Task Allocation in Ant Colonies

Alejandro Cornejo[1], Anna Dornhaus[2], Nancy Lynch[3], and Radhika Nagpal[1]

[1] Harvard University
School of Engineering and Applied Sciences
[2] University of Arizona
Ecology and Evolutionary Biology
[3] Massachusetts Institute of Technology
CSAIL

Abstract. In this paper we propose a mathematical model for studying the phenomenon of division of labor in ant colonies. Inside this model we investigate how simple task allocation mechanisms can be used to achieve an optimal division of labor.
We believe the proposed model captures the essential biological features of division of labor in ant colonies and is general enough to study a variety of different task allocation mechanisms. Within this model we propose a distributed randomized algorithm for task allocation that imposes only minimal requirements on the ants; it uses a constant amount of memory and relies solely on a primitive binary feedback function to sense the current labor allocation. We show that with high probability the proposed algorithm converges to a near-optimal division of labor in time which is proportional to the logarithm of the colony size.

1 Introduction

Task allocation in ant colonies is the process used by individual ant workers in a colony to select which task to perform in order to ensure the survival of the colony. Depending on the species the tasks typically include things like collecting food, feeding and caring for the offspring and defending the nest against intruders or parasites. The number of individuals allocated to each task varies over time in response to changes in the demand for different tasks.

From a biology perspective, there is an extensive body of empirical work studying the phenomenon of division of labor in ant colonies across different ant species [1–5]. Biologists have also proposed a number of individual behaviors that could produce the collective division of labor [1, 6–9]. It has been suggested that ant workers might select tasks based on different features, including their age [10, 11], body size [12, 13], genetic background [14], position in the nest [13, 15, 16], nutrition [17], in response to signals from other ants [10, 18], or by comparing internal response thresholds to the need to perform a particular task [19–21].

Most of these works were born out of a desire to understand what are the algorithms that may be used by ants, and not to compare the performance of different algorithms. In addition, although in biology there is much work on optimization, there is not a tradition of explicitly separating the system model, the problem statement, and the algorithm being considered. This has made it difficult to generalize the results across different insect species. Most of the modeling studies do not attempt to compare how well different strategies perform, and obtaining any empirical observations for this purpose would be extremely hard (if not impossible). As a result, despite the wealth of empirical results, the trade-offs between different division of labor strategies remain poorly understood.

Our contributions are twofold. First, we describe a very general mathematical formulation for the problem of task allocation in ant colonies. Second, we propose a distributed algorithm that solves the task allocation problem efficiently and imposing only minimal assumptions on the capabilities of the individual ants. The algorithm we propose is modeled after a common strategy thought to be used by ants.

The model for task allocation proposed in this paper is meant to provide a rigorous theoretical framework under which new and existing mechanisms for division of labor can be formulated, studied and compared. In particular it allows modeling ant colonies consisting of workers that have different characteristics and formulate strategies that have different notions about of what is an optimal task allocation.

The randomized distributed algorithm for task allocation we propose requires constant memory with respect to the colony size, and assumes that ants can only sense the need for a particular task through a primitive binary feedback function. In more detail, we assume that each ant can use the information available in the environment (including sensing pheromones or other signaling methods between ants) to determine whether a particular task requires more workers or not. We show that using only this information and a constant amount of memory, ants can converge to a near-optimal task allocation with high probability, in a number of steps which is logarithmic in the size of the colony.

1.1 Related Work

As we argued before, from a biology standpoint there is a large body of work studying division of labor in ant colonies [1–5]. However existing models do not address the question of how well task allocation performs, and it is difficult to quantitatively compare the different proposed strategies.

We are not the first to leverage the tools of distributed computation to understand social insects and model biological processes. In particular the problem of ants foraging for food was recently studied in the distributed computing community [22–25]. In more detail, by modeling the environment as an infinite two-dimensional grid, upper [22] and lower [23] bounds were provided for the problem of ants collaboratively foraging for food on the environment without using communication. These bounds were recently generalized and improved in [25]

In a different vein, the parallels between the process of sensory organ precursor selection in the fruit fly and the problem of finding a maximal independent set have been used to design novel bio-inspired algorithms for fundamental problems in distributed computation [26].

The problem of task allocation is an important and well-studied topic in computer science, particularly in the distributed computing community. One of the oldest incarnations, is that of multiprocessor scheduling [27, 28], a recent survey is available in [29]. Another related problem is that of the centralized k-server problem in which a collection of requests must be assigned to servers [30], distributed versions of this problem have also been considered [31].

The general formulation of task allocation considers some number p of processors that must cooperatively perform n tasks. The existing literature covers a great number of variants of this problem, processors and tasks might be identical or different; tasks might arrive online or may be known to the processors in advance; all processors may not be able to perform all tasks, or tasks may have to be performed in a certain order; processors may be susceptible to fail-stop (or other) failures; processors may communicate through shared memory, message passing, or may not rely on communication; etc. A recent and thorough review on the problem of distributed task allocation is available in [32].

There are several subtle differences between existing distributed task allocation algorithms and the task allocation algorithm for ant colonies that we consider. Namely the algorithm we consider only uses constant memory, does not require that ants to be capable of sensing the demand of each task or to estimate global parameters (like the size of the colony). A more fundamental difference between the problem statements is that the number of tasks we consider is a small constant with respect to the number of ants, and we are concerned with the allocation of proportions of workers to different tasks and not the allocation of individual workers to specific tasks.

2 System model

We start by describing a mathematical model for the task allocation problem in ant colonies. For the sake of simplicity, throughout our discussion we use the term *energy* loosely without specifying the unit; however, any energy unit can be used (e.g., Joule) as long as it is used consistently throughout the definitions.

Task allocation problem

The task allocation problem concerns a set A of ants, and a set T of tasks. The size of an ant colony $|A|$ depends on the species and varies slowly through the lifetime of the colony, but can be considered fixed during short time intervals. Depending on the ant species typical colony sizes range from 10 to millions; in some species, each colony goes through this entire range through its lifetime. The number of tasks $|T|$ and their typical energy requirements also depend on the ant species considered. However the number of tasks performed by a particular species is constant and is not a function of the size of the colony.

For a task $\tau \in T$, a time $t \in \mathbb{R}_{\geq 0}$ and an ant $a \in A$ we define the following quantities.

– $d(\tau, t)$ is the energy demand for task τ at time t. This "demand" reflects how much work is currently needed in this task, and can be determined by environmental conditions, the colony size, the fraction of brood (ant offspring) vs workers, etc; although we do not attempt to uncover the relationship between the demand and these parameters.
– $e(\tau, a, t)$ is the energy that ant a can supply to task τ at time t when it is engaged in this task. This captures the effectiveness of an ant at a specific task, and may be determined by its morphological characteristics, as well its previous experience.

Task assignment

A task assignment is a function y that assigns at each time t and for each ant $a \in A$ either one task $\tau \in T$ or no task. Formally $y(a, t) \in T \cup \{\perp\}$.

Given an assignment y we define $Y(\tau, t)$ as the set of ants assigned to task $\tau \in T$, and $I(t)$ is the set of ants that are not assigned to any task at time t. Formally that is:

$$\begin{aligned} Y(\tau, t) &= \{a \in A : y(a, t) = \tau\} \\ I(t) &= \{a \in A : y(a, t) = \perp\} \end{aligned}$$

Thus by definition at any time t we have that $A = I(t) \cup \bigcup_{\tau \in T} Y(\tau, t)$. We will say an ant is *idle* at time t if it belongs to the set $I(t)$[4].

For succinctness we define the energy supply and the energy need at each task (both are defined in terms of a task assignment).

– $w(\tau, t) = \sum_{a \in Y(\tau,t)} e(\tau, a, t)$ is the energy supplied to task τ at time t, and it is the sum of the energy supplied by the individual ants assigned to task τ at time t.
– $q(\tau, t) = d(\tau, t) - w(\tau, t)$, if negative represents a surplus of energy at task τ, if positive represents a deficit of energy at task τ, and if zero then the task τ is in equilibrium.

A *satisfying* task assignment is one where no task has an energy deficit, formally at time t a *satisfying* task assignment is one where $q(\tau, t) \leq 0$ for all $\tau \in T$. We say a task allocation problem is satisfiable if it has at least one satisfying task assignment.

We are mostly interested in satisfiable task allocation problems where the energy available at the colony far exceeds the energy demands of the tasks. This is likely consistent with what has been observed in real ant colonies [4], where even during periods where tasks have a very high demand (such as nest

[4] We believe this formulation is superior to the alternative of considering an additional "idle task", since task allocation algorithms will need to deal with this task differently, and it would unnecessarily complicate the definition of the demand and effectiveness functions to account for the "idle task".

migration) an important fraction of the ants remain idle where the rest perform the tasks necessary for the survival of the colony.

Ideally, we would like to find task assignments that achieve equilibrium for every task (i.e., where the energy demand equals the energy supply). However due to rounding issues this is not always possible, even when restricted to satisfiable task allocation problems. For instance, consider the case where the energy demand for each task is an odd number and the energy that can be exerted by each ant on any task is an even number. In this case, regardless of the number of ants assigned to each task, no task can be at equilibrium.

For this reason, we instead seek task assignments that minimize the squared difference between the energy demands and the energy supplied. Formally, an *optimal* task assignment is one that minimizes $\sum_{\tau \in T} q(\tau, t)^2$. Clearly a task assignment where all tasks are at equilibrium is optimal, but the opposite need not be true.

3 Restricted system model

The model described in Section 2 was intentionally defined to be as broad as possible. This decision was made with the purpose of allowing others to study different kinds of task allocation algorithms and ant behaviors.

The following sections refine this broad model by imposing some additional restrictions. The algorithm we describe in Section 4 considers this restricted version.

Complexity of individual variation. Variation of the individual ant workers in the colony is captured by the parameter $e(\tau, a, t)$. This parameter can model complex effects such as a particular ant being worst at a task because of its physical characteristics, or an ant getting better at a particular task through experience. Unfortunately capturing even the simplest forms of individual variations in the colony using this parameter quickly results in an intractable task allocation formulation.

For instance, consider the case where, for each ant $a \in A$, the parameter $e(\tau, a, t)$ is the same for every task $\tau \in T$ and for every time $t \in \mathbb{R}_{\geq 0}$. In other words, the effectiveness of an ant does not depend on the task it performs and does not vary through time, but different ants may have different levels of effectiveness. This is the simplest form for the parameter $e(\tau, a, t)$ that still allows each individual in the colony to have a different level effectiveness.

In this case, even if we assume a centralized full-information setting, the problem of finding an optimal assignment of ants to tasks can be shown to be NP-complete. This remains true even if there are only wo tasks with equal demand since the set partition problem, known to be NP-complete [33], can be reduced to the task allocation problem. In general this problem is also hard to approximate and thus finding near-optimal solutions remains NP-hard [34]

The problem can be made tractable by placing additional restrictions, for instance by limiting the possible values of the demand for each task or the restricting different levels of effectiveness of ants. Regardless, real ant colonies may not converge on optimal or even near-optimal solutions, as long as the solutions they have arrived at through the evolutionary process allow colonies to survive and do better than their competitors.

For the above reasons, and in an effort to consider algorithms that more closely resemble those used by real ant colonies, in this paper we restrict our attention to task allocation problems without individual variation. There is empirical biological evidence that supports this decision: at least in some ant species [3], variation in task-specific efficiency among workers is not utilized by colonies. In other words, worker allocation to tasks in some cases is unrelated to their ability to perform them particularly well.

Concretely in the rest of the paper we assume that $e(\tau, a, t)$ is known and constant for all $\tau \in T, a \in A$ and $t \in \mathbb{R}^+$. Therefore for simplicity throughout the rest of the paper we will assume that both the energy demand $d(\tau, t)$ and energy supply $w(\tau, t)$ are measured in ant units.

Synchronous task allocation. For the purposes of this paper we will assume that time proceeds in synchronous lock-step rounds $i = 1, 2, \ldots$. A task allocation algorithm is a program that runs locally at each ant to produce a task assignment. During each round $i \in \mathbb{R}$, every ant $a \in A$ works at task $y(a, i)$. Before transitioning to the next round each ant may communicate with other ants, sense different parameters of the environment and perform some local computation to decide on what task to work on at the next round. The next section describes in detail exactly which environmental parameters are capable of sensing and/or communicating.

4 Allocation through binary feedback

In this section we propose and analyze a randomized distributed algorithm for task allocation that requires only a very primitive binary feedback function. In particular the algorithm we describe does not require that ants know the size of the colony $|A|$, the energy demand $d(\tau, i)$ of a task $\tau \in T$, or the energy supplied $w(\tau, i)$ to a task $\tau \in T$.

Instead, we assume that through local communication and by sensing their environment and their own state, each ant can determine for each round i and for each task $\tau \in T$ whether τ has too few or too many ants assigned to it. In other words, ants can determine only whether a task has a deficit or surplus of energy, but they are not able to quantify it. Specifically we assume that at each round i and for each task $\tau \in T$ ants can sense only a binary feedback function $f(\tau, i)$ where:

$$f(\tau, i) = \begin{cases} +1 & q(\tau, i) \geq 0, \\ -1 & q(\tau, i) < 0. \end{cases}$$

Recall that $q(\tau, i) = d(\tau, i) - w(\tau, i)$ is the difference between the energy demand for task τ at round i and the energy supplied for task τ at round i. Therefore the feedback function f is positive when the demand is greater than or equal to the energy supply, and is negative otherwise. Alternative feedback functions that do not distinguish sharply between having a surplus or deficit of energy are discussed in Section 5.

This restricted binary feedback model prevents the colony from trying to reach a perfect allocation in expectation in a single step, and thus the colony must rely on a progressive refinement strategy. Moreover, this binary feedback function does not provide enough information for the colony to determine if the demand is matched exactly by the energy supplied to that task, and having oscillations of the energy supply around the demand is inevitable.

4.1 Algorithm description

This subsection describes a randomized distributed algorithm that relies only on the aforementioned binary feedback function to converge to a near-optimal task allocation.

Ants executing the algorithm can be in one of the five states Resting, FirstReserve, SecondReserve, TempWorker and CoreWorker, they maintain a task $currentTask \in T \cup \{\perp\}$ and a table ϱ of potentials for each task $\tau \in T$. Initially ants start in the Resting state with $currentTask = \perp$ and a potential of zero for every task $\forall \tau \in T, \varrho[\tau] = 0$. The paragraphs below describe each state in detail as well as the role of the potential table ϱ and the task $currentTask$.

Ants in the Resting state are idle and use the potential table ϱ to choose a *currentTask* and to determine when to start working (i.e., transition to a working state). The potential for every task is a two bit value $\{0, 1, 2, 3\}$ which is updated based on the output of the binary feedback function; specifically tasks with a deficit get their potential increased, and tasks with a surplus get a potential of zero. The *candidateList* is defined as those tasks potential of 3, and with constant probability ants in the Resting state will choose a task from the *candidateList* uniformly at random and transition to the TempWorker state. This is in the same spirit of the response-threshold strategies proposed by biologists [20].

Ants in the TempWorker state and CoreWorker state work on the task specified by *currentTask* (ants in all other states are idle). Specifically, ants in the TempWorker state transition to the FirstReserve state if there is a surplus of energy in *currentTask*, and otherwise transition to the CoreWorker state. Ants in the CoreWorker state transition to the TempWorker state if there is a surplus of energy in *currentTask*, and otherwise remain in the CoreWorker state. The result is that when there is a surplus of energy all ants in the TempWorker state will become idle before any ants in the CoreWorker state.

Ants in the FirstReserve state and SecondReserve are state idle, but unlike ants in the Resting state (which are also idle) if they start working they

Algorithm 1 Binary-Threshold Algorithm

```
state ← RESTING, $currentTask \leftarrow \perp$, $\varrho[\tau] = 0, \forall \tau \in T$          ▷ Initialize
case state = RESTING
    $\forall \tau \in T, \varrho[\tau] \leftarrow \begin{cases} 0 & \text{if } f(\tau, i) < 0 \\ \max(\varrho[\tau]+1, 3) & \text{if } f(\tau, i) > 0 \end{cases}$
    candidateList ← $\{\tau \in T \mid \varrho[\tau] = 3\}$
    if candidateList $\neq \varnothing$ then
        with probability $\frac{1}{2}$ do
            $\forall \tau \in T, \varrho[\tau] \leftarrow 0$
            $currentTask$ ← random task from candidateList
            state ← TEMPWORKER
        end with
    end if
case state = FIRSTRESERVE
    if $f(currentTask, i) < 0$ then
        state ← RESTING
    else
        with probability $\frac{1}{2}$ do
            state ← TEMPWORKER
        otherwise
            state ← SECONDRESERVE
        end with
    end if
case state = SECONDRESERVE
    if $f(currentTask, i) < 0$ then
        state ← RESTING
    else
        state ← TEMPWORKER
    end if
case state = TEMPWORKER
    if $f(currentTask, i) < 0$ then
        state ← FIRSTRESERVE
    else
        state ← COREWORKER
    end if
case state = COREWORKER
    if $f(currentTask, i) < 0$ then
        state ← TEMPWORKER
    end if
end case
```

will do so at the task they were last working on. Ants in the FirstReserve state transition to the Resting state if there is a surplus of energy in *currentTask*, and otherwise they transition to the TempWorker state with constant probability or join the SecondReserve state. Ants in the SecondReserve state transition to the Resting state if there is a surplus of energy *currentTask*, and otherwise transition to the TempWorker state. Having two reserve states, one with a deterministic transition and the other with a probabilistic transition, allow us to recruit ants only from the reserved states instead of the entire idle population, which prevents big oscillations. Indeed, biologists [35–39] have provided compelling evidence that ants prefer to work on tasks they have become experienced in.

We remark that two reserve states is the minimum required for our recruitment process to work. The requirement that the potential of a task to reaches three before a resting ant starts working guarantees that resting ants do not start working on a task until ants in the reserved states for that task have already started working.

The number of bits of memory required by the algorithm to store the *state*, the *currentTask* and the potential table ϱ is $3 + \log_2 |T| + 2|T| \in \Theta(|T|)$, which is linear on the number of tasks and independent of the colony size $|A|$.

For simplicity whenever ants make a decision with constant probability, we will assume they make the decision with probability $\frac{1}{2}$ (i.e., as if each ant was flipping an independent unbiased coin). However by changing only the constants in the different lemmas and propositions, the same analysis works if ants are using any other constant probability.

4.2 Algorithm analysis

Before describing the properties satisfied by the proposed algorithm we introduce some additional notation. For a round i we denote with $\textsc{Rest}_i$, $\textsc{First}_i$, $\textsc{Second}_i$, $\textsc{Temp}_i$ and $\textsc{Core}_i$ the set of ants that at the beginning of round i are in the state Resting, FirstReserve, SecondReserve, TempWorker and CoreWorker respectively. Additionally for a round i and a task τ we denote with $\textsc{First}_i(\tau)$, $\textsc{Second}_i(\tau)$, $\textsc{Temp}_i(\tau)$ and $\textsc{Core}_i(\tau)$ the set of ants that at the beginning of round i have τ as their *currentTask* and are in the state FirstReserve, SecondReserve, TempWorker and CoreWorker state respectively.

For the energy supplied by the colony to converge to the energy demanded by a task we must require that the demand remains "constant" for a sufficiently long period of time. Specifically, the demand of task τ is constant at a round i iff $d(\tau, i-1) = d(\tau, i) = d(\tau, i+1)$. Similarly, the demand for a task τ is constant during an interval $[i, i+k]$ iff for all $j \in [i, i+k]$ the demand for task τ is constant at round j.

Proof roadmap. In the following paragraphs we outline the steps used to prove that the proposed algorithms converges quickly to a near-optimal allocation.

First, we show (Lemma 1) that given an interval of constant demand where the number of ants in the colony is sufficient to satisfy the demand, then as long as no task has too big a surplus during the interval, then the probability that none of the tasks with a deficit becomes satisfied during the interval is exponentially small in the interval length.

Next (Lemma 4) we show that in an interval of constant demand, once a task transitions from having a deficit of ants to having a surplus of ants, this transition will keep happening every two or three rounds during that interval. We refer to these transitions as *oscillations*. We also show that each time an oscillation happens, with constant probability it is a constant fraction smaller than the previous oscillation in the interval.

We leverage the previous result to show (Lemma 5) that in an interval of constant demand with oscillations, the probability that by the end of the interval the size of the oscillations is greater than one ant is exponentially small on the interval length.

Finally we show (Theorem 6) that during an interval of constant demand of logarithmic length, with high probability by the end of the interval the number of workers assigned to each task only differs from the demand by at most one ant. Due to space constraints we only outline the proofs.

Informally, the following lemma shows that during an interval of constant demand and given sufficiently many ants, then if a set of tasks is satisfied without surplus and a set of tasks is unsatisfied, then it is likely that an additional task will become satisfied.

Lemma 1. Fix a constant $\varepsilon \in (0,1)$ and let $k \in \Theta(\log \frac{1}{\varepsilon})$.

If the demands for all tasks are constant during the interval $[i, i+k]$, and $|A| \geq \sum_{\tau \in T} (d(\tau, i) + 1)$, and there is a set of tasks $C \subseteq T$ such that $\forall \tau \in T \setminus C$ $d(\tau, i) > w(\tau, i)$ and $\forall \tau \in C, \forall j \in [i, i+k]$ $d(\tau, j) \leq w(\tau, j) \leq d(\tau, j) + 1$, then

$\Pr\left[\forall \tau \in T \setminus C, \, \forall j \in [i, i+k] \, w(\tau, j) < d(\tau, j)\right] \leq \varepsilon$.

The proof of this lemma argues that given the conditions assumed, then in order for all task to remain unsatisfied during the entire interval, at least one ant must repeatedly decide not to work at any task, despite the fact that there are tasks that need work; and the probability of this happening is exponentially small on the interval length.

For the remaining part of the analysis we will make use of Hoeffding's inequality. In addition we will also leverage the following proposition, which can be shown easily as a consequence of Hoeffding's inequality.

Proposition 2. Let $X_1, \ldots, X_k$ be independent random variables where $\Pr[X_i = 1] = \Pr[X_i = 0] = \frac{1}{2}$ and let $X = \sum_{i=1}^{k} X_i$. If $k \geq 2$ then $\Pr\left[\frac{1}{8}k \leq X \leq \frac{7}{8}k\right] \geq \frac{1}{2}$.

To simplify the analysis we introduce the following definition.

Definition 1. An *oscillation* happens for task τ at round i if $f(\tau, i) < 0$ and $f(\tau, i+1) > 0$.

Observe that if the demand for task τ is constant at round i and an oscillation occurs for task τ at round i, then the fact that $f(\tau, i) < 0$ implies that $d(\tau, i) < |\textsc{Core}_i(\tau) \cup \textsc{Temp}_i(\tau)|$ and the algorithm guarantees that by round $i+1$ any temporary worker becomes idle and only core workers remain. This together with $f(\tau, i+1) > 0$ implies that $|\textsc{Core}_i(\tau)| < d(\tau, i)$. This is summarized in the next proposition.

Proposition 3. If there is an oscillation for task τ at round i and the demand for τ was constant at round i then $|\textsc{Core}_i(\tau)| < d(\tau, i) \leq |\textsc{Core}_i(\tau) \cup \textsc{Temp}_i(\tau)|$.

Observe that during an oscillation of task τ at round i the ants in $\textsc{Temp}_i(\tau)$ transition between working and being idle, while the ants in $\textsc{Core}_i(\tau)$ keep working through the oscillation, and there are never more than $\textsc{Core}_i(\tau) \cup \textsc{Temp}_i(\tau)$ ants working during the oscillation. Therefore given an oscillation for task τ at round i we say $|\textsc{Temp}_i(\tau)|$ is the *magnitude* of the oscillation, $|\textsc{Core}_i(\tau)|$ is the *low-value* of the oscillation, and $|\textsc{Core}_i(\tau) \cup \textsc{Temp}_i(\tau)|$ is the *high-value* of the oscillation.

The next lemma shows that if an oscillation happens during an interval of constant demand, it will happen again in two or three rounds. Moreover with constant probability the magnitude of the oscillation will become a constant fraction smaller than the previous oscillation by either increasing the low-value or decreasing the high-value.

Lemma 4. If there is an oscillation for task τ at round i and the demand for τ is constant during the interval $[i, i+4]$ then:

1. $\textsc{Second}_{i+2}(\tau) \subseteq \textsc{Temp}_i(\tau)$ and if $|\textsc{Temp}_i(\tau)| \geq 2$ then with probability at least $\frac{1}{2}$ we have $\frac{1}{8}|\textsc{Temp}_i(\tau)| \leq |\textsc{Second}_{i+2}(\tau)| \leq \frac{7}{8}|\textsc{Temp}_i(\tau)|$.
2. There is an oscillation for task τ either at
   a) round $i+2$ where $\textsc{Temp}_{i+2} = \textsc{Temp}_i(\tau) \setminus \textsc{Second}_{i+2}(\tau)$ and $\textsc{Core}_{i+2}(\tau) = \textsc{Core}_i(\tau)$, or
   b) round $i+3$ where $\textsc{Temp}_{i+3} = \textsc{Second}_{i+2}(\tau)$ and $\textsc{Core}_{i+3}(\tau) = \textsc{Temp}_i(\tau) \cup \textsc{Core}_i(\tau) \setminus \textsc{Second}_{i+2}(\tau)$.

The proof follows through a straightforward but detailed analysis of the state transitions made by the ants on the different states of the algorithm during the interval, relying on Proposition 2 to show that the magnitude of oscillations decreases by a constant fraction with constant probability.

The next lemma shows that given an interval with a constant demand and where the number of resting ants is greater than the number of ants required by a particular task, then the probability that the magnitude of the oscillations does not converge to one is exponentially small in the interval length.

Lemma 5. Fix $\varepsilon \in (0,1)$ and let $k \in \Theta(\log |A| + \log \frac{1}{\varepsilon})$. If the demand for task τ is constant in the interval $[i, i+k]$ with oscillations for task τ at round i and $i+k$ then $\Pr\left[|\textsc{Temp}_{i+k}(\tau)| > 1\right] \leq \varepsilon$.

This lemma can be shown as a consequence of the results of Lemma 4 and following a standard concentration of measure argument.

We are finally ready to prove the main theorem of this paper. Namely, we show that if the colony has enough ants to satisfy the demand, then during an interval with a length which is logarithmic on the colony size and linear on the number of tasks, the probability that the difference between the energy supplied to each task, and the energy required by each task, is greater than one for any task, is exponentially small on the size of the colony.

Theorem 6. *Fix* $\varepsilon \in (0,1)$ *and let* $k \in \Theta(|T|(\log \frac{1}{\varepsilon} + \log |A| + \log |T|))$.

If demand is constant for all tasks in the interval $[i, i+k]$ *and* $|A| \geq \sum_{\tau \in T}(1 + d(\tau, i))$, *then* $\Pr[\forall \tau \in T\, d(\tau, i+k) - w(\tau, i+k)| \leq 1] \geq 1 - \varepsilon$.

To prove this theorem we go through a case analysis, showing that unless the work supplied by the colony has converged to be within one ant of the demand for every task, an additional task will converge after at most $\Theta(\log |A| + \log \frac{1}{\varepsilon})$; the additive $\log |T|$ term is a result of a union bound over the probability of not converging for each task.

Given that in real ant colonies we expect $|T|$ to be a constant, then by letting $\varepsilon = 1/|A|$ we get the following corollary.

Corollary 1. *Let* $k \in \Theta(\log |A|)$. *If the demand is constant for all tasks in the interval* $[i, i+k]$ *and the number of ants is enough to satisfy the demand, then with high probability ants converge to an allocation where* $|d(\tau, i+k) - w(\tau, i+k)| \leq 1$.

5 Conclusions and future work

We've shown that given certain assumptions, a fairly simple task allocation algorithm can achieve near-optimal performance in time which is logarithmic on the colony size. The assumptions we had to make were that individual workers did not differ in their ability to perform tasks, that workers could sense whether each task required more workers or not, and that workers use a small amount of memory about recently performed tasks and task demand. No sophisticated communication or sensing, nor long-term memory were required. Interestingly, we also argued that once variation among workers (in ability to perform tasks) is introduced, the task allocation problem becomes NP-hard. This may be the reason that such variation, when it is not correlated with clear categorizing factors such as body size, is not always optimally utilized by ants [3].

It is inherent in our algorithm, and probably any algorithm that does not allow ants to measure precisely how many workers are needed to fulfill a task, that the number of workers engaged in a task fluctuates around the optimum. In other words, the algorithm does not reach an optimal allocation immediately, but approaches it over time. Our model assumes that workers go through three stages between being fully uncommitted and idle (Resting) and fully committed and working on a task (CoreWorker). These stages introduces some resistance in workers to being too frequently reallocated among different tasks, and thus reduces oscillations.

As future work, we would like to explore different, weaker binary feedback functions. For instance, one could consider a binary function which is only correct with probability $1-\gamma$ for some $\gamma < \frac{1}{2}$. An alternative binary feedback function could return accurate values when the difference between the demand and the supply is different by a significant mount, but returns random values when the energy supply and demand are close to being in equilibrium. We conjecture that with little modifications the same algorithm would perform well in these circumstances, but it would provide slightly weaker guarantees (bigger oscillations) and require a more technical analysis.

In this paper the role of communication has been abstracted and is captured by the binary feedback function. We would also like to explore possible mechanisms by which said function could be implemented using explicit means of communication, such as pheromone or other signaling mechanisms.

What are the effects of varying the number of tasks on the properties of the algorithm. Specifically, the memory requirements of the proposed algorithm are linear in the number of tasks, and so is its convergence time. Can either of these be decreased to be logarithmic on the number of tasks?

References

[1] Scott Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.

[2] Thomas D Seeley. *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press, 2009.

[3] Anna Dornhaus. "Specialization does not predict individual efficiency in an ant". In: *PLoS biology* 6.11 (2008), e285.

[4] Anna Dornhaus et al. "Why do not all workers work? Colony size and workload during emigrations in the ant Temnothorax albipennis". In: *Behavioral Ecology and Sociobiology* 63.1 (2008), pp. 43–51.

[5] Noa Pinter-Wollman et al. "How is activity distributed among and within tasks in Temnothorax ants?" In: *Behavioral Ecology and Sociobiology* 66.10 (2012), pp. 1407–1420.

[6] Madeleine Beekman, David JT Sumpter, and Francis LW Ratnieks. "Phase transition between disordered and ordered foraging in Pharaoh's ants". In: *Proceedings of the National Academy of Sciences* 98.17 (2001), pp. 9703–9706.

[7] D Sumpter and S Pratt. "A modelling framework for understanding social insect foraging". In: *Behavioral Ecology and Sociobiology* 53.3 (2003), pp. 131–144.

[8] MR Myerscough and BP Oldroyd. "Simulation models of the role of genetic variability in social insect task allocation". In: *Insectes Sociaux* 51.2 (2004), pp. 146–152.

[9] Michele C Lanan et al. "The trail less traveled: individual decision-making and its effect on group behavior". In: *PloS one* 7.10 (2012), e47976.

[10] Gene E Robinson. "Regulation of division of labor in insect societies". In: *Annual review of entomology* 37.1 (1992), pp. 637–665.

[11] Marc A Seid and James FA Traniello. “Age-related repertoire expansion and division of labor in Pheidole dentata (Hymenoptera: Formicidae): a new perspective on temporal polyethism and behavioral plasticity in ants”. In: *Behavioral Ecology and Sociobiology* 60.5 (2006), pp. 631–644.

[12] Edward O Wilson. “Caste and division of labor in leaf-cutter ants (Hymenoptera: Formicidae: Atta)”. In: *Behavioral Ecology and Sociobiology* 7.2 (1980), pp. 157–165.

[13] Jennifer M Jandt and Anna Dornhaus. “Spatial organization and division of labour in the bumblebee *Bombus impatiens*”. In: *Animal Behaviour* 77.3 (2009), pp. 641–651.

[14] Gene E Robinson, Christina M Grozinger, and Charles W Whitfield. “Sociogenomics: social life in molecular terms”. In: *Nature Reviews Genetics* 6.4 (2005), pp. 257–270.

[15] AB Sendova-Franks and NR Franks. “Spatial relationships within nests of the ant *Leptothorax unifasciatus* and their implications for the division of labour”. In: *Animal Behaviour* 50.1 (1995), pp. 121–136.

[16] Chris Tofts and Nigel R Franks. “Doing the right thing: ants, honeybees and naked mole-rats”. In: *Trends in ecology & evolution* 7.10 (1992), pp. 346–349.

[17] Amy L Toth and Gene E Robinson. “Worker nutrition and division of labour in honeybees”. In: *Animal Behaviour* 69.2 (2005), pp. 427–435.

[18] Deborah M Gordon. “The organization of work in social insect colonies”. In: *Complexity* 8.1 (2002), pp. 43–46.

[19] Anja Weidenmüller. “The control of nest climate in bumblebee (Bombus terrestris) colonies: interindividual variability and self reinforcement in fanning response”. In: *Behavioral Ecology* 15.1 (2004), pp. 120–128.

[20] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. “Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies”. In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 263.1376 (1996), pp. 1565–1569.

[21] Fabien Ravary et al. “Individual experience alone can generate lasting division of labor in ants”. In: *Current Biology* 17.15 (2007).

[22] Ofer Feinerman et al. “Collaborative search on the plane without communication”. In: *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. ACM. 2012, pp. 77–86.

[23] Ofer Feinerman and Amos Korman. “Memory lower bounds for randomized collaborative search and implications for biology”. In: *Distributed Computing*. Springer, 2012, pp. 61–75.

[24] Amos Korman. “Theoretical distributed computing meets biology”. In: *Proceedings of the 4th International Workshop on Theoretical Aspects of Dynamic Distributed Systems*. ACM. 2012, pp. 7–7.

[25] Calvin Newport Christoph Lenzen Nancy Lynch and Tsvetomira Radeva. “Trade-offs between Selection Complexity and Performance when Searching the Plane without Communication”. In: *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. Springer, 2014.

[26] Yehuda Afek et al. “A biological solution to a fundamental distributed computing problem”. In: *science* 331.6014 (2011), pp. 183–185.

[27] Chung Laung Liu and James W Layland. “Scheduling algorithms for multiprogramming in a hard-real-time environment”. In: *Journal of the ACM (JACM)* 20.1 (1973), pp. 46–61.

[28] Michael L. Dertouzos and Aloysius K. Mok. “Multiprocessor online scheduling of hard-real-time tasks”. In: *Software Engineering, IEEE Transactions on* 15.12 (1989), pp. 1497–1506.

[29] Robert I Davis and Alan Burns. “A survey of hard real-time scheduling for multiprocessor systems”. In: *ACM Computing Surveys (CSUR)* 43.4 (2011), p. 35.

[30] Amos Fiat, Yuval Rabani, and Yiftach Ravid. “Competitive k-server algorithms”. In: *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on.* IEEE. 1990, pp. 454–463.

[31] Yair Bartal and Adi Rosen. “The distributed k-server problem-a competitive distributed translator for k-server algorithms”. In: *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on.* IEEE. 1992, pp. 344–353.

[32] Chryssis Georgiou and Alexander A Shvartsman. “Cooperative Task-Oriented Computing: Algorithms and Complexity”. In: *Synthesis Lectures on Distributed Computing Theory* 2.2 (2011), pp. 1–167.

[33] R Garey Michael and David S Johnson. “Computers and Intractability: A guide to the theory of NP-completeness”. In: *WH Freeman & Co., San Francisco* (1979).

[34] David B Shmoys and Éva Tardos. “An approximation algorithm for the generalized assignment problem”. In: *Mathematical Programming* 62.1-3 (1993), pp. 461–474.

[35] Jean-Louis Deneubourg et al. “Self-organization mechanisms in ant societies (II): learning in foraging and division of labor”. In: *From individual to collective behavior in social insects* (1987), pp–177.

[36] Daniel Merkle and Martin Middendorf. “Dynamic polyethism and competition for tasks in threshold reinforcement models of social insects”. In: *Adaptive Behavior* 12.3-4 (2004), pp. 251–262.

[37] RC Plowright and CMS Plowright. “Elitism in social insects: a positive feedback model”. In: *Interindividal Behavioral Variability in Social Insects* (1988), pp. 419–431.

[38] Guy Theraulaz, Eric Bonabeau, and JN Denuebourg. “Response threshold reinforcements and division of labour in insect societies”. In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 265.1393 (1998), pp. 327–332.

[39] Frederic Tripet and Peter Nonacs. “Foraging for Work and Age-Based Polyethism: The Roles of Age and Previous Experience on Task Choice in Ants”. In: *Ethology* 110.11 (2004), pp. 863–877.