

Anytime Heuristic for Weighted Matching Through Altruism-Inspired Behavior

Panayiotis Danassis, Aris Filos-Ratsikas, Boi Faltings

Artificial Intelligence Laboratory (LIA), École Polytechnique Fédérale de Lausanne (EPFL)

{panayiotis.danassis,aris.filosratsikas,boi.faltings}@epfl.ch

ABSTRACT

We present a novel anytime heuristic (ALMA), inspired by the human principle of altruism, for solving the assignment problem. ALMA is decentralized, completely uncoupled, and requires no communication between the participants. We prove an upper bound on the convergence speed that is polynomial in the desired number of resources and competing agents per resource; crucially, in the realistic case where the aforementioned quantities are bounded independently of the total number of agents/resources, the convergence time remains *constant* as the total problem size increases.

We have evaluated ALMA under three test cases: (i) an anti-coordination scenario where agents with similar preferences compete over the same set of actions, (ii) a resource allocation scenario in an urban environment, under a constant-time constraint, and finally, (iii) an on-line matching scenario using *real* passenger-taxi data. In all of the cases, ALMA was able to reach high social welfare, while being orders of magnitude faster than the centralized, optimal algorithm. The latter allows our algorithm to scale to realistic scenarios with hundreds of thousands of agents, e.g., vehicle coordination in urban environments.

KEYWORDS

Coordination and Cooperation; Resource Allocation; Multi-agent Learning

1 INTRODUCTION

One of the most relevant problems in multi-agent systems (MAS) is finding an optimal allocation between agents. This pertains to role allocation (e.g., team formation for autonomous robots [GA13]), task assignment (e.g., employees of a factory, taxi-passenger matching [VCGA12]), resource allocation (e.g., parking spaces and/or charging stations for autonomous vehicles [GC13]), etc. What follows is *applicable to any such scenario*, but for concreteness we will refer to the allocation of a set of resources to a set of agents, a setting known as the *assignment problem*, one of the most fundamental combinatorial optimization problems [Mun57].

When designing algorithms for assignment problems, a significant challenge emerges from the nature of real-world applications, which is often distributed and information-restrictive. For the former part, a variety of decentralized algorithms have been developed [GLM10, IS17, ZSP08, BNBA12], all of which, though, require polynomial in the problem size number of messages. However, inter-agent interactions often repeat no more than a few hundreds of times. Moreover, sharing plans and preferences creates high overhead, and there is often a lack of responsiveness and/or communication between the participants [SKKR10]. Achieving fast convergence and high efficiency in such information-restrictive

settings is extremely challenging. Yet, humans are able to routinely and robustly coordinate in similar everyday scenarios. One driving factor that facilitates human cooperation is the principle of *altruism* [CR02, NS05, Gin00]. Inspired by human behavior, the proposed heuristic is modeled on the principle of altruism. This results to fast convergence to highly efficient allocations, without any communication between the agents.

A distinctive characteristic of ALMA is that agents make decisions locally, based on (i) the contest for resources that *they* are interested in, (ii) the agents that are interested in the *same* resources. If each agent is interested in only a *subset* of the total resources, ALMA converges in time polynomial in the maximum size of the subsets; not the total number of resources. In particular, if the size of each subset is a constant fraction of the total number of resources, then the convergence time is *constant*, in the sense that it does not grow with the problem size. The same is not true for other algorithms (e.g., the optimal centralized solution) which require time polynomial in the *total* number of agents/resources, even if the aforementioned condition holds. The condition holds by default in many real-world applications; agents have only local knowledge of the world, there is typically a cost associated with acquiring a resource, or agents are simply only interested in resources in their vicinity (e.g., urban environments). This is important, as the proposed approach avoids having to artificially split the problem in subproblems (e.g., by placing bounds or spatial constraints) and solve those separately, in order to make it tractable. Instead, ALMA utilizes a natural domain characteristic, instead of an artificial optimization technique (i.e., artificial bounds). Coupled to the convergence time, the decentralized nature of ALMA makes it applicable to large-scale, real-world applications (e.g., IoT devices, intelligent infrastructure, autonomous vehicles, etc.).

1.1 Our Results

Our main contributions in this paper are:

(1) We introduce a novel, anytime **ALtruistic MAtching heuristic (ALMA)** for solving the assignment problem. ALMA is decentralized, completely uncoupled (i.e., each agent is only aware of his own history of action/reward pairs [Tal13]), and requires no communication between the agents.

(2) We prove that if we bound the maximum number of resources an agent is interested in, and the maximum number of agents competing for a resource, the expected number of steps for any agent to converge is independent of the total problem size. Thus, we do not require to artificially split the problem, or similar techniques, to render it manageable.

(3) We provide a thorough empirical evaluation of ALMA on both synthetic and real data. In particular, we have evaluated ALMA under three test cases: (i) an anti-coordination scenario where agents

with similar preferences compete over the same set of actions, (ii) a resource allocation scenario in an urban environment, under a constant-time constraint, and finally, (iii) an on-line matching scenario using *real* passenger-taxi data. In all of the cases, ALMA achieves high social welfare (total satisfaction of the agents) as compared to the optimal solution, as well as various other algorithms.

1.2 Related Work

The assignment problem consists of finding a maximum weight matching in a weighted bipartite graph and it is one of the best-studied combinatorial optimization problems in the literature. The first polynomial time algorithm (with respect to the total number of nodes, and edges) was introduced by Jacobi in the 19th century [BJ65, Oll09], and was succeeded by many classical algorithms [Mun57, EK72, Ber79] with the *Hungarian* algorithm of [Kuh55] being the most prominent one (see [Su15] for an overview). The problem can also be solved via linear programming [Dan90], as its LP formulation relaxation admits integral optimal solutions [PS82]. In Section 3.3, we will apply ALMA on a non-bipartite setting, which corresponds to the more general *maximum weight matching* problem on general graphs. To compute the optimal in this case, we will use the *blossom algorithm* of [Edm65] (see [LP09]).

In reality, a centralized coordinator is not always available, and if so, it has to know the utilities of all the participants, which is often not feasible. In the literature of the assignment problem, there also exist several decentralized algorithms (e.g., [GLM10, IS17, ZSP08, BNBA12] which are the decentralized versions of the aforementioned well-known centralized algorithms). However, these algorithms require polynomial computational time and polynomial number of messages (such as cost matrices [IS17], pricing information [ZSP08], or a basis of the LP [BNBA12], etc.). Yet, agent interactions often repeat no more than a few hundreds of times. To the best of our knowledge, a decentralized algorithm that requires no message exchange (i.e., no communication network) between the participants, and achieves high efficiency, like ALMA does, has not appeared in the literature before. Let us stress the importance of such a heuristic: as autonomous agents proliferate, and their number and diversity continue to rise, differences between the agents in terms of origin, communication protocols, or the existence of sub-optimal, legacy agents will bring forth the need to collaborate without any form of explicit communication [SKKR10]. Finally, inter-agent communication creates high overhead as well.

ALMA is inspired by the decentralized allocation algorithm of [DF19]. Using such a simple learning rule which only requires environmental feedback, allows our approach to scale to hundreds of thousands of agents. Moreover, it does not require global knowledge of utilities; only local knowledge of personal utilities (in fact, we require knowledge of pairwise differences which are far easier to estimate).

2 ALTRUISTIC MATCHING HEURISTIC

In this section, we define ALMA and prove its convergence properties. We begin with the definition of the assignment problem, and its interpretation in our setting.

2.1 The Assignment Problem

The assignment problem consists of finding a maximum weight matching in a weighted bipartite graph, $\mathcal{G} = \{\mathcal{N} \cup \mathcal{R}, \mathcal{E}\}$. In the studied scenario, $\mathcal{N} = \{1, \dots, N\}$ agents compete to acquire $\mathcal{R} = \{1, \dots, R\}$ resources. We assume that each agent n is interested in a subset of the total resources, i.e., $\mathcal{R}^n \subset \mathcal{R}$. The weight of an edge $(n, r) \in \mathcal{E}$ represents the utility ($u_n(r)$) agent n receives by acquiring resource r . Each agent can acquire at most one resource, and each resource can be assigned to at most one agent. The goal is to maximize the social welfare (sum of utilities), i.e.,

$$\begin{aligned} \max_{\mathbf{x} \geq 0} \quad & \sum_{(n,r) \in \mathcal{E}} u_{n,r} x_{n,r} \\ \text{subject to} \quad & \sum_{r|(n,r) \in \mathcal{E}} x_{n,r} = 1, \forall n \in \mathcal{N} \\ & \sum_{n|(n,r) \in \mathcal{E}} x_{n,r} = 1, \forall r \in \mathcal{R} \end{aligned} \quad (1)$$

2.2 Learning Rule

This section describes the proposed heuristic (**ALMA**: **ALtruistic MAtching heuristic) for weighted matching**. We make the following two assumptions: First, we assume (possibly noisy) knowledge of personal preferences by each agent. Second, we assume that agents can observe feedback from their environment. This is used to inform collisions and detect free resources. It could be achieved by the use of visual, auditory, olfactory sensors etc., or by any other means of feedback from the resource (e.g., by sending an occupancy message). Note here that these messages would be between the requesting agent and the resource, not between the participating agents themselves, and that it suffices to send only 1 bit of information (e.g., 0, 1 for occupied / free respectively).

ALMA learns the right action through repeated trials as follows. Each agent sorts his available resources (possibly $\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing order of utility $(r_1, r_2, \dots, r_i, r_{i+1}, \dots, r_{R^n})$. The set of available actions is denoted as $\mathcal{A} = \{Y, A_{r_1}, \dots, A_{r_{R^n}}\}$, where Y refers to yielding, and A_r refers to accessing resource r . Each agent has a strategy (g_n) that points to a resource and it is initialized to the most preferred one. As long as an agent has not acquired a resource yet, at every time-step, there are two possible scenarios. If $g_n = A_r$ (strategy points to resource r), then agent n attempts to acquire that resource. If there is a collision, the colliding parties back-off with some probability. Otherwise, if $g_n = Y$, the agent chooses a resource r for monitoring. If the resource is free, he sets $g_n \leftarrow A_r$. Alg. 1 presents the pseudo-code of ALMA, which is followed by every agent individually. The back-off probability and the next resource to monitor are computed individually and locally based on the current resource and each agent's utilities, as will be explained in the following section. Finally, note that if the available resources change over time, the agents simply need to sort again the currently available ones.

2.3 Back-off Probability & Resource Selection

Let \mathcal{R} be totally ordered in decreasing utility under $<_n$, $\forall n \in \mathcal{N}$. If more than one agent compete for resource r_i (step 4 of Alg. 1), each of them will back-off with probability that depends on their utility

loss of switching to their respective remaining resources. The loss is given by Eq. 2.

$$\text{loss}_n^i = \frac{\sum_{j=i+1}^k u_n(r_i) - u_n(r_j)}{k-i} \quad (2)$$

where $k \in \{i+1, \dots, R^n\}$ denotes the number of remaining resources to be considered. For $k = i+1$, the formula only takes into account the utility loss of switching to the immediate next best resource, while for $k = R^n$ it takes into account the average utility loss of switching to all of the remaining resources. In the remainder of the paper we assume $k = i+1$, i.e., $\text{loss}_n^i = u_n(r_i) - u_n(r_{i+1})$. The actual back-off probability can be computed with any monotonically decreasing function f on loss_n , i.e., $P_n(r_i, <_n) = f_n(\text{loss}_n^i)$. In the evaluation section, we have used two such functions, a linear (Eq. 3), and the logistic function (Eq. 4). The parameter ϵ places a threshold on the minimum / maximum back-off probability for the linear curve, while γ determines the steepness of the logistic curve.

$$f(\text{loss}) = \begin{cases} 1 - \epsilon, & \text{if } \text{loss} \leq \epsilon \\ \epsilon, & \text{if } 1 - \text{loss} \leq \epsilon \\ 1 - \text{loss}, & \text{otherwise} \end{cases} \quad (3)$$

$$f(\text{loss}) = \frac{1}{1 + e^{-\gamma(0.5 - \text{loss})}} \quad (4)$$

Using the aforescribed rule, agents that do not have good alternatives will be less likely to back-off and vice versa. The ones that do back-off select an alternative resource and examine its availability. The resource selection is performed in sequential order, i.e., $S_n(r_{\text{prev}}, <_n) = r_{\text{prev}+1}$, where r_{prev} denotes the resource selected by that agent in the previous round. We also examined the possibility of using a weighted or uniformly at random selection, but achieved inferior results.

2.4 Altruism-Inspired Behavior

ALMA is inspired by the human principle of altruism. We would expect an altruistic person to give up a resource either to someone who values it more, if that resulted in an improvement of the well-being of society [CR02], or simply to be nice to others [Sim16]. Such behavior is especially common in situations where the backing-off subject has equally good alternatives. For example, in human pick-up teams, each player typically attempts to fill his most preferred position. If there is a collision, a colliding player might back-off because his teammate is more competent in that role, or because he has an equally good alternative, or simply to be polite; the player backs-off now and assumes that role at some future game. From an alternative viewpoint, following such an altruistic convention leads to a faster convergence which outweighs the loss in utility. Such conventions allow humans to routinely and robustly coordinate in large scale and under dynamic and unpredictable demand. Behavioral conventions are a fundamental part of human societies [Lew08], yet they have not appeared meaningfully in empirical modeling of multi-agent systems. Inspired by human behavior, ALMA attempts to reproduce these simple rules in an artificial setting.

Algorithm 1 ALMA: Altruistic Matching Heuristic.

Require: Sort resources ($\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing order of utility $r_1, r_2, \dots, r_i, r_{i+1}, \dots, r_{R^n}$.

Require: Initialize $g_n \leftarrow A_{r_1}$, and $r_{\text{prev}} \leftarrow r_1$.

```

1: procedure ALMA
2:   if  $g_n = A_r$  then
3:     Agent  $n$  attempts to acquire resource  $r$ . Set  $r_{\text{prev}} \leftarrow r$ .
4:     if Collision( $r$ ) then
5:       back-off (set  $g_n \leftarrow Y$ ) with probability  $P_n(r, <_n)$ .
6:     else ( $g_n = Y$ )
7:       Agent  $n$  monitors  $r \leftarrow S_n(r_{\text{prev}}, <_n)$ . Set  $r_{\text{prev}} \leftarrow r$ .
8:       if Free( $r$ ) then set  $g_n \leftarrow A_r$ .

```

2.5 Convergence

Agents who have not acquired a resource ($g_n = Y$) will not claim an occupied one. Additionally, every time a collision happens, there is a positive probability that some agents will back-off. As a result, the system will converge. The following theorem proves that the expected convergence time is logarithmic in the number of agents N and quadratic in the number of resources R .

THEOREM 2.1. *For N agents and R resources, the expected number of steps until the system of agents following Alg. 1 converges to a complete matching is bounded by (5), where $p^* = f(\text{loss}^*)$, and loss^* is given by Eq. 6.*

$$O\left(R \frac{2 - p^*}{2(1 - p^*)} \left(\frac{1}{p^*} \log N + R \right)\right) \quad (5)$$

$$\text{loss}^* = \arg \min_{\text{loss}_n^r} \left(\min_{r \in \mathcal{R}, n \in \mathcal{N}} (\text{loss}_n^r), 1 - \max_{r \in \mathcal{R}, n \in \mathcal{N}} (\text{loss}_n^r) \right) \quad (6)$$

PROOF. To improve readability, we will only provide a sketch of the proof. Please see the appendix for the complete version. The proof is based on [CF11, DF19].

We first assume that every agent, on every collision, backs-off with the same constant probability p . We start with the case of having N agents competing for 1 resource and model our system as a discrete time Markov chain. Intuitively, this Markov chain describes the number of individuals in a decreasing population, but with two caveats: the goal (absorbing state) is to reach a point where only one individual remains, and if we reach zero, we restart. We prove that the expected number of steps until we reach a state where either 1 or 0 agents compete for that resource is $O\left(\frac{1}{p} \log N\right)$.

Moreover, we prove that with high probability, $\Omega\left(\frac{2(1-p)}{2-p}\right)$, only 1 agent will remain (contrary to reaching 0 and restarting the process of claiming the resource), no matter the initial number of agents. Having proven that, we move to the general case of N agents competing for R resources.

At any time, at most N agents can compete for each resource. We call this period a round. During a round, the number of agents competing for a specific resource monotonically decreases, since that resource is perceived as occupied by non-competing agents. Let the round end when either 1 or 0 agents compete for the resource. This will require $O\left(\frac{1}{p} \log N\right)$ steps. If all agents backed-off, it will

take on average R steps until at least one of them finds a free resource. We call this period a break. In the worst case, the system will oscillate between a round and a break. According to the above, one oscillation requires in expectation $\mathcal{O}\left(\frac{1}{p} \log N + R\right)$ steps. If $R = 1$, as mentioned in the previous paragraph, in expectation there will be $\frac{2-p}{2(1-p)}$ oscillations. For $R > 1$ the expected number of oscillations is bounded by $\mathcal{O}\left(R \frac{2-p}{2(1-p)}\right)$. Thus, we conclude that if all the agents back-off with the same constant probability p , the expected number of steps until the system converges to a complete matching is $\mathcal{O}\left(R \frac{2-p}{2(1-p)} \left(\frac{1}{p} \log N + R\right)\right)$.

Next, we drop the constant probability assumption. Intuitively, the worst case scenario corresponds to either all agents having a small back-off probability, thus they keep on competing for the same resource, or all of them having a high back-off probability, thus the process will keep on restarting. These two scenarios correspond to the inner ($\frac{1}{p}$) and outer ($\frac{2-p}{2(1-p)}$) probability terms of bound (5) respectively. Let $p^* = f(\text{loss}^*)$ be the worst between the smallest or highest back-off probability any agent $n \in \mathcal{N}$ can exhibit, i.e., having loss^* given by Eq. 6. Using p^* instead of the constant p , we bound the expected convergence time according to bound (5). \square

It is worth noting that the back-off probability p^* in bound (5) does not significantly affect the convergence time. For example, using Eq. 3 with a quite small $\epsilon = 0.01$, the resulting quantities would be at most $100R \log N$, and $50R^2$. Most importantly, though, this is a rather loose bound (e.g., agents would rarely back-off with probabilities as extreme as p^*).

Apart from the convergence of the whole system, we are interested in the expected number of steps any individual agent would require in order to acquire a resource. In real-world scenarios, there is typically a cost associated with acquiring a resource. For example, a taxi driver would not be willing to drive to the other end of the city to pick up a low fare passenger. As a result, each agent is typically interested in a subset of the total resources, i.e., $\mathcal{R}^n \subset \mathcal{R}$, thus at each resource there is a bounded number of competing agents. Let R^n denote the maximum number of resources agent n is interested in, and N^r denote the maximum number of agents competing for resource r . By bounding these two quantities (i.e., we consider R^n and N^r to be constant functions of N, R), Corollary 2.2 proves that the expected number of steps any individual agent requires in order to claim a resource is independent of the total problem size (i.e., N , and R), or, in other words, that the convergence time is *constant* in these quantities.

COROLLARY 2.2. *Let $R^n = |\mathcal{R}^n|$, such that $\forall r \in \mathcal{R}^n : u_n(r) > 0$, and $N^r = |\mathcal{N}^r|$, such that $\forall n \in \mathcal{N}^r : u_n(r) > 0$. The expected number of steps until an agent $n \in \mathcal{N}$ following Alg. 1 successfully acquires a resource is bounded by (7), where $p_n^* = f(\text{loss}^*)$ and loss^* is given by Eq. 8, independent of the total problem size N, R .*

$$\mathcal{O}\left(\max_{n' \in \cup_{r \in \mathcal{R}^n} \mathcal{N}^r} \frac{2 - p_n^*}{2(1 - p_n^*)} \left(\frac{1}{p_n^*} \log(\max_{r \in \mathcal{R}^n} N^r) + \max_{n' \in \cup_{r \in \mathcal{R}^n} \mathcal{N}^r} R^{n'}\right)\right) \quad (7)$$

$$\text{loss}^* = \arg \min_{\text{loss}_n^r} \left(\min_{r \in \mathcal{R}^n, n \in \mathcal{N}^r} (\text{loss}_n^r), 1 - \max_{r \in \mathcal{R}^n, n \in \mathcal{N}^r} (\text{loss}_n^r) \right) \quad (8)$$

PROOF. The expected number of steps until an agent $n \in \mathcal{N}$ successfully acquires a resource is upper bounded by the total convergence time of the sub-system he belongs to, i.e., the sub-system consisting of the sets of \mathcal{R}^n resources and $\cup_{r \in \mathcal{R}^n} \mathcal{N}^r$ agents. In such scenario, at most $\max_{r \in \mathcal{R}^n} N^r$ agents can compete for any resource. Using Theorem 2.1 for $\max_{r \in \mathcal{R}^n} N^r$ agents, $\max_{n' \in \cup_{r \in \mathcal{R}^n} \mathcal{N}^r} R^{n'}$ resources, and worst-case loss^* given by any agent in $\cup_{r \in \mathcal{R}^n} \mathcal{N}^r$ (i.e., Eq 8) results in the desired bound. Note that agents do not compete for already claimed resources (step 8 of Alg. 1), thus the convergence of an agent does not require the convergence of agents of overlapping sub-systems. \square

3 EVALUATION

In this section we evaluate ALMA under various test cases. For the first two, we focus on convergence time and relative difference in social welfare (SW), i.e., $(\text{achieved} - \text{optimal})/\text{optimal}$. In every reported metric, except for the social welfare, we report the average value out of 128 runs of the same problem instance. Error bars represent one standard deviation (SD) of uncertainty. As a measure of social welfare, we report the cumulative regret of the aforementioned 128 runs, i.e., for $i \in [1, 128]$ runs, the reported relative difference in social welfare is $(\sum_i \text{achieved} - \sum_i \text{optimal})/\sum_i \text{optimal}$. This was done to improve visualization of the results in smaller problem sizes, where really small differences result in high SD bars (e.g., if $\text{achieved} = 1 \times 10^{-5}$, and $\text{optimal} = 2 \times 10^{-5}$, the relative difference would be -50% for practically the same matching). The optimal matchings were computed using the Hungarian algorithm¹. The third test case is an on-line setting, thus we report the achieved SW (not the relative difference to the optimal), and the empirical competitive ratio (average out of 128 runs, as before). All the simulations were run on 2x Intel Xeon E5-2680 with 256 GB RAM. In Section 3.1 we use the logistic function (Eq. 4) with $\gamma = 2$, while in Sections 3.2 & 3.3 we use the linear function (Eq. 3) with $\epsilon = 0.1$.

It is important to stress that our goal is not to improve the convergence speed of a centralized, or decentralized algorithm. Rather, the computation time comparisons of Sections 3.1 & 3.2 are meant to ground the actual speed of ALMA, and argue in favor of its applicability on large-scale, real-world scenarios. Given the nature of the problem, we elected to use a specialized algorithm to compute the optimal solution, rather than a general LP-based technique (e.g., the Simplex method). Specifically, we opted to use the Hungarian algorithm which, first, has proven polynomial worse case bound, and second, as our simulations will demonstrate, can handle sufficiently large problems.

3.1 Test Case #1: Uniform, and Noisy Common Preferences

3.1.1 Setting. As a first evaluation test case, we cover the extreme scenarios. The first pertains to an anti-coordination scenario in which agents with similar preferences compete over the same set of actions [DF18]. For example, autonomous vehicles would prefer the least congested route, bidding agents participating in multiple auctions would prefer the ones with the smallest number of participants, etc. We call this scenario ‘noisy common preferences’ and

¹We used Kevin L. Stern’s $\mathcal{O}(N^3)$ implementation: <https://github.com/KevinStern/>.

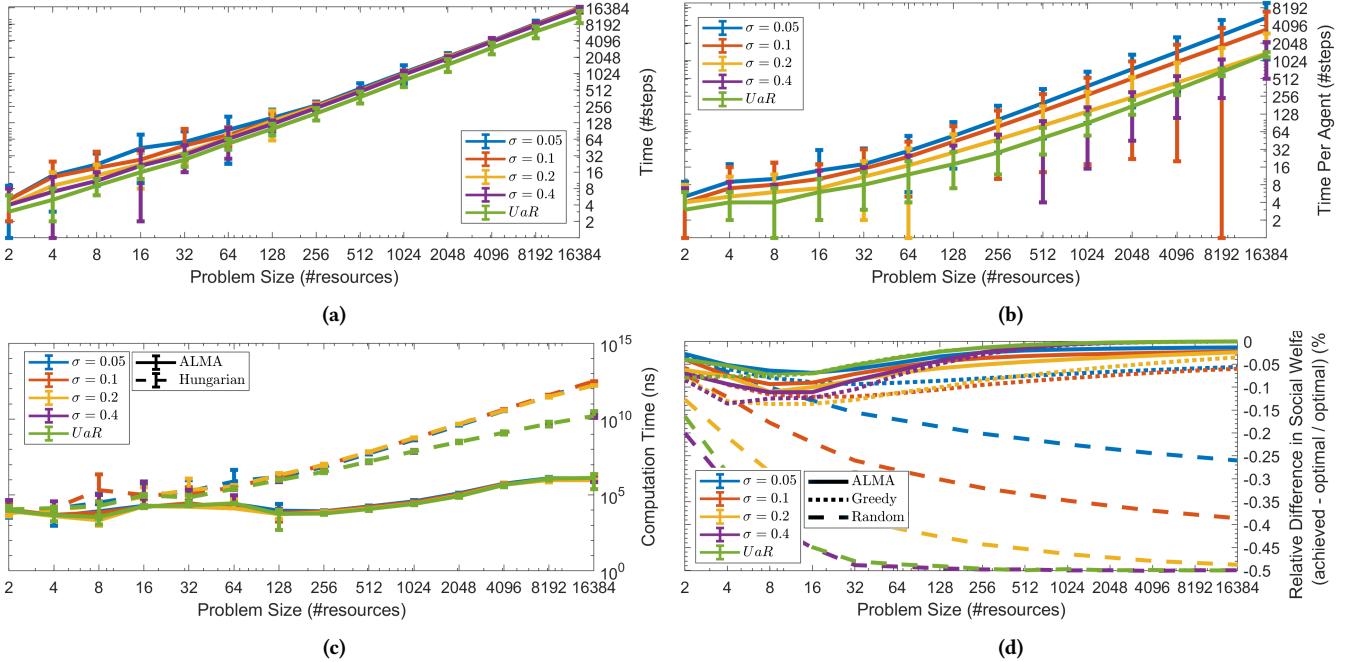


Figure 1: From left to right, top to bottom: (1a) Total convergence time (#steps), (1b) average time (#steps) for an individual agent to successfully acquire a resource, (1c) Computation time (ns), and (1d) Relative difference in social welfare (%), for increasing number of resources, and $N = R$. Fig. 1a, 1c, and 1b are in double log. scale, while Fig. 1d is in single log. scale.

model the utilities as follows: $\forall n, n' \in \mathcal{N}, |u_n(r) - u_{n'}(r)| \leq \text{noise}$, where the noise is sampled from a zero-mean Gaussian distribution, i.e., noise $\sim \mathcal{N}(0, \sigma^2)$ ². In the second scenario the utilities are initialized uniformly at random (UaR) for each agent and resource.

3.1.2 Convergence Time. Starting with Fig. 1a, we can see that the convergence time for the system of agents following Alg. 1 is linear to the number of resources R . From the perspective of a single agent, Fig. 1b shows that on average he will successfully acquire a resource significantly ($> 2\times$) faster than the total convergence time. This suggest that there is a small number of agents which take longer in claiming a resource and which in turn delay the system's convergence. We will exploit this property in the next section to present the anytime property of ALMA. Fig. 1c shows that ALMA requires approximately 4 to 6 orders of magnitude less computation time than the centralized Hungarian algorithm. Furthermore, ALMA seems to scale more gracefully, an important property for real world applications. Note also that in real-world applications we would have to take into account communication time, communication reliability protocols, etc., which create additional overhead for the Hungarian or any other algorithm for the assignment problem.

3.1.3 Efficiency. The relative difference in social welfare (Fig. 1d) reaches asymptotically zero as R increases. For a small number of resources, ALMA achieves the worst social welfare, approximately 11% worse than the optimal. Intuitively this is because when we have a small number of choices, a single wrong matching can have a significant impact to the final social welfare, while as the number of

resources grow, the impact of an erroneous matching is mitigated. For 16384 resources we lose less than 2.5% of the optimal. As a reference, Fig. 1d depicts the centralized greedy, and the random solutions as well. The greedy solution goes through the participating agents randomly, and assigns them their most preferred unassigned resource. In this scenario, the random solution loses up to 50% of the optimal SW, while the greedy solution achieves similar results to ALMA, especially in high noise situations. This is to be expected, since first, all agents are interested in all the resources, and second, as the noise increases, the agents' preferences become more distinguishable, more diverse. ALMA is of a greedy nature as well, albeit it utilizes a more intelligent back-off scheme. Contrary to that, the greedy solution does not take into account the utilities between agents, thus there are scenarios where ALMA would significantly outperform the greedy (e.g., see Section 3.3). Finally, recall that ALMA operates in a significantly harder domain with no communication, limited feedback, and time constraints. In contrast, the greedy method requires either a central coordinator or message exchange (to communicate users' preferences and resolve collisions).

3.2 Test Case #2: Resource Allocation in a Cartesian Map with Manhattan Distances

Adopting a simple rule allows the applicability of ALMA to large scale multi-agent systems. In this section we will analyze such a scenario. Specifically we are interested in resource allocation in urban environments (e.g., parking spots / charging stations for

²Similar results achieved using uniform noise, i.e., $\sim \mathcal{U}(-v, v)$.

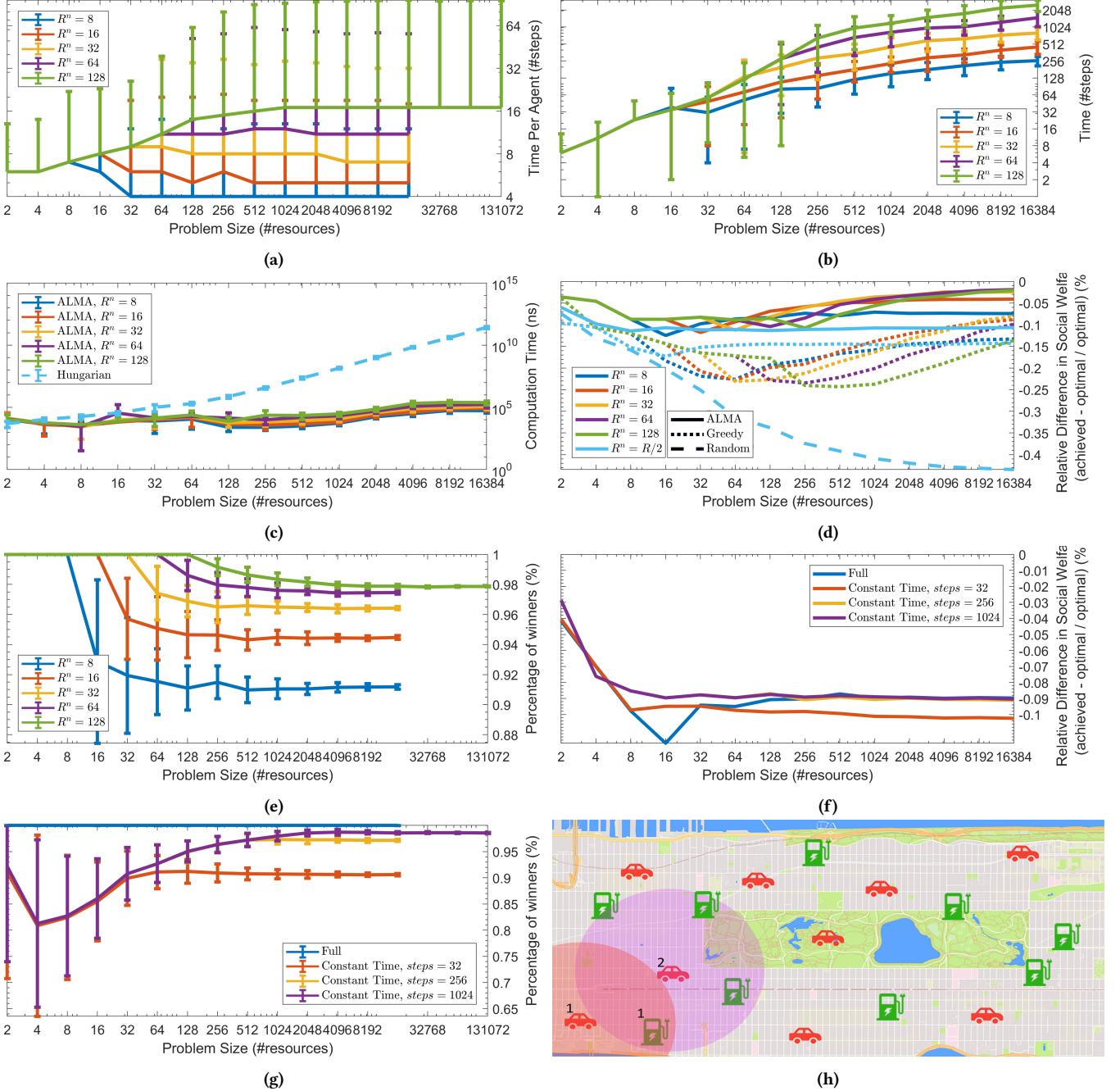


Figure 2: Left to right, top to bottom: (2a) Average time (#steps) for an agent to acquire a resource, (2b) Total convergence time (#steps), (2c) Computation time (ns), (2d) Relative difference in SW (%), (2e) Percentage of 'winners', (2f) Relative difference in SW (%) in interrupted execution, (2g) Percentage of 'winners' in interrupted execution. The aforementioned are for increasing number of resources, and $N = R$. Fig. (2a), (2b), and (2c) are in double log scale, while the rest are in single log scale. Fig. (2h) presents an example of the studied resource allocation scenario in an urban environment. We assume grid length of $\sqrt{4 \times N}$.

autonomous vehicles, taxi - passenger matchings, etc.). The aforementioned problems become ever more relevant due to rapid urbanization, and the natural lack of coordination in the usage of resources [Var16]. The latter result in the degradation of response (e.g., waiting time) and quality metrics in large cities [Var16].

3.2.1 Setting. Let us consider a Cartesian map representing a city on which are randomly distributed vehicles and charging stations, as depicted in Fig. 2h. The utility received by a vehicle n for using a charging station r is proportional to the inverse of their distance, i.e., $u_n(r) = 1/d_{n,r}$. Since we are in an urban environment, let $d_{n,r}$ denote the Manhattan distance. Typically, there is a cost each agent is willing to pay to drive to a resource, thus there is a cut-off distance, upon which the utility of acquiring the resource is zero (or possibly negative). This is a typical scenario encountered in resource allocation in urban environments, where there are spatial constraints and local interactions.

The way such problems are typically tackled, is by dividing the map to sub-regions, and solving each individual sub-problem. For example, Singapore is divided into 83 zones based on postal codes [CN11], and taxi drivers' policies are optimized according to those [NKL17, VCGA12]. On the other hand, not placing bounds means that the current solutions will not scale. To the best of our knowledge, we are the first to propose an anytime heuristic for resource allocation in urban environments that can scale in constant time without the need to artificially split the problem. Instead, ALMA exploits the two typical characteristics of an urban environment: the anonymity in interactions and homogeneity in supply and demand [Var16] (e.g., assigning any of two equidistant charging stations to a vehicle, would typically result to the same utility). This results in a simple learning rule which, as we will demonstrate in this section, can scale to hundreds of thousands of agents.

3.2.2 Convergence Time. To demonstrate the latter, we placed a bound on the maximum number of resources each agent is interested in, and on the maximum number of agents competing for a resource, specifically $R^n = N^r \in \{8, 16, 32, 64, 128\}$. According to Corollary 2.2, bounding these two quantities should result in convergence in constant time, regardless of the total problem size (R, N). The latter is corroborated by Fig. 2a, which shows that the average number of time-steps until an agent successfully claims a resource remains constant as we increase the total problem size. Same is true for the system's convergence time (Fig. 2b), which caps as R increases. The small increase is due to outliers, as Fig. 2b reports the convergence time of the last agent. This results to approximately 7 orders of magnitude less computation time than the centralized Hungarian algorithm (Fig. 2c), and this number would grow boundlessly as we increase the total problem size. Moreover, as mentioned, in an actual implementation, any algorithm for the assignment problem would face additional overhead due to communication time, reliability protocols, etc.

3.2.3 Efficiency. Along with the constant convergence time, ALMA is able to reach high quality matchings, achieving less than 7.5% worse social welfare (SW) than the optimal (Fig. 2d). The latter refers to the small bound of $R^n = 8$. As observed in Section 3.1, with a small number of choices, a single wrong matching can have a significant impact to the final social welfare. By increasing

the bound to a more realistic number (e.g., $R^n = 32$), we achieve less than 2.5% worse SW. In general, for $R > 2$ resources and different values of R^n , ALMA achieves between 1.9 – 12% loss in SW, while the greedy approach achieves 8.0 – 24% and the random 7.3 – 43.4%. The behavior of the graphs depicted in Fig. 2d for $R^n \in \{8, 16, 32, 64, 128\}$ indicate that, as the problem size (R) increases, the social welfare reaches its lowest value at $R = 2 \times R^n$. To investigate the latter, we have included a graph for increasing R^n (instead of constant to the problem size), specifically $R^n = R/2$. ALMA achieves a constant loss in social welfare (approximately 11%). The greedy approach achieves loss of 14%, while the random solution degrades towards 44% loss.

Compared to Test Case #1, this is a significantly harder problem for a decentralized algorithm with no communication and no global knowledge of the resources. The set of resources each agent is interested in is a proper subset of the set of the total resources, i.e., $\mathcal{R}^n \subsetneq \mathcal{R}$ (or could be $R < N$). Furthermore, the lack of communication between the participants, and the stochastic nature of the algorithm can lead to deadlocks, e.g., in Fig. 2h, if vehicle 2 acquires resource 1, then vehicle 1 does not have an available resource in range. Nonetheless, ALMA results in an almost complete matching. Fig. 2e, depicts the percentage of 'winners' (i.e., agents that have successfully claimed a resource r such that $u_n(r) > 0$). The aforementioned percentage refers to the total population (N) and not the maximum possible matchings (potentially $< N$). As depicted, the percentage of 'winners' is more than 90%, reaching up to 97.8% for $R^n = 128$. We also employed ALMA in larger simulations with up to 131072 agents, and equal resources. As seen in Fig. 2e, the percentage of winners remains stable at around 98%. Even though the size of the problem prohibited us from running the Hungarian algorithm (or an out-of-the-box LP solver) and validating the quality of the achieved matching, the fact that the percentage of winners remains the same suggests that the relative difference in SW will continue on the same trend as in Fig. 2d. Moreover, the average steps per agent to claim a resource remains, as proven, constant (Fig. 2a). The latter validate the applicability of ALMA in large scale applications with hundreds of thousands of agents.

3.2.4 Anytime Property. In the real world, agents are required to run in real time, which imposes time constraints. ALMA can be used as an anytime heuristic as well. To demonstrate the latter, we compare four configurations: the 'full' one, which is allowed to run until the systems converges, and three 'constant time' versions which are given a time budget of 32, 256, and 1024 time-steps. In this scenario, we do not impose a bound on R^n, N^r , but we assume a cut-off distance, upon which the utility is zero. The cut-off distance was set to 0.25 of the maximum possible distance, i.e., as the problem size grows, so do the R^n, N^r . On par with Test Case #1, the full version converges in linear time. As depicted in Fig. 2f, the achieved SW is less than 9% worse than the optimal. The inferior results in terms of SW compared to Test Case #1 are because this is a significantly harder problem due to the aforementioned deadlocks. On the other hand though, ALMA benefits from the spatial constraints of the problem. The average number of time-steps an individual agent needs to successfully claim a resource is significantly smaller, which suggest that we can enforce computation time constraints. Restricting to only 32, 256, and 1024 time-steps, results in 1.25%, 0.12%,

and 0.03% worse SW than the unrestricted version, respectively. Even in larger simulations with up to 131072 agents, the percentage of winners (Fig. 2g) remains stable at 98.6%, which suggests that the relative difference in SW will continue on the same trend as in Fig. 2f (we do not suggest that this is the case in any domain. For example, in the noisy common preferences domain of Test Case #1, the quality of the achieved matching decreases boundlessly as we decrease the allotted time. Nevertheless, the aforescribed domain is a realistic one, with a variety of real-world applications). Finally, the repeated nature of such problems suggests that even in the case of a deadlock, the agent which failed to win a resource, will do so in some subsequent round.

3.3 Test Case #3: On-line Taxi Request Match

In this section we present a motivating test case involving ride-sharing, via on-line taxi request matching, using *real* data of taxi rides in New York City. Ride-sharing (or carpooling), offers great potential in congestion relief and environmental sustainability in urban environments. In the past few years, several commercially successful ride-sharing companies have surfaced (e.g., Uber, Lyft, etc.), giving rise to a new incarnation of ride-sharing: dynamic ride-sharing, where passengers are matched in real-time. Ride-sharing, though, results to some passenger disruption due to loss in flexibility, security concerns, etc. Compensation comes in the form of monetary incentives, as it allows passengers to share the travel expenses, and thus reduce the cost. Ride-sharing companies account for a plethora of factors, like current demand, prime time pricing, the cost of the route without any ride-sharing, the likelihood of a match, etc. Yet, a fundamental factor of the cost of any shared ride, no matter if it is a company or a locally-organized car sharing scheme, is the traveled distance.

In this test case, we attempt to maximize the total distance saved, by matching taxi requests of high overlap. Fig. 3 provides an illustrative example. There are two passengers (depicted as yellow and red) with high overlap routes. Each can drive on their own to their respective destinations (dashed yellow and red line respectively), or share a ride (green line) and reduce travel costs.

Dynamic ride-sharing is an inherently *on-line* setting, as a matching algorithm is unaware of the requests that will appear in the future and needs to make decisions for the requests before they ‘expire’ (a similar setting was studied in [ABD⁺18]). ALMA is highly befitting for such a scenario, as it involves large-scale matchings under dynamic demand, it is highly decentralized, and partially observable.

3.3.1 Setting. We use a dataset³ of all taxi requests (ρ) in New York City during one week ({01-01-16 0:00 - 01-07-16 23:59}, 34077 requests in total). The data include pickup and drop-off times, and geolocations. Requests appear (become open) at their respective pickup time, and wait k_ρ time-steps to find a match. Let a time-step be one minute. After k_ρ time-steps we call request ρ , critical. If a critical request is not matched, we assume they drive off to their destination in a single passenger ride. Let *open*, *critical* denote the sets of open, and critical requests respectively, and let *current* = *open* \cup *critical*. To compute k_ρ we assume the following: There is



Figure 3: Example of the studied taxi request matching scenario.

a minimum $minW$, and a maximum $maxW$ waiting time set by the ride-sharing company, i.e., $minW \leq k_\rho \leq maxW, \forall \rho$. Moreover, since each passenger specifies his destination in advance, we can compute the trip time (l_ρ). Assuming people are willing to wait for a time that is proportional to their trip time, let $k_\rho = q \times l_\rho$, where $q \in [0, 1]$. The parameters $minW$, $maxW$, and q can be set by the ride-sharing company. We report results on different values for all of the above parameters. For each pair ρ_1, ρ_2 of requests, we compute the driving distance ($d_{\rho_1, \rho_2} = \min$ of all possible combinations of driving between ρ_1, ρ_2 ’s pickup and drop-off locations) that would be traveled if ρ_1, ρ_2 are matched, i.e., if they share the same taxi. Subsequently, the utility of matching ρ_1 to ρ_2 (distance saved) is $u_{\rho_1}(\rho_2) = d_{\rho_1, \rho_2}$ (km).

Given the on-line nature of the setting, it might be beneficial to use the following *non-myopic heuristic*: avoid matching low utility pairs, as long as the requests are not critical, since more valuable pairs might be presented in the future. Thus, if $u_{\rho_1}(\rho_2) < d_{min}$, and $\rho_1, \rho_2 \notin \text{critical}$, we do not match ρ_1, ρ_2 . In what follows, we select for each algorithm and for each simulation the value $d_{min} \in \{0, 500, 1000, 1500, 2000, 2500\}$ that results in the highest score. To compute the actual trip time, and driving distance, we have used the Open Source Routing Machine⁴, which computes shortest paths in road networks.

3.3.2 Computation of the optimal matching. In this scenario, each request has a dual role, being both an agent and a resource, i.e., we have a non-bipartite graph. For computing the optimal (maximum weight) matching for the employed on-line heuristics, we use the blossom algorithm of [Edm65], which computes a maximum weight matching in general graphs. This enables us to compute the best possible matching among current requests, i.e., requests that have not ‘expired’ at the time of the computation. In fact, the following observation allows us to compute the optimal *off-line* matching as well, i.e., the best possible matching over the whole time interval. Let a_ρ be the pick-up time of request ρ and let $e_\rho = a_\rho + k_\rho$ be the time when it becomes critical. Then, we can redefine the utility of a matching as:

$$u_{\rho_1}(\rho_2) = \begin{cases} d_{\rho_1, \rho_2}, & \text{if } a_i \leq e_j \text{ and } e_i \leq e_j \\ -1, & \text{otherwise} \end{cases} \quad (9)$$

i.e., the utility is the distance saved if both requests are simultaneously *active* when matched and -1 otherwise. The latter effectively results in this pair never being matched in an optimal solution. We remark that this clairvoyant matching is not feasible in the on-line setting and serves as a benchmark against which we can compare

³<https://www.kaggle.com/debanjanpaul/new-york-city-taxi-trip-distance-matrix/>

⁴<http://project-osrm.org/>

the performance of on-line algorithms, as is common in the literature of competitive analysis [BEY05]. The measure of efficiency, as compared to the off-line optimal, will be the *empirical competitive ratio*, i.e., the ratio of the social welfare of the on-line algorithm over the welfare of the optimal, as measured empirically for our dataset.

3.3.3 The Blossom Algorithm vs Linear Programming. Contrary to the other test cases, the fact that the graph is not bipartite has implications on how linear programming can be used to compute the optimal solution. In particular, in all of the presented test cases (#1, #2, and #3), one can formulate the problem of computing the optimal solution as an Integer Linear Program (ILP) and then solve it using some general solver like CPLEX⁵. Yet, the computational complexity of solving the aforementioned ILP varies amongst the different test cases.

Solving integer linear programs is generally quite computationally demanding. Thus, it is common to resort to solving the *LP relaxation* (where the integrality constraints have been ‘relaxed’ to fractional constraints), which can be computed in polynomial time.

In the case of bipartite graphs (such as test cases #1 and #2), the standard LP relaxation admits integer solutions, i.e., solutions to the actual maximum weight bipartite matching problem (because the constraint matrix is totally unimodular). In the case of general (non-bipartite) graphs, this is no longer the case, thus we have to resort to solving the *LP relaxation*. It is known that the (fractional) optimal solution to the LP relaxation might be better than the (integral) optimal solution to the original ILP formulation (i.e., it has an *integrality gap* which is larger than 1, in fact 2). In other words, solving the LP relaxation will not provide solutions for the maximum weight matching problem but for an ‘easier version’ with fractional matchings. Thus, comparing against that solution could only give very pessimistic ratios, when the real empirical competitive ratios are much better.

One can derive a different ILP formulation of the problem using more involved constraints (called ‘blossom’ constraints [Edm65]), whose relaxation admits integer solutions (i.e., the integrality gap is now 1). However, the latter would result in an exponential number of constraints, and one would need to employ the Ellipsoid method with an appropriately chosen separation oracle to solve it in polynomial time (see [FOW02, page 4] for more details]. Overall, the employment of the combinatorial algorithm of [Edm65] for finding the maximum weight matching is a cleaner and more efficient solution.

3.3.4 Benchmarks. Each request runs ALMA independently. ALMA waits until the request becomes critical, and then matches it by running Alg. 1, where $\mathcal{N} = \text{critical}$, and $\mathcal{R} = \text{current}$. In this non-bipartite scenario, if an agent is matched under his dual role as a resource, he is immediately removed. As we explained earlier, it is infeasible for an on-line algorithm to compute the optimal matching over the whole period of time. Instead, we consider *just-in-time* and *in batches* optimal solutions. Specifically, we compare to the following [AESW11, ABD⁺18]:

⁵<https://www.ibm.com/analytics/cplex-optimizer>

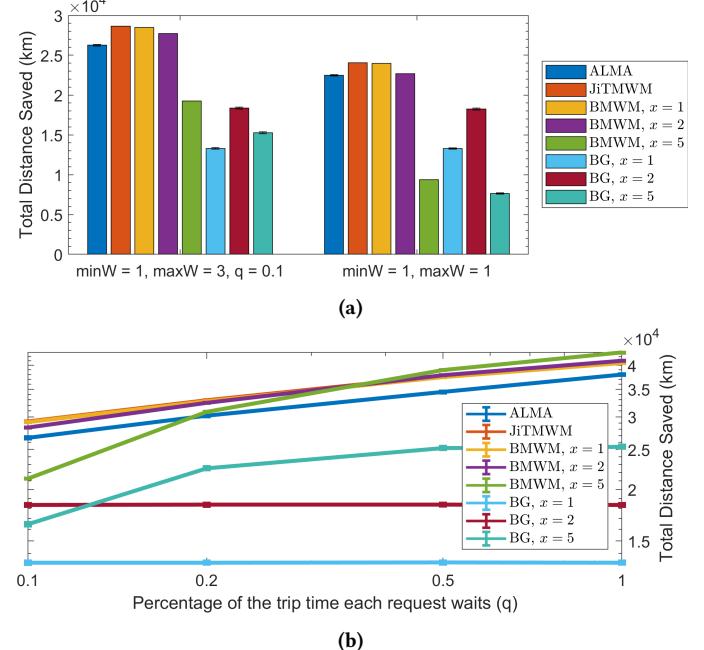


Figure 4: Total distance saved (km) for various values of $\text{minW}, \text{maxW}, q$. (4a, left) Pragmatic scenario, (4a, right) Requests become critical in just one time-step, (4b) Various levels of waiting time (q) assuming no bounds, i.e., $\text{minW} = 0, \text{maxW} = \infty$, (double log. scale).

- **Just-in-time Max Weight Matching (Jitmwm):** Waits until a request becomes critical and then computes a maximum-weight matching of all the current requests, i.e $\mathcal{N} = \mathcal{R} = \text{current}$.
- **Batching Max Weight Matching (BMWm):** Waits x time-steps and then computes a maximum-weight matching of all the current requests, i.e $\mathcal{N} = \mathcal{R} = \text{current}$.
- **Batching Greedy (BG):** Waits x time-steps and then greedily matches current requests, i.e $\mathcal{N} = \mathcal{R} = \text{current}$ (ties are broken randomly). Unmatched open requests are removed. For batch size $x = 1$ we get the simple greedy approach where requests are matched as soon as they appear.

3.3.5 Efficiency. Starting with the social welfare, Fig. 4 presents the total distance saved (km) for various values of minW , maxW , and q . ALMA loses 8.3% of SW in the pragmatic scenario of Fig. 4a (left), and 6.5% when the requests become critical in just one time-step (Fig. 4a (right)). If no bounds are placed on the minimum and maximum waiting time (i.e., $\text{minW} = 0, \text{maxW} = \infty$), ALMA exhibits loss of 8 – 11.5%, for various values of q (Fig. 4b). The above are compared to the best performing benchmark on each scenario (Jitmwm, or BMWm, $x = 5$). Moreover, it significantly outperforms every greedy approach. In the first scenario the BGs lose between 35.8 – 53.5%, in the second between 24 – 68.2%, and in the third between 31.5 – 69%.

Once more, it is worth noting that ALMA requires just a broadcast of a single bit to indicate the occupancy of a resource, while the

Table 1: Empirical Competitive Ratio.

$(minW, maxW, q)$	ALMA	JiTWMW	BMWWM, $x = 1$	BMWWM, $x = 2$	BMWWM, $x = 5$	BG, $x = 1$	BG, $x = 2$	BG, $x = 5$
(1, 3, 0.1)	0.7890	0.8621	0.8568	0.8283	0.5883	0.3991	0.5422	0.4603
(1, 1, -)	0.8491	0.9243	0.9190	0.8663	0.3810	0.5112	0.6891	0.3038
(0, ∞ , 0.1)	0.7835	0.8528	0.8486	0.8211	0.6221	0.3900	0.5299	0.4840
(0, ∞ , 0.2)	0.7546	0.8207	0.8158	0.8000	0.7616	0.3399	0.4604	0.5688
(0, ∞ , 0.5)	0.6939	0.7439	0.7368	0.7448	0.7668	0.2731	0.3714	0.5133
(0, ∞ , 1.0)	0.6695	0.7390	0.7254	0.7343	0.7706	0.2440	0.3306	0.4606

compared approaches require either *message exchange* for sharing the utility table, or the use of a *centralized authority*. For example, the greedy solution would require message exchange to communicate users' preferences and resolve collisions in a decentralized setting, and every batching approach would require a common centralized synchronization clock.

Table 1 presents the empirical competitive ratio for the first day of the week of the employed dataset. As we can see, even in the extreme, unlikely scenarios where we assume that people would be willing to wait for more than 10 or 20 minutes (which correspond to large values of q), ALMA achieves high relative efficiency, compared to the off-line (infeasible) benchmark. These scenarios are favorable to the off-line optimal, because requests stay longer in the system and therefore the algorithm takes more advantage of its foreseeing capabilities (hence, the drop in the competitive ratios is observed in *all* of the employed algorithms). In particular, ALMA achieves an empirical competitive ratio of 0.67 for $q = 1$ and even better ratios for more realistic scenarios (as large as 0.85). The just-in-time and batch versions of the maximum weight matching perform slightly better, but this is to be expected, as they compute the maximum weight matching on the graphs of the current requests. In spite of the unpredictability of the on-line setting, and the dynamic nature of the demand, ALMA is consistently able to exhibit high performance, in all of the employed scenarios.

4 CONCLUSION

Algorithms for solving the assignment problem, whether centralized or distributed, have runtime that increases with the total problem size, even if agents are interested in a small number of resources. Thus, they can only handle problems of some bounded size. Moreover, they require a significant amount of inter-agent communication. Humans on the other hand are routinely called upon to coordinate in large scale in their everyday lives, and are able to fast and robustly match with resources under dynamic and unpredictable demand. One driving factor that facilitates human cooperation is the principle of altruism. Inspired by human behavior, we have introduced a novel anytime heuristic (ALMA) for weighted matching in both bipartite and non-bipartite graphs. ALMA is decentralized and requires agents to only receive partial feedback of success or failure in acquiring a resource. Furthermore, the running time of the heuristic is constant in the total problem size, under reasonable assumptions on the preference domain of the agents. As autonomous agents proliferate (e.g., IoT devices, intelligent infrastructure, autonomous vehicles, etc.), having robust algorithms

that can scale to hundreds of thousands of agents is of utmost importance.

The presented results provide an empirical proof of the high quality of the achieved solution in a variety of scenarios, including both synthetic and *real* data, time constraints and on-line settings. Furthermore, both the proven theoretical bound (which guarantees constant convergence time), and the computation speed comparison (which grounds ALMA to a proven fast centralized algorithm), argue for its applicability to large scale, real world problems. As future work, it would be interesting to identify meaningful domains in which ALMA can provide provable worst-case performance guarantees, as well as to empirically evaluate its performance on other real datasets, corresponding to important real-world, large-scale problems.

REFERENCES

- [ABD⁺18] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. Maximum weight online matching with deadlines. *arXiv preprint arXiv:1808.03526*, 2018.
- [AESW11] Niels Agatz, Alan L Erera, Martin WP Savelsbergh, and Xing Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Procedia-Social and Behavioral Sciences*, 17:532–550, 2011.
- [Ber79] DP Bertsekas. A distributed algorithm for the assignment problem. *Laboratory for Information and Decision Systems Working Paper, Massachusetts Institute of Technology, Cambridge, MA*, 1979.
- [BEY05] Allan Borodin and Ron El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.
- [BJ65] CW Borchardt and CGJ Jacobi. De investigando ordine systematis aequationum differentialium vulgarium cujuscunque. *Journal für die reine und angewandte Mathematik*, 64:297–320, 1865.
- [BNBA12] Mathias Bürger, Giuseppe Notarstefano, Francesco Bullo, and Frank Allgöwer. A distributed simplex algorithm for degenerate linear programs and multi-agent assignments. *Automatica*, 48(9):2298–2304, 2012.
- [CF11] Ludek Cigler and Boi Faltings. Reaching correlated equilibria through multi-agent learning. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 509–516. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- [CN11] S. F. Cheng and T. D. Nguyen. Taxisim: A multiagent simulation platform for evaluating taxi fleet operations. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 14–21, Aug 2011.
- [CR02] Gary Charness and Matthew Rabin. Understanding social preferences with simple tests*. *The Quarterly Journal of Economics*, 117(3):817–869, 2002.
- [Dan90] George B Dantzig. *Origins of the simplex method*. ACM, 1990.
- [DF18] Panayiotis Danassis and Boi Faltings. Learning in ad-hoc anti-coordination scenarios. *AAAI Spring Symposium Series*, 2018.
- [DF19] Panayiotis Danassis and Boi Faltings. Courtesy as a means to coordinate. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19. International Foundation for Autonomous Agents and Multiagent Systems, 2019.
- [Edm65] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.
- [EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972.

- 1972.
- [FOW02] Uriel Feige, Eran Ofek, and Udi Wieder. Approximating maximum edge coloring in multigraphs. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 108–121. Springer, 2002.
 - [GA13] Tyler Gunn and John Anderson. Dynamic heterogeneous team formation for robotic urban search and rescue. *Procedia Computer Science*, 19:22 – 31, 2013. The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013).
 - [GC13] Yanfeng Geng and Christos G Cassandras. New “smart parking” system based on resource allocation and reservations. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1129–1139, 2013.
 - [Gin00] Herbert Gintis. Strong reciprocity and human sociality. *Journal of theoretical biology*, 206(2):169–179, 2000.
 - [GLM10] Stefano Giordani, Marin Lujak, and Francesco Martinelli. A distributed algorithm for the multi-robot task allocation problem. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 721–730. Springer, 2010.
 - [IS17] S. Ismail and L. Sun. Decentralized hungarian-based approach for fast and scalable task allocation. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 23–28, June 2017.
 - [Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.
 - [Lew08] David Lewis. *Convention: A philosophical study*. John Wiley & Sons, 2008.
 - [LP09] László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.
 - [Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.
 - [NKL17] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Collective multiagent sequential decision making under uncertainty. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
 - [Nor98] James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.
 - [NS05] Martin A Nowak and Karl Sigmund. Evolution of indirect reciprocity. *Nature*, 437(7063):1291, 2005.
 - [Oll09] François Ollivier. Looking for the order of a system of arbitrary ordinary differential equations. *Applicable Algebra in Engineering, Communication and Computing*, 20(1):7–32, 2009.
 - [PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.
 - [Reg92] Vernon Rego. Naive asymptotics for hitting time bounds in markov chains. *Acta Informatica*, 29(6):579–594, 1992.
 - [Sim16] Jay Simon. On the existence of altruistic value and utility functions. *Theory and Decision*, 81(3):371–391, 2016.
 - [SKKR10] Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, July 2010.
 - [Su15] Hsin-Hao Su. Algorithms for fundamental problems in computer networks. 2015.
 - [Tal13] Mohammad Sadegh Talebi. Uncoupled learning rules for seeking equilibria in repeated plays: An overview. *CoRR*, abs/1310.5660, 2013.
 - [Var16] Pradeep Varakantham. Sequential decision making for improving efficiency in urban environments. 2016.
 - [VCGA12] Pradeep Reddy Varakantham, Shih-Fen Cheng, Geoff Gordon, and Asrar Ahmed. Decision support for agent populations in uncertain and congested environments. 2012.
 - [ZSP08] Michael M Zavlanos, Leonid Spesivtsev, and George J Pappas. A distributed auction algorithm for the assignment problem. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 1212–1217. IEEE, 2008.

A APPENDIX

A.1 Proof of Theorem 2.1

Theorem 2.1. For N agents and R resources, the expected number of steps until the system of agents following Alg. 1 converges to a complete matching is bounded by (10), where $p^* = f(\text{loss}^*)$ and loss^* is given by Eq. 11.

$$O\left(R \frac{2-p^*}{2(1-p^*)} \left(\frac{1}{p^*} \log N + R\right)\right) \quad (10)$$

$$\text{loss}^* = \arg \min_{\text{loss}_n^r} \left(\min_{r \in \mathcal{R}, n \in \mathcal{N}} (\text{loss}_n^r), 1 - \max_{r \in \mathcal{R}, n \in \mathcal{N}} (\text{loss}_n^r) \right) \quad (11)$$

In this section we provide a formal proof of Theorem 2.1.⁶ To facilitate the proof, we will initially assume that every agent, on every collision, backs-off with the same constant probability, i.e.:

$$P_n(r, \prec_n) = p > 0, \forall n \in \mathcal{N}, \forall r \in \mathcal{R} \quad (12)$$

A.1.1 Case #1: Multiple Agents, Single resource ($R = 1$). We will describe the execution of the proposed learning rule as a discrete time Markov chain (DTMC)⁷. In every time-step, each agent performs a Bernoulli trial with probability of ‘success’ $1 - p$ (remain in the competition), and failure p (back-off). When N agents compete for a single resource, a state of the system is a vector $\{0, 1\}^N$ denoting the individual agents that still compete for that resource. But, since the back-off probability is the same for everyone (Eq. 12), we are only interested in how many agents are competing and not which ones. Thus, in the single resource case ($R = 1$), we can describe the execution of the proposed algorithm using the following chain:

Definition A.1. Let $\{X_t\}_{t \geq 0}$ be a DTMC on state space $S = \{0, 1, \dots, N\}$ denoting the number of agents still competing for the resource. The transition probabilities are as follows:

$$\begin{aligned} Pr(X_{t+1} = N | X_t = 0) &= 1 && \text{restart} \\ Pr(X_{t+1} = 1 | X_t = 1) &= 1 && \text{absorbing} \\ Pr(X_{t+1} = j | X_t = i) &= \binom{i}{j} p^{i-j} (1-p)^j && i > 1, j \leq i \end{aligned}$$

(all the other transition probabilities are zero)

Intuitively, this Markov chain describes the number of individuals in a decreasing population, but with two caveats: The goal (absorbing state) is to reach a point where only one individual remains, and if we reach zero, we restart.

Before proceeding with Theorem 2.1’s convergence proof, we will restate Mityushin’s Theorem [Reg92] for hitting time bounds in Markov chains, define two auxiliary DTMCs, and prove two auxiliary lemmas.

THEOREM A.2. (Mityushin’s Theorem [Reg92]) Let $A = \{0\}$ be the absorbing state of a Markov chain $\{X_t\}_{t \geq 0}$. If $\mathbb{E}(X_{t+1}|X_t = i) < \frac{i}{\beta}$, $\forall i \geq 1$ and some $\beta > 1$, then:

⁶The proof is an adaptation of the convergence proof of [CF11] and [DF19].
⁷For an introduction on Markov chains see [Nor98].

$$\mathbb{E}(T_i^A) < \lceil \log_\beta i \rceil + \frac{\beta}{\beta - 1} \quad (13)$$

where T_i^A denotes the hitting time of a state in A , starting from state i .

Definition A.3. Let $\{Y_t\}_{t \geq 0}$ be a DTMC on state space $S = \{0, 1, \dots, N\}$ with the following transition probabilities (two absorbing states, 0 and 1):

$$\begin{aligned} Pr(Y_{t+1} = 0 | Y_t = 0) &= 1 && \text{absorbing} \\ Pr(Y_{t+1} = 1 | Y_t = 1) &= 1 && \text{absorbing} \\ Pr(Y_{t+1} = j | Y_t = i) &= \binom{i}{j} p^{i-j} (1-p)^j && i > 1, j \leq i \end{aligned}$$

(all the other transition probabilities are zero)

Definition A.4. Let $\{Z_t\}_{t \geq 0}$ be a DTMC on state space $S = \{0, 1, \dots, N\}$ with the following transition probabilities (state 0 the only absorbing state):

$$\begin{aligned} Pr(Z_{t+1} = 0 | Z_t = 0) &= 1 && \text{absorbing} \\ Pr(Z_{t+1} = j | Z_t = i) &= \binom{i}{j} p^{i-j} (1-p)^j && i \geq 1, j \leq i \end{aligned}$$

(all the other transition probabilities are zero)

LEMMA A.5. The expected hitting time of the set of absorbing states $A = \{0\}$, starting from state $Z_0 = N$, of the DTMC $\{Z_t\}$ of Definition A.4 is bounded by $O\left(\frac{1}{p} \log N\right)$.

PROOF. If the DTMC $\{Z_t\}$ is in state $Z_t = i$, the next state Z_{t+1} is drawn from a binomial distribution with parameters $(i, 1-p)$. Thus, the expected next state is $\mathbb{E}(Z_{t+1}|Z_t = i) = i(1-p)$. Using Theorem A.2 with $\beta = \frac{1}{1-p}$ results in the required bound:

$$\mathbb{E}(T_N^A) = O\left(\frac{1}{p} \log N\right) \quad (14)$$

□

COROLLARY A.6. The expected hitting time of the set of absorbing states $A = \{0, 1\}$, starting from state $Y_0 = N$, of the DTMC $\{Y_t\}$ of Definition A.3 is bounded by $O\left(\frac{1}{p} \log N\right)$.

PROOF. The expected hitting time of the absorbing state of $\{Z_t\}$ is an upper bound of the expected hitting time of $\{Y_t\}$. This is because any path that leads into state 0 in $\{Z_t\}$ either does not go through state 1 (thus happens with the same probability as in $\{Y_t\}$), or goes through state 1. But, state 1 in $\{Y_t\}$ is an absorbing state, hence in the latter case the expected hitting time for $\{Y_t\}$ would be one step shorter. □

Let h_i^A denote the hitting probability of a set of states A , starting from state i . We will prove the following lemma.

LEMMA A.7. *The hitting probability of the absorbing state $\{1\}$, starting from any state $i \geq 1$, of the DTMC $\{Y_t\}$ of Definition A.3 is given by Eq. 15. This is a tight lower bound.*

$$h_i^{\{1\}} = \Omega\left(\frac{2(1-p)}{2-p}\right), \forall i \geq 1 \quad (15)$$

PROOF. For simplicity we denote $h_i \triangleq h_i^{\{1\}}$. We will show that for $p \in (0, 1)$, $h_i \geq \lambda = \frac{2(1-p)}{2-p}$, $\forall i \geq 1$ using induction. First note that since state $\{0\}$ is an absorbing state, $h_0 = 0$, $h_1 = 1 \geq \lambda$ and that $\lambda \in (0, 1)$.

The vector of hitting probabilities $h^A = (h_i^A : i \in S = \{0, 1, \dots, N\})$ for a set of states A is the minimal non-negative solution to the system of linear equations 16:

$$\begin{cases} h_i^A = 1, & \text{if } i \in A \\ h_i^A = \sum_{j \in S} p_{ij} h_j^A, & \text{if } i \notin A \end{cases} \quad (16)$$

By replacing p_{ij} with the probabilities of Definition A.3, the system of equations 16 becomes:

$$\begin{cases} h_i^A = 1, & \text{if } i \in A \\ h_i^A = \sum_{j=0}^i \binom{i}{j} p^{i-j} (1-p)^j h_j^A, & \text{if } i \notin A \end{cases} \quad (17)$$

Base case:

$$\begin{aligned} h_2 &= (1-p)^2 h_2 + 2p(1-p)h_1 + p^2 h_0 = \frac{2p(1-p)}{1-(1-p)^2} \\ &= \frac{2(1-p)}{2-p} \geq \lambda \end{aligned}$$

Inductive step: We assume that $\forall j \leq i-1 \Rightarrow h_j \geq \lambda$. We will prove that $h_i \geq \lambda, \forall i > 2$.

$$\begin{aligned} h_i &= \sum_{j=0}^i \binom{i}{j} p^{i-j} (1-p)^j h_j \\ &= p^i h_0 + ip^{i-1} (1-p) h_1 + \sum_{j=2}^{i-1} \binom{i}{j} p^{i-j} (1-p)^j h_j \\ &\quad + (1-p)^i h_i \\ &\geq p^i h_0 + ip^{i-1} (1-p) h_1 + \sum_{j=2}^{i-1} \binom{i}{j} p^{i-j} (1-p)^j \lambda \\ &\quad + (1-p)^i h_i \\ &= ip^{i-1} (1-p) + [1 - p^i - (1-p)^i - ip^{i-1} (1-p)] \lambda \\ &\quad + (1-p)^i h_i \\ &\Rightarrow h_i = \frac{ip^{i-1} (1-p) + [1 - p^i - (1-p)^i - ip^{i-1} (1-p)] \lambda}{1 - (1-p)^i} \\ &\Rightarrow h_i = \lambda - \frac{p^i}{1 - (1-p)^i} \lambda + \frac{ip^{i-1} (1-p)}{1 - (1-p)^i} (1 - \lambda) \end{aligned}$$

We want to prove that:

$$\begin{aligned} h_i \geq \lambda &\Rightarrow \\ \frac{ip^{i-1} (1-p)}{1 - (1-p)^i} (1 - \lambda) &\geq \frac{p^i}{1 - (1-p)^i} \lambda \Rightarrow \\ ip^{i-1} (1-p) &\geq [p^i + ip^{i-1} (1-p)] \lambda \Rightarrow \\ \frac{ip^{i-1} (1-p) + p^i - p^i}{p^i + ip^{i-1} (1-p)} &\geq \lambda \Rightarrow \\ 1 - \frac{p^i}{p^i + ip^{i-1} (1-p)} &\geq \lambda \Rightarrow \\ 1 - \frac{p^i}{p^i + ip^{i-1} (1-p)} &\geq \frac{2(1-p)}{2-p} \Rightarrow \\ 1 - \frac{p^i}{p^i + ip^{i-1} (1-p)} &\geq 1 - \frac{p}{2-p} \Rightarrow \\ \frac{p^i}{p^i + ip^{i-1} (1-p)} &\leq \frac{p}{2-p} \Rightarrow \\ p^i (2-p) &\leq p[p^i + ip^{i-1} (1-p)] \Rightarrow \\ p^i (2-p) &\leq p^i [p + i(1-p)] \Rightarrow \\ 2 - 2p - i + ip &\leq 0 \Rightarrow \\ 2 - i - p(2-i) &\leq 0 \Rightarrow \\ (2-i)(1-p) &\leq 0 \Rightarrow \\ 2 - i &\leq 0 \end{aligned}$$

which holds since $i > 2$.

The above bound is also tight since $\exists i \in S : h_i = \lambda$, specifically $h_2 = \lambda$. \square

Now we can prove the following theorem that bounds the convergence time of the DTCM of Definition A.1, which corresponds to the proposed learning rule for the case of a single available resource ($R = 1$) and constant back-off probability.

THEOREM A.8. *The expected hitting time of the set of absorbing states $A = \{1\}$ of the DTMC $\{X_t\}$ of Definition A.1, starting from any initial state $X_0 \in \{0, 1, \dots, N\}$, is bounded by:*

$$O\left(\frac{2-p}{2p(1-p)} \log N\right) \quad (18)$$

PROOF. Using Lemma A.7 we can derive that the DTMC $\{X_t\}$ needs in expectation $\frac{1}{\lambda} = \frac{2-p}{2(1-p)}$ passes until it hits state 1. Each pass requires $O\left(\frac{1}{p} \log N\right)$ steps (Corollary A.6). Thus, the expected hitting time of state $A = \{1\}$ is $O\left(\frac{2-p}{2p(1-p)} \log N\right)$. \square

A.1.2 Case #2: Multiple Agents, Multiple resources ($R > 1$).

THEOREM A.9. *For N agents and R resources, assuming a constant back-off probability for each agent, i.e., $P_n(r, \prec_n) = p > 0, \forall n \in \mathcal{N}, \forall r \in \mathcal{R}$, the expected number of steps until the system of agents following of Alg. 1 converges to a complete matching is bounded by (19).*

$$O\left(R \frac{2-p}{2(1-p)} \left(\frac{1}{p} \log N + R\right)\right) \quad (19)$$

PROOF. At most N agents can compete for each resource. We call this period a round. During a round, the number of agents competing for a specific resource monotonically decreases, since that resource is perceived as occupied by non-competing agents. Let the round end when either 1 or 0 agents compete for the resource. Corollary A.6 states that in expectation this will require $\mathcal{O}\left(\frac{1}{p} \log N\right)$ steps.

If all agents backed-off, it will take on average R steps until at least one of them finds a free resource. We call this period a break.

In the worst case, the system will oscillate between a round and a break. According to the above, one oscillation requires in expectation $\mathcal{O}\left(\frac{1}{p} \log N + R\right)$ steps. If $R = 1$, Lemma A.7 states that in expectation there will be $\frac{1}{\lambda} = \frac{2-p}{2(1-p)}$ oscillations. For $R > 1$ the expected number of oscillations is bounded by $\mathcal{O}\left(R \frac{2-p}{2(1-p)}\right)$. Thus, we derive the required bound (19). \square

A.1.3 Dynamic back-off Probability. So far we have assumed a constant back-off probability for each agent, i.e., $P_n(r, \prec_n) = p > 0, \forall n \in \mathcal{N}, \forall r \in \mathcal{R}$. In this section we will drop this assumption. Let $\psi = \max(\log N, R)$. Bound (19) becomes:

$$\mathcal{O}\left(\psi^2 \frac{2-p}{2(1-p)} \left(\frac{1}{p} + 1\right)\right) \quad (20)$$

Intuitively, the worst case scenario corresponds to either all agents having a small back-off probability, thus they keep on competing for the same resource, or all of them having a high back-off probability, thus the process will keep on restarting. These two scenarios correspond to the inner ($\frac{1}{p}$) and outer ($\frac{2-p}{2(1-p)}$) probability terms of bound (20) respectively. We can rewrite the right part of bound (20) as:

$$\frac{2-p}{2(1-p)} \left(\frac{1}{p} + 1\right) = \frac{1}{p} + \frac{1}{1-p} + \frac{1}{2} = \tau \quad (21)$$

As seen by Eq. 21, τ assumes its maximum value on the two extremes, either with a high ($p \rightarrow 1^-$), or a low ($p \rightarrow 0^+$) back-off probability, i.e., $\lim_{p \rightarrow 1^-} \tau = \lim_{p \rightarrow 0^+} \tau = \infty$. Let $p^* = f(\text{loss}^*)$ be the worst between the smallest or highest back-off probability any agent $n \in \mathcal{N}$ can exhibit, i.e., having loss^* given by Eq. 23. Using p^* instead of the constant p , we bound the expected convergence time according to bound (22).

$$\mathcal{O}\left(R \frac{2-p^*}{2(1-p^*)} \left(\frac{1}{p^*} \log N + R\right)\right) \quad (22)$$

$$\text{loss}^* = \arg \min_{\text{loss}_n^r} \left(\min_{r \in \mathcal{R}, n \in \mathcal{N}} (\text{loss}_n^r), 1 - \max_{r \in \mathcal{R}, n \in \mathcal{N}} (\text{loss}_n^r) \right) \quad (23)$$

This concludes the proof of Theorem 2.1.