Multi-resource Fair Allocation with Bounded Number of Tasks in Cloud Computing Systems

Weidong Li, Xiaolu Zhang, Xuejie Zhang
Yunnan University, Kunming, 650091, PR China
E-mail: \{weidong, zxl, xjzhang\}@ynu.edu.cn

October 7, 2014

Abstract. We study a problem of multi-resource fair allocation with bounded number of tasks, and propose the lexicographically max-min dominant share (LMMDS) fair allocation mechanism, which is a generalization of the popular dominant resource fairness (DRF) mechanism. We prove that LMMDS satisfies sharing incentives, group strategy-proofness, envy-freeness, and Pareto efficiency, by exploiting the properties of the optimal solution. In addition, we design a non-trivial optimal algorithm to find a LMMDS fair allocation whose running time is linear in the number of users $n$, when the number of types of resources $m$ is bounded. Finally, we prove that the approximation ratio of LMMDS (or DRF) is infinity for the general case, and exactly $n$ for a special case, improving the previous best lower bound $m$.

Keywords: Lexicographically max-min dominant share; Dominant resource fairness; Multi-resource fair allocation; Approximation ratio.
1 Introduction

Multi-resource fair (or efficient) allocation is a fundamental problem in any shared computer system including cloud computing systems. As pointed by Ghodsi et al. [6], the traditional slot-based scheduler for state-of-the-art cloud computing frameworks (for example, Hadoop) can lead to poor performance, unfairly punishing certain workloads. Ghodsi et al. [6] are the first to suggest a compelling alternative known as the dominant resource fairness (DRF) mechanism, which is to maximize the minimum dominant share of users, where the dominant share is the maximum share of any resource allocated to that user. DRF is generally applicable to multi-resource environments where users have heterogeneous demands, and is now implemented in the Hadoop Next Generation Fair Scheduler [1].

In recent three years, DRF has attracted much attention and been generalized to many dimensions. Joe-Wong et al. [14] designed a unifying multi-resource allocation framework that captures the trade-offs between fairness and efficiency, which generalizes the DRF measure. Parkes et al. [9] extended DRF in several ways, including the presence of zero demands and the case of indivisible tasks. Bhattacharya et al. [1] adapted the definition of DRF to support hierarchies. Zeldes and Feitelson [15] proposed an online algorithm based on bottlenecks and global priorities. Kash, Procaccia and Shah [8] developed a dynamic model of fair division and proposed some dynamic resource allocation mechanisms based on DRF. Wang et al. [13] generalized the DRF measure into the cloud computing systems with heterogeneous servers. Psomans and Schwartz [11], Friedman, Ghodsi, and Psomas [5] studied the multi-resource allocation of discrete tasks on multiple machines.

Notably, Dolev et al. [4] suggested another notion of fairness, called Bottleneck-Based Fairness (BBF), which guarantees that each user either receives all he wishes for, or else gets at least his entitlement on some bottleneck resource. Gutman and Nisan [7] situated DRF and BBF in a common economics framework and provided a unifying polynomial-time algorithm to find a fair allocation. Very recently, Bonald and Roberts [3] argued

\footnote{1http://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-yarn-site/FairScheduler.html}
that proportional fairness is preferable to DRF, especially assuming that the population
of jobs in progress is a stochastic process. We refer to [10] to find other related works.

As mentioned by Wang, Liang and Li [13], the users have a finite number of tasks in
a real-world cloud computing system. They [13] studied the multi-resource allocation
problem in heterogeneous cloud computing systems with bounded number of tasks, and
proposed a generalized version of the well-known water-filling algorithm. However, the
running time of the water-filling is pseudo-polynomial. This motivates us to design a more
efficient algorithm. In this paper, we consider the the multi-resource allocation problem
in a single server with bounded number of tasks, which is a special case of the model
studied in [13]. We propose the lexicographically max-min dominant share (LMMDS)
fair allocation mechanism, and design a non-trivial polynomial time optimal algorithm.

2 Problem Definition

In a cloud computing system, assume that we are given a server with \( m \) resources and \( n \)
users. As in [6, 9], assume that each user \( u_j \) has a publicly known weight \( w_{ij} \) for resource
\( i \), which represents the amount of resource \( i \) contributed by user \( u_j \) to the resource pool.
Without loss of generality, assume that \( \sum_{j=1}^{n} w_{ij} = 1 \), for \( i = 1,2,\ldots, m \). Each user \( u_j \)
requires \( r_{ij} \)-fraction of resource type \( i \). For convenience, assume \( r_{ij} > 0 \) for all \( i, j \). Let
\( x_j \) be the number of tasks processed on the server for user \( u_j \). The resource requirement
constraints are

\[
\sum_{j=1}^{n} r_{ij} x_j \leq 1, \quad i = 1,2,\ldots, m. \tag{1}
\]

The weighted share of resource \( i \) for user \( u_j \) is \( r_{ij} x_j / w_{ij} \), and the dominant share for user
\( u_j \) is defined as

\[
D_j = \max\{ r_{ij} x_j / w_{ij} \mid i = 1,2,\ldots, m \}.
\]

The dominant resource fairness (DRF) mechanism [6] seeks to maximize the number
of allocated tasks \( x_j \), under the constraint that the dominant shares of the users are
equalized, i.e.,

\[ D_1 = D_2 = \cdots = D_n. \] (2)

The DRF allocation is equivalent to the solution for the following linear programming:

\[
\text{max } (x_1, x_2, \ldots, x_n)
\]

subject to (1)(2).

As mentioned in [13], the number of tasks need to be processed on the server is bounded in the realistic multi-resource environment, i.e.,

\[ x_j \leq B_j, \text{ for } j = 1, 2, \ldots, n. \] (3)

**Example 1.** Consider a system with of 18CPUs, 36GB RAM, and two users, where user \( u_1 \)'s task requires (1 CPU, 4GB), and user \( u_2 \)'s task requires (3 CPU, 1GB). Assume that \( w_{ij} = 1/2 \) for all \( i, j \). The DRF mechanism [6] will allocate (6 CPU, 24GB) to user \( u_1 \) and (12 CPU, 4GB) to user \( u_2 \), giving the allocation in Figure 1(a). With this allocation, each user ends up with the same dominant share 2/3. Assume the numbers of tasks for users need to be processed are \( B_1 = 5 \) and \( B_2 = 3 \), respectively. The server will allocate (5 CPU, 20GB) to user \( u_1 \) and (9 CPU, 3GB) to user \( u_2 \) to maximize system utilization, giving the allocation in Figure 1(b).

![Comparison](image-url)
For the bounded case, we observe that:

(a) The dominant shares may not be equal. In the above example, the dominant share of user $u_1$ is $\max\{5 \cdot 1/18, 5 \cdot 4/36\} = 5/9$, while the dominant share of user $u_2$ is $\max\{3 \cdot 3/18, 3 \cdot 1/36\} = 1/2$.

(b) There may not exist a bottleneck resource for the bounded case, where the bottleneck resource is the resource $i$ satisfying $\sum_{j=1}^{n} r_{ij} x_j = 1$. In the above example, both CPUs and RAM are not saturated.

Therefore, for the multi-resource fair allocation problem with bounded number of tasks, we want to allocate the resources as fair (may not equal) as possible, and the objective is to maximize system utilization $(x_1, x_2, \ldots, x_n)$. The main problem we face is how to define “fair”. The well-known lexicographically max-min fairness [12] is a good choice. Given a feasible allocation, we compute the dominant share for each user. Let $D = (D_1, D_2, \ldots, D_n)$ be the dominant-share vector, where $D_j$ is the dominant share of user $u_j$ for $j = 1, 2, \ldots, n$.

**Definition 1.** A dominant-share vector $D = (D_1, D_2, \ldots, D_n)$ is called (lexicographically max-min) LMM-optimal, if for every feasible dominant-share vector $D' = (D'_1, D'_2, \ldots, D'_n)$, $D_\tau$ is lexicographically greater than $D'_\tau$, where $D_\tau$ ($D'_\tau$) is the vector obtained by arranging the dominant shares in $D$ ($D'$) in order of increasing magnitude.

For example, given two feasible dominant-share vectors $D = (0.8, 0.9, 0.4)$ and $D' = (0.6, 0.7, 0.5)$, $D$ is not LMM-optimal, as $D'_\tau = (0.5, 0.6, 0.7)$ is lexicographically greater than $D_\tau = (0.4, 0.8, 0.9)$.

Noting that the system utilization $(x_1, x_2, \ldots, x_n)$ increases generally with increasing dominant-share vector $(D_1, D_2, \ldots, D_n)$, we propose the *lexicographically max-min dominant share* (LMMDS) mechanism for the multi-resource fair allocation problem with bounded number of tasks, which is to find the LMM-optimal dominant-share vector $D$, subject to (1)(3).
3 Fairness Properties

Let \( i_j \) be the weighted dominant resource of user \( u_j \) where \( i_j \in \arg\max r_{ij}/w_{ij} \). Similarly to [6, 9], the following are important and desirable properties of a fairness allocation with bounded number of tasks:

1. **Sharing incentive (SI).** For all users \( u_j \): either \( x_j = B_j \) or there exists a resource \( i \) such that \( r_{ij}x_j \geq w_{ij} \).

2. **Group Strategy-proofness (GSP).** No user can schedule more tasks by forming a coalition with others to misreports their requirements \( r_{ij} \) or the number of tasks \( B_j \).

3. **Envy-freeness (EF).** For all users \( u_j \): either \( x_j = B_j \) or there exists a resource \( i \) such that

\[
\frac{r_{ij}x_j}{w_{ij}} \geq \frac{r_{ik}x_k}{w_{ik}}.
\]

4. **Pareto efficiency (PE).** It should not be possible to increase the number of tasks processed on the server without decreasing the allocation of at least another user.

Sharing incentive means that the number of tasks processed for each user \( u_j \) is no less than the case where a \( r_{ij} \)-fraction of each resource \( i \) is allocated to every user. Group strategy-proofness means no user can get a better allocation by lying about \( r_{ij} \) or \( B_j \). Envy-freeness requires that for every user \( u_j \) such that \( x_j < B_j \), she does not envy user \( u_k \) when the allocation of \( u_k \) is scaled by \( w_{ij}/w_{ik} \). Pareto efficiency is to maximize system utilization subject to satisfying the other properties.

Consider the LMM-optimal dominant-share vector \( D = (D_1, D_2, \ldots, D_n) \) and the corresponding utility vector \( (x_1, x_2, \ldots, x_n) \), where \( x_j = w_{ij}D_j/r_{ij} \leq B_j \) for \( j = 1, 2, \ldots, n \).

**Lemma 1.** If \( x_k < B_k \) for user \( u_k \), the \( D_k \geq D_j \) for every \( j = 1, 2, \ldots, n \).

**Proof.** If there is a user \( u_h \) satisfying \( D_h = x_hr_{ih}/w_{ih} > x_kr_{ik}/w_{ik} = D_k \), we construct a feasible dominant-share vector \( D' = (D'_1, D'_2, \ldots, D'_n) \), where

\[
D'_j = \begin{cases} 
(x_h - \epsilon_1)r_{ih}/w_{ih}, & j = h; \\
(x_k + \epsilon_2)r_{ik}/w_{ik}, & j = k; \\
D_j, & j \neq k, h.
\end{cases}
\]
Here, $\epsilon_1$ is a small positive number such that $D'_h > D'_k$, and $\epsilon_2 = \epsilon_1 \min_i r_{ih}/r_{ik}$. This implies that $D_h > D'_h > D'_k > D_k$. Clearly, $\epsilon_2 r_{ik} \leq \epsilon_1 r_{ih}$ for every $i$, which implies that $D'$ is a feasible dominant-share vector. It is easy to verify that $D'_r$ is lexicographically greater than $D_r$, which contradicts the fact that $D$ is the LMM-optimal dominant-share vector.

**Theorem 1.** LMMDS satisfies the SI property.

**Proof.** If $x_j = B_j$ for every $j$, all users are satisfied. If there is a user $u_k$ such that $x_k < B_k$, there exists at least one saturated resource. If not, we can allocate $(x_k + \epsilon)(r_{1k}, r_{2k}, \ldots, r_{mk})$ to user $u_k$ without changing the other $x_j$s, where $\epsilon = \min_i (1 - \sum_{j=1}^{n} r_{ij} x_j)/r_{ik}$. Then, we obtain a new dominant-share vector $D' = (D'_1, D'_2, \ldots, D'_n)$, where $D'_k = (x_k + \epsilon) r_{ik} k$ and $D'_j = D_j$ for $j \neq k$. Clearly, $D'_r$ is feasible and lexicographically greater than $D_r$, which contradicts the fact that $D$ is the LMM-optimal dominant-share vector.

For each user $u_k$ such that $x_k < B_k$, if there is a resource $i$ satisfying $r_{ik} x_k \geq w_{ik}$, then $u_k$ is satisfied. Otherwise, $D_k < 1$ and for an arbitrary saturated resource $i'$, we have $r_{i'k} x_k < w_{i'k}$. As $\sum_{j=1}^{n} w_{i'j} = 1$ and the resource $i'$ is saturated, i.e., $\sum_{j=1}^{n} r_{i'j} x_{i'j} = 1$, there is a user $u_h$ satisfying $r_{i'k} x_h > w_{i'k}$, implying that $D_h > 1$. Let $\epsilon_1$ be a small positive number less than $(x_h r_{i'k} - w_{i'k})/r_{i'k}$, and $\epsilon_2 = \epsilon_1 \min_i r_{ih}/r_{ik}$. This implies that $(x_h - \epsilon_1) r_{i'k}/w_{i'k} > 1$ and $\epsilon_2 r_{ik} \leq \epsilon_1 r_{ih}$ for every $i$. Construct a feasible dominant-share vector $D' = (D'_1, D'_2, \ldots, D'_n)$, where $D'_j$ is defined as in (4). It is easy to verify that $D' = (D'_1, D'_2, \ldots, D'_n)$ is a feasible dominant-share vector and $D'_r$ is lexicographically greater than $D_r$, which contradicts the fact that $D$ is the LMM-optimal dominant-share vector. Thus, for every user $u_j$, either $x_j = B_j$ or there exists a resource $i$ such that $r_{ij} x_j \geq w_{ij}$.

**Theorem 2.** LMMDS satisfies the GSP property.

**Proof.** Denote by $\bar{D} = (\bar{D}_1, \bar{D}_2, \ldots, \bar{D}_n)$ the LMM-optimal dominant-share vector when a coalition of users $U_1 \subseteq \{u_1, u_2, \ldots, u_n\}$ misreports requirement $\bar{r}_{ik}$ instead of $r_{ik}$ and $\bar{B}_k$ instead of $B_k$ for all $u_k \in U_1$. For an arbitrary user $u_k$, if $x_k = B_k$, user $u_k$ cannot increase its utility by altering the demand vector or $B_k$. Thus, we assume that $x_k < B_k$ for each user $u_k \in U_1$. 7
Denote by the \((\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)\) the corresponding numbers of tasks processed in the modified system, where \(\bar{D}_k = \max_i \bar{x}_k \bar{r}_{ik} / w_{ik}\) for each user \(u_k \in U_1\) and \(\bar{D}_j = \bar{x}_j r_{ij} / w_{ij}\) for \(u_j \notin U_1\). For an arbitrary user \(u_k \in U_1\), assume that the true utility \(\bar{x}_{k}^{\text{true}}\) of user \(u_k\) is increased in the modified system. Thus,

\[ x_k < \bar{x}_{k}^{\text{true}} = \min_i \frac{\bar{x}_k \bar{r}_{ik}}{r_{ik}}, \]

which implies that

\[ x_k r_{ik} < \bar{x}_{k}^{\text{true}} r_{ik} \leq \bar{x}_k \bar{r}_{ik}, \text{ for } i = 1, 2, \ldots, m, \text{ and } D_k < \bar{D}_k. \quad (5) \]

Consider an arbitrary saturated resource \(i\) in the original system (there must exist, as in the proof of Theorem 1). As \(\sum_{j=1}^{n} r_{ij} x_j = 1, \sum_{j:u_j \in U_1} \bar{r}_{ij} \bar{x}_j + \sum_{j:u_j \in U \setminus U_1} r_{ij} \bar{x}_j \leq 1, \) and \(x_k r_{ik} < \bar{x}_k \bar{r}_{ik}\) for each user \(u_k \in U_1\), there is a user \(u_h \in U \setminus U_1\) such that \(r_{ih} x_h > r_{ih} \bar{x}_h\), which implies that

\[ \bar{x}_h < x_h \leq B_h, \text{ and } \bar{D}_h = \frac{\bar{x}_h r_{ih} h}{w_{ih}} < \frac{x_h r_{ih} h}{w_{ih}} = D_h. \quad (6) \]

In the original system, if \(D_k < D_h\), consider the feasible dominant-share vector \(\bar{D}' = (\bar{D}_1', \bar{D}_2', \ldots, \bar{D}_n')\) defined in (4). It is easy to verify that \(\bar{D}'\) is lexicographically greater than \(\bar{D}\), which contradicts the fact that \(\bar{D}\) is the LMM-optimal dominant-share vector in the original system. Hence, \(D_k \geq D_h\). Combining (5) and (6), we have

\[ \bar{D}_k > D_k \geq D_h > \bar{D}_h. \]

Let

\[
\bar{D}'_j = \begin{cases} 
\max_i (\bar{x}_k - \epsilon_1) \bar{r}_{ik} / w_{ik}, & j = k; \\
(\bar{x}_h + \epsilon_2) r_{ih} h / w_{ih}, & j = h; \\
\bar{D}_j, & j \neq k, h,
\end{cases}
\]

where \(\epsilon_1, \epsilon_2\) are small positive numbers closely to 0 defined as in the proof of Lemma 1. Clearly, \(\bar{D}' = (\bar{D}_1', \bar{D}_2', \ldots, \bar{D}_n')\) is feasible and lexicographically greater than \(\bar{D}\), which
contradicts the fact that \( \mathbf{D} \) is the LMM-optimal dominant-share vector in the modified system. Therefore, every user \( u_k \in U_1 \) cannot increase her dominant share by altering the demand vector or \( B_k \).

**Theorem 3.** LMMDS satisfies the EF property.

**Proof.** For an arbitrary user \( u_k \), if \( x_k = B_k \), \( u_k \) does not envy any user. If \( x_k < B_k \), by Lemma 1, we have \( D_k \geq D_j \) for every user \( u_j \). If user \( u_k \) envies another user \( u_h \), we must have a strictly higher weighted share of every resource than that of user \( u_k \), i.e., \( x_k r_{ik} / w_{ik} < x_h r_{ih} / w_{ih} \), for \( i = 1, 2, \ldots, m \), which implies that \( D_k < D_h \). A contradiction.

**Theorem 4.** LMMDS satisfies the PO property.

**Proof.** Consider the user \( u_k \) such that \( x_k < B_k \). If we can allocate \((x_k + \epsilon)(r_{1k}, r_{2k}, \ldots, r_{mk})\) to user \( u_k \) without changing the allocation of the other users, it will contradict the fact that \( \mathbf{D} \) is the LMM-optimal dominant-share vector.

## 4 LMMDS Scheduling Algorithm

Since the number of tasks for each user \( u_j \) is bounded by \( B_j \), in any feasible allocation, the maximum dominant share for user \( u_j \) is \( D_j^{\text{max}} = r_{ij} B_j / w_{ij} \). For a real number \( D \geq 0 \), let \( A(D) = (D_1, D_2, \ldots, D_n) \) be the dominant-share vector such that \( D_j = \min\{D_j^{\text{max}}, D\} \) for each user \( u_j \).

**Lemma 2.** There exists a real number \( D^* \) such that \( A(D^*) \) is the LMM-optimal dominant-share vector.

**Proof.** Let \( (D_1, D_2, \ldots, D_n) \) be a LMM-optimal dominant-share vector and \( D^* = \max_j D_j \). If there is a user \( u_k \) satisfying \( D_k \neq \min\{D_k^{\text{max}}, D^*\} \), we have \( D_k < D_k^{\text{max}} \) and \( D_k < D^* \). Consider the user \( u_h \) satisfying \( D_h = D^* > D_k \). The dominant-share vector \( \mathbf{D}' = (D'_1, D'_2, \ldots, D'_n) \) defined as in (4) satisfies

\[
D_k < D'_k < D'_h < D_h, \text{ and } D'_j = D_j, \text{ for } j \neq k, h.
\]

This contradicts the fact \( (D_1, D_2, \ldots, D_n) \) is a LMM-optimal dominant-share vector.
According to Lemma 2, a simple way is to use the binary method to find the maximum $D^*$ such that $A(D^*) = (D_1, D_2, \ldots, D_n)$ is a feasible dominant-share vector satisfying (1) and (3). However, this naive algorithm is not efficient enough. Our algorithm is described as follows. Using Blum et al.’s method [2] to find the median value $D$ in $\{D_{1}^{\text{max}}, D_{2}^{\text{max}}, \ldots, D_{n}^{\text{max}}\}$, let $D_j = \min\{D, D_j^{\text{max}}\}$ and $x_j = w_{ij}D_j/r_{ijj}$ for every user $u_j$, where $i_j$ is the dominant resource for user $u_j$. Clearly, if $\sum_{j=1}^{n} r_{ijj}x_j \leq 1$ for every $i = 1, 2, \ldots, m$, we have $D^* \geq D$. Otherwise, we have $D^* < D$. We distinguish the following two cases:

**Case 1. $D^* \geq D$.** For each user $u_j$ satisfying $D_j = D_j^{\text{max}}$, which implies $D_j^{\text{max}} \leq D \leq D^*$, by Lemma 2, we have $D_j = \min\{D_j^{\text{max}}, D\} = D_j^{\text{max}}$. Delete user $u_j$ and decrease the corresponding resources consumed by $u_j$ from the instance.

**Case 2. $D^* < D$.** For each user $u_j$ satisfying $D_j = D$, which implies $D_j^{\text{max}} \geq D > D^*$, by Lemma 2, we have $D_j = \min\{D_j^{\text{max}}, D^*\} = D^*$. Thus, these users satisfying $D_j = D$ have the same dominant share $D^*$ in the LMM-optimal solution, and can be merged into a “dummy” user $u_{\text{dum}}$. Let $U_{\text{dum}}$ be the set of users satisfying $D_j = D$. Given a possible “dominant share” $D_{\text{dum}}$ of the dummy user $u_{\text{dum}}$, for each user $u_j$ in $U_{\text{dum}}$, it consumes $r_{ijj}x_j$-fraction resource of type $i$, where $x_j = D_{\text{dum}}w_{ijj}/r_{ijj}$. Thus, the dummy user $u_{\text{dum}}$ consumes $D_{\text{dum}}\mu_i$-fraction resource of type $i$, where

$$\mu_i = \sum_{j:u_{ij} \in U_{\text{dum}}} r_{ijj}w_{ijj}/r_{ijj}.$$  

Note that the number of users is reduced by half. Again, use Blum et al.’s method [2] to find the median value $D$ in $\{D_j^{\text{max}}|u_j \text{ is the remaining user and is not the dummy user}\}$. Then, compare $D$ and $D^*$ similarly to the above discussion, and reduce the number of users by half correspondingly until that there is only one dummy user. Finally, we will find the user $u_\tau$ with maximum value $D_{\tau}^{\text{max}}$ such that $D_{\tau}^{\text{max}} \leq D^*$. For each user $u_j$ satisfying $D_j^{\text{max}} \leq D_{\tau}^{\text{max}} \leq D^*$, set $D_j = D_j^{\text{max}}$. For each user $u_j$ in $U_{\text{dum}}$ such that
$D_j^{\text{max}} > D_r^{\text{max}}$, consider the following linear programming (LP):

$$\text{max} \quad D \sum_{j: u_j \in U_{\text{dum}}} D \cdot r_{ij} \cdot \frac{w_{ij} D_{j}^{\text{max}}}{r_{ij}} \leq 1 - \sum_{j: D_j^{\text{max}} \leq D_r^{\text{max}}} r_{ij} \cdot \frac{w_{ij} D_{j}^{\text{max}}}{r_{ij}}, \text{ for } i = 1, 2, \ldots, m.$$  

It is easy to verify that $D^*$ is the optimal solution to the above LP and

$$D^* = \min \frac{1 - \sum_{j: D_j^{\text{max}} \leq D_r^{\text{max}}} r_{ij} \cdot \frac{w_{ij} D_{j}^{\text{max}}}{r_{ij}}}{\sum_{j: u_j \in U_{\text{dum}}} r_{ij} \cdot \frac{w_{ij} D_{j}^{\text{max}}}{r_{ij}}} = \min \frac{1 - \sum_{j: D_j^{\text{max}} \leq D_r^{\text{max}}} r_{ij} \cdot \frac{w_{ij} D_{j}^{\text{max}}}{r_{ij}}}{\mu_i}.$$  

For each user $u_j \in U_{\text{dum}}$, set $D_j = D^*$, and then we obtain the LMM-optimal solution $D = (D_1, D_2, \ldots, D_n)$.

Algorithm 1 shows the pseudo-code for LMMDS scheduling algorithm.

---

Algorithm 1 LMMDS pseudo-code

Step 1. Initialization.

- $\mathbf{rc} = (1, 1, \ldots, 1)$ \hfill $\triangleright$ the remaining resource capacities.
- $D_j = 0$ \hfill $\triangleright$ user $u_j$’s dominant share
- $U = \{u_1, u_2, \ldots, u_n\}$ \hfill $\triangleright$ the remaining user set
- $D = \{D_1^{\text{max}}, D_2^{\text{max}}, \ldots, D_n^{\text{max}}\}$ \hfill $\triangleright$ the possible critical value set
- $U_{\text{dum}} = \emptyset$ \hfill $\triangleright$ the set of users merged into the dummy user
- $D_{\text{dum}} = 0$ \hfill $\triangleright$ the dominant share of the users in $U_{\text{dum}}$
- $\mu_i = 0, i = 1, 2, \ldots, m$ \hfill $\triangleright$ the weights of the dummy user

Step 2. Using the method in [2] to find the median $D$ in $D$, set $D_{\text{dum}} = D$ and $D_j = \min\{D, D_j^{\text{max}}\}$ for $u_j \in U$. If $D_j$ satisfies

$$D_{\text{dum}} \mu_i + \sum_{j: u_j \in U} r_{ij} \cdot \frac{w_{ij} D_j}{r_{ij}} \leq \mathbf{rc}_i, \text{ for } i = 1, 2, \ldots, m,$$

set

$$\mathbf{rc}_i \leftarrow \mathbf{rc}_i - \sum_{j: u_j \in U, D_j = D_j^{\text{max}}} r_{ij} \cdot \frac{w_{ij} D_j}{r_{ij}}, \text{ for } i = 1, 2, \ldots, m;$$

$$U \leftarrow U \setminus \{u_j | D_j = D_j^{\text{max}}\}.$$  

11
Else, set
\[
\mu_i \leftarrow \mu_i + \sum_{j: u_j \in U, D_j < D_j^{max}} \frac{r_{ij}}{r_{ij}} \cdot \frac{w_{ij}}{r_{ij}};
\]
\[
U_{dum} \leftarrow U_{dum} \cup \{u_j | D_j < D_j^{max}, u_j \in U\};
\]
\[
U \leftarrow U \setminus \{u_j | D_j < D_j^{max}\}.
\]

**Step 3.** If \( U \neq \emptyset \), goto **Step 2**; else for each user \( u_j \in U_{dum} \), set
\[
D_j \leftarrow \min_i \frac{1 - \sum_{j: D_j^{max} \leq D_j} \frac{w_{ij}}{r_{ij}} \cdot \frac{D_j^{max}}{r_{ij}}}{\mu_i}.
\]

**Step 4.** Output \( D_j \), for \( j = 1, 2, \ldots, n \).

At each iteration \( k \), since we have at most \( n/2^{k-1} + 1 \) users, deciding whether \( D \geq D^* \) can be done within \( O(mn/2^{k-1}) \) time. Thus, the overall running time is \( O(m(n + n/2 + n/2^2 + \ldots + 1)) = O(mn) \), which is linear in \( n \) when \( m \) is bounded number.

## 5 Approximation ratios

Given an allocation, we define its (utilitarian) social welfare as \( \sum_j x_j \). As defined in [9], the **approximation ratio** of a mechanism is the worst-case ratio between the social welfare of the optimal solution and the social welfare of the mechanism’s solution. Parkes, Procaccia, and Shah [9] show that the approximation ratio of DRF is at least \( m \). Noting that the LMMDS mechanism is exactly the DRF mechanism when \( B_j = +\infty \) for \( j = 1, 2, \ldots, n \), the approximation ratio of the LMMDS mechanism is at least \( m \), too. Consider a setting with one resource and two users. Assume that \( w_{11} = r_{11} = \epsilon, w_{12} = r_{12} = 1 - \epsilon, \) and \( B_1 = B_2 = +\infty \). The LMMDS (or DRF) mechanism produces a fair allocation with \( x_1 = x_2 = D_1 = D_2 = 1 \). It is easy to verify that the social welfare of the optimal solution is \( 1/\epsilon \) obtained by allocating all resources to user \( u_1 \). Thus, the approximation ratio is \( 1/2\epsilon \), which approaches infinity when \( \epsilon \to 0 \). Thus,
Theorem 6. When $B_j = +\infty$ for every $j$, the approximation ratio of LMMDS (or DRF) is infinity.

Assuming that each user contributes equal amount for every type of resource [6], i.e., $w_{ij} = 1/n$ for every $i, j$, we obtain a tight approximation ratio.

Theorem 7. When $w_{ij} = 1/n$ and $B_j = +\infty$ for every $i, j$, the approximation ratio of LMMDS (or DRF) is exactly $n$.

Proof. Let $(x_1, x_2, \ldots, x_n)$ and $(x_1^*, x_2^*, \ldots, x_n^*)$ be a LMMDS (or DRF) solution and a social welfare maximized solution, respectively. As $(x_1, x_2, \ldots, x_n)$ satisfies SI, for each user $u_j$, there is a resource $i$ such that $u_j$ receives at least $w_{ij} = 1/n$ -fraction of it. Then, for each $j = 1, 2, \ldots, n$, we have $x_j \leq nx_j^*$, implying that $\sum_{j=1}^{n} x_j \leq n \sum_{j=1}^{n} x_j^*$.

Consider a setting with one resource and $n$ users. For $j = 1, 2, \ldots, n-1$, the requirement of user $u_j$ is $r_{1j} = 1/n$, and the requirement of user $u_n$ is $r_{1n} = 1/n^k$, where $k$ is a positive integer. The optimal allocation will give all of resource to user $u_n$, for a social welfare $n^k$. In contrast, under LMMDS (or DRF) each user will receive a $1/n$-fraction of the resource, for a social welfare $n^{k-1} + n - 1$. When $k$ grows larger, the approximation ratio of LMMDS (or DRF) approaches

$$\lim_{k \to \infty} \frac{n^k}{n^{k-1} + n - 1} = \lim_{k \to \infty} \frac{n}{1 + \frac{1}{n^{k-2}} - \frac{1}{n^{k-1}}} = \frac{n}{1 + \frac{1}{n^{k-2}} - \frac{1}{n^{k-1}}} = n.$$

Thus, for every $\delta > 0$, LMMDS (or DRF) cannot have an approximation ratio better than $n - \delta$ for the social welfare, which implies that the approximation ratio of LMMDS (DRF) is exactly $n$.

Indeed, removing the conditions $B_j = +\infty$ for every $j$, we can also obtain the same result.

Corollary 1. When $w_{ij} = 1/n$ for every $i, j$, the approximation ratio of any SI mechanism for the multi-resource fair allocation with bounded number of tasks is exactly $n$. 

13
6 Conclusions

We have presented the LMMDS mechanism, which generalized the well-known DRF. We believe that LMMDS can be found more applications in realistic settings. One important direction is to generalize our algorithm to multiple heterogeneous servers [13].

Acknowledgements

The work is supported in part by the National Natural Science Foundation of China [Nos. 11301466, 61170222].

References

[1] A.A. Bhattacharya, D. Culler, E. Friedman, A. Ghodsi, S. Shenker, and I. Stoica, Hierarchical scheduling for diverse datacenter workloads. In Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC’13, Article No. 4, 2013.

[2] M. Blum, R.W. Floyd, V. Pratt, R.R. Rivest, R.E. Tarjan, Time bounds for selection, Journal of Computer and System Sciences 7(4) 448-461, 1973.

[3] T. Bonald, J. Roberts, Enhanced cluster computing performance through proportional fairness. arXiv:1404.2266, 2014.

[4] D. Dolev, D. G. Feitelson, J. Y. Halpern, R. Kupferman, and N. Linial, No justified complaints: on fair sharing of multiple resources. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS’12, pp. 68-75, 2012.

[5] E. Friedman, A. Ghodsi, C-A. Psomas, Strategyproof allocation of discrete jobs on multiple machines, in Proceedings of the fifteenth ACM conference on Economics and computation, pp. 529-546, 2014.

[6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, Dominant resource fairness: Fair allocation of multiple resource types. In Proceedings of the
8th USENIX Conference on Networked Systems Design and Implementation, NSDI’11, pp. 24-24, Berkeley, CA, USA, 2011.

[7] A. Gutman and N. Nisan, Fair allocation without trade. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS’12, pp. 719-728, 2012.

[8] I. Kash, A. Procaccia, and N. Shah, No agent left behind: Dynamic fair division of multiple resources, In Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS’13, pp. 351-358, 2013.

[9] D.C. Parkes, A.D. Procaccia, and N. Shah, Beyond dominant resource fairness: extensions, limitations, and indivisibilities. In ACM Conference on Electronic Commerce, pp. 808-825, 2012.

[10] A.D. Procaccia, Cake cutting: Not just child’s play, Communications of the ACM, 2013.

[11] C.-A. Psomas and J. Schwartz, Beyond beyond dominant resource fairness: indivisible resource allocation in clusters. Tech Report Berkeley, 2013.

[12] N. Megiddo, Optimal flows in networks with multiple sources and sinks. Mathematical Programming 7(3), pp. 97-107, 1974.

[13] W. Wang, B. Li, and B. Liang, Dominant resource fairness in cloud computing systems with heterogeneous servers, In Proceedings of IEEE INFOCOM, 2014.

[14] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. In Proceedings of IEEE INFOCOM, pp. 1206-1214, 2012.

[15] Y. Zeldes and D. G. Feitelson, On-line fair allocations based on bottlenecks and global priorities. In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, ICPE 13, pp. 229-240, 2013.