Generalized Machine Activation Problems *

Jian Li † Samir Khuller ‡

Abstract

In this paper we consider a generalization of the machine activation problem introduced recently ["Energy efficient scheduling via partial shutdown" by Khuller, Li and Saha (ACM-SIAM 2010 Symp. on Discrete Algorithms)] where the unrelated parallel machine scheduling problem is studied with machine activation cost. This is the standard unrelated parallel machine scheduling problem with a machine dependent activation cost that is incurred, if any job is assigned to the machine. The problem asks for a choice of machines to activate, and a schedule of all jobs on the active machines subject to the makespan constraint. The goal is to minimize the total activation cost.

Our main generalization consists of a general activation cost model, where the activation cost for a machine is a non-decreasing function of its load. We develop a greedy algorithm that yields a fractional assignment of jobs, such that at least $n -$ jobs are assigned fractionally and the total cost is at most $1 + \ln(n=)$ times the optimum. Combining with standard rounding methods yields improved bounds for several machine activation problems.

In addition, we study the machine activation problem with d linear constraints (these could model makespan constraints, as well as other types of constraints). Our method yields a schedule with machine activation cost of $O(\frac{1}{} \log n)$ times the optimum and a constraint violation by a factor of $2d +$. This result matches our previous bound for the case $d = 1$.

As a by-product, our method also yields a $\ln n + 1$ approximation factor for the non-metric universal facility location problem for which the cost of opening a facility is an arbitrary non-decreasing function of the number of clients assigned to it. This gives an affirmative answer to the open question posed in earlier work on universal facility location.

*Research supported by NSF Awards CCF-0728839 and CCF-0937865 (HECURA), and a Google Research Award.

†University of Maryland, College Park, MD 20742. E-mail : lijian@cs.umd.edu .

‡University of Maryland, College Park, MD 20742. E-mail : samir@cs.umd.edu.

1 Introduction

In this work we consider the generalized machine activation problem defined as follows. We are given a set M of m unrelated machines and a set J of n jobs. Let $p_{i;j}$ and $a_{i;j}$ be the processing time and cost, respectively, for processing job j on machine i. Given an assignment of jobs to machines we refer to the **load** of machine i as the total processing time of all jobs that are assigned to machine i. We are also given a non-decreasing, piecewise linear and left-continuous machine activation cost function $!_i : \mathbb{R}^+ \to \mathbb{R}^+$ for each machine $i \in M$. A feasible integral solution is an assignment $: J \to M$ of each job to a machine. Let u be the machine load vector under assignment , i.e., $u_i = \sum_{(j)=i} p_{i;j}$.

The cost of a feasible solution consists of two parts: One is the machine activation cost given by $\sum_{i \in M} !_i(u_i)$, while the other is the assignment cost given by $\sum_{j \in J} a_{(j)j}$. We call this problem the **generalized machine activation** (*GMA*) problem. We can use the cost function to model the makespan constraint T by letting $!_i(0) = 0$, $!_i(x) = c_i$ for $0 < x \leq T$, and $!_i(x) = \infty$ for $x > T$, where c_i is the cost for opening machine i. We call the problem with such cost functions the **machine activation with assignment cost** (MAAC) problem. Furthermore, if there are no assignment costs, the problem reduces to the basic version of the machine activation problem which we denote by MA. (Both the MAAC and MA problems were introduced in [23].) Besides the capability of capturing the makespan constraints, this more general activation cost can also be used to model the situation that many current computing devices, such as CPUs and disk arrays, have more flexible power management controls than simple on/off switches, thus permitting many intermediate levels of power consumption.

Our first result is a simple greedy algorithm for GMA that finds a **fractional** assignment in which at least $n -$ jobs are fractionally assigned with total cost at most $\ln \frac{n}{} + 1$ times the optimum solution (Theorem 3.1). By applying the rounding scheme by Shmoys and Tardos [33], we obtain a $(2; (1 + o(1)) \ln n)$ bicriteria approximation for MAAC, where an $(;)$-approximation means if there exists a schedule of makespan T and cost C we can find one with

makespan at most T and cost at most C. This improves the $(3+\ ;O(\frac{1}{}\ln n))$-approximation in [23] and the $(2+\ ;2\ln(2n=\ )+5)$-approximation in [14]. For MA, the algorithm reduces to the greedy algorithm for submodular covering problem and using the Shmoys-Tardos scheme yields a $(2;\ln n\ +\ 1)$-approximation, matching the results in [23]. Improving the factor of 2 for minimizing makespan is a well known open problem. We also obtain a $(1+o(1))\ \ \ln n$-approximation for GMA if $!\ _i()$ satisfies $!\ _i(2x)\ \leq\ !\ \ _i(x)$ for all i (e.g. for concave $!$, we have $\ \ =2$).

Typically, a fractional assignment can be obtained from a generalized flow computation for an instance of the unrelated machine scheduling problem. In the special case where assignment costs can be modeled by a standard flow computation in the given GMA instance, the same greedy algorithm (with some simplifications) is able to find a feasible integral assignment and the total cost is at most $\ln n+1$ times the optimum solution (Theorem 4.3). The result directly yields an optimal $\ln n\ +\ 1$-approximation for the non-metric universal facility location (UniFL) problem in which the cost function for a facility is an arbitrary non-decreasing function of the number of clients assigned to it. This answers an open question proposed in [19, 26]. We note that a logarithmic approximation is essentially optimal in light of Feige's result [13]: For any $\ >\ 0$, there is no $(1\ -\ )\ln n$-approximation for set cover unless $NP\ \subseteq\ DTIME\ [n^{O(\log\log\ n)}]$.

Note that GMA does not fit into the well studied submodular covering framework [38]. Bar-Ilan et al. [3] considered the problem of incorporating the assignment cost to the submodular covering framework (see Section 4.3). Their framework, in which the assignment cost is captured by a standard network flow computation, can be used to model a number of network design problems including the capacitated facility location problem. Our work extends this work by having a more general cost function and assignment model, i.e., generalized flows. Our algorithm is simpler than their multi-phase greedy approach and our bound is also tighter. We summarize our results in Figure 1.

Our main algorithm is a simple greedy algorithm. Indeed, in prior work we introduced a greedy framework [23] in which at each step a new machine is activated. We would like to activate the machine which has minimum cost to benefit ratio. However, computing the benefit of a machine in this context is a hard problem. The key idea there is to estimate the benefit by a fractional relaxation and compute a fractional schedule that is rounded at the end. In this work we significantly generalize this framework to define a greedy method that works with fractional activations at each step. In our analysis, we make extensive use of the conformal decomposition lemma (for generalized flows, and more generally for LP solutions), which enables a charging argument to show the relationship between the cost to benefit ratio and the optimum. Note that the charging argument for the basic set cover problem was extended for several generalizations such as facility location, set multicover, capacitated set cover [12]. To illustrate the use of the conformal decomposition lemma, we also give a short alternate proof of the supermodularity of minimum cost generalized flow. This simplifies the proof by Fleischer that is based on LP duality [14] [1].

Besides one single makespan constraint, a more realistic machine may restrict the set of tasks that can be assigned to it in a variety of other ways. Several prior works study the problem of combining the degree constraint (at most a certain number of jobs can be assigned to a particular machine), or more generally some linear constraints, into the traditional machine scheduling problem (e.g. [39, 37, 25, 32]). Motivated by similar considerations, we study the machine activation problem with d linear constraints on each machine – we refer to this problem as Machine Activation with Linear Constraints (*MALC*). We show that we can find a schedule such that the machine activation cost is at most $O(\frac{1}{}\log n)$ times the optimum and each constraint may be violated by at most a factor of $2d+\ $. Note that when $d=1$, our result matches the $(2+\ ;O(\frac{1}{}\log n))$-approximation obtained by LP rounding in [23]. Our algorithm is also based on LP rounding and makes use of ideas from [25, 23].

1.1 Related Work

Our work is closely related to the submodular covering framework [38]. In an instance of the problem, we are given a ground set N, with each element e in N having a cost c_e and a nonnegative monotone submodular function $f\ :\ 2^N\ \rightarrow\ \mathbb{R}^+\ \cup\ \{0\}$. The objective is to find a subset $S\subseteq N$ of the minimum cost such that $f(S)=f(N)$. It is well known that the problem admits logarithmic factor approximations [38, 15]. Indeed, the fractional version of MA is can be cast as a submodular covering problem [23, 14] , by noticing that maximum generalized flow is submodular (proved in [14]). However, MAAC and the more general GMA do not fit into the submodular covering framework due to the assignment cost. Fleischer [14] defined the supermodular packing problem and used it to capture the assignment cost for MAAC. It is not clear to how to

[1] In the context of network ows, the supermodularity (or submodularity) has a close relationship with the notions of substitutes and complements, which have been a subject of study in operation research community. See the references in [14] for prior work.

	MA	MAAC	UniFL	GSC
previous results	$(2; \ln n + 1)$ [23] $(2 + \ ; \ln(n= \ ) + 1)$ [14]	$(3 + \ ; O(\frac{1}{} \ln n))$ [23] $(2 + \ ; 2\ln(2n= \ ) + 5)$ [14]	open [19, 26]	$O(\log nM \ )$[3]
our results	$(2; \ln n + 1)$	$(2; (1 + o(1)) \ln n)$	$\ln n + 1$	$(*) \ln \lvert b \rvert + 1$

Figure 1: Comparison of our results to previous results. n is the number of jobs (or clients). M is the largest integer in the instance. $(*)$ If $|b|$ is not bounded by a polynomial of input size, the ratio is $(1 + o(1)) \ln |b|$.

use the the same idea in GMA because of the general machine cost functions. Fleischer [14] also studied the completion time version of the machine activation problem where there is a completion time constraint instead of of the makespan constraint and obtained a logarithmic approximation.

A considerable amount of work concerns the problem of rounding a fractional solution x obtained by solving the linear-(or convex-)programming relaxation to an integral one X such that multiple linear constraints are satisfied approximately in the rounded solution. Typically, the constraints are a collection of k linear functions f_i and we want to minimize the "discrepancy" terms $|f_i(x) - f_i(X)|$ for most/all i. If k is not a constant, the terms usually depend on n or k [4, 1]. Recently, Saha and Srinivasan [32] used a random-walk rounding technique to show that there is an efficient algorithm that returns a b-matching maintaining the capacity constraints on all machines nodes and $|f_i(X) - f_i(x)| < \ _i \ \forall i$ where f_i is a linear function defined on edges incident on node i and $\ _i$ is an upper bound on the coefficients. This appears to be the first result that gives a bound independent of n for more than one constraint per node. MALC can be seen as a generalization of their problem by having activation costs for machines and d constraints on each machine. Our bound on the violation of the constraints is also independent of n.

There is a huge literature on the facility location problem and its many variants. For the uncapacitated facility location in metric space, small constant factor approximation algorithms can be obtained by many approximation algorithmic techniques (e.g. [34, 21, 9, 2, 20, 10]) The current best approximation ratio is 1.5 [8]. Guha and Khuller [18] showed that there is no 1:463 approximation unless $NP \subseteq DTIME[n^{O(\log\log n)}]$. Hajiaghayi et al. [19] first studied the facility location problem with concave facility cost functions. Mahdian and Pal obtained the first constant approximation for arbitrary non-decreasing facility cost functions [26] where the problem was named universal facility location (*UniFL*). The focus of both works is the metric case and a logarithmic approximation for the non-metric case was posed as an open question. The current best ratio for metric universal facility location is 6.702 by Vygen [36]. A number of well-studied problems are very special cases of universal facility location. These include uncapacitated facility location, set cover with soft or hard capacities, facility location with soft constraints [2, 11, 20, 27] or hard capacities [24, 29, 11, 40], the linear-cost facility location problem [27, 28], the concave-cost facility location [19] and the load-distance balancing problem [6].

The special case of MA where the job assignment can be computed via a max-flow has been also studied in the context of deploying internet transit access points in a wireless sensor network [30] and designing a payment system in a trust network [16]. The algorithms developed in the above two works are also greedy and they use a charging argument based on the path decomposition of flow that is similar to ours in spirit.

2 Preliminaries

Scheduling unrelated machines can be modeled as a generalized flow problem. We review some basic concepts about generalized flows. A generalized flow problem is the same as a standard flow problem, except with a gain factor $_e > 0$ on each arc e. This means that if a unit of flow enters arc e, $_e$ units of flow leave arc e. We say arc e is a normal arc if $_e = 1$. The cost of the flow f is $\sum_{e \in E} f_e c_e$ where c_e is the cost per unit of flow on arc e, and f_e is the flow entering arc e. To see that how a generalized flow is used to compute a fractional assignment, we add a sink t to the bipartite graph $G = (M \cup J; E)$. We add a normal arc $(j; t)$ for each job j. Each arc $(j; t)$ has a capacity 1. Each arc $(i; j); i \in M; j \in J$ has a capacity p_{ij}, gain factor $1=p_{ij}$, and cost $c_{ij} = a_{ij} = p_j$. Each node $i \in M$ is a flow source and its flow excess is determined during the execution of our algorithm. The total fractional number of satisfied jobs is the amount of flow that the sink t receives and the fractional assignment cost is just the cost the corresponding flow. The gain of a cycle is the product of the gain factors of arcs in that cycle. A ow generating (ow-absorbing) cycle is a cycle whose gain factor is larger (smaller) than one. A bicycle consists of a flow-generating cycle, a flow-absorbing cycle, and a (possibly trivial) path from the first cycle to the second. A circulation is a feasible gen-

eralized flow on a network where there is no flow excess or demand at any node. A **circuit** is a circulation that sends positive flow only along the arcs of a bicycle or unit-gain cycle. We shall use the following **conformal decomposition lemma** extensively.

Lemma 2.1. ([17]) A generalized circulation f can be decomposed into circuits $g_1, \ldots, g_k$ with $k \leq |E|$ such that $f = \sum_{i=1}^{k} g_i$ and $g_i(e) > 0$ only if $f(e) > 0$.

Clearly, the existence of a negative cost circuit in the residual graph implies that the current circulation can be improved by pushing flow around the circuit. The converse of this fact is also true: A generalized circulation f is optimal if and only if the residual graph contains no negative cost circuit (e.g. [17]).

Sometimes, it is easy to think of a source (a sink) as a flow-generating (flow-absorbing) self-loop. For example, if source s has a flow excess x, we replace s with a self loop which has a gain factor 2 and capacity x. Therefore, a generalized flow in our context can be treated as a generalized circulation and the decomposition lemma is still applicable. In our algorithm, the machine activation cost is counted separately from the flow cost, so the cost on these self loops is zero. We finally note that, in a fractional assignment, the load of machine i equals to the amount of flow leaving node i.

3 The Greedy Algorithm

Our main result in this section is the following theorem.

Theorem 3.1. There is a polynomial time approximation algorithm that can find a fractional assignment such that at least $n - \alpha$ jobs are (fractionally) satisfied and the cost is at most $\ln(\frac{n}{\alpha})+1$ times the optimal cost for *GMA*.

First, we need some notation and more preparation on generalized flows. Arc (v, w) has capacity $u_{vw} \geq 0$, gain factor $\gamma_{vw} > 0$, and per unit flow cost $c_{vw} \geq 0$. Let $\mathbf{u} \in \mathbb{R}^{|M|}$ be the flow excess vector where u_i is the flow excess on machine node i. Let f_{vw} be the flow value entering arc (v, w). For a flow f, we use $c(f)$ to denote the cost of the flow, i.e., $c(f) = \sum_{vw} c_{vw} f_{vw}$. Note that u_i puts an upper bound on the load of machine i. Recall that we can think of a machine node i with flow excess x as a flow-generating self-loop with gain factor 2 and capacity x. We shall also call $\mathbf{u}$ **capacity vector** sometimes, where $u_i = u_{ii}$, the capacity of the self-loop (i, i). Let $G[\mathbf{u}]$ be the generalized flow network with capacity vector $\mathbf{u}$. We define $\beta(\mathbf{u}, f_t)$ to be the optimal cost of sending at least f_t units of flow to t in $G[\mathbf{u}]$, which can be determined by the following LP:

$$\beta(\mathbf{u}, f_t) = \min. \sum_{vw} c_{vw} f_{vw} \tag{3.1}$$

Algorithm 1 : GMA-GREEDY(G, α, δ)

```
1 u = 0; f_t = 0;
2 while f_t < n − α do
3 |   Choose machine i and Δ ≥ 0; Δ ≥ δ s.t.
4 |       (β(u, f_t + Δ) − β(u, f_t))/Δ is minimized;
5 |   u = u + Δ e_i;
6 |   f_t = f_t + Δ;
7 |   u = CLEANUP(u, f_t);
```

$$\text{s. t.} \sum_{w \in \Gamma^+(v)} f_{vw} - \sum_{w \in \Gamma^-(v)} \gamma_{wv} f_{wv} = 0 \ \forall v \in V \setminus t;$$

$$\sum_{w \in \Gamma^-(t)} \gamma_{wt} f_{wt} \geq f_t;$$

$$0 \leq f_{vw} \leq u_{vw} \ \forall (v, w)$$

where $\Gamma^-(v) = \{w \mid (w, v) \in E\}$ and $\Gamma^+(v) = \{w \mid (v, w) \in E\}$. This first constraint is the flow conservation constraint, i.e., the total flow entering a node equals that leaving the node. The second says the total flow entering the sink t is at least f_t. In this LP, we use flow-generating self-loops on machine nodes. Therefore, we do not need any other constraint to model flow excesses.

In this paper, we assume that $\beta(\mathbf{u}, 0) = 0$ for any $\mathbf{u} \geq 0$ and $\beta(\mathbf{u}, f_t) \geq 0$ for any $\mathbf{u} \geq 0$ and $f_t \geq 0$. It is a useful fact that $\beta(\mathbf{u}, f_t)$ is a nondecreasing function of f_t (a standard property for linear programming).

Let $C(\mathbf{u}, f_t) = \sum_{i \in M} \ell_i(u_i) + \beta(\mathbf{u}, f_t)$. Define

$$\beta(\mathbf{u}, f_t) = \frac{1}{\delta} C(\mathbf{u} + \delta e_i, f_t + \Delta) - C(\mathbf{u}, f_t)$$

for any $i \in M$ and $\Delta > 0$, where e_i is the ith elementary vector: $e_i(i) = 1$ and $e_i(j) = 0$ for $j \neq i$.

Our greedy algorithm starts with $\mathbf{u} = 0$. In each iteration, we first increase the capacity of some machine i by δ and push Δ more units of flow to t such that $\beta(\mathbf{u}, f_t)$ is minimized. To ensure the algorithm runs in polynomial time, we require that the increment Δ is at least a given parameter δ ($\delta = \frac{1}{\text{poly}(n)}$ will be fixed later). Then, we perform a clean-up procedure which produces a maximal excess vector which could send as much flow to the sink with no larger cost (Property 3.1). This property is useful in the analysis of Lemma 3.4. The algorithm terminates when the sink t receives more than $n - \alpha$ units of flow, i.e., more than $n - \alpha$ fractional clients are satisfied. It is easy to see that there are a polynomial number of iterations. We specify below how each iteration can be implemented in polynomial time. We denote the algorithm by **GMA-GREEDY** (see Algorithm 1).

Finding the minimum ratio : Now, we elaborate on how to find the minimum value in each iteration in polynomial time. For a fixed machine i, it is easy to see that is the optimal value of the following linear-fractional program (LFP) :

$$\begin{aligned}
\text{min.} \quad & \frac{1}{-} \; !_i(u_i + \;) - !_i(u_i) + \sum_{vw} c_{vw} f_{vw} - \; (u; f_t) \\
\text{s. t.} \quad & \sum_{w \in \; ^+(v)} f_{vw} - \sum_{w \in \; ^-(v)} {}_{wv} f_{wv} = 0 \;\; \forall v \in V \setminus \{t\}; \\
& \sum_{w \in \; ^-(t)} {}_{wt} f_{wt} \geq f_t + \;; \\
& 0 \leq f_{vw} \leq u_{vw} \;\; \forall (v; w) \in E \setminus \{(i; i\,)\}; \\
& 0 \leq f_{ii} \leq u_{ii} + \;; \quad \geq 0; \quad \geq
\end{aligned}$$

We can assume $!_i(u_i + \;)$ is a linear function of . This is without loss of generality since we can solve the above FLP for every piece of the function $!_i(\cdot)$ (recall $!(\cdot)$ is piecewise linear) and take the minimum. Any FLP of the form $\max: \frac{c^T x + d}{e^T x + f}$ s.t. $Ax = b; x \geq 0$ can be converted to an equivalent LP by standard homogenization [7] as follows: W.l.o.g. assume $e^T x + f > 0$. We introduce a new variable z and let $z = \frac{1}{e^T x + f}$. Then, the FLP can be written as $\max: zc^T x + dz$ s.t. $zAx = zb; ze^T x + fz = 1; x \geq 0$. Substituting zx with new variables y, we get an equivalent linear program $\max: c^T y + dz$ s.t. $Ay = zb; e^T y + fz = 1; y; z \geq 0$.

The CLEANUP procedure: Now, we specify the details of the CLEANUP procedure. Given b and f_t, the goal of the procedure is to find an excess vector u such that the following property holds:

Property 3.1. (1) $C(u; f_t) \leq C(b; f_t)$; (2) There does not exist any $i \in M$ and ≥ 0 such that $C(u + \; e_i; f_t) < C(u; f_t)$.

In other words, we want to find a maximal excess vector such that the total cost of sending f_t flow to t is no larger than $C(b; f_t)$. The algorithm iterates over all machine nodes. For each machine node i, the algorithm attempts to add the maximum amount of excess to u_i such that the total cost does not increase. This amount can be determined by the following LP:

Algorithm 2 : CLEANUP($b; f_t$)

```
for each i ∈ M do
    Let   be the optimal solution of LP(3.2);
    b = b +  e_i ;
Return b;
```

$$\begin{aligned}
(3.2) \quad \text{maximize} \;\; & ; \quad \text{subject to} \\
& !_i(b_i + \;) - !_i(b_i) + \sum_{vw} c_{vw} f_{vw} - \; (b; f_t) \leq 0; \\
& \sum_{w \in \; ^+(v)} f_{vw} - \sum_{w \in \; ^-(v)} {}_{wv} f_{wv} = 0 \;\; \forall v \in V \setminus \{t\}; \\
& \sum_{w \in \; ^-(t)} {}_{wt} f_{wt} \geq f_t; \\
& 0 \leq f_{vw} \leq u_{vw} \quad \forall (v; w) \in E \setminus \{(i; i\,)\}; \\
& 0 \leq f_{ii} \leq u_i + \;; \quad \geq 0
\end{aligned}$$

The first constraint dictates that the cost does not go up. The last one increase the capacity of the flow-generating self-loop $(i; i\,)$ to $(u_i + \;)$. In Lemma 3.3, we show that it suffices to increment the excess once for every machine in order to ensure Property 3.1.

3.1 Analysis of the Approximation Ratio

We need a few lemmas. We use $\mathcal{R}(G; f\,)$ to denote the residual graph of G w.r.t. f and $\mathcal{R}(u; f\,)$ as a shorthand notation for $\mathcal{R}(G[u]; f\,)$. First, we state some simple facts that will be useful in various places. The proofs of the first three are straightforward and the fourth can be easily seen from the first three.

Lemma 3.1.

1. If f is a feasible circulation in G and f' is a feasible circulation in $\mathcal{R}(G; f\,)$. Then, $f + f'$ is feasible in G.
2. Suppose f and f' are feasible circulations in G. Then, $f - f'$ is a feasible circulation in $\mathcal{R}(G; f')$ [2].
3. Suppose f, a feasible circulation in G, can be decomposed to conformal circuits $g_1; g_2; \ldots; g_k$. For any $S \subseteq [k]$, $\sum_{i \in S} g_i$ is a feasible circulation on G.
4. Suppose f and f' are feasible circulations in G. $f - f'$ is decomposed into conformal circuits $g_1; g_2; \ldots; g_k$. For any $S \subseteq [k]$, $f' + \sum_{i \in S} g_i$ and $f - \sum_{i \in S} g_i$ are feasible circulations in G.

[2] 1 unit of flow on $(v; w)$ is equivalent to $_{vw}$ units of flow on $(w; v)$.

Recently, Fleischer showed that (u) is supermodular [14]. The proof was based on analyzing the changes of the dual variables. We provide a simpler direct proof using conformal decomposition.

Lemma 3.2. (Fleischer [14]) $(:; f_t)$ is a supermodular function. That is, that for any excess vector u, $i \in M$, positive scalars $;$, and $u' = u + e_j$, the following inequality holds:

$$(u; f_t) - (u + e_i; f_t) \geq (u'; f_t) - (u' + e_i; f_t):$$

Proof. We drop the second parameter f_t for ease of notation. Let $f; f^{\ i}; f^{\ j}; f^{\ ij}$ denote the optimal flow corresponding to (u); $(u + e_i)$; (u') and $(u' + e_i)$, respectively. Using Lemma 2.1, $\bar{f}_1 = f^{\ j} - f$ can be decomposed into conformal circuits. First, we can see all circuits have non-positive cost. Otherwise, by Lemma 3.1.4, subtracting those with positive costs from $f^{\ j}$ results in a feasible flow on $G[u + e_j]$ of less cost, contradicting the optimality of $f^{\ j}$. Second, no circuit has positive flow on any flow-generating loop other than $(j; j\ )$ since otherwise we can add those circuits to f and get a feasible flow of less cost, contradicting that f is optimal. Repeating the same argument, $\bar{f}_2 = f^{\ ij} - f^{\ j}$ can be also decomposed into conformal circuits whose costs are non-positive and no circuit has positive flow on any flow generating loop other than $(i; i\ )$.

$\bar{f}_1 + \bar{f}_2 = f^{\ ij} - f$ is a feasible flow on $\mathcal{R}(u + e_i + e_j; f\ )$. We decompose $\bar{f}_1 + \bar{f}_2$ into conformal circuits and partition them into two groups. The circuits in the first group g_1 (the second group g_2) has positive flow on the flow generating loop $(j; j\ )$ $((i; i\ ))$. Other circuits can be assigned to either group arbitrarily. Since $f + g_1$ is a feasible flow on $G[u + e_i + e_j\ ]$ (by Lemma 3.1.4), thus on $G[u + e_j\ ]$, we have $c(g_1) \geq c(\bar{f}_1)$. Therefore,

$$c(g_2) = c(\bar{f}_1 + \bar{f}_2) - c(g_1) \leq c(\bar{f}_2):$$

Also we can see that $f + g_2$ is a feasible flow on $G[u + e_i\ ]$. Hence,

$$\begin{aligned}(u + e_i) - (u) \leq c(g_2) \leq c(\bar{f}_2)\\ = c(f^{\ ij}) - c(f^{\ j}) = (u' + e_i) - (u'):\end{aligned}$$

This completes the proof □

Lemma 3.3. The excess vector u returned by CLEANUP $(b; f_t)$ satisfies Property 3.1.

Proof. The third constraint in LP(3.2) guarantees that $C(u; f_t)$ does not increase over iterations. This proves the first part. Now we show the second part. Suppose the lemma is not true and there exists $i \in M$ and > 0 such that $C(u + e_i; f_t) < C(u; f_t)$. We denote the excess vector obtained right after the ith iteration by $u^{(i)}$. It is obvious that $C(u^{(i)} + e_i; f_t) > C(u^{(i)}; f_t)$ for any > 0, since otherwise we would have a larger optimal value for LP(3.2). Thus, we have that

$$\begin{aligned}C(u^{(i)}; f_t) - C(u^{(i)} + e_i; f_t) < 0\\ < C(u; f_t) - C(u + e_i; f_t):\end{aligned}$$

Subtracting $!_i(u_i^{(i)}) - !_i(u_i^{(i)} +) = !_i(u_i) - !_i(u_i +)$ from both sides, we can get

$$(u^{(i)}; f_t) - (u^{(i)} + e_i; f_t) < (u; f_t) - (u + e_i; f_t):$$

which contradicts the supermodularity of . □

For any two vectors a and b, we use $\max(a; b)$ to denote the vector $(\max(a_1; b_1); : : : ; \max(a_m; b_m))$. The following lemma is the key to establishing the approximation ratio. The statement of the lemma is somewhat reminiscent of the standard relationship between the cost-benefit ratio and the optimum for set cover, which can be shown by a simple charging argument. However, the charging argument here is not as straightforward and needs to be done based on a conformal decomposition.

Lemma 3.4. Suppose u, f_t satisfy the second part of Property 3.1. For any $e; \tilde{f}_t$ such that $\tilde{f}_t - f_t \geq$, the following holds:

$$\min_{i;\ \geq} (i; ;\ ;\ u; f_t) \leq 1 + \frac{n}{} \frac{C(e; \tilde{f}_t)}{\tilde{f}_t - f_t}:$$

Proof. Let f , $\tilde{f}$ be the generalized flows corresponding to $(u; f_t)$ and $(e; \tilde{f}_t)$, respectively. Consider the flow $\tilde{f} - f$. Apply Lemma 2.1, $\tilde{f} - f$ can be decomposed into conformal circuits. By Lemma 3.1.2, we can see that $\tilde{f} - f$ is a feasible flow on $\mathcal{R}(\max(u; e); f\ )$, so is any conformal circuit obtained from the decomposition. Now, we group those circuits as follows. If the circuit contains the flow-generating self-loop $(i; i\ )$, we assign it to group g_i. Thanks to the structure of the circuit, each circuit contains at most one flow-generating self-loop. We note it may also contain at most one flow-absorbing self-loop (a negative flow on an flow-generating arc can be seen as a positive flow on the arc of the opposite (flow-absorbing) direction). Other circuits containing no self-loop are assigned arbitrarily. We also use g_i to denote the flow formed by the sum of the circuits in g_i. Let $\Delta u_i = \max(e_i; u_i) - u_i$ and $\Delta !_i = \max(!_i(e_i); !_i(u_i)) - !_i(u_i)$. Let $(g_i)_t = \sum_{wt} {}_{wt} g_i(w; t)$ be the the contribution of g_i to the sink t and $S = \{i \mid (g_i)_t < \}$.

Due to Property 3.1 and the fact that $C(u; f_t)$ is non-decreasing in f_t, we can see that $\Delta!_i + c(g_i)$ must be nonnegative for every $i \in M$. Therefore, we have that

$$\begin{aligned}
\frac{C(u;\tilde{f}_t)}{\tilde{f}_t - f_t} &\geq \frac{\sum_i \Delta!_i + \sum_i (u;\tilde{f}_t) - (u;f_t)}{\sum_i (g_i)_t} \\
&= \frac{\sum_i \Delta!_i) + \sum_i c(g_i)}{\sum_{i\in S}(g_i)_t + \sum_{i\in M\setminus S}(g_i)_t} \\
&\geq \frac{\sum_{i\in M\setminus S} \Delta!_i) + c(g_i)}{\sum_{i\in M\setminus S}(g_i)_t + n} \\
&\geq 1 - \frac{n}{\sum_{i\in M\setminus S}(g_i)_t} \frac{\sum_{i\in M\setminus S} \Delta!_i) + c(g_i)}{\sum_{i\in M\setminus S}(g_i)_t} \\
&\geq 1 - \frac{n}{} \min_i \frac{\Delta!_i + c(g_i)}{(g_i)_t};
\end{aligned}$$

The third inequality holds since $\sum_{i\in M\setminus S}(g_i)_t + n \geq$. Notice that g_i is a feasible flow on $\mathcal{R}(u + \Delta u_i e_i; f)$ by Lemma 3.1.4. Therefore $f + g_i$ is feasible on $G[u + \Delta u_i e_i]$ by Lemma 3.1.1 and has $(g_i)_t$ more units of flow received by t. Thus $\frac{\Delta!_i + c(g_i)}{(g_i)_t}$ is an upper bound of $\min_{;} \quad (i; ; ; u; f_t)$. □

Suppose the algorithm terminates at iteration k. The cost of our solution is $SOL \leq \sum_{i=1}^{k}$ $_i$ $_i$ where $_i$ and $_i$ are the minimum ratio and the amount of flow pushed to t, respectively, in iteration i. Let u^* be the optimal excess vector and $OPT = C(u^*; n)$. Consider the differential equation $\frac{df(x)}{dx} = \frac{OPT}{n-x}$: The solution for this equation is $f(x) = \ln \frac{n}{n-x}$ OPT for $x < n$, under boundary condition $f(0) = 0$. Let $= \sum_{j=1}^{i-1} 1 + \frac{n}{}$. Consider piecewise function $h(x) =$ $_i$ for $\sum_{j=1}^{i-1}$ $_j \leq x < \sum_{j=1}^{i}$ $_j$. It is easy to see that $h(x) \leq \frac{OPT}{n-x} = \frac{df(x)}{dx}$ from Lemma 3.4. Therefore,

$$\begin{aligned}
SOL &\leq \int_0^{n-} h(x)dx + \int_{n-}^{n} {}_k dx \\
&\leq \int_0^{n-} \frac{df(x)}{dx}dx + \int_{n-}^{n} \frac{OPT}{}dx \\
&\leq f(n-\ ) + \ OPT:
\end{aligned}$$

By letting $= \frac{}{n^2}$, we get $= (1 + O(\frac{1}{n}))$ and

$$SOL \leq \ \ln \frac{n}{} + O(1)\ OPT:$$

Remark: In fact the same algorithm works even if the underlying network is a general graph instead of a bipartite graph. Therefore, we can define GMA to be the more general problem: We are given a generalized network where some nodes are machine nodes. Our goal is to put u_i units of flow excess on machine i, for all i, such that n units of flow can be pushed to the sink t and the total cost $C(u; n)$ is minimized.

4 Applications

4.1 Machine Activation with Assignment Cost

We run the GMA-GREEDY algorithm for some $<$ 1 fixed later. All machines that have a non-zero flow excess are activated. Since $!_i(x) = \infty$ for $x > T$, the load on each machine is at most T. Suppose x is the min-cost fractional assignment given the set of the machines activated by GMA-GREEDY . In other words, $x_{ij} = f_{ij\ ij} = f_{ij}/p_j$ where f is the corresponding generalized flow. We then round the fractional assignment to an integral one using the scheme by Shmoys and Tardos [33].

The Shmoys-Tardos scheme constructs a bipartite graph $B = (M' \cup J; E)$ based on f, where J is the set of jobs and $M' = \{m_{is} \mid i \in M; s = 1; 2; : : : ; \lceil \sum_j x_{ij} \rceil\}$. Any matching in B corresponds to an assignment: edge $(m_{is}; j); m_{is} \in M'; j \in J$ is present in the matching iff we assign job j to machine i. B has the following properties: (1) There is a fractional matching in B that has value $\sum_{ij} x_{ij}$ and cost $\sum_{ij} x_{ij} c_{ij}$, (2) For any integral matching $\bar{x}$ in B, $\sum_{s;j} p_{ij} \bar{x}_{is;j} \leq T + p_{max}$ for any i. We refer interested readers to [33] for more details. Since $\sum_{ij} x_{ij} > n - > n - 1$, there exists an integral matching such that all jobs are matched. The second property implies that any integral assignment corresponding to a matching on B has makespan at most $T + p_{max}$. Finally, we can show that the cost of the integral matching, which is the assignment cost, is $\sum_{is;j} \bar{x}_{is;j} a_{ij} \leq \frac{1}{1-} \sum_{ij} x_{ij} a_{ij}$. In fact, this can be easily seen as follows: Let (x) be the optimal cost of sending x units of flow from s to t in a standard network with integral capacities. It is well-known that can be computed by repeatedly augmenting the shortest s-t paths (w.r.t cost). Thus, we can see that (x) is concave and piecewise linear and each linear piece is over an integral interval. Our claim follows the linearity of (x) on $[n-1; n]$. Therefore, the total cost is

$$\begin{aligned}
&\sum_{i\in M} !_i(u_i) + \sum_{is;j} \bar{x}_{is;j}\ a_{ij} \leq \frac{1}{1-} \sum_{i\in M} !_i(u_i) + \sum_{ij} x_{ij}\ a_{ij} \\
&\leq \frac{1}{1-}\ \ln n + \ln \frac{1}{} + O(1)\ OPT(T):
\end{aligned}$$

By letting $= \frac{1}{\ln^2 n}$, we get the following theorem.

Theorem 4.1. *We can find a solution, in polynomial time, to MAAC such that the makespan is at most* $2T$ *and the total cost is at most* $(1 + o(1)) \ln n\, OPT(T)$.

Using this theorem, we can immediately obtain a $(1+o(1))\ \ln n$-approximation for GMA if $!_i()$ satisfies $!_i(2x) \leq\ !\ _i(x)$ for $x \geq 0$ and all i. Note that positive non-decreasing concave functions satisfy the property for $\ = 2$.

For the basic version of the machine activation (MA) problem without assignment cost, we just set $\ = 1$ for GMA-GREEDY . Notice that the CLEANUP procedure is not necessary since there is no flow cost and opening new machines can only increase the cost. We do not need to set up any lower bound for $\$ to guarantee a polynomial running time because there are at most m iterations. In fact, the algorithm reduces to the simple greedy algorithm proposed in [23]. Moreover, since there is no flow cost, we do not need to pay the extra factor of $\frac{1}{1-}$ and the total opening cost is at most $\ln n + 1$ times the optimal solution.

Theorem 4.2. GMA-GREEDY nds in polynomial time a solution to *MA* such that the makespan is at most $2T$ and the total cost is at most$(\ln n + 1)\mathrm{OPT}(T)$.

4.2 Universal Facility Location A universal facility location (UniFL) instance consists of a bipartite graph $B = (M \cup J; E)$, where M is the set of facilities and J is the set of cities. Each facility is associated with a non-decreasing function $!_i() : \mathbb{Z}^+ \to \mathbb{R}^+$, which indicates that the facility cost for facility i is $!_i(x)$ if i serves x cities. There is also a connection cost c_{ij} incurred if j is assigned to i. We do not assume c_{ij} s satisfy triangle inequality. Our goal is to find an assignment of all cities to facilities such that the total cost, the sum of the facility costs and connection costs, is minimized.

For consistence of terminology, we shall just use machines and jobs in place of facilities and cities respectively. It is easy to see that UniFL is just a special case of GMA. We notice that once the capacity of each facility is fixed, the assignment cost can be computed from a standard network flow computation. Since a network with integral capacities always has an integral min-cost flow and the machine cost functions are only defined on integers, we can restrict the flow increments to only integers. In fact, the CLEANUP step is also not necessary (will be clear from the analysis). We remark the greedy algorithm after the simplifications is nothing but the set-cover greedy, however, no analysis was known before. Even for the capacitated facility location problem, a very special case of UniFL, the best known upper bound is $O(\log nM)$ [3] where M is the largest weight of the instance and the algorithm is a multi-phase greedy algorithm which is also more complicated than our algorithm UniFL-GREEDY .

Algorithm 3 : UniFL-GREEDY(G)

```
1 u = 0;
2 while |u| < n do
3 |   Choose machine i and  ∈ ℤ+ s.t.
4 |       (i; ; u) = (C(u+ e_i; |u+ e_i|) − C(u; |u|)) / ( ) is
  |   minimized;
5 |   u = u +  e_i;
```

Theorem 4.3. UniFL-GREEDY is a polynomial time $\ln n + 1$-approximation for *UniFL*.

Now, we analyze UniFL-GREEDY for UniFL. Notice that in our algorithm, once we increase the flow excess on a node by , the flow sent to t should also increase by . Thus, at any stage, the excess on any machine node is exhausted, or equivalently, the flow on the flow-generating self loop matches its capacity, which may not be true for generalized flows. To capture this change, we modify the semantics of $(u; f_t)$ to denote the cost of the optimal flow f in $G[u]$ such that f_t flow is received by t and the machine self-loops are saturated. In another word, $(u; f_t)$ is the optimal solution of LP(3.1) with additional constraints $f_{ii} = u_i\ \forall i$. If we deal with standard network flows, we can use (u) and $C(u)$ as shorthand notations for $(u; |u|)$ and $C(u; |u|)$ respectively, where $|u| = \sum_i u_i$. Again, we let u^* be the excess vector for the optimal solution.

To establish the approximation bound, we need to show Lemma 4.2 which is an analogue of Lemma 3.4. The following technical lemma is useful in Lemma 4.2. For generality, we prove this lemma for generalized flows, although we only need standard flows in this subsection.

Lemma 4.1. For any excess vector$u$ and $f_t < n$, there is an excess vector$e$ such that

1. $u \preceq e \preceq \max(u; u^*)$;
2. $(e; n) - \ (u; f_t) \leq\ (u^*; n)$.

Proof. Let f, f^* be the generalized flows corresponding to $(u; f_t)$ and $(u^*; n)$, respectively. First, it is easy to see that $f^* - f$ is a feasible flow on $\mathcal{R}(\max(u; u^*); f)$. By Lemma 2.1, we can decompose $f^* - f$ into circuits. We partition these circuits into two groups: the first group contains all circuits with t being their flow-absorbing cycle and the second contains the rest. Let g and h be the sum of the circuits in the first and the second groups, respectively. Roughly speaking, h is a flow going from loops with positive $u^* - u$ values to loops with negative $u^* - u$ values, while g flows from loops with positive

$u^* - u$ values to t. By Lemma 3.1.4, $f + g$ is a feasible flow in $G[\max(u; u^*)]$. Let $\tilde{f} = f + g$: It is obvious that $\tilde{f}$ sends n units of flow to t. Let $\tilde{e}$ be the excess vector of $\tilde{f}$, i.e., $\tilde{e} = \{\tilde{f}_{ii}\}_{i \in M}$. The feasibility of $\tilde{f}$ on $G[\max(u; u^*)]$ implies $\tilde{e} \preceq \max(u; u^*)$. Since each circuit in g contains only one flow-absorbing loop, which is t, we can see that the excess vector of g, $\{g_{ii}\}_{i \in M} \succeq 0$. Thus, $\tilde{e} = u + \{g_{ii}\}_{i \in M} \succeq u$ [3].

Now, we only need to check the second requirement. For contradiction, we suppose $(\tilde{e}; n) > (u^*; n) + (u; f_t) = c(f^*) + c(f)$. First we can see that $c(g) + c(h) = c(f^*) - c(f)$. Therefore, $(\tilde{e}; n) \leq c(\tilde{f}) = c(f + g) = c(f) + c(g) = c(f^*) - c(h)$. Hence, we have $c(f^*) + c(f) < (\tilde{e}; n) \leq c(f^*) - c(h)$ or equivalently $c(f) + c(h) = c(f + h) < 0$. But, by Lemma 3.1.4, $f + h$ is also a feasible flow in $G[\max(u; u^*)]$ that has f_t units of flow entering t. Therefore, $c(f + h)$ can not be negative. The contradiction proves the lemma. □

Using an argument similar to that in Lemma 3.4, we can show the following lemma. The key difference is to use the flow obtained from Lemma 4.1, instead of the optimal flow use in the proof of Lemma 3.4.

Lemma 4.2. For any u such that $(u) < +\infty$ and $|u| < n$, the following holds:

$$\min_{i;} (i; ; u) \leq \frac{C(u^*)}{n - |u|}:$$

Proof. Let f, $\tilde{f}$ be the generalized flows corresponding to (u) and $(\tilde{e})$, respectively, where $\tilde{e}$ is the excess vector obtained in Lemma 4.1. Consider the flow $\tilde{f} - f$ and apply the conformal decomposition lemma. We group the obtained circuits in the same way as in Lemma 3.4. Since $\tilde{e} \geq u$, each group must be a collection of (possibly non-simple) paths that carry some units of flow from a facility (machine) node to the sink. Therefore, the cost to flow ratio of any such group can be used as an upper bound of $\min_{i;} (i; ; u)$. The rest of the proof proceeds the same way as Lemma 3.4. Note that we do not need to pay the additional factor involving and . □

Using Lemma 4.2 and the same argument as in Section 3, we can easily show Theorem 4.3.

4.3 Generalized Submodular Covering

In this section, we discuss our improvement on the generalized submodular covering (GSC) problem [3], defined as follows. The programs of GSC are of the following form.

$$\text{minimize} \quad \sum_{i=1}^{m} !_i y_i + \sum_{j=1}^{q} c_j x_j$$

$$\text{subject to } Az \succeq b; \text{where } z = (y_1; \ldots; y_m; x_1; \ldots; x_q);$$
$$y_i \in \{0; 1\}; 0 \leq x_j \leq u_j \ \forall i; j$$

Moreover, b consists of nonnegative integers, $!_i$ and c_j are nonnegative and A is a $k \times (m+q)$ matrix, where the entries in the $k \times m$ submatrix A' are nonnegative integers and the $k \times q$ submatrix A'' is an "incidence matrix". Namely, each column of A'' contains exactly one "1", one "-1" and "0" elsewhere.

Now, we construct an equivalent GMA instance which consists of $m + k + 1$ nodes. The first m nodes $v_1; \ldots; v_m$ are machines, the last node t is the sink and each of the rest $w_1; \ldots; w_k$ corresponds to a row in A. For each v_i and w_j, there is an arc from v_i to w_j with capacity $A_{j;i}$. For each w_j, there is an arc from w_j to t with capacity b_j. For the j th column of A'', there is an arc with capacity u_j and cost c_j from the node w_a to w_b if the ath and bth row of that column are -1 and 1 respectively. All arcs are just normal arcs, i.e., the gain factor is 1. The cost function on machine i is $!_i(0) = 0$ and $!_i(x) = !_i$ for $x > 0$. The goal is to open some machines and push $|b| = \sum_j b_j$ units of flow into t such that the total cost is minimized.

As we remarked in Section 3, our results for GMA still hold even if the underlying network is a general graph. In fact, we can just use the greedy algorithm developed for UniFL, since the assignment cost $\sum_j c_j x_j$ in GSC can be also captured by a standard network flow computation. The only change is to replace n, the amount of flow needed to send to t, by $|b|$. If $|b|$ is polynomial in the input size, the algorithm clearly runs in polynomial time and the approximation bound extends trivially.

Theorem 4.4. If $|b|$ is bounded by a polynomial of the input size, there is a polynomial time $\ln |b| + 1$-approximation for *GSC*.

Now, we briefly sketch the greedy algorithm for the case when $|b|$ is not polynomially bounded in the GSC problem. The only change we need to make is that in Algorithm 3, we restrict the increment to be an integer at least $\max\{\frac{|b|-|u|}{n^2}; 1\}$ in each iteration. Using the same proof as in Lemma 3.4, we can show that

1. if $|b| - |u| > n^2$, then

$$\min_{i;} (i; ; u) \leq \left(1 + \frac{n}{|b| - |u|}\right) \frac{C(u^*)}{|b| - |u|};$$

[3] By default, we use $(i; i)$ to denote the flow-generating arc. If we write $g_{ii} < 0$, that means a positive flow on the arc of the opposite (flow-absorbing) direction.

2. if $|b| - |u| \le n^2$, then $= 1$ and

$$\min_{i;} \ (i; \ ; \ u) \le \frac{C(u^*)}{|b| - |u|}:$$

In either case, we have that $\min_{i;} \ (i; \ ; \ u) \le 1 + \frac{1}{n} \ \frac{C(u^*)}{|b|-|u|}$. Hence, the overall approximation ratio is $(1 + \frac{1}{n})(\ln |b| + 1) = (1 + o(1)) \ln |b|$. To see the polynomial running time, we notice that the demand $|b| - |u|$ decreases by a factor of $1 - \frac{1}{n^2}$ in each iteration. Therefore, there are at most $-\log_{1-\frac{1}{n^2}} |b| \approx n^2 \log |b|$ iterations.

Theorem 4.5. There is a polynomial time $(1 + o(1)) \ln |b|$-approximation for *GSC*, even when $|b|$ is not polynomially bounded.

Theorem 4.4 and 4.5 immediately imply tighter approximations for several network design problems that can be modeled by GSC, such as the average cost center problem, the fault tolerant facility location problem and the (non-metric) capacitated facility location problem. See [3] for their definitions and how to formulate them using GSC.

5 Machine Activation with Linear Constraints

In this section, we consider the machine activation problem where, for each machine, there is a constant number d of linear constraints. We denote this problem by MALC. The following integral linear program gives us the optimum schedule.

$$\begin{aligned} (5.3) \qquad \text{minimize} \quad & \sum_{i \in M} \ !_i y_i \\ \text{subject to} \quad & \sum_{j \in J} p_{ijk} \, x_{ij} \ \le T_{ik} \, ; \quad \forall i \in M; k = 1; : : : ; d \\ & \sum_{i \in M} x_{ij} \ = 1; \quad \forall j \in J \, ; \quad x_{ij} \ \le y_i \quad \forall i; j \ ; \\ & x_{ij} \ ; y_i \in \{0; 1\} \quad \forall i; j \end{aligned}$$

For each machine $i \in M$, we have a constant number d of linear constraints and we use k to index the constraints. We force $x_{ij} = 0$ if $p_{ijk} > T_{ik}$ for some k. By relaxing the constraint $x_{ij} \ ; y_i \in \{0; 1\}$ to $x_{ij} \ ; y_i \in [0; 1]$, we obtain the linear program relaxation. The main result of this section is the following:

Theorem 5.1. For any constant > 0, there is a polynomial time randomized algorithm P that returns an integral schedule $X_{ij} \ ; Y_i$, such that (1) $\sum_{j \in J} p_{ijk} \, X_{ij} \le (2d + \ )T_{ik}$ for each opened machine i ($P(Y_i = 1)$ and each $1 \le_P k \le d$ w.p. 1. (2) $\mathbb{E}[\sum_{i \in M} !_i Y_i] \le O(\frac{1}{} \log n) \sum_{i \in M} !_i y_i$.

If $d = 1$, we obtain a $(2 + \ ; O(\frac{1}{} \log n))$-approximation for the basic version of MA. This matches the approximation ratio obtained by the rounding scheme in [23].

We need some notations first. Let $(x; y)$ be the fractional optimal solution. We assume it is a **basic solution**, i.e. it corresponds to a vertex of the polytope of LP(5.3). If $x_{ij} = y_i$, we say edge $(i; j)$ is **tight**, otherwise we call it **floating**. A job j is called **-tight** if $\sum_{i:(i;j) \text{ is tight}} x_{ij} \ge$. We call the graph induced by all floating edges **the floating graph** and denote it by $\mathcal{F}$.

Our algorithm gradually modifies $\mathcal{F}$ and x_{ij} values. y_i values are unchanged throughout. During the iterations, we set floating x_{ij} either to y_i or to 0. $\mathcal{F}$ always contains only floating edges. Let $@(i)$ be the set of edges incident on node i in $\mathcal{F}$ and $\deg_{\mathcal{F}}(i) = |@(i)|$. Let $\mathcal{F}[S]$ denote the subgraph of $\mathcal{F}$ induced by the subset S of vertices. If $\deg_{\mathcal{F}}(v) = 1$, we say node v is a **singleton**.

The algorithm consists of three phases. See Algorithm 4 for the details. We can think of as a small constant, e.g., 0:1. Phase 1 contains a repeat loop. The loop repeatedly does the following: (1) throwing -tight jobs out of $\mathcal{F}$ and setting x values to zero for all floating edges incident on it; (2) making all edges incident on a low degree machine ($\deg \le 2d - 1$) or a singleton job tight. We show in Lemma 5.2, that $\mathcal{F}$ has a special structure after executing the loop. Then, if the y value is less than for any non-isolated machine node, we remove its adjacent edges and go back to the repeat loop. After Phase 1, $\mathcal{F}$ has a special structure and all non-isolated machines have y values at least . Phase 2 further modifies the structure of $\mathcal{F}$ by making use of these properties. The details of Phase 2 are provided after Lemma 5.2. After Phase 2, we have either $x_{ij} = y_i$ or $x_{ij} = 0$ for all $i; j$, and all jobs are -tight. In Phase 3, we scale up the fractional solution by a factor of $\frac{1}{}$ and get a instance of set cover where a machine i covers all jobs j with $x_{ij} = y_i$. We finally apply a standard rounding scheme.

Lemma 5.1. Let $(x; y)$ be any vertex solution of LP(5.3). Throughout the execution of the algorithm, for any subsets $S_1 \subseteq M$ and $S_2 \subseteq J$,

$$|E(\mathcal{F}[S_1 \cup S_2])| \le d|S_1| + |S_2|:$$

Proof. It suffices to only consider $\mathcal{F}$ at the beginning since we never introduce more floating edges during the execution. Suppose the lemma is not true. Consider the following system of linear equations with variables $x_{i;j} \ ; (i; j) \in \mathcal{F}[S_1 \cup S_2]$:

$$\sum_{j \in S_2} p_{ijk} \, x_{ij} \ = \sum_{j \in S_2} p_{ijk} \, x_{ij} \ ; \quad \forall i \in S_1; k = 1; : : : ; d$$