Fair Resource Allocation for Heterogeneous Tasks

Koyel Mukherjee, Partha Dutta, Gurulingesh Raravi, Thangaraj Balasubramaniam, Koustuv Dasgupta, Atul Singh
Xerox Research Center India, Bangalore, India 560105
Email: {Koyel.Mukherjee, Partha.Dutta, Gurulingesh.Raravi}@xerox.com
{Rajasubramaniam.T, Koustuv.Dasgupta, Atul.Singh}@xerox.com

***Abstract*—We consider the problem of fair allocation of resources to tasks where a resource has to be assigned to at most one task entirely without any fractional allocation. The system is heterogeneous in the sense that the cost may vary across resources, and different tasks may have different resource demand. Due to heterogeneity of resource costs, the cost of allocation for a task in isolation, without any other competing task, may differ significantly from its allocation cost when the task is allocated along with other tasks. In this context, we consider the problem of allocating resources to tasks, while ensuring that the cost is distributed fairly across the tasks, namely, the ratio of allocation cost of a task to its isolation cost is minimized over all tasks. We show that this fair resource allocation problem is strongly NP-Hard even when resources are of unit size by a reduction from 3-partition. Our main results are a $2+O(\epsilon)$ approximation LP rounding based algorithm for the problem when resources are of unit capacity, and a near-optimal greedy algorithm for a more restricted version.**

The above fair allocation problem arises in various context, such as, allocating computing resources for reservation requests from clients in a data center, allocating resources to computing tasks in grid computing, or allocating personnel for projects in service delivery organizations.

***Keywords*-Resource allocation, Approximation algorithm, LP rounding, Greedy algorithm**

I. Introduction

In this work we consider a heterogeneous system where each resource has an associated cost and a capacity (or size), and a resource can be allocated entirely to at most one task, without any fractional allocation. Every task has a demand (or capacity requirement) and we would like to minimize the resource allocation cost for meeting this demand. In isolation, i.e., when there is exactly one task in the system, an ideal allocation would select a set of resources that would meet the task's demand while minimizing its total cost. This case of a task in isolation is identical to the minimization knapsack problem [6], a known NP-Hard problem, and we call the corresponding minimum cost of a task, its *isolation cost*. However, in presence of multiple tasks, due to heterogeneity of resources, the resource allocation cost for a task may significantly differ from its isolation cost. In this scenario, we would like to minimize the cost for multiple tasks in a manner that is fair across tasks. In particular, when multiple tasks are present, we want to allocate the resources in a way that minimizes over all tasks, the ratio of the allocation cost of each task to its isolation cost.

Fair resource allocation is one of the core problems in parallel and distributed computing which arises in multiple settings, as illustrated next. Consider a multi-tenant data center where multiple tenants request to reserve certain computing capacity over the set of available (physical or virtual) machines. Here the tasks are the reservation requests with their respective demands, and the resources are the machines, each with its computing capacity and cost. We would like to ensure a resource allocation that is fair across tenants, in terms of the costs of resources allocated. From a tenant's point of view, such a fair allocation is preferable than the case where the data center operator tries to minimize the total cost over all resources. A similar problem can be seen in a geographically distributed grid computing environment in which different users request computing resource reservations, where the resource costs can be the monetary cost, or the cost of network communication. Finally, the problem of allocating personnel to project in large service delivery organizations, such that any increase in project cost, due to the presence of other tasks, is fairly distributed, can also be modeled as the above fair resource allocation problem.

A. Problem Definition and System Model

We study the following problem. Given a set of tasks where each task has a certain resource requirement (also called demand), and a set of resources where each resource has a certain resource supply (also called capacity or size of the resource) and a cost, find a *fair* allocation of tasks to resources such that the resource requirements of all the tasks are met and each resource is allocated entirely to at most one task. The fairness is defined with respect to the ratio of the cost of a task when allocated along with other tasks, to the optimal cost of the task, when allocated in isolation (i.e., when other tasks are not present). Our goal is to minimize the maximum of this ratio across all tasks. Henceforth, we refer to this problem as the *fair allocation problem*.

Formally, we are given a set of tasks $\mathcal{P}$, and we have access to a set of resources $\mathcal{E}$. Each task j is characterized by its resource requirement denoted by D_j. Each resource i is characterized by two parameters: a resource supply s_i and a cost c_i. Upon allocating a resource i to a task j, the task incurs a cost of c_i and its resource requirement is reduced by s_i. In order for a task j to successfully execute, it must be ensured that the total size of resources allocated to it is at least D_j. We require all assignments to be integral (0 or 1), i.e., each resource must be entirely assigned to a task and hence cannot be fractionally assigned to multiple tasks. We denote by I_j the *isolation cost* of a task j, that is the cost that the task would incur if this is the only task that needs to be allocated resources and no other tasks were present in the system. We denote by C_j the cost incurred by the task j when resources are allocated to it along with other tasks in the system; we refer to this cost as the *actual cost* of the task. The problem objective is to minimize $\gamma = \max_{i \in \mathcal{P}} \frac{C_i}{I_i}$.

B. Related Work

Fairness of resource allocation has been well-studied both from a theoretical and practical point of view. Fair sharing of network resources (such as a wired network links and wireless spectrum) has been extensively studied, and various indices and algorithms for fairness have been developed, such as [5], [7]. More recent work on sharing of system resources has focused on sharing resources in data centers [4], [9], and on a variant of max-min fairness. Our work differs from these [4], [5], [7], [9] in following two ways. Firstly, we consider a system where there are sufficient resources to satisfy requirements of all tasks, but the cost of the resources may vary, which in turn results in increase in cost of task in presence of other tasks. Most earlier work considers a system where the number of resources is limited and may not be able to fully satisfy the demands of all tasks. With the rapid increase in the computing capacity of data centers, our assumption of a large number of available resources from one or more data centers, albeit with different (rental) costs, is increasingly becoming more relevant. Secondly, we give a provable fairness guarantee for tasks, where fairness is quantified by the ratio of actual cost of a task to its isolation cost, which has not been investigated earlier.

The combinatorial optimization problem studied in this paper is a mixed packing and covering problem [11], where we need to cover the demand of task using resources subject to packing the selected resources within a certain cost budget. Young [11] studies the fractional version of the problem, which is not NP-hard, and provides efficient sequential and parallel algorithms for the problem. Chakaravarthy et al. [2] study a version of mixed packing and covering problems, where they solve a knapsack cover problem subject to cardinality constraints, and then extend it to multiple matroid constraints. In terms of the resource allocation problem that we study in this paper, the work of Chakaravarthy et al. [2] can be used to find the minimum cost resource allocation for *one* task, such that the total size of the resources allocated meets the total resource requirement of the task, and the number of resources allocated is at most a pre-specified number. This is slightly different compared to our problem where we try to ensure *fairness* of resource allocation costs across *multiple tasks*, such that the total resource requirement of every task is met. Escoffier et al. [3] study some multi-agent optimization problems where the goal is to maximize the satisfaction of the least satisfied agent, where the satisfaction of an agent is defined as the ratio between his utility for the given solution and his maximum possible utility. For some NP-hard problems, assuming a feasible solution exists, they give polynomial algorithms with approximation factors dependent on the *number* of agents and/or on other problem parameters. Though the objective of the problem studied is quite similar to ours, the underlying problem we study is different, and we provide *constant* approximations, not dependent on the problem size. Moreover, our techniques are fundamentally different from Escoffier et al. [3].

Online algorithms for the fractional mixed packing and covering problem are investigated by Azar et al. [1], where the packing constraints are known offline, while the covering constraints get revealed online. Their objective is to minimize the maximum multiplicative factor by which any constraint is getting violated, while meeting all the covering constraints. They give a polylogarithmic competitive ratio, and a nearly tight lower bound for the fractional problem. In contrast, we study an integral, offline version of the above problem, for which we give constant approximations.

C. Our Contributions

This paper makes the following contributions.

1) We first show that the fair allocation problem is NP-Hard in the strong sense even when the resources are of unit size.

2) We formulate the fair allocation problem as a mixed packing and covering problem and give a $2 + O(\epsilon)$ approximation algorithm, based on LP rounding when resources are unit sized.
3) We further show that the LP considered has an integrality gap of 2, hence the bound of $2 + O(\epsilon)$ is essentially tight for this LP.
4) Finally, for a restricted version of the problem, we give a near-optimal greedy algorithm.

We would like to note that, although we assume that resources cannot be fractionally allocated to a task, it is straightforward to extend our methods to a setup where a resource can be allocated in multiples of a certain given fraction $\frac{1}{k}$, by creating k copies of the resource each with $\frac{1}{k}$ of the cost and size of the original resource.

D. *Organization of the Paper*

The rest of the paper is organized as follows. In Section II, we show that the fair allocation problem is strongly NP-hard even for unit size resources. Then we formulate the problem as an integer linear program and show that a naive relaxation has an unbounded integrality gap in Section III, for unit sizes. We also show that when resources have arbitrary sizes and costs, any trivial modifications of the LP considered will still result in an unbounded integrality gap. Hence, we focus on the unit size resource problem the next section onwards. We give a $2 + O(\epsilon)$ approximate LP rounding algorithm for this problem in Section IV, following which we show that the LP considered has an integrality gap of 2 in Section V which essentially shows that our bound is tight. In Section VI, we present a greedy algorithm, that in a restricted scenario, achieves near optimal performance and finally Section VII concludes the paper.

II. NP-HARDNESS

In this section, we show that the problem of fair resource allocation to multiple tasks, is strongly NP-hard even when the resources are of unit size and have different costs and the tasks are all identical in the sense that the resource requirement of each task is same. The reduction is from 3-Partition.

Theorem 1: The feasibility problem of the fair resource allocation to multiple tasks is strongly NP-hard.

Proof: Consider an instance of the 3-partition problem. There are $n = 3m$ integers $\{a_i | i \in [1, \ldots, n]\}$ such that the sum of the integers $\sum_{i \in [1,\ldots,n]} a_i = mB$, and each integer a_i is strictly between $\frac{B}{4}$ and $\frac{B}{2}$, i.e., $\frac{B}{4} < a_i < \frac{B}{2}$. The feasibility question is whether there exists a partition of the n integers into m partitions such that the sum of the integers in each partition is exactly equal to B. Note that this would require every partition to have exactly 3 integers. Now, let us define a fair resource allocation problem, where we have m tasks, each requiring 3 units of resource. We create $n = 3m$ resources, each of unit size, where the cost of each resource is $c_i = a_i$. Let us order the integers in the 3-partition instance in non-decreasing order of their sizes. Let the sum of the 3 smallest integers be I. The isolation cost of every task is therefore equal to I. The feasibility question we ask is whether $\gamma = \frac{B}{I}$ is feasible. If there exists a feasible solution to the 3-partition instance, that implies that $\gamma = \frac{B}{I}$ is feasible, since this corresponds to 3 units of resources per task, and the cost incurred by every task is B. At the same time, if there exists a feasible $\gamma = \frac{B}{I}$ for the fair resource allocation problem, then this implies that the 3-partition instance is feasible. Every task has received 3 resources, and the cost incurred by every task must be $\leq \gamma I = \frac{B}{I} I = B$. Since we have allocated every resource, the total sum of the costs incurred by all m tasks is mB, hence no task can cost $< B$, as that would make another task cost $> B$, which cannot happen, γ being feasible. As a result, this requires every task to cost exactly B. Therefore, the resource allocations to the m tasks corresponds to feasible m partitions of the integers in 3-partition, where the sum of the integers in every partition is exactly B. ■

Minimize γ subject to the following constraints:

I1.	$\sum_{i \in \mathcal{E}} x_{i,j} \times c_i \leq \gamma \times I_j$	$\forall j \in \mathcal{P}$
I2.	$\sum_{i \in \mathcal{E}} x_{i,j} \times s_i \geq D_j$	$\forall j \in \mathcal{P}$
I3.	$\sum_{j \in \mathcal{P}} x_{i,j} \leq 1$	$\forall i \in \mathcal{E}$
I4.	$x_{i,j} \in \{0, 1\}$	$\forall i \in \mathcal{E}, j \in \mathcal{P}$

Fig. 1. An integer linear program for the fair allocation problem.

III. ILP FORMULATION, LP RELAXATION AND INTEGRALITY GAP

In this section, we formulate the problem as Integer Linear Program (ILP), then relax it to a Linear Program (LP) and show that the problem in the generic case where resources may have different costs and sizes has an unbounded integrality gap.

Recall that the aim is to find an allocation of resources to tasks that minimizes the ratio of actual cost of allocation to the isolation cost, for all tasks. Formally, we want to minimize γ, where $\gamma = \max_{j \in \mathcal{P}} \frac{C_j}{I_j}$. This is subject to fulfilling the resource requirement D_j for all tasks $j \in \mathcal{P}$. We formulate this problem as an ILP, shown in Figure 1.

In the formulation given in Figure 1, γ represents an upper bound on the ratio of C_j to I_j across all tasks $j \in \mathcal{P}$, and since we are minimizing it, γ is the smallest possible value for the required objective, i.e., $\min \max_{j \in \mathcal{P}} \frac{C_j}{I_j}$, that can be achieved by any integral allocation. In the formulation , the inequality $I1$ ensures that the cost incurred by a task j due to resource allocation is at most $C_j = \gamma I_j$, inequality $I2$ ensures that the total resource allocated to a task is no less than its resource requirement D_j, inequality $I3$ ensures that no resource must be over allocated, and finally $I4$ ensures that every resource must be integrally allocated to a task, if at all.

A naive linear relaxation of this ILP formulation would relax $I4$ to $x_{i,j} \geq 0$. However, such a relaxation has an unbounded integrality gap as illustrated next. Consider m tasks, each with a resource requirement of $D_j = 1$, and m resources of which $m-1$ resources have a cost 1 and one resource has a cost m. All resources are of unit-size, i.e., $s_i = 1$. Any integral allocation would have to allocate the high cost resource to one of the tasks, hence incurring a $\gamma = m$. However, the linear program would allocate $\frac{1}{m}$ of each resource to each task. This will meet the resource requirement of every task since $m \cdot \frac{1}{m} = 1$, while the cost incurred by every task is $m\frac{1}{m} + \frac{m-1}{m} = 2 - \frac{1}{m}$. Therefore, the integrality gap is $\geq \frac{m}{2-\frac{1}{m}} > \frac{m}{2} \rightarrow \infty$ as $m \rightarrow \infty$.

To overcome this, we use the *parametric pruning* technique, similar to the seminal work of Lenstra et al. [8]. We guess the optimal value of γ, and solve a feasibility linear program. (Later in the section, we show that γ can be guessed in a logarithmic number of iterations.) For each guess of γ, we solve the feasibility linear program shown in Figure 2, where in order to avoid giving an unfair advantage to the linear program, we allow the LP to assign a resource i to a task j, only if $c_i \leq \gamma \times I_j$.

C1.	$\sum_{i \in \mathcal{E} \mid c_i \leq \gamma I_j} x_{i,j} \times c_i \leq \gamma I_j$	$\forall j \in \mathcal{P}$
C2.	$\sum_{i \in \mathcal{E} \mid c_i \leq \gamma I_j} x_{i,j} \times s_i \geq D_j$	$\forall j \in \mathcal{P}$
C3.	$\sum_{j \in \mathcal{P}} x_{i,j} \leq 1$	$\forall i \in \mathcal{E}$
C4.	$x_{i,j} \geq 0$	$\forall i \in \mathcal{E}, j \in \mathcal{P}$

Fig. 2. An LP relaxation of ILP shown in Figure 1.

However, if the resources are of arbitrary sizes, then the linear program shown in Figure 2 still has an unbounded gap[1].

Further, simple parametric pruning along the size dimension, specifically, allowing the LP to allocate a resource i to a task j, only if $s_i \leq D_j$, still cannot remove the gap. Consider the LP shown in Figure 3. This LP still has an unbounded integrality gap, and trivial modification of this LP cannot remove this gap. We show this with an example. Consider m tasks where each task has a resource requirement of m. Let there be $2m-1$ resources of which $m-1$ resources have a cost of m and a size of m, and m resources have a cost of m and a size of 1. Any integral feasible solution would have to assign the $m-1$ resources of size m to $m-1$ tasks and the remaining m resources of unit size to another task. The task receiving the m unit size resources incurs a cost of m^2, whereas the isolation cost of every task is m, which means any integral solution would incur a γ of m. Since the isolation cost of every task is m, and the resource requirement of every task is m, and further since $\forall i, j : s_i \leq D_i$, every resource is feasible to be allocated to every task by the LP, for every $\gamma \geq 1$. A feasible LP solution therefore, allocates each of the $m-1$ resources of size m to each of the m tasks to an extent of $\frac{1}{m}$, and one unit size resource integrally to each task. The total resource received by every task is therefore $(m-1)m\frac{1}{m} + 1 = m$. The total cost incurred by every task is $(m-1)m\frac{1}{m} + m = 2m-1$. Therefore, the LP will return a feasible solution for a $\gamma = \frac{2m-1}{m} < 2$, whereas any integral solution will incur a γ of m, giving an integrality gap of $\frac{m}{2} \rightarrow \infty$, as $m \rightarrow \infty$. Note that if we try to modify the LP by restricting the allocation of resources which have a large ratio of cost to size, compared to the ratio of isolation cost to the resource requirement of a task, may render the LP infeasible, when a feasible solution exists, or suboptimal. Hence, in the rest of the paper, we will only consider the case where all resources are of unit size, but may have different costs.

C1.	$\sum_{i \in \mathcal{E} \mid c_i \leq \gamma I_j \, \wedge \, s_i \leq D_j} x_{i,j} \times c_i \leq \gamma I_j$	$\forall j \in \mathcal{P}$
C2.	$\sum_{i \in \mathcal{E} \mid c_i \leq \gamma I_j \, \wedge \, s_i \leq D_j} x_{i,j} \times s_i \geq D_j$	$\forall j \in \mathcal{P}$
C3.	$\sum_{j \in \mathcal{P}} x_{i,j} \leq 1$	$\forall i \in \mathcal{E}$
C4.	$x_{i,j} \geq 0$	$\forall i \in \mathcal{E}, j \in \mathcal{P}$

Fig. 3. Further restricted LP relaxation (still with an unbounded integrality gap for arbitrary sizes.

Guessing γ: We next show that we can guess a near optimal γ in a logarithmic number of iterations. The minimum value of γ is given by $\gamma_{min} = 1$. For every guess of γ, we allow a resource i to be allocated to a task j, only if $c_i \leq \gamma I_j$, i.e., $c_i \leq C_j$.

Without loss of generality, we assume that the minimum cost of any resource is 1, since the costs are rational numbers, and can always be made integral by scaling the problem. Similarly, we assume that the sizes of the resources are 1, and the resource requirement for every task D_j is integral.

Observation 2: The maximum value of γ is $\gamma_{max} = c_{max}$, where $c_{max} = \max_{i \in \mathcal{E}} c_i$.

Proof: Suppose that the resource requirement of a task τ_j with the highest value of γ is D_j. In the worst case, the task is allocated those resources that have the highest cost, hence $C_j \leq D_j \times c_{max}$. However, the isola-

[1]Consider a system with 2 tasks, each with a resource requirement of $D_j = 1$, and 2 resources of which resource 1 has a size = 2 and a cost 1, and resource 2 has a size 1 and a cost $L \gg 1$. It can be seen that the isolation cost of both tasks is 1. Any integral allocation would have to allocate the high cost resource to one of the tasks, resulting in a $\gamma = L$. However, the linear program would return a feasible solution for $\gamma = 1$, by allocating fraction $\frac{1}{2}$ of resource 1 to both tasks.

tion cost of the task can be expressed as $I_j \geq D_j$, since minimum cost of any resource is ≥ 1. Therefore, the maximum value of γ is given by $\gamma \leq \frac{D_j \times c_{max}}{D_j} = c_{max}$. ■

With that, we can do a binary search for γ in the range $[\gamma min, \gamma_{max}] = [1, c_{max}]$, using a resolution of ϵ and guess the optimal γ in $\log_2 (\frac{c_{max}}{\epsilon})$ iterations.

The next theorem shows that a near optimal value of γ can be found by such a method.

Theorem 3: Let $\gamma_{OPT-INT}$ be the objective function value for an integral optimal solution (where the optimality is with respect to the fairness objective). Let γ_{OPT-LP} be the smallest γ for which the linear program LP is feasible. Then $\gamma_{OPT-LP} \leq \gamma_{OPT-INT} + \epsilon$.

Proof: Let the smallest γ for which the LP is feasible be γ_{OPT-LP}. By definition of γ_{OPT-LP} and by the property of binary search it must hold that for $\gamma_{OPT-LP} - \epsilon$, the LP must have been infeasible. Therefore, there exists no feasible resource allocation satisfying the specified constraints in the LP, such that the cost for every task j is $\leq (\gamma_{OPT-LP} - \epsilon) I_j$ and allocation is $\geq D_j$, and every resource is allocated to unity. This implies that in any feasible (integral or otherwise) allocation of resources satisfying the resource requirement D_j for every task j, there must exist a task j' such that $C_{j'} > (\gamma_{OPT-LP} - \epsilon) I_{j'}$. Since any the optimal integral allocation is a feasible allocation (satisfying the LP constraints), this implies that there exists a task j'' in the optimal integral allocation, such that the cost incurred by j'' is $C_{j''} > (\gamma_{OPT-LP} - \epsilon) I_{j'}$, therefore, $\gamma_{OPT-INT} \geq \frac{C_{j''}}{I_{j''}} > (\gamma_{OPT-LP} - \epsilon)$. This completes the proof. ■

In the remainder of the paper, we assume that the resources are of unit size. Note that, with unit-sized resources, the isolation cost I_j of a task j, with demand D_j, is simply the sum of the cost of D_j lowest cost resources. We also assume that there are sufficient resources in the system to satisfy the demand of all tasks simultaneously, but possibly with different allocation costs.

IV. LP Rounding Algorithm

In this section, we present a polynomial LP rounding algorithm with a $2 + O(\epsilon)$ approximation to the fair resource allocation problem. The ϵ factor is due to the binary search performed over the space of γ with a resolution ϵ. *We later (in Section V) show that for unit size resources, even the LP has an integrality gap approaching* 2*, hence our rounding algorithm is essentially tight.*

The rounding proceeds as follows. First, we convert a feasible LP solution to another feasible solution where every resource, if allocated, is allocated up to unity. In other words, no resource has an overall fractional allocation. Once we have done this conversion, we essentially have a perfect fractional matching, where every resource is allocated to unity, and every task's resource requirement is exactly satisfied, while the cost of every task j is $\leq \gamma I_j$. Now, we convert this to an integral, feasible, perfect matching, such that the total cost of the resources allocated to every task j is at most twice γI_j, where the LP was feasible for γ. This procedure is described in detail in Section IV-A.

We now show how to convert a feasible LP solution to a perfect fractional matching, where every resource is allocated to unity, and every task's resource requirement is exactly satisfied, while the cost of every task j is $\leq \gamma I_j$.

Lemma 4: Let γ_{LP-OPT} be the smallest value for which the LP is feasible. Then there exists a feasible LP solution σ for the same value of γ, in which any resource that has a non-zero allocation, is allocated to an extent of 1; formally, if $\exists i \in \mathcal{E}, \exists j \in \mathcal{P}$ such that $x_{ij} > 0$ then it holds that $\sum_{j \in \mathcal{P}} x_{ij} = 1$.

Proof: Informally, the proof is based on the following reasoning. A more formal proof follows the informal discussion. Depending on whether one or more resources are fractionally allocated, we need to consider two cases.

If there exists one resource i that is fractionally allocated (i.e., $x_i = \sum_{j \in \mathcal{P}} x_{i,j} < 1$) then in a feasible solution, some tasks must be over-allocated. This is because (i) total resource requirement $D = \sum_{j \in \mathcal{P}} D_j$ of tasks is integral since individual resource requests D_j of each task are integral, and (ii) resources are unit size, (iii) only one resource is fractionally assigned, therefore, any feasible solution must have assigned $\geq D$ resources to unity. Hence, there must be over-allocation of resources and this over allocation sums to x_i which can be eliminated by readjusting the resource allocation of tasks without violating the feasibility of the solution thereby making all the resource allocations integral.

If there exists multiple resources that are fractionally assigned then considering a pair of such resources at a time and by readjusting their resource allocations (decreasing one of the fractional allocation and increasing the other fractional allocation) systematically converts *at least* one of the fractional resource allocations into an integral allocation without violating the feasibility of the solution. Repeating this process either eliminates all the fractional allocations or leaves one fractional allocation in which case we can use the procedure described for the first case.

A formal proof follows now. We prove the claim by contradiction. Suppose that the claim is not true. Then we need to consider two cases.

Case 1: A single resource has an allocation $x_i = \sum_{j \in \mathcal{P}} x_{i,j} < 1$. We have already argued that at least D resources must have been allocated to unity, otherwise, the total resource requirement of D cannot be met. For every task j for which resource i has a non-zero allocation to, we first reduce $x_{i,j}$ by $\sum_{k \in \mathcal{E}} x_{k,j} - D_j$, without any loss of feasibility. Now, let $\mathcal{P}'$ be the set of tasks for which resource i has a non-zero allocation after this modification. All the tasks in $\mathcal{P}'$ are now exactly satisfied in terms of their resource requirements, i.e., $\sum_{k \in \rangle} x_{k,j} = D_j \ \forall j \in \mathcal{P}'$. Let us consider two scenarios depending on the cost c_i of the fractional resource i.

Scenario 1.a. First consider the scenario, where all of the resources allocated to unity, cost at most c_i. Since $\sum_{k \in \rangle, j \in \mathcal{P}} x_{k,j} > \sum_{j \in \mathcal{P}} D_j$, there must exist a task j', where $\sum_{k \in \rangle} x_{k,j'} > D_{j'}$. Let the highest cost resource allocated to task j' be i'. Since $c_i \geq c_{i'}$ and resource i has a non-zero allocation to task j, resource i' is feasible to be allocated to j. Now, we decrease the allocation of resource i' to task j' by $\delta = \min \left(x_{i',j'}, x_{i,j}, \sum_{k \in \mathcal{E}} x_{k,j'} - D_{j'} \right) > 0$ and increase the allocation of resource i' to task j by δ, and decrease the allocation of resource i allocation to task j by δ. The resultant allocation is still feasible in terms of resource requirements to all tasks, and the cost of every task is at most the cost of the earlier allocation. Since by definition, $\delta > 0$, we have decremented $x_{i,j}$. As long as $x_{i,j} > 0$, there is always a task j'' that has been over allocated, and we repeat the above transformations, till we have removed the allocation of resource i to task j, or, $x_{i,j} = 0$. If $x_i > 0$, then there must exist another task p to which resource i has a non-zero allocation. We repeat the above steps for resource i and task p. We continue till we have removed the allocation of i to every project, without violating feasibility, and removing the fractional resource. This takes a polynomial number of transformations.

Scenario 1.b. Now consider the scenario where there exists a resource that has been allocated to unity and whose cost is greater than c_i. From the set of resources allocated to unity, let k be the highest cost resource. Clearly, $c_k > c_i$. Now consider a task j' to which resource k is allocated. Increase the allocation of resource i to task j' by $\delta = \min (x_{k,j'}, 1 - x_i)$, while decreasing allocation of resource k to task j' by δ. In the process, we have not violated feasibility since the resource requirement is still met every where, every resource is allocated at most to unity, and the costs of the tasks after this transformation is at most what they were earlier. After this transformation, if we have made resource i allocated up to unity, then resource k becomes the new fractionally allocated resource having the highest cost of any non-zero allocated resource, and hence we can now apply the procedure of Scenario 1.a to reduce its allocation to 0. Otherwise, i.e., if resource i is not allocated to unity then $x_{k,j'} = 0$. Note that, even after this transformation, $x_k > 0$ since $\delta < 1$. Therefore, there must exist another task j'' where resource k has a non-zero allocation. We repeat the above process, till we make resource i allocated to unity. This will always be possible since resource k is allocated to unity, and $1 - x_i < 1$. Now, resource k becomes the new fractionally allocated resource, and it costs the most among all the resources allocated, hence this reduces to Scenario 1.a. We repeat the transformations outlined in Scenario 1.a, till we reduce x_k to 0.

Case 2: More than one resource has an allocation < 1. Let us consider a pair of resources (i, i') both of which are fractionally allocated. Without loss of generality, let $c_i \leq c_{i'}$. Suppose $x_i + x_{i'} \leq 1$. In this case, for every task j that resource i' has a non-zero allocation to, we make $x_{i,j} = x_{i',j}$ and reduce $x_{i',j}$ to 0. In this way, we have reduced the number of resources with fractional allocation by at least 1. We repeat this for every pair of fractional resources, till we are left with at most one fractional resource. Then we perform the transformations described in Case 1. Now, suppose $x_i + x_{i'} > 1$. Again, we repeat the above procedure for every task that resource i' has a non-zero allocation to, till we come across a task, where $x_{i',j} > 1 - x_i$, where x_i refers to the current fractional allocation of resource i after the above transformations. In this case, we set $x_{i,j} = 1 - x_i$, and $x_{i',j} = x_{i',j} - x_{i,j}$. Now, we have again reduced the number of fractional resources by 1. We repeat the above described procedures for every pair, till we are left with at most one fractional resource, which corresponds to Case 1.

This completes the proof. ■

We now describe the rounding algorithm.

A. Description of the LP Rounding Algorithm

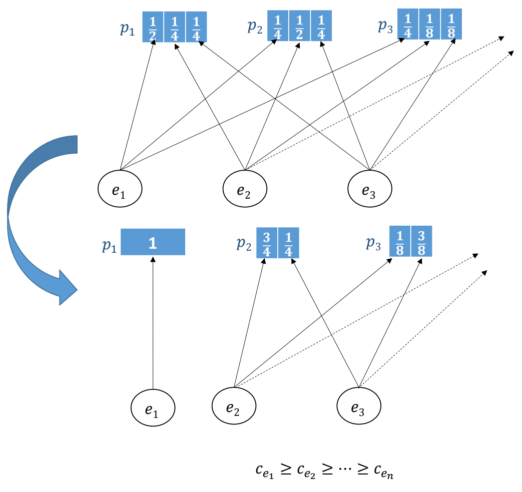
The solution σ is a set of connected components in a bipartite graph, with one vertex partition $V_1 = \{i \in \mathcal{E} | \sum_{j \in \mathcal{P}} x_{i,j} > 0\}$ being the set of resources with non-zero allocation. In fact, after the polynomial transformation described in Lemma 4, $V_1 = \{i \in \mathcal{E} | \sum_{j \in \mathcal{P}} x_{i,j} = 1\}$. The other partition V_2 is the set of tasks $\mathcal{P}$. We have already argued that we have a perfect fractional matching in this bipartite graph, which we will now convert to a perfect integral matching, without violating the cost constraints too much.

From σ, we first remove all the edges that are integral, allocating the corresponding resource to the corresponding task, without any loss in feasibility. Let $V_1' = \mathcal{E}'$ be the set of resources who are yet unallocated and $V_2' = \mathcal{P}'$ be the set of tasks, that have still not been fully allocated, after removing the integral edges. Since all the resource requirements are integral, and all resources are allocated to unity, the total number of resource available is no less than the total resource requirement over all tasks, i.e., $|V_1'| \geq \sum_{p \in V_2'} D_p$, i.e., we still have a perfect fractional matching.

We order the resource vertices in non-increasing order of their costs, and assume henceforth that they are numbered accordingly, namely $e_1, e_2, \dots, e_{|V_1'|}$, where $c_{e_1} \geq c_{e_2} \geq \dots c_{e_{|V_1'|}}$. The tasks are ordered in any arbitrary order, $p_1, p_2, \dots, p_{|V_2'|}$. Now, we use a procedure, which is somewhat inspired by the analysis of the generalized assignment problem, in the seminal work of Shmoys and Tardos [10] to transform the current fractional matching to an integral matching. The algorithm proceeds as follows. Start with the highest cost resource and allocate it integrally to any task that it has a non-zero allocation to. Now, adjust the allocation of the resources to maintain feasibility. Then, proceed to completely allocate this task. Once done, pick the next highest cost resource, that is not yet integrally allocated and repeat the procedure. We describe this in detail in the following paragraphs.

Let p_j be a task where resource e_1 has a non-zero allocation, in other words, (e_1, p_j) is an edge in this connected component. We allocate resource e_1 to resource p_j integrally (i.e., $x_{e_1,p_j} = 1$), and remove e_1 from the set V_1'. Now, we find the lowest resource index e_i such that $\sum_{e_k | k \in \{1,\dots,i\}} x_{e_k,p_j} \geq 1$, and $\sum_{e_k | k \in \{1,\dots,i-1\}} x_{e_k,p_j} < 1$. We remove the edge x_{e_i,p_j} and split the node i into two nodes i_1 and i_2, and add two edges $x_{e_{i_1},p_j} = \sum_{e_k | k \in \{1,\dots,i\}} x_{e_k,p_j} - 1$, and $x_{e_{i_2},p_j} = x_{e_i,p_j} - x_{e_{i_1},p_j}$.

We have made $x_{e_1,p_j} = 1$. Therefore to maintain feasibility, we have to make $x_{e_1,p_q} = 0 \; \forall p_q \in \mathcal{P}', q \neq j$. Note that all the resources e_k, $1 < k \leq i$, have costs $\leq c_{e_1}$, and hence are feasible to be allocated to any task that resource e_1 was allocated to. Now, consider a task j' to which resource e_1 had a non-zero allocation. We know that all of the resources $e_k, 2 \leq k \leq i$ are feasible to be allocated to j'. Hence, we add an edge from resource e_2 to task j' of value $x_{e_2,j'} = \min(x_{e_2,j}, x_{e_1,j'})$ to j' and decrement $x_{e_2,j}$ by $x_{e_2,j'}$. Now, if $x_{e_2,j'} = x_{e_1,j'}$, we consider the next task j'' that resource e_1 had an allocation to, and repeat the above process, with e_2 if $x_{e_2,j} > 0$. Otherwise, if $x_{e_2,j} = 0$, and $x_{e_2,j'} < x_{e_1,j'}$, we pick e_3 do the same procedure. We continue till either we have removed all the edges (e_k, p_j) for all $1 \leq k < i$ and $x_{e_{i_1},p_j}$, or we have have removed all edges x_{e_1,p_q} for all $q \neq j$, in a



p_1
$\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{4}$
p_2
$\frac{1}{4}$ $\frac{1}{2}$ $\frac{1}{4}$
p_3
$\frac{1}{4}$ $\frac{1}{8}$ $\frac{1}{8}$
e_1
e_2
e_3
p_1
1
p_2
$\frac{3}{4}$ $\frac{1}{4}$
p_3
$\frac{1}{8}$ $\frac{3}{8}$
e_1
e_2
e_3
$c_{e_1} \geq c_{e_2} \geq \cdots \geq c_{e_n}$

Fig. 4. Illustration of one iteration of the LP rounding algorithm.

polynomial number of transformations ($O(|V_1'| + |V_2'|)$). Note that $x_{e_{i_1},p_j} + \sum_{e_k | k \in \{2,\dots,i\}} x_{e_k,p_j} = 1 - x_{e_1,p_j}$, therefore, both the events will happen simultaneously, after which we have a feasible solution, where every resource in V_1' is allocated up to unity and every task in V_2' is fully allocated. If task p_j had only one resource requirement, then we have allocated task p_j integrally, while not violating the cost constraint of p_j, as $c_{e_1} \leq C_j$. Otherwise, if $D_j > 1$, then we still have unmet resource requirement in task p_j.

Now, we find the lowest resource index $u > 1$, that has a non-zero allocation to task p_j, i.e., $x_{e_u,p_j} > 0$ and repeat the above. If $x_{e_{i_2},p_j} > 0$, then $u = i$, otherwise, $u > i$. We allocate resource u integrally to task p_j, remove the resource from V_1', and repeat the above procedure. We continue till p_j is allocated D_j integral resources.

Once task p_j is fully allocated integrally, we remove it from V_2', and find the next highest cost resource e_s remaining in V_1'. Let p_r be a task that e_s has a non-zero allocation to. We perform the same transformations on resource e_s and task p_r and continue, till both $V_1' = \emptyset$ and $V_2' = \emptyset$. Figure 4 provides an example for one iteration of the rounding algorithm.

Observe that we allocate only one resource integrally to a task at a time, and adequately remove some assignments to maintain feasibility of total allocation of any resource, and resource allocation of any task. Since, initially $|V_1|' \geq \sum_{p_k \in V_2'} D_{p_k}$, and we are not violating the size constraints on resource allocations at any iteration, we will always find a resource to allocate to a task yet unsatisfied, that is, if $V_2 \neq \emptyset$, that implies $V_1 \neq \emptyset$. Not only that, at any iteration, when we change the allocations, we maintain feasibility in terms of the assignment restrictions on the resources; in other words, we allow a resource to be allocated to a task, only if the cost of the resource is at most the total cost allowed for the task. Moreover, at any iteration, when we are processing the resources allocated to a task q, we are removing D_q vertices from V_1' and 1 vertex from V_2'. Therefore, in at most $|V_1|$ iterations, we will have allocated every resource and and every task integrally.

B. *Cost Incurred due to Rounding Algorithm*

In this section, we argue that the resultant integral allocation will be at most $2C_j$ for every task j. Note that the first resource e_1 that is allocated to any task p_j has a non-zero allocation to p_j, hence, $c_{e_1} \leq C_j$. Let the next resource to be allocated be e_u, where $u > 1$. Also, let i be the lowest index such that $\sum_{e_k|1\leq k\leq i} x_{e_k,p_j} \geq 1$. Note that, $u \geq i$. According to the procedure outlined earlier, we must have added two vertices i_1 and i_2 to the graph, in place of i, and replaced the edge x_{e_i,p_j} by $x_{e_{i_1},p_j}$ and $x_{e_{i_2},p_j}$, such that $\sum_{e_k|1\leq k<i} x_{e_k,p_j} + x_{e_{i_1},p_j} = 1$, and $x_{e_{i_2},p_j} = x_{e_i,p_j} - x_{e_{i_1},p_j}$. In this case, $C_1 = \sum_{e_k|1\leq k<i} c_{e_k} x_{e_k,p_j} + c_{e_i} x_{e_{i_1},p_j}$. Clearly, $C_1 \geq c_{e_i} \sum_{e_k|1\leq k<i} x_{e_k,p_j} + c_{e_i} x_{e_{i_1},p_j} = c_{e_i}$ as $\sum_{e_k|1\leq k<i} x_{e_k,p_j} + x_{e_{i_1},p_j} = 1$. Therefore, $c_{e_i} \leq C_1$, hence we charge its cost to C_1 at no additional cost.

Similarly, let the next resource to be integrally allocated to j is e_v, $v > u$. This means $\sum_{e_k|u\leq k\leq v} x_{e_k,p_j} > 1$, and let e_w be the lowest index such that $\sum_{e_k|u\leq k\leq w} x_{e_k,p_j} \geq 1$. Note that $v \geq w$. Again, following the same of procedure of adding two vertices (say w_1 and w_2) and replacing the edges as described above, we obtain $c_{e_w} \leq C_2$ and hence we charge its cost to C_2 at no additional cost where $C_2 = \sum_{e_k|u\leq k<w} c_{e_k} x_{e_k,p_j} + c_{e_w} x_{e_{w_1},p_j}$

The total cost $C_j = C_1 + C_2 + \ldots C_{D_j}$, where C_is are defined above, and we have charged the cost of the k^{th} resource allocated to p_j, to C_{k-1}, where $k \in \{2, \ldots, D_j\}$, therefore, the total cost of these resources is $\leq C_j$. The first resource to be allocated has a cost $c_{e_1} \leq C_j$. Therefore, the cost of the total integral allocation to task p_j is at most $2C_j$. We repeat the above argument for every task.

C. *Approximation Ratio of LP Rounding Algorithm*

Theorem 5: Given a γ for which a feasible LP solution exists, there exists a polynomial time algorithm that gives an integral feasible resource allocation to all tasks, such that the cost of any task p_j, is at most $2\gamma I_j$, where I_j is its isolation cost.

Proof: The proof follows from the discussion in Sections IV-A and IV-B. ■

Theorem 6: There exists a polynomial $2 + O(\epsilon)$ approximation algorithm to the integral fair resource allocation problem.

Proof: From Theorem 5, we know that given a γ, for which a feasible LP solution exists, we can find an integral feasible solution, where every task costs at most $2C_j = 2\gamma I_j$. From Theorem 3, we know that the lowest value for which the LP is feasible is, $\gamma_{LP-OPT} \leq \gamma_{OPT-INT} + \epsilon$, and it can be found in at most $\log\left(\frac{c_{max}}{\epsilon}\right)$ iterations, for any fixed $\epsilon > 0$. Hence the resultant value of γ incurred by our algorithm is $\leq 2\gamma_{LP-OPT} \leq 2\gamma_{OPT-INT} + 2\epsilon$. This proves the theorem. ■

V. Integrality Gap for Unit Size Resources

In this section we show that the LP has an integrality gap approaching 2.

Theorem 7: The LP has an integrality gap $\rightarrow 2$, hence the LP rounding algorithm of Section IV is tight.

Proof: Consider an instance with m tasks, each with a resource requirement of m. Let there be m^2 resources of which $m^2 - 1$ resources are of cost 1 and another resource is of cost m. Any feasibly integral solution would have to allocate the resource of cost m to one of the tasks thereby incurring a cost of $2m-1$. However, the isolation cost I of every task is m. Therefore, any integral feasible solution would incur a $\gamma \geq \frac{2m-1}{m} = 2 - \frac{1}{m}$. For the LP, every resource is feasible to be allocated to every task for every $\gamma \geq 1$, since the isolation cost of every task is m and the resource requirement of every task is m. Let us consider $\gamma = 1 + \frac{1}{m}$. A feasible LP solution would allocate to every task, $m^2 - m$ unit cost resources integrally, $m - 1$ unit cost resources to an extent of $\frac{1}{m}$, and $\frac{1}{m}$ of the resource of cost m. With such an allocation, every task receives a resource allocation of m units, and every resource is allocated up to 1. Note that the overall number of resources allocated is $m^2 - m + m - 1 + 1 = m^2$. The cost incurred by every task is $(m-1) + (m-1)\frac{1}{m} + m\frac{1}{m} = m + 1 - \frac{1}{m} < m+1$, which is feasible since $\gamma = \frac{m+1}{m}$. Therefore, the integrality gap is at least $\frac{2-\frac{1}{m}}{1+\frac{1}{m}} \rightarrow 2$, when $m \rightarrow \infty$. Hence the proof. ■

VI. Greedy algorithm

In this section, we consider a restricted version of the problem, where the costs of the resources, though arbitrary, vary smoothly across resources. The resources are unit sized as in the previous sections. We give a greedy algorithm which will give a near optimal fairness in resource allocation, when all the tasks have the same resource requirement. We first define some notations.

Definition 8: A set of resources are called *homogenous*, if, when ordered in non-increasing order of their costs, the difference in costs between two consecutive resources is very small, $\leq \epsilon$ for some small fixed $\epsilon > 0$. In other words, assuming the resources are numbered according to their position in the sorted order, for any $i \in \{1, \ldots, |\mathcal{E}|\}$, $c_{i+1} - c_i \leq \epsilon$. We assume that $c_i \geq 1$.

Definition 9: A set of tasks are called *identical* in terms of their resource requirements, if their resource requirements are identical. In other words, for every pair of tasks $j, j' \in \mathcal{P}$, $D_j = D_{j'} = D$, where $D \geq 1$.

In this version of the problem, we assume the resources are homogeneous and give a greedy algorithm which allocates resources to the tasks in iterations, till every task is fully allocated its total resource requirement. In every iteration, the task to allocate a resource is chosen according to a rule which is a function of the total resource requirement of the task, remaining resource requirement and the cost incurred so far by the task due to allocated resources. To the chosen task, the algorithm allocates the next available resource in the sorted order. The choosing rule is as follows: pick the task j with the largest value of $f(j)$, i.e., $j = \arg\max_{k \in \mathcal{P}} f(k)$, where $f(j)$ is defined as: $f(j) = \frac{D'_j + \alpha C'_j}{D_j}$, where D_j is the total resource requirement of task j, D'_j is the remaining resource requirement or current deficit, C'_j is the current cost incurred by j by the resources already allocated. Note that the isolation cost of a task depends on the total resource requirement of the task, and is higher for tasks with high resource requirements. The intuition behind choosing this function is that we want to favor the tasks that have already incurred a high cost relative to the total resource requirement (hence, the isolation cost), and also the tasks for which the remaining resource requirement is high relative to the total resource requirement. By favoring a task, we mean allocating it lower cost resources. This is done to balance the ratio of actual costs to isolation costs across resources. The value of α is chosen so that the greedy algorithm behaves in a certain manner to ensure fairness of cost across tasks. Specifically, $\alpha = \frac{1}{\sum_{i \in \mathcal{E}} c_i}$.

The pseudo-code of the greedy algorithm is presented in Algorithm 1.

We will prove that when the resources are homogeneous and tasks are identical, the greedy algorithm will give a near optimal fairness ratio.

Let us denote the number of tasks $|\mathcal{P}| = n$, in the following analysis for notational ease, and the resource requirement of each as D. Since we have n tasks, each requiring D resources, the greedy algorithm will require nD iterations. Let us define the set of consecutive n iterations $[kn+1, kn+2, \ldots, (k+1)n]$ as the k^{th} block iteration. Clearly we have D block iterations.

We next prove that the greedy algorithm will allocate resources to the tasks in an alternating round-robin manner, such that in the i^{th} block iteration, where $i \in \{1, \ldots, D\}$, every task will be allocated exactly one resource.

Lemma 10: Every task is allocated a resource in every block iteration (in other words, consecutive n iterations).

Algorithm 1: Greedy algorithm for fair resource allocation.

```
Input : $\mathcal{P}$: set of tasks;   $D_j$: total resource requirement in
        task $j \in \mathcal{P}$   $D'_j$: current resource deficit in task
        $j \in \mathcal{P}$ (initially set to the original resource requirement
        of $j$;   $C_j$: current cost incurred by task $j \in \mathcal{P}$
        (initially set to 0);   $\mathcal{E}$: set of resources;   $c_i$: cost of
        a resource $i \in \mathcal{E}$
Output: $x_{ij}$: assignment of resources to tasks.
1  Set $D'_j \leftarrow D_j$ and $C'_j \leftarrow 0, \forall j \in \mathcal{P}$.
2  Sort all resources in $\mathcal{E}$ such that
   $c_i \leq c_{i+1}\ \forall i \in \{1, \ldots, |\mathcal{E}| - 1\}$
3  Set $x_{i,j} \leftarrow 0, \forall i \in \mathcal{E}, j \in \mathcal{P}$.
4  Set $i \leftarrow 1$;   while $\mathcal{P} \neq \emptyset$ do
5      Choose the task $j$, such that
       $j = \arg\max_{k \in \mathcal{P}} \left(\frac{D'_k + \alpha C'_k}{D_k}\right)$
6      Set $x_{i,j} \leftarrow 1$ and allocate resource $i$ to task $j$.
7      Set $D'_j \leftarrow D'_j - 1$ and $C'_j \leftarrow C'_j + c_i$
8      Set $i \leftarrow i + 1$. if $D'_j = 0$ then
9          $\mathcal{P} \leftarrow \mathcal{P} \setminus j$.
10     end
11 end
```

Proof: Let us consider the first block iteration. Suppose a task j is not allocated at all in block iteration 1. Since a block iteration consists of n rounds, this implies that some task j' is was allocated ≥ 2 resources in this iteration. Consider the round in the first block iteration, when j' was chosen for a second allocation. According to the choosing rule, therefore, $j' = \arg\max_{k \in \mathcal{P}} f(k)$, which implies $\left(\frac{D'_{j'} + \alpha C'_{j'}}{D}\right) \geq \left(\frac{D'_j + \alpha C'_j}{D}\right)$. But $D'_j = D$ whereas $D'_{j'} = D - 1$, $C'_j = 0$ and $C'_{j'} = c$, where c is the cost of the resource allocated to j'. Clearly, $\frac{D'_j + \alpha C'_j}{D} = 1$. On the other hand, $\frac{D'_{j'} + \alpha C'_{j'}}{D} = 1 - \frac{1}{D} + \frac{1}{D} \frac{c}{\sum_{k \in \mathcal{E}} c_k} < 1$. This gives a contradiction. Therefore, every task gets an allocation in the first block iteration.

Now, let us assume by induction hypothesis that this holds for the first i block iterations, where $i < D$. Now, all tasks have have i unit resources. If a task j does not get an allocation in the iteration $i+1$, then this implies that some task j' is was allocated ≥ 2 resources in this iteration. Consider the round in the $i+1$ block iteration, when j' was chosen for a second allocation. Since $j' = \arg\max_{k \in \mathcal{P}} f(k)$, $\left(\frac{D'_{j'} + \alpha C'_{j'}}{D}\right) \geq \left(\frac{D'_j + \alpha C'_j}{D}\right)$. Note that $D'_{j'} = D - (i+1)$, whereas $D'_j = D - i$. Because of our choice of α, $\alpha \frac{C'_{j'}}{D} < \frac{1}{D}$, therefore, $\alpha \frac{C'_{j'}}{D} - \alpha \frac{C'_{j'}}{D} < \frac{1}{D}$. Also, $\frac{D'_j}{D} = 1 - \frac{i}{D}$, whereas $\frac{D'_{j'}}{D} = 1 - \frac{i+1}{D}$. Therefore, $f(j') - f(j) < 0$. This gives a contradiction. Hence, we have proved inductively the statement of the lemma. ■

Observation 11: When all tasks have received the same number of resources, the next task to be allocated a resource is the task that has incurred the highest cost so far.

This follows from the function used by greedy as the choosing rule.

Lemma 12: At the end of every block iteration i, the cost incurred by any task j, C'_j is at most $C_{i,OPT} + n\epsilon$, where $C_{i,OPT} = \frac{1}{n}\sum_{e \in \{1,\ldots,i\cdot n\}} c_e$ is the optimal fair allocation cost when all the tasks are identical, each with a resource requirement of i, and the resources are homogeneous.

Proof: At the end of any block iteration i, let the costs incurred by tasks be C'_j $j \in [1,\ldots,n]$. Since any optimal solution would allocate the first $i \cdot n$ resources from the sorted list to the n tasks, the total cost incurred across all tasks in the greedy solution till iteration i is the same as that of any optimal solution. Therefore, $C_{i,OPT} = \frac{1}{n}\sum_{j\in[1,\ldots,n]} C'_j$. Let $C_{min,i}$ denote the lowest cost incurred by any task at the end of block iteration i and $C_{max,i}$ denote the highest cost incurred by any task. Clearly, $C_{i,OPT} \geq C_{min,i}$ by definition.

We will prove by induction that for any pair of tasks (j, j'). at the end of block iteration i, $|C'_j - C'_{j'}| \leq n\epsilon$. At the end of block iteration 1, $C_{min,1} = c_1$, and $C_{max,1} = c_n \leq c_1 + n\epsilon$, (where c_p is the cost of the p^{th} resource in the sorted order), by our assumption of homogeneous resources. Hence, the base case holds. Now, we assume by induction hypothesis, that the above holds for all block iterations $\{1,\ldots,i-1\}$. In the block iteration i, according to Observation 11 and Lemma 10, the task with the highest cost is allocated first, followed by the next highest cost, and so on, till the lowest cost task gets allocated last. Suppose at the end of block iteration i, $C'_{j,i} \geq C'_{j',i}$, where $C'_{j,i}$ denotes the cost incurred by j at the end of iteration i, hence j will allocated before j'. But from induction hypothesis, $C'_{j,i} \leq C'_{j',i} + n\epsilon$. Therefore, $C'_{j,i+1} - C'_{j',i+1} \leq c' - c'' + n\epsilon$, where c' is the cost of the resource allocated to j in block iteration $i+1$, and c'' is the cost of the resource allocated to j'' in block iteration $i+1$. Note that $c' \leq c''$ due to the sorted order and from Observation 11. Therefore, $C'_{j,i+1} - C'_{j',i+1} \leq n\epsilon$. At the same time, $C'_{j',i+1} - C'_{j,i+1} = C'_{j',i} + c'' - C'_{j,i} - c' \leq c'' - c'$, since $C'_{j',i} \leq C'_{j,i}$. But we know from homogeneous property of resources. $c'' - c' \leq n\epsilon$, therefore, $C'_{j,i+1} - C'_{j',i+1} \leq n\epsilon$. This holds for any pair of resources (j, j'). This completes the proof by induction.

We have proved that for any pair of tasks (j, j'). at the end of block iteration i, $|C'_j - C'_{j'}| \leq n\epsilon$. Therefore, $C_{max,i} \leq C_{min,i} + n\epsilon \leq C_{i,OPT} + n\epsilon$. ∎

Theorem 13: For homogeneous resources and identical tasks, the greedy algorithm gives a $1+\epsilon$ approximation to the fair resource allocation problem.

Proof: Let $C_{max,D}$ be the highest cost incurred by any task at the end of nD iterations, and C_{OPT} be the optimal fair allocation cost. For any optimal solution, $\gamma_{OPT} = \frac{C_{OPT}}{I}$ where I is the isolation cost of any resource. From Lemma 12, $C_{max,D} \leq C_{OPT} + n\epsilon$. Therefore, the resultant $\gamma_{greedy} \leq \frac{C_{OPT}+n\epsilon}{I}$. Hence, $\frac{\gamma_{greedy}}{\gamma_{OPT}} \leq 1 + \frac{n\epsilon}{C_{OPT}}$. We know that $C_{OPT} = nDc_{avg}$ where c_{avg} the average cost of the first nD resources. Therefore, $\frac{\gamma}{\gamma_{OPT}} \leq 1 + \frac{n\epsilon}{nDc_{avg}} < 1 + \epsilon$, since $D \geq 1$ and $c_{avg} \geq c_1 \geq 1$. ∎

VII. Concluding Remarks

This paper studied the problem of fair allocation of resources with heterogeneous costs to tasks with heterogeneous resource requirements. We show that the problem is strongly NP-Hard even with unit sized resources, and present an LP rounding approximation algorithm for this version of the problem and a near-optimal greedy algorithm for a special case in which costs of resources do not differ much and resource requirements of tasks are identical. As part of future work, we plan to study the problem where resources have limited heterogeneity in costs, as well as, the online version of the problem.

References

[1] Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 85–100, 2013.

[2] Venkatesan T Chakaravarthy, Anamitra Roy Choudhury, Sivaramakrishnan R Natarajan, and Sambuddha Roy. Knapsack cover subject to a matroid constraint. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 24. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[3] Bruno Escoffier, Laurent Gourvès, and Jérôme Monnot. Fair solutions for some multiagent optimization problems. *Autonomous agents and multi-agent systems*, 26(2):184–201, 2013.

[4] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: max-min fair sharing for datacenter jobs with constraints. In *Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys)*, pages 365–378, 2013.

[5] Raj Jain, Dah-Ming Chiu, and W Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report TR-301, DEC, 1984.

[6] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[7] H. J. Kushner and P. A. Whiting. Convergence of proportional-fair sharing algorithms under general conditions. *IEEE Transactions on Wireless Communications*, 3(4):1250–1259, 2004.

[8] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46(3):259–271, 1990.

[9] Alan Shieh, Srikanth Kandula, Albert G. Greenberg, Changhoon Kim, and Bikas Saha. Sharing the data center network. In *NSDI*, 2011.

[10] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474, 1993.

[11] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–546, 2001.