Self-Stabilizing Task Allocation In Spite of Noise

Anna Dornhaus[1], Nancy Lynch[2], Frederik Mallmann-Trenn*[2], Dominik Pajak[2], and Tsvetomira Radeva[2]

[1]University of Arizona, Department of Ecology and Evolutionary Biology
[2]MIT, CSAIL

Abstract

We study the problem of distributed task allocation inspired by the behavior of social insects, which perform task allocation in a setting of limited capabilities and noisy environment feedback. We assume that each task has a *demand* that should be satisfied but not exceeded, i.e., there is an optimal number of ants that should be working on this task at a given time. The goal is to assign a near-optimal number of workers to each task in a distributed manner and without explicit access to the values of the demands nor the number of ants working on the task.

We seek to answer the question of how the quality of task allocation depends on the accuracy of assessing whether too many (`overload`) or not enough (`lack`) ants are currently working on a given task. Concretely, we address the open question of solving task allocation in the model where each ant receives feedback that depends on the *deficit* defined as the (possibly negative) difference between the optimal demand and the current number of workers in the task. The feedback is modeled as a random variable that takes value `lack` or `overload` with probability given by a sigmoid of the deficit. Each ants receives the feedback independently, but the higher the overload or lack of workers for a task, the more likely it is that all the ants will receive the same, correct feedback from this task; the closer the deficit is to zero, the less reliable the feedback becomes. We measure the performance of task allocation algorithms using the notion of *regret*, defined as the absolute value of the deficit summed over all tasks and summed over time.

We propose a simple, constant-memory, self-stabilizing, distributed algorithm that quickly converges from any initial distribution to a near-optimal assignment. We also show that our algorithm works not only under stochastic noise but also in an adversarial noise setting. Moreover we show a lower bound on the regret for any constant-memory algorithm, which matches, up to a constant factor, the regret achieved by our algorithm.

keywords— distributed task allocation; self-stabilization; randomized algorithms; load balancing, social insects

*mallmann@mit.edu

1 Introduction

Task allocation in social insect colonies is the process of assigning workers to tasks such as foraging, scouting, nursing, *etc.* in a way that maximizes reproductive success of the colony. Each ant can work on each of the tasks but the demands of the tasks are different and might also vary over time. The ants probably neither know the number of ants needed nor can count the current number of ants working on a given task. Therefore the allocation has to be performed based only on the feedback from the tasks. The feedback, in biology called 'task stimulus', corresponds for example to sensing a too-high temperature in the nest, seeing light through a hole in the nest-wall, or smelling a pheromone produced by hungry brood. Despite using limited communication, local observations, and noisy sensing, many ant species are known to excel at task allocation. How do ants perform task allocation and what can we learn from their behavior?

In [CDLN14] the authors proposed a solution to the problem of task allocation in the case where the feedback received by the ants is always correct. More precisely, in [CDLN14], each task $j \in [k]$ has a demand $d^{(j)}$, that represents the number of workers needed at that task. If the *load*, *i.e.,* the number of ants working on the task, exceeds the demand, then all ants receive feedback `overload`. Conversely, if the load is below or equals the demand of the task, then all ants receive feedback `lack`. Such a feedback function is rather unrealistic in ant colonies due to its sharp transition between `overload` and `lack`— it requires each ant to be able to tell the difference between $d^{(j)}$ and $d^{(j)}+1$ number of workers at a task. The authors in [CDLN14] therefore pose the open problem of considering a *weaker*, noisy binary-feedback version—the focal point of this paper.

Similar to the model introduced in [CDLN14] we assume that each ant periodically receives a binary feedback, indicating whether for a given task there is a `lack` or `overload` of workers. However we want to consider noisy feedback hence we assume that the binary feedback is a random variable taking values `lack` or `overload` with probabilities depending on the sigmoid of the *deficit* of a given task. Deficit of a task is defined as the demand minus the number of workers at the task.

Similarly to [CDLN14], we assume synchronous rounds: at the beginning of each round each ant receives binary feedback of the load of each task. The ants then concurrently make a decision of whether to join a task or to leave their current task.

In order to measure how well an algorithm performs in this setting, we borrow the notion of *regret* from the area of machine learning. Intuitively, the regret in our setting measures the sum over all rounds over all tasks of the absolute values of the deficits (demand minus load) of the tasks. Here we penalize overload and underload equally: an underload corresponds to work that is not being done and each ant exceeding the demand of a task corresponds to work being wasted (or, even worse, sometimes the excessive number of workers in a task may block each other and decrease the efficiency [DNDF06; CGR13]). Note, that we do not charge any cost for switching tasks.

Our main result is a simple, constant-memory Algorithm Ant that utilizes the oscillations in the number of workers in each task in order to achieve a stable allocation in the synchronous model. Intuitively, from a perspective of a fixed task j, there exists value c such that if $(1-c)\cdot d^{(j)}$ ants work on task j, then all ants with high probability receive feedback `lack` and if the load is $(1+c)\cdot d^{(j)}$ then the feedback is `overload` for all the ants with high probability. Observe that oscillating the load between $(1-c)\cdot d^{(j)}$ and $(1+\gamma)\cdot d^{(j)}$ ensures stability because all the ants receive the same feedback. Moreover the load in the task is only $cd^{(j)}$ from the optimal. The smallest possible c such that it ensures stability of every task is an important parameter in our analysis, we call it *critical value* and denote by γ^* (see Section 2 for a formal definition). We show that our Algorithm Ant has regret proportional to γ^* times the sum of the demands of all the tasks. We also lower bound any constant-memory algorithm in terms of our regret function. The lower bound shows that the regret of any constant-memory algorithm is at least linear in the sum of demands times the critical value.

We then extend our first algorithm to derive a more theoretical algorithm Algorithm Precise Sigmoid that uses more memory but achieves smaller regret. This algorithm, together with a lower bound, establish a theoretical tradeoff between the memory available at each ant and the optimal regret that can be achieved by an algorithm using this memory.

Our results show, that almost-optimal task allocation can be achieved in the synchronous model with noisy sensing.

2 Model

We have a collection of n ants and k tasks where each task $j \in [k]$ has a fixed demand $d^{(j)}$ meaning that the task requires $d^{(j)}$ many ants assigned to it. Let $\mathbf{d}$ be the demand vector. Let $W_t^{(j)}, j \in [k]$ denote the load of task j at time t, *i.e.,* the number of ants performing the task. For task $j \in [k]$ we define the deficit as $\Delta_t^{(j)} = d^{(j)} - W_t^{(j)}$ and a negative deficit signifies an overload. The initial distribution of ants is arbitrary. We assume that the demands do not change, but our results trivially extend to changing demand due to the self-stabilizing nature of our algorithms.

We seek to model the noise in the sensing such that the following axioms are fulfilled. First, in a case of a very large deficit almost all ants will notice this (w.h.p.[1] all ants receive feedback `lack`). Similarly, in case of a high overload (negative deficit) almost all ants will notice this as well. Second, whenever exactly the correct number of ants are working on a given task, then the 'uncertainty' in this task is the largest and each ant receives `lack` and `overload` with equal probability.

Sigmoid feedback. We define round t to be the time interval between times $t-1$ and t. At the beginning of round t, each ant receives feedbacks $F_t^{(j)}(i)$ from for all tasks $j \in [k]$. The noisy feedback is modeled by a sigmoid function $s(x) = \frac{1}{1+e^{-\lambda x}}$, for fixed $\lambda \in \mathbb{R}$:

$$F_t^{(j)}(i) = \begin{cases} \texttt{lack} & \text{with probability } s(\Delta_{t-1}^{(j)}) \\ \texttt{overload} & \text{otherwise} \end{cases}.$$

Note that feedback variables depend on deficits from the previous round $\Delta_{t-1}^{(j)}$. It is not crucial for our results to have a sigmoid function; in fact all our results apply for any monotone antisymmetric function f with exponential decay and $\lim_{x\to-\infty} f(x) = 0$ and $\lim_{x\to\infty} f(x) = 1$ and $f(0) = 1/2$.

Model of an algorithm. Each ant is assumed to have a local memory that is linear in the number of tasks but independent of n. After collecting the vector of feedbacks each ant independently (without communication), in parallel makes a decision whether to change its assignment. The decision of the ant is based on the memory state of the ant and the collected feedbacks $F_t^{(j)}(i)$ for all $j \in [k]$. Working ant may decide to leave its task or remain working and idle ant may decide to join some task or remain idle. The decision can also consist in changing the memory state. The decisions made by the ants at the beginning of round t determine the assignment of the ants to the tasks for the duration of round t.

Critical value. The critical value is the value which ensures that all the ants receive w.h.p. the correct feedback provided that the overload/lack is far enough away from 0.

Definition 2.1 (Critical value and grey zone)**.** *We define the* critical (feedback) value *as follows depending on the model. Let* $y(x) = \min_{c\in\mathbb{R}}\{s(-c \cdot d^{(j)}) \leq x\colon \text{ for all } j \in [k]\}$. *We define the critical (feedback) value to be* $\gamma^* = y(1/n^8)$. *Note that, due to the antisymmetry of the sigmoid,* $s(-\gamma^* \cdot d^{(j)}) = 1 - s(\gamma^* \cdot d^{(j)})$.

We define for each task $j \in [k]$ *the* grey zone *to be* $g_j = [-\gamma^* d^{(j)}, \gamma^* d^{(j)}]$.

[1] With high probability means here with probability at least $1 - 1/n$.

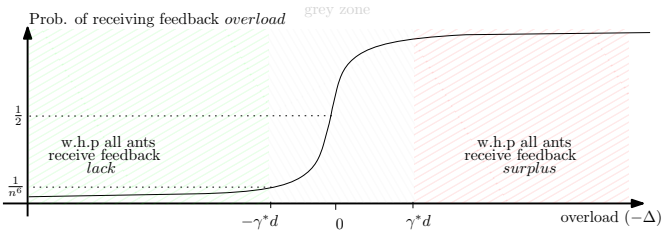

Figure 1: Whenever the overload is in the green (red, respectively) region, all ants will receive w.h.p. the feedback `lack` (`overload`, respectively). Whenever the overload is in the grey region (falling tilling pattern), and the closer the overload is to 0, the more unpredictable is the feedback received by the ants.

Regret metric, closeness and farness. In order to compare the quality of task allocation strategies we use the following notion of regret $r(t)$, defined for time t as: $r(t) = \sum_{j\in[k]} |d^{(j)} - W_t^{(j)}| = \sum_{j\in[k]} |\Delta_t^{(j)}|$. And we define the (total) regret up to time t as: $R(t) = \sum_{i\le t} r(i)$.

The idea behind the regret metric is that if over large scales of time the deficits is only rarely large, then this should not affect the performance (*i.e.,* the survival of the colony) too much. It is worth pointing out that we penalize lack and overload (measuring a waste of energy) equally and we leave it as a future direction to use different weights.

We say that an algorithm produces an assignment that is *c-close* if the average regret of the allocation is at most c times the critical value and the sum of demands, *i.e.,* $\lim_{t\to\infty} \frac{R(t)}{t} \le \left(c\gamma^* \sum_{j\in[k]} d^{(j)}\right) + O(1)$. Similarly, we say an the assignment of an algorithm is *c-far* if the average regret of is at least multiplicative in c times the critical value and the sum of demands, *i.e.,* $\lim_{t\to\infty} \frac{R(t)}{t} \ge \left(c\gamma^* \sum_{j\in[k]} d^{(j)}\right)$.

3 Results and Discussion

In this section we present simplified statements of our results and we refer the reader to the full version for precise statements. Our results are threefold:

(i) In our main theorem we show that the somewhat realistic Algorithm Ant with parameter γ results w.h.p. in an assignment that is $5\frac{\gamma}{\gamma^*}$-close provided that $\gamma \ge \gamma^*$.

(ii) Second, we develop an algorithm (Algorithm Precise Sigmoid) that is slightly more artificial but achieves arbitrary closeness: for any $\varepsilon \in (0,1)$, the algorithm produces w.h.p. an assignment that is ε-close. The algorithm uses $O(\log(1/\varepsilon))$ memory and synchronous phases of length $O(1/\varepsilon)$.

(iii) Finally, we show that this bound is in fact tight: any algorithm using $c\log(1/\varepsilon)$ memory, for small enough c will produce, with overwhelming probability, an assignment that is ε-far for almost all time steps.

4 Algorithm Ant and Proof

In our Algorithm Ant, time is grouped into phases consisting of two rounds. On a high level, in each phase each of the ants collects two samples (one in each round). The sample is a vector consisting of feedbacks (`lack` or `overload`) from each task in this round. To obtain the first sample, all ants assigned to a task work on it. For the second sample all working ants leave their tasks with a small probability. If, for some task, both samples indicate an overload, each ant will permanently leave

the task with a small probability. If for some task both samples indicate an underload, then ants that are currently idle will join this task (or if there are multiple such tasks then each idle ant joins one chosen uniformly at random). This two-sample approach can be seen as computing the slope of the regret function in each task. The ants collectively, in a distributed fashion, increase or decrease the loads in the tasks by a small fraction depending on the computed "direction of the gradient" and learning parameter γ. In some sense, the algorithm bears similarities to gradient descent; in our case a noisy and distributed version. We show that this algorithm achieves a task allocation that is up to a constant factor optimal with respect to the aforementioned lower bound.

Proof Idea. We divide time into intervals of length n^4 and bound the regret in each such interval. The idea of our proof is to analyze the regret in three different regions. For this we split the regret $R(t)$ into three functions $R^+(t), R^{\approx}(t)$ and $R^-(t)$. Intuitively $R^+(t)$ is the 'overload regret' *i.e.,* regret for those tasks j and time steps t for which the deficit is large (larger than $c\gamma d^{(j)}$ for some constant c). From the perspective of any task j we can show several lemmas bounding the dynamics of the load of this task during the considered interval of length n^4. We can show that if the value of the load exceeds $(1+\gamma)d^{(j)}$ then, if we look only at the first sample in each phase then it remains to be at least $(1+\gamma)d^{(j)}$ for a polynomial number of rounds. Moreover since it is above $(1+\gamma)d^{(j)}$ then by the definition of the critical value ($\gamma \geq \gamma^*$) the load cannot increase (because first sample indicates `overload` and it is outside the grey zone). Finally for some c if the load is larger than $(1+c\gamma)d^{(j)}$ then we can show that the load decreases (both samples show `overload`). And this decrease is in each such step by a factor of approximately $(1-1/\gamma)$ thus we can bound the total regret associated with task j from steps with large deficit using a geometric sum by $O(n/\gamma)$. Hence in total $R^+(t) = O(kn/\gamma)$.

Function $R^-(t)$ is the 'underload regret' *i.e.,* regret for tasks j and time steps t for which the deficit is large negative (smaller than $-c\gamma d^{(j)}$ for some constant c). We bound it by $O(nk)$ using a series of potential arguments. We do this by arguing that in any phase w.h.p. either the 'lack' of the resources decreases by a significant amount or the number of tasks that are underloaded does. It turns out that both quantities are monotonically decreasing every second round and will w.h.p. quickly reach 0.

Finally, $R^{\approx}(t)$ 'oscillation regret' can simply, by definition, be bounded by $O(\gamma \sum_{j\in[k]} d^{(j)})$ per time step; in fact this is, up to constants, tight as our lower bounds show. Together this shows that $R(t) = O(nk/\gamma + \gamma \sum_{j\in[k]} d^{(j)})$.

We also show that in every interval of length n^4 in at most $O(k \log n/\gamma)$ steps there exists a task j for which the deficit has absolute value of at least $5\gamma d^{(j)}$. This shows that our Algorithm Ant guarantees that in all except only a very small fraction of rounds load in every task almost exactly matches the demand.

References

[CDLN14] A. Cornejo, A. R. Dornhaus, N. A. Lynch, and R. Nagpal. "Task Allocation in Ant Colonies". In: *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*. 2014, pp. 46–60. URL: https://doi.org/10.1007/978-3-662-45174-8_4 (cit. on p. 1).

[CGR13] T. J. Czaczkes, C. Grüter, and F. L. Ratnieks. "Negative feedback in ants: crowding results in less trail pheromone deposition". In: *Journal of the Royal Society Interface* 10.81 (2013), p. 20121009 (cit. on p. 1).

[DNDF06] A. Dussutour, S. C. Nicolis, J.-L. Deneubourg, and V. Fourcassié. "Collective decisions in ants when foraging under crowded conditions". In: *Behavioral Ecology and Sociobiology* 61.1 (2006), pp. 17–30 (cit. on p. 1).