Lexicographic min-max fairness in task assignments

Geoffrey Ding and Hamsa Balakrishnan

***Abstract*— Assignment problems and their variants are ubiquitous across resource allocation applications. While they traditionally focus on minimizing costs or maximizing utility, fairness is also an important consideration, especially for task assignment in multi-agent systems. We propose algorithms for assigning tasks to agents that consider lexicographic min-max fairness, a stronger notion of fairness than min-max fairness, which minimizes the maximum cost to any single agent. We apply our proposed approaches to both one-to-one and one-to-many assignment problems. Due to the computational challenges of one-to-many task assignments, we develop tractable approaches to achieve approximate fairness. Finally, we use the proposed methods to evaluate the trade-offs between efficiency and fairness through numerical experiments.**

I. Introduction

The assignment problem, which involves the optimal matching of objects between two sets, has become a mainstay of operations research textbooks due to its importance in resource allocation [1]. The growth of multi-agent systems has led to renewed interest in the problem of assigning tasks to agents [2], [3]. While efficiency (i.e., minimizing total cost or maximizing total utility) has been the traditional focus of the assignment problem, fairness is also an important consideration, especially in multi-agent systems. For example, fairness may be desired in assignments of workers to jobs or evacuees to shelters. However, there are many metrics for fairness, and prior work has shown that fairness in resource allocation generally comes at the expense of efficiency [4].

A common notion of fairness is min-max fairness, which minimizes the cost of the worst case allocation (i.e., the maximum cost incurred by any agent). Min-max fairness is equivalent to the bottleneck assignment problem for one-to-one assignments [1]. One-to-many assignments are of interest in several applications, e.g., when an agent can perform multiple tasks [5], [6]. Min-max fairness in this setting is equivalent to the *Santa Claus problem*, which deals with allocating presents to children to maximize the value to the least happy child [7]. A stronger variation of min-max fairness is to not only minimize the highest cost to any agent, but to then progressively minimize the second-highest cost without increasing the highest cost to any agent, the third-highest cost without increasing either of the two highest costs, and so on. This concept, previously considered in the context of one-to-one assignments, is called *lexicographical min-max fairness* [1]. We refer to it as *lexifairness* for short and consider it in one-to-many task assignment settings.

The NASA University Leadership Initiative (grant #80NSSC21M0071) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity. G. Ding was additionally supported by a National Science Foundation Graduate Research Fellowship.

G. Ding and H. Balakrishnan are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA gding,hamsa@mit.edu

A recent focus of work on balancing fairness and efficiency involves matching drivers and passengers in ridesharing. Measures of fairness for this problem include min-max fairness [8], an extension of the Fagin-Williams share [9], and average pairwise difference in payoff [10]. However, these works measure fairness with a single number, which cannot capture lexifairness.

Bandwidth allocation in communication networks also must balance efficiency and fairness. This problem has motivated the concept of α-fairness [11], which uses a utility function of the form $U_\alpha(x) = \sum_i x_i^{(1-\alpha)}/(1-\alpha)$, where x is a vector of bandwidth allocations. Varying $\alpha \geq 0$ results in different notions of fairness, with $\alpha \to \infty$ yielding a max-min fairness that not only maximizes the minimum utility of any agent, but also progressively maximizes the utilities of the other agents. As such, it is the continuous analog of lexifairness for discrete assignments, since bandwidth is continuous. Empirically, however, naïvely applying α-fairness to an assignment problem produces inefficient solutions.

We propose algorithms for finding lexifair solutions to one-to-one and one-to-many assignment problems. For the latter problem, we propose a computationally tractable approach that approximates lexifairness by considering the total number, rather than the cost, of tasks assigned to an agent. Finally, we evaluate the trade-offs between efficiency and fairness in these settings through numerical experiments.

A. The assignment problem

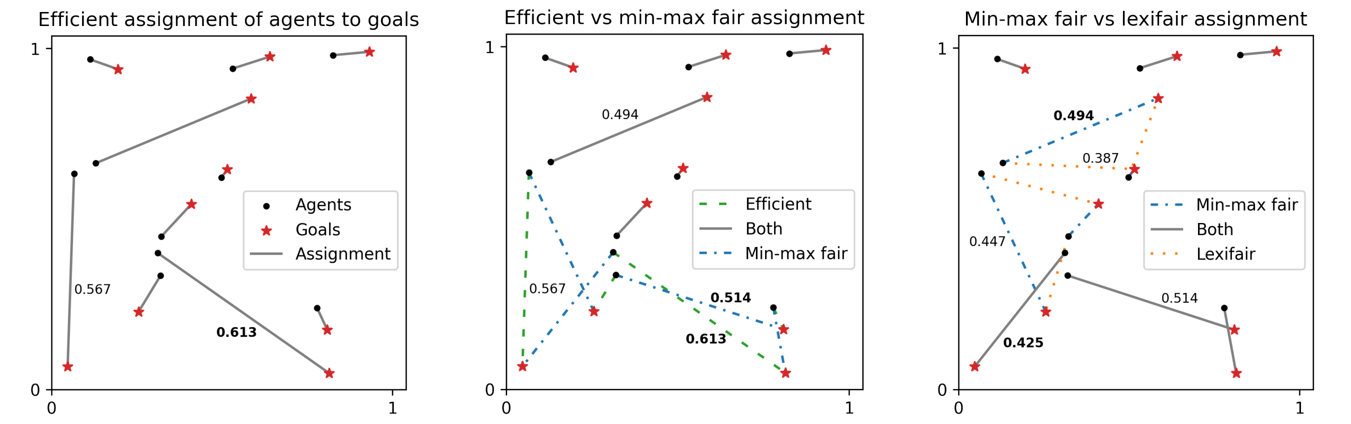
Consider a scenario with n agents and n tasks, where each agent must perform exactly one task and each task must be done by exactly one agent. The cost of agent i performing task j is c_{ij}, and we would like to assign tasks to agents to minimize the total cost. For simplicity, let us assume the costs of all agent-task pairings are unique. The *efficient* assignment is the solution to the following optimization problem:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \tag{1b}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \tag{1c}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j, \tag{1d}$$



(a) Efficient assignment of agents to goals, with the highest cost to an agent in bold.

(b) Efficient vs. min-max fair assignment, with the highest cost to an agent in bold.

(c) Min-max fair vs. lexifair assignment, with the *second*-highest cost to an agent in bold.

Fig. 1: Efficient, min-max fair, and lexifair assignments.

where x_{ij} is 1 if agent i performs task j and 0 otherwise. In the rest of this paper, we mention explicitly where $x_{ij} = 1$; all other values of x_{ij} are assumed to be 0. Equation (1b) ensures that all tasks are completed, while (1c) ensures that each agent performs only one task.

Consider the environment in Fig. 1, where agents (black circles) must travel to goals (red stars); we also refer to goals as tasks. Suppose the cost of assigning agent i to task j is equal to the distance between the two. Fig. 1a depicts the efficient assignment, where the edges indicate the assignment of goals to agents that minimizes the total cost.

B. *Fair assignments*

An efficient assignment may not be fair. For example, consider the following cost matrix:

$$c = \begin{bmatrix} 9 & 8 & 7 \\ 5 & 2 & 3 \\ 6 & 4 & 1 \end{bmatrix} \tag{2}$$

Choosing $x_{11} = x_{22} = x_{33} = 1$ is the most efficient solution, with a total cost of 13, but it is also unfair—agent 1 incurs a cost of 9, while agents 2 and 3 only incur costs of 2 and 1, respectively. In scenarios where the agents are non-cooperative, we may be interested in a fair assignment.

Min-max fairness aims for equity by minimizing the maximum cost to any agent. Solving the following optimization problem yields the min-max fair solution:

$$\begin{aligned}
\min \quad & z && && \text{(3a)}\\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1 && \forall j && \text{(3b)}\\
& \sum_{j=1}^{n} x_{ij} = 1 && \forall i && \text{(3c)}\\
& c_{ij} x_{ij} \leq z && \forall i, j && \text{(3d)}\\
& x_{ij} \in \{0,1\} && \forall i, j, && \text{(3e)}
\end{aligned}$$

where c_{ij} and x_{ij} are as above, and z is the maximum cost incurred by any agent. For the cost matrix given in (2), solving (3) yields the solution $x_{13} = x_{22} = x_{31} = 1$, with agents incurring costs of $(7, 2, 6)$ and a total cost of 15. This is less efficient ($15 > 13$), but it is more fair because the maximum cost incurred by any agent is lower ($7 < 9$).

Fig. 1b shows the min-max fair assignment of agents to goals. Agent-goal pairs that appear in the efficient assignment but not the fair one are shown in dashed green, those that appear in the fair assignment but not the efficient one are shown in dash-dotted blue, and those that appear in both are shown in solid gray. While the total cost increases from 2.335 (for the efficient assignment) to 2.542 (for the min-max fair assignment), the maximum cost incurred by any agent decreases from 0.613 to 0.514, as labeled in bold.

Lexifairness may—and typically does—come at a greater cost to efficiency than min-max fairness. For the cost matrix from (2), the lexifair assignment is $x_{13} = x_{21} = x_{32} = 1$, with agents incurring costs of $(7, 5, 4)$ and a total cost of 16. This is less efficient than the min-max fair assignment, but now the second-highest cost is also minimized ($5 < 6$).

C. *Trade-offs between fairness and efficiency*

The *price of fairness* is the decrease in efficiency (or increase in cost) of a fair assignment relative to the corresponding efficient one. It is defined as

$$\mathrm{POF}(c) = \frac{\sum_{i,j} c_{ij} x_{ij}^{\text{fair}} - \sum_{i,j} c_{ij} x_{ij}^{\text{eff}}}{\sum_{i,j} c_{ij} x_{ij}^{\text{eff}}}, \tag{4}$$

where x^{fair} is a fair solution and x^{eff} is an efficient solution. For the costs in (2), the price of min-max fairness is $(15 - 13)/13 = 0.15$, and the price of lexifairness is $(16 - 13)/13 = 0.23$.

Common approaches for evaluating the trade-off between efficiency and fairness are (1) multi-objective optimization, which optimizes a convex combination of fairness and efficiency (weighted by λ and $1-\lambda$, respectively, for $\lambda \in [0, 1]$); (2) α-fairness, where varying α yields different notions of fairness and $\alpha = 0$ corresponds to the efficient solution; and (3) thresholding, which uses the min-max fair assignment if the loss in efficiency is less than $\Delta(n-1)$, where Δ is a predetermined parameter and n is the number of agents, and the efficient assignment otherwise.

Each of the above approaches uses a parameter to navigate the efficiency-fairness trade-off: λ in multi-objective optimization, α for α-fairness, and Δ for thresholding. Our proposed method also parametrizes fairness. However, in contrast to these approaches, we do so using the number of agents that are treated fairly.

The remainder of this paper is structured as follows: Section II proposes algorithms for determining a lexicographically min-max fair task assignment and contrasts it with the traditional min-max fair assignment. We evaluate how allowing a single agent to complete multiple tasks impacts efficiency and fairness in Section III, and consider "partially fair" assignments in Section IV. We include some additional points of discussion in Section V. Section VI concludes and suggests possible directions for further research.

II. Lexicographic min-max fairness in one-to-one assignments

We first consider one-to-one assignment settings (i.e., with n agents and n tasks). Letting $c_i = \sum_{j=1}^{n} c_{ij} x_{ij}$ be the cost to agent i, the lexifair assignment x has the following property for all assignments x':

$$\exists i \text{ s.t. } c'_i < c_i \implies \exists k \text{ s.t. } (c_k \geq c_i) \wedge (c'_k > c_k). \quad (5)$$

In other words, if some agent i has a lower cost in x' than in x, it must come at the expense of another agent k that has at least as high of a cost as i in x.

A. Determining a lexifair assignment

A lexifair assignment may be found by repeatedly solving the min-max fair assignment problem, fixing one agent and one task at each iteration. Solving the problem yields an objective value z that corresponds to some c_{ij} of maximal cost in the assignment. The associated agent i is assigned to task j, and this is fixed in future iterations. Then, we update c_{ij} to be 0 to ignore it in future iterations. These steps are repeated n times until all n agents and tasks are assigned. Algorithm 1 shows this procedure.

Algorithm 1: Algorithm for lexifair assignment

```
Input: $c \in \mathbb{R}_{\geq 0}^{n \times n}$
Output: $x^* \in \{0,1\}^{n \times n}$, a lexifair assignment
$x^* \leftarrow 0^{n \times n}$;
for $k = 1, \ldots, n$ do
    Solve min-max fair assignment (3) with $x_{ij} \geq x^*_{ij}$;
    $z \leftarrow$ objective value of assignment problem;
    $x^*_{ij} \leftarrow 1$ where $c_{ij} = z$;
    $c_{ij} \leftarrow 0$ ;      /* Does not count in future
     iterations. */
end
```

We sketch a proof of correctness for this algorithm. Without loss of generality, let agent k be fixed in the k^{th} iteration of the algorithm and z^*_k be its cost. By construction of the min-max fair problem, agent 1 cannot reduce its cost in any other assignment. Now, suppose agents $1, \ldots, k-1$ are fairly and optimally assigned, so there are $n-k$ unassigned tasks after iteration k. Agent k cannot decrease its cost by taking any of those tasks, again by construction of the min-max fair problem. Then, suppose it could decrease its cost by taking one of the previously assigned tasks, reassigning the corresponding agent i to another task. To preserve fairness, since $z^*_i > z^*_k$, we cannot reassign agent i to a task of cost higher than z^*_i. On the other hand, reassigning it to a task of lower cost would contradict our assumption of an optimal assignment for agent i. Therefore, this algorithm determines the lexifair assignment.

Alternatively, as before, we solve n iterations of (3). After each iteration, we fix $x_{ij} = 1$ for which $c_{ij} = z$, where z is the solution to (3). Then, we remove the constraint of (3d) associated with (i,j) and proceed with the next iteration. This approach also determines a lexifair assignment.

B. Comparison with min-max fairness

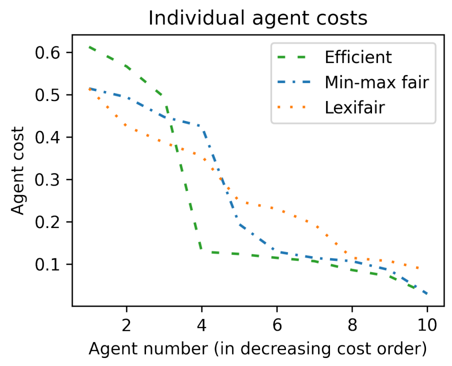
Lexifairness answers the question of how tasks should be assigned to agents that do *not* incur the maximum cost. While the min-max fair assignment given by solving (3) does not guarantee a unique solution, lexifairness does. One approach for the min-max fair assignment, done in Fig. 1b, is to minimize total cost while minimizing the maximum individual cost.

Fig. 1c shows the differences between the min-max fair assignment (blue dash-dots) and the lexifair assignment (orange dots) for the example in Fig. 1a. The agent-goal pair of second-highest cost in each assignment is labeled in bold, and the costs are 0.494 and 0.425 in the min-max fair and lexifair assignments, respectively. The agent with the second-highest cost has a lower cost in the lexifair assignment than in the min-max fair assignment, so we do not consider any additional agents; if the second-highest cost were the same for both assignments, then we would look at the agent with the third-highest cost, and so on until the lexifair assignment yields an agent with a lower (ordered) cost than the min-max fair assignment. Much like how the efficient and min-max fair assignments may be identical, it is also possible that the min-max fair and lexifair assignments are identical.

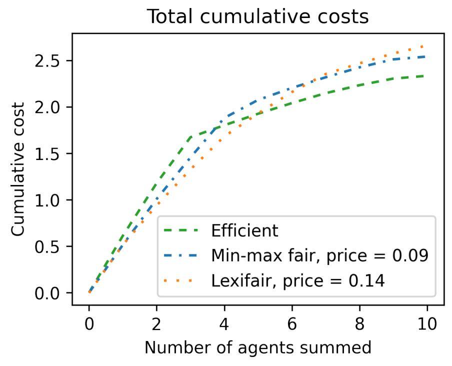
For another perspective on the efficient, min-max fair, and lexifair assignments, we plot agents' costs in decreasing order in Fig. 2a, and cumulative costs in Fig. 2b. There are two expected—but important—observations here: (1) The total cost of the lexifair assignment is greater than that of the min-max fair assignment, i.e., the stronger notion of fairness is achieved at the expense of efficiency; and (2) the cost to the agent incurring the second-highest cost is lower in the lexifair assignment than in the min-max fair assignment, i.e., we care not just about the agent with the highest cost but also the agent with the second-highest cost, and so on.

C. Lexifair assignment as a network flow problem

Motivated by the frequent formulation of assignment problems as network flows for computational tractability,



(a) Individual agent costs



(b) Cumulative costs

Fig. 2: Individual and cumulative costs of efficient, min-max fair, and lexifair assignments.

we propose an alternative algorithm to determine a lexifair assignment. We initialize a bipartite graph with n agent and n task vertices. Then, in order of increasing cost, we add edges until the assignment problem is feasible. The last added edge has cost equal to the objective value in iteration k of Algorithm 1. That edge is included in all future initializations of the graph. This process is repeated n times to obtain n edges that form the lexifair assignment. Because this algorithm models the assignment problem as a network flow, it can be solved in polynomial time, e.g., by using n iterations of the network simplex algorithm [12].

The proof of correctness for this algorithm is simpler than for Algorithm 1. Once again, without loss of generality, let agent k be the agent assigned in the k^{th} iteration of the algorithm and z_k^* be its cost. $z_k < z_k^*$ is impossible because the addition of the edge corresponding to z_k^* is the first instance of a feasible solution. Meanwhile, any $z_k > z_k^*$ is not min-max fair because it can be lowered to z_k^* without adversely impacting any other agent of equal or higher cost. Therefore, this network flow-based approach gives the lexifair assignment in a one-to-one setting.

III. One-to-Many Fair Assignments

Thus far, we have only considered cases with n agents and n tasks. We now examine situations with m tasks, potentially with $m \neq n$, and where each agent may perform multiple tasks (or no tasks). For simplicity, we assume that the total

Algorithm 2: Lexifair assignment using network flows

```
Input: $c \in \mathbb{R}_{\geq 0}^{n\times n}$
Output: $x \in \{0,1\}^{n\times n}$, a lexifair assignment
$x \leftarrow 0^{n\times n}$ ;          /* No values assigned. */
$U \leftarrow \{1,\dots,n\}$;
$V \leftarrow \{1,\dots,n\}$;
$Edges \leftarrow [(i,j) \mid i \in U, j \in V]$ of unit capacity in order
  of ascending $c_{ij}$;
$s \leftarrow$ source node of supply $n$;
$t \leftarrow$ sink node of supply $-n$;
for $k = 1,\dots,n$ do
    $E \leftarrow \{(s,i) \mid i \in U\} \cup \{(j,t) \mid j \in V\}$ of unit
      capacity and 0 cost;
    $E \leftarrow E \cup \{(i,j) \in U \times V \mid x[i,j] = 1\}$ of unit
      capacity and cost $c_{ij}$;
    $\ell \leftarrow 1$;
    while network flow on $(\{s,t\} \cup U \cup V, E)$
      (equivalently, bipartite matching $(U,V,E)$) is
      infeasible do
        $e \leftarrow Edges[\ell]$;
        $E \leftarrow E \cup \{e\}$;
        $\ell \leftarrow \ell + 1$;
    end
    $x_e \leftarrow 1$;
end
```

costs of all subsets of tasks for an agent are unique; this assumption avoids the need to tiebreak among assignments.

A. Optimization formulations

The efficient solution is given by

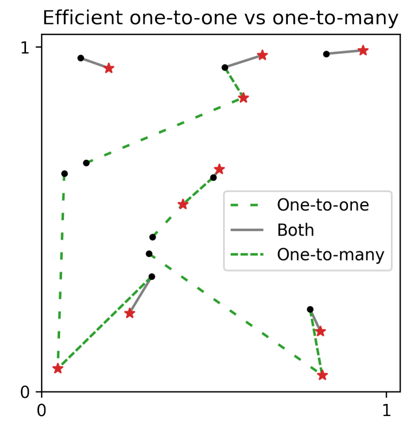
$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \tag{6a}$$
$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \tag{6b}$$
$$x_{ij} \in \{0,1\} \qquad \forall i,j, \tag{6c}$$

where unlike (1), we no longer require each agent to perform exactly one task, as in (1c). This problem can be solved via a greedy approach, with each task assigned to the agent that can complete it at the lowest cost. Fig. 3a illustrates the differences between an efficient one-to-one assignment (sparser dashes) and an efficient one-to-many assignment (denser dashes). Each task is assigned to the closest agent, with some agents assigned multiple tasks and others none.

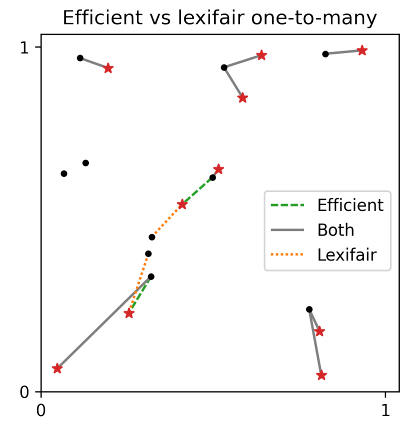
The lexifair assignment is obtained by iteratively solving

$$\min \quad z \tag{7a}$$
$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \tag{7b}$$
$$\sum_{j=1}^{m} c_{ij}x_{ij} \leq z \qquad \forall i \tag{7c}$$
$$x_{ij} \in \{0,1\} \qquad \forall i,j, \tag{7d}$$

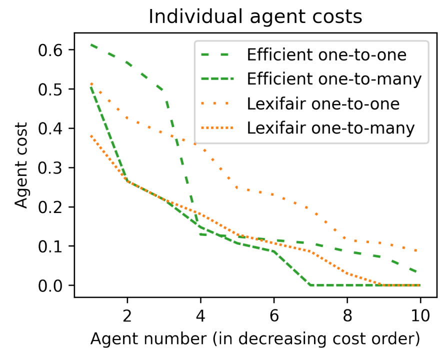
where (7c) aggregates the costs of all tasks performed by an agent when considering fairness. However, this means we can no longer use Algorithm 2 to determine a lexifair assignment,



(a) Efficient one-to-one and efficient one-to-many assignments.



(b) Efficient one-to-many and lexifair one-to-many assignments.



(c) Individual agent costs for various assignments.

Fig. 3: Comparisons of efficient and lexifair one-to-one and one-to-many assignments.

as selecting a costlier edge may enable other agents to reduce their costs. The problem would become feasible as soon as there is an edge to all tasks, but assigning one task does not prevent an agent from performing additional tasks.

Suppose we use the algorithm from Section II-C to solve the one-to-many case for (2). The assignment would assign, in order, x_{21}, x_{22}, and x_{33} to 1, for a min-max cost of 7. On the other hand, the true min-max fair solution is given by $x_{21} = x_{32} = x_{33} = 1$, with a min-max cost of 5. Therefore, Algorithm 2 cannot be directly applied to the one-to-many assignment problem. Fig. 3b shows the differences between an efficient one-to-many assignment (dense green dashes) and a lexifair one-to-many assignment (dense orange dots) of agents to goals. For both tasks that are assigned to new agents, the original agent had been performing multiple tasks.

Finding the lexifair solution in the one-to-many setting requires repeatedly solving the min-max fair problem (7), i.e., the Santa Claus problem. However, the latter problem is NP-hard to approximate better than within a factor of 2 [13]. Researchers have therefore pursued polynomial-time approximation algorithms and heuristics [14], [15].

B. *The value of one-to-many assignments*

Allowing agents to perform multiple tasks generally reduces costs. When optimizing for efficiency, a task can be assigned to the agent that can do it at the lowest cost, while an assignment cannot become less lexifair because we have increased the feasible space. Although a *specific* agent may incur a higher cost, a lexifair one-to-many assignment will still be fairer than a lexifair one-to-one assignment.

We illustrate this by plotting the individual agent costs of efficient and lexifair assignments in one-to-one and one-to-many settings in Fig. 3c. As expected, the one-to-many assignments are more efficient *and* more fair regardless of the objective. Here, the efficient one-to-many assignment is even fairer than the lexifair one-to-one solution. Finally, the total costs decrease by 43.1% and 47.5%, respectively, for the efficient and lexifair one-to-many assignments.

IV. Approximations of Lexifairness

Next, we consider varying levels of fairness by changing the number of agents treated fairly. Furthermore, we propose computationally tractable approaches to approximating lexifairness in one-to-many settings.

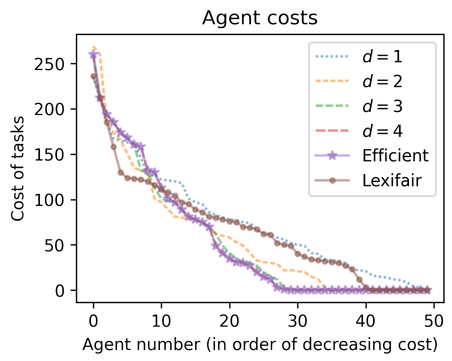
A. *Task count restriction*

A partially-fair assignment for the one-to-many case can be found by modifying Algorithm 2. We first include edges in our network for all agent-task pairs. Instead of edges of unit capacity from the source node to each agent, we add edges of integral capacity $1 \leq d \leq n$, representing a situation in which each agent can perform up to d tasks; we call this a d-degree fair allocation. The resulting network flow problem still must consider which task is assigned to which agent. For example, if an agent has the lowest cost for every task but can do at most two tasks, the optimization must decide which tasks should be done by that agent and which by others. This solution can be found in polynomial time.

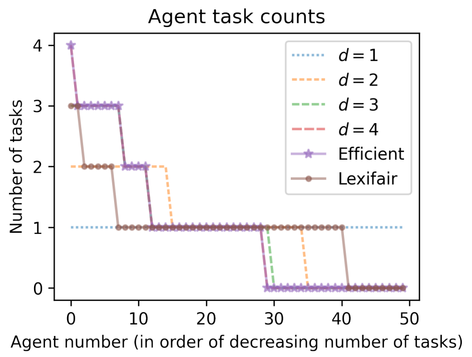
The restriction of task count is only meaningful in one-to-many settings, so we use that as the model for comparing different approaches to fairness and their approximations. We use a randomized 50-by-50 cost matrix of integers from 0 to 2,499. Fig. 4a compares the distribution of agent costs for different levels of d-degree fairness to the distribution of agent costs in the lexifair and efficient assignments. Fig. 4b illustrates the degree distribution of agents, in order of decreasing numbers of tasks completed.

B. *Partial optimization*

Because both of our proposed algorithms solve for a lexifair assignment iteratively, another natural way to approximate lexifairness is to stop after any iteration and use the assignment at that point. A k-agent fair allocation, where $1 \leq k \leq n$, first performs k iterations of min-max fair assignment, fixing the agent with the highest cost after each iteration and ignoring it in subsequent iterations. It then determines an efficient assignment for the remaining $n-k$

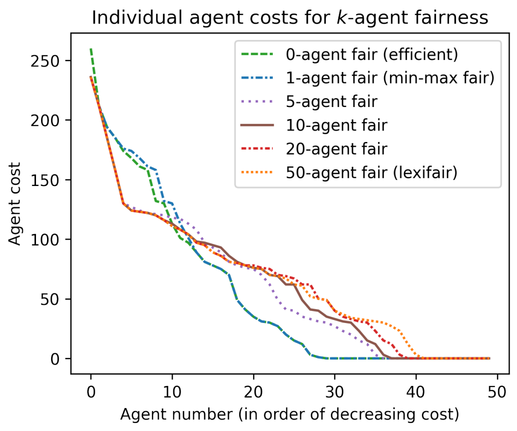


(a) Agent costs for d-degree fairness.

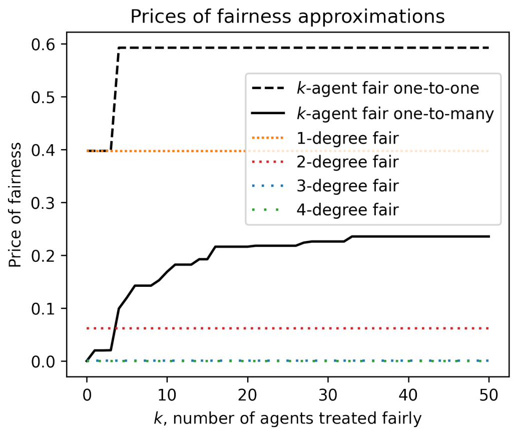


(b) Task counts for d-degree fairness.

Fig. 4: Properties of d-degree fairness.



(a) Agent costs for k-agent fairness.



(b) Price of d-degree and k-agent fairness.

Fig. 5: Properties of k-agent fairness.

agents. To preserve lexifairness, the efficient assignment must not assign any agent a cost higher than that of the k^{th} agent. Under k-agent fairness, the efficient assignment is 0-agent fair, the min-max fair assignment is 1-agent fair, and the lexifair assignment is n-agent fair.

Fig. 5a compares k-agent fair assignments for selected values of k. Since an n-agent fair assignment is the lexifair assignment, a k-agent fair assignment should overlap with the lexifair assignment for (at least) the first k agents. By the definition of lexifairness, the first agent for which the k-agent fair and the lexifair assignments differ in agent cost must have a higher cost in the k-agent fair assignment than in the lexifair assignment. In Fig. 5b, we plot the price of k-agent fairness for all values of k. For comparison, we also include horizontal lines representing d-degree fairness for relevant values of d. The cost of the 4-degree fair assignment is the same as that of the efficient one-to-many assignment because the latter assigns an agent to at most four tasks in this instance, and a d-degree fair assignment optimizes efficiency subject to task count constraints. Similarly, the cost of the 1-degree fair assignment is the same as that of the efficient one-to-one assignment since the problems are equivalent.

It is possible for optimizations with two different values of k to yield identical assignments (and therefore costs); this tends to be the case especially as $k \to n$, as can be seen in Fig. 5b. Conversely, without the assumption of unique costs of subsets of tasks, two optimizations with identical values of k may yield different assignments.

V. Discussions

We perform an experiment with randomized instances to assess average efficiency and fairness, and we discuss how lexifairness may be extended to other applications.

A. Fairness over many trials

The Gini coefficient is used in economics to measure inequality; we use it to measure fairness. A Gini coefficient of $G = 0$ represents perfect fairness, with all agents incurring an identical cost, while $G = 1$ represents perfect unfairness, with one agent bearing all costs.

We generate 500 random 50×50 cost matrices (Sec. IV) and find the efficient and lexifair assignments each time. Then, we sort individual costs in increasing order and average the 500 costs in each position to find the corresponding mean cost. We use these mean costs to plot the Lorenz curves in Fig. 6. The fair assignment is on average 13% more costly than the efficient assignment, but it is more equitable, with $G^{\text{fair}} = 0.37 < 0.47 = G^{\text{eff}}$.

The Gini coefficient should be applied with caution in this context. The fairest possible scenario, with the 50 agents incurring costs from 0 to 49, still has a Gini coefficient of 1/3, not the ideal 0. Drawing costs independently from a uniform

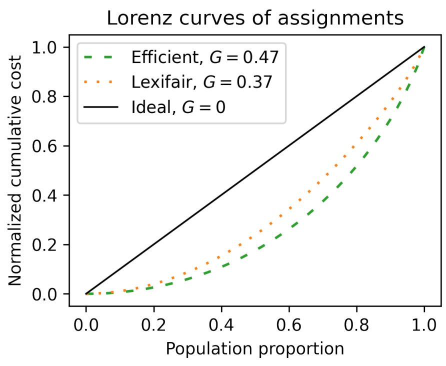


Fig. 6: Lorenz curves for efficient and lexifair solutions. Cumulative costs are normalized to facilitate comparison.

distribution instead of choosing unique values would likely achieve lower Gini coefficients, denoting fairer outcomes.

B. Additional applications

We use the navigation setting (Fig. 1) as a motivating example, but lexifair assignments have many other applications, e.g., assigning jobs to workers. Differing worker abilities lead to heterogeneous job costs, and equitable worker effort may be desirable. To individual employees, minimizing the total job cost may matter less than minimizing individual costs.

Emergency shelter allocation is another application where optimizing fairness may be more important than efficiency. Suppose there are n shelters in a city and m neighborhoods, indexed by j and each with population p_j. Then, planning an emergency response may entail assigning individuals or neighborhoods to shelters. This formulation is amenable to the d-degree fair model: In the associated network, the source and sink have supplies of $\pm \sum_j p_j$, the edge between neighborhood j and the sink has capacity p_j, and all other edges have unlimited capacity. Then, a d-degree fair assignment can be thought of as "load balancing" individuals across shelters. Meanwhile, a lexifair assignment can be justified with the explanation that assignment to a nearer shelter would require another individual who must already travel a greater distance to go even farther. In this case, the roles of "agent" and "task" are reversed—shelters correspond to agents and neighborhoods correspond to goals, even though shelters are fixed locations and individuals in neighborhoods move and incur costs in reality.

The matching problem could also leverage d-degree fairness, which can be interpreted as enforcing lexifairness on the number of tasks assigned to an agent rather than on the total cost of tasks. Suppose we wish to complete a set of tasks, but not every agent can perform every task (i.e., the bipartite graph is not complete). Then, if we start from $d = 1$ and increase it until the problem is feasible, we find the maximum number of tasks a single agent must perform in order for the agents in aggregate to be able to complete all tasks. After removing the agent of maximum degree and its tasks, we can repeat the procedure until all tasks are matched.

VI. Conclusion and Future Work

In this paper, we consider the problem of finding a lexicographically min-max fair assignment of tasks to agents. We propose two algorithms for determining a lexifair assignment when each agent performs one—and only one—task, and we show how it differs from the min-max fair assignment. We illustrate how allowing an agent to perform multiple tasks can improve both efficiency and fairness. We then suggest methods to determine a partially fair yet partially efficient assignment. We also discuss the distribution of agent costs, additional applications of our approaches, and the computational complexity of the proposed approaches.

A key limitation of this work is the assumption that all subsets of tasks have unique costs (i.e., no tiebreaking is required). Branching with breadth-first search may resolve ties, but duplicate costs could still result in a factorial number of branches. Tiebreaking efficient assignments may also yield unique results. Finally, another direction to consider is fairness and efficiency in a team setting.

References

[1] D. W. Pentico, "Assignment problems: A golden anniversary survey," *European Journal of Operational Research*, vol. 176, no. 2, 2007.

[2] Q. Baert, A.-C. Caron, M. Morge, and J.-C. Routier, "Fair multi-agent task allocation for large datasets analysis," *Knowledge and Information Systems*, vol. 54, pp. 591–615, 2018.

[3] T. A. Wood, M. Khoo, E. Michael, C. Manzie, and I. Shames, "Collision avoidance based on robust lexicographic task assignment," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, 2020.

[4] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," *Operations research*, vol. 59, no. 1, pp. 17–31, 2011.

[5] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5816–5823, 2021.

[6] M. Malencia, V. Kumar, G. Pappas, and A. Prorok, "Fair robust assignment using redundancy," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4217–4224, 2021.

[7] N. Bansal and M. Sviridenko, "The Santa Claus problem," in *Proceedings of the thirty-eighth annual ACM Symposium on Theory of computing*, pp. 31–40, 2006.

[8] N. S. Lesmana, X. Zhang, and X. Bei, "Balancing efficiency and fairness in on-demand ridesourcing," *Advances in neural information processing systems*, vol. 32, 2019.

[9] Z. Chen, P. Cheng, L. Chen, X. Lin, and C. Shahabi, "Fair task assignment in spatial crowdsourcing," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, 2020.

[10] Y. Zhao, K. Zheng, J. Guo, B. Yang, T. B. Pedersen, and C. S. Jensen, "Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 265–276, IEEE, 2021.

[11] F. Kelly, "Charging and rate control for elastic traffic," *European transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.

[12] J. B. Orlin, "A polynomial time primal network simplex algorithm for minimum cost flows," *Mathematical Programming*, vol. 78, pp. 109–129, 1997.

[13] I. Bezáková and V. Dani, "Allocating indivisible goods," *ACM SIGecom Exchanges*, vol. 5, no. 3, pp. 11–18, 2005.

[14] A. Asadpour and A. Saberi, "An approximation algorithm for max-min fair allocation of indivisible goods," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 114–121, 2007.

[15] S. Davies, T. Rothvoss, and Y. Zhang, "A tale of Santa Claus, hypergraphs and matroids," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2020.