilities depend upon the pheromone potentials which themselves are continuously updated by the Markov chains. There is therefore little hope that one could estimate *very precisely* the rate of convergence of our ant model. A general asymptotic study of the convergence (where one considers that 1) the pheromone values have converged and 2) the ants have reached their stationary distribution) may be pursued and this is a possible subject for future research.

Nonetheless, by studying the literature on the cover time, we were able to find the following interesting result [1]:

PROPOSITION 2. *On a regular n-vertex graph*[2]*, consider m independent balanced random walks*[3]*, each started at a uniform random vertex. Let C^m be the time until every vertex has been hit by some walk*[4]*. Then as the size of the graph $n \to \infty$ and while the number of walks satisfies $m \geq 6 \log n$, we have:*

$$E\left[C^{[m]}\right] \leq \frac{(25 + o(1))n^2 \log^2 n}{m^2}.$$

The interesting point is the $\frac{1}{m^2}$ dependence: this implies that (as $n \to \infty$ and $m \geq 6 \log n$) the cover time is bounded by a function which is inversely quadratic in the number of walks. On simple graphs (the $n - cycle$), [1] explains that the above bound is sharp, so in such a case, the cover time is inversely quadratic in the number of walks.

Using the above discussion, we can "translate" the above proposition into something that is meaningfull for our ant framework:

PROPOSITION 3. *Consider the ant model described in section 2.1 on a toric grid environment with n cells, with an exploration rate $\epsilon = 1$ and initialize the ants uniformly on the environment. When the size of the environment $n \to \infty$ and while the number m of ants satisfies $m \geq 6 \log n$, the time for the pheromone values to reduce their distance to their limit by a factor β is bounded by $\frac{(25+o(1))n^2 \log^2 n}{m^2}$ that is a function that is inversely quadratic in the number m of ants.*

We consider a *toric* environment so that the corresponding graph is regular. We take $\epsilon = 1$ so that the ants follow a balanced random walk. Also notice then that the uniform initialization is the stationary distribution of the random walks, so that the distribution of ants is stationary over time. Our proposition is thus simply a corollary of proposition 2.

The bound is sharp when the environment is a n-cycle graph, that is a long (cycled) corridor: in such a case, the rate of convergence can be quadratic in the number of ants. We there have a superlinear rate of convergence. The experiments we made suggest that superlinearity also happens for other values of ϵ and general environments. Extending the above results to these more general setting constitutes future research.

[2]A regular graph is a graph where all nodes have the same degree, i.e: each node is connected to the same number of nodes.

[3]A balanced random walk on the graph is such that transitions are uniform distributions on neighbors.

[4]$E[C^m]$ is thus the cover time of the graph by these parallel random walks

5. CONCLUSIONS AND DISCUSSION

We have presented a class of ant models that can be related to the framework of optimal control, and proved that they converge in some sense. We have also studied the rate of convergence with respect to the number of ants: we showed, through experimental and analytical arguments, that the rate of convergence is superlinear in the number of ants. At first sight, superlinearity may look like an impressive property. But there is no magic: the fundamental reason why we have such a superlinear rate is due to the fact that small populations of our ants are *especially* inefficient at visiting the whole state space, and therefore making the pheromone potentials converge. In fact, the convergence analysis of contraction mapping that we used clearly suggests (compare equations 6 and 8) that the fastest method for computing the optimal value function is the synchronous version: at each step, the value is approaching its limit with the linear rate γ. The relative slowness of the asynchronous version has to be understood as the price for decentralization.

The ant model we have presented in this paper is very flexible. It can incorporate many variations. For instance, one could consider that there are several food sources, each containing a finite quantity of food, which decreases over time as ants come and go. This quantity could be incorporated in the corresponding optimal control model, through the reward function on the food states. The reward would then evolve over time, and the pheromones, which are continously being updated, would keep on following the corresponding "moving" optimal value function. One could also use different parameters for the back and forth trips of the ants: this could model different strategies depending on the fact that the carries food or not. Finally, if the environment is static, one could make the exploration rate ϵ slowly tend to 0, so that the ants eventually converge to the deterministic optimal policies. Studying good ways for "freezing" the ant behavior through the exploration parameter ϵ constitutes future research.

It is in fact easy to see that one could construct similar ant algorithms for almost any optimal control problem. As long as the problem is formulated as a MDP, we know that it can be solved by an asynchronous version of Value Iteration, and building the corresponding ant system is immediate: we just need to have ants move around the state space and do the Bellman update everywhere they go. If we care about the constraints that "ants make their decision only using local information" then this approach will work as long as the transition in the MDP are also *local* in the state space. We can even go further: All our analysis (the convergence and the rate of convergence) relies on the "contraction mapping" property. This suggests that any problem that involves the computation of a contraction mappings (Contraction mappings can for instance appear in zero crossing problems, constrained and unconstrained optimization [10]) on a finite space has a natural (superlinear) ant-like solution: ants

move around on this finite space and do local contraction updates.

To conclude, let us mention two pieces of work that present algorithms that are strongly related to ours. In [11] the authors also tackle foraging problem with multiple pheromone landscapes by exploiting the optimal control theory (in fact they refer to reinforcement learning [15] which is also formulated through MDPs). In [13], the author presents an algorithm that compute shortest path using an asynchronous implementation of the Bellman-Ford algorithm: the local update is of the form $U(x) \leftarrow 1 + \min_{x' \in \mathcal{N}(x)} U(x')$. Modulo the variable change $U \leftrightarrow \frac{\log(V)}{\log(\beta)}$ this is equivalent to our pheromone update with $\alpha = 1$, that is to equation 2. There are significant differences between these work and ours: 1) they focus only on the foraging problem whereas we have just argued that in principle, any optimal control problem (and any computation of contraction mapping) could be tackled by ant-like algorithms. 2) They do not provide any convergence proof but only extensive experimental data. 3) Last but not least, they say nothing about the rate of convergence with respect the number of ants. In other words, our present work can be considered as a theoretical deepening of [11, 13].

6. REFERENCES

[1] D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. Monograph in preparation, 1996.

[2] G. Beni and J. Wang. Swarm intelligence. In R. press., editor, *Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, 1989.

[3] D. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice hall, 1989.

[4] D. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[6] C. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotic and Systems*, 10(7):931–946, October 1993.

[7] M. Dorigo and C. Blum. Ant colony optimization theory: a survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005.

[8] U. Feige and Y. Rabinovich. Deterministic approximation of the cover time. *Random Struct. Algorithms*, 23(1):1–22, 2003.

[9] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, 1988.

[10] D. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, New York, 1989.

[11] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[12] M. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.

[13] O. Simonin. Construction of numerical potential fields with reactive agents. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

[14] R. Sutton. Artificial intelligence as a control problem: Comments on the relationship between machine learning and intelligent control. In *IEEE International Symposium on Intelligent Control*, pages 500–507, 1988.

[15] R. Sutton and A. Barto. *Reinforcement Learning, An introduction*. BradFord Book. The MIT Press, 1998.