

Data Center Scheduling, Generalized Flows, and Submodularity

Lisa Fleischer*

Abstract

Inspired by problems in data center scheduling, we study the submodularity of certain scheduling problems as a function of the set of machine capacities and the corresponding implications. In particular, we

- give a short proof that, as a function of the excess vector, maximum generalized flow is submodular and minimum cost generalized flow is supermodular;
- extend Wolsey’s approximation guarantees for submodular covering problems to a new class of problems we call supermodular packing problems;
- use these results to get tighter approximation guarantees for several data center scheduling problems.

1 Introduction

Inspired by problems in data center scheduling introduced in [12], we study the submodularity of certain scheduling problems as a function of the set of machine capacities and the corresponding implications. In particular, we

1. give a short proof that, as a function of the excess vector, maximum generalized flow is submodular and minimum cost generalized flow is supermodular (Section 2);
2. extend Wolsey’s approximation guarantees for submodular covering problems to a new class of problems we call supermodular packing problems (Section 3);
3. use these results to get tighter approximation guarantees for several data center scheduling problems (Section 4).

In the *minimum cost data center problem with makespan constraint* (DCM), there is a set of data centers D with $|D| = m$ and a set of jobs J with $|J| = n$. The processing time of job j at data center i is p_{ij} . The cost to open data center i is c_i . The goal is to pick a minimum cost subset of D subject to being able to perform all the jobs on those open centers by a time T , where the time needed to process a subset S_i at data center i is $\sum_{j \in S_i} p_{ij}$. Motivation for data center scheduling problems of this type is discussed in [12].

*Computer Science, Dartmouth, Hanover, NH 03755. Email: lkf@cs.dartmouth.edu. This work supported in part by NSF grant CCF-0728869.

In the *job profit* version (JDC), each job j has a potential profit ϕ_j which is obtained only if the job is completed in full. The goal is to minimize opening costs plus lost profits of jobs not scheduled.

In the *assignment-cost* version (ADCM), there is also a cost c_{ij} for performing job j at data center i , and the goal is to minimize sum of opening costs and assignment costs, subject to performing all jobs at their assigned centers by time T .

In the *completion time* version (DCC), instead of the bound T on makespan, we have a bound K on total completion time: if $h_j(S)$ is the time job j completes in schedule S , then we want $\sum_j h_j(S) \leq H$.

For a fixed set of open data centers, the minimum total completion time schedule can be computed in polynomial time via a minimum cost flow [3, 11]. For a fixed set of open data centers, the makespan problems are NP-hard [13]. However, the number of jobs that can be fractionally completed by time T can be computed using a maximum generalized flow computation, and the assignment cost of this fractional schedule can be computed using a minimum cost generalized flow computation.

A generalized flow problem is like a standard flow problem, except with flow-multipliers on the arcs, called gains or losses. A gain factor of $\gamma > 0$ on arc e means that if f units of flow enter arc e , γf units leave arc e . Flow multipliers are used to model financial transactions, conversion of materials or goods in manufacturing, loss due to leakage or theft or spoilage in transshipment, or, in the case of scheduling problems, processing times.

The maximum generalized flow problem asks to maximize the flow reaching the sink. The minimum cost generalized flow problem asks to satisfy the specified demand at the sink or sinks, at minimum flow cost.

We use linear programming duality to demonstrate that the maximum generalized flow value as a function of the set of sources and sinks in a network is submodular. In particular, in Section 2, we give a short proof that the objective function for the minimum cost generalized flow is supermodular in the demand vector. A maximum generalized flow problem has an objective function that is the negative of a special minimum cost objective function. Since the negative of a supermodular function is

submodular, this implies that the maximum generalized flow is submodular.

To model a data center problem as a generalized flow problem, the data centers are the sources, and the jobs correspond to arcs of capacity and gain one directed into the sink, and there are arcs from data center to job with gain factor equal to the inverse of the processing time. Since the maximum flow volume as a function of the set of sources is submodular, the version of (DCM) that allows us to split jobs among different data centers is a submodular covering problem. We show that the problem (JDC) is also a submodular covering problem. To handle (DCC) and the version of (ADCM) that allows us to split jobs among different data centers, we define a class of problems called supermodular packing problems in Section 3. This class is applicable to (DCC) and (ADCM) since minimum flow cost is a supermodular function of the set of sources.

A well-known method to obtain approximately optimal solutions to submodular covering problems is the greedy algorithm, which is analyzed by Wolsey [22]. In terms of (DCM), this algorithm chooses data centers based on the smallest ratio of cost to added maximum flow value, to find a set of data centers that can fractionally serve the jobs J . We show this can yield a fractional solution with opening cost at most $(1 + \ln \frac{n}{\epsilon})\text{OPT}_{\text{DCM}}$ that completes by time $T(1+\epsilon)$. Here OPT is the cost of the minimum cost fractional solution, which is a lower bound on the true optimal cost. To get an integral solution, we can then apply the rounding techniques of Lenstra, Shmoys, and Tardos, for unrelated machine scheduling problems [13] to obtain a solution with cost $(1 + \ln \frac{n}{\epsilon})\text{OPT}_{\text{DCM}}$ that completes by time $T(2 + \epsilon)$. A similar analysis yields similar results for (JDC).

To solve (DCC), we show in Section 3 that Wolsey’s analysis for the submodular covering problem can be easily generalized to work for an appropriately defined supermodular packing problem. Since the corresponding unrelated machine scheduling problem is solvable in polynomial time, this yields an exact approximation algorithm that has average completion time at most H , and opening cost at most $1 + \ln(nH)$ times optimal.

Applying the analysis for supermodular covering to obtain bounds for (ADCM) on the sum of opening costs and assignment costs of a feasible fractional assignment requires some additional work. The issue is that our definition of supermodular packing assumes that the problem is trivially feasible (for sufficiently high flow cost). Thus, our analysis has to argue about the cost objective as well as the feasibility. In Section 4, we show that the greedy algorithm, together with appropriate pre- and post-processing including the by the rounding algorithm of Shmoys and Tardos [19] obtains a solution

with cost $(4 + 2 \ln \frac{n}{\epsilon})\text{OPT}_{\text{ADCM}}$ that completes by time $T(2 + \epsilon)$.

Our results on (DCM) and (ADCM) improve on the bicriteria approximation guarantees of the randomized rounding algorithm analyzed by Khuller, Li, and Saha [12].

Additional Related Work. There is a series of work for standard network flows (without losses and gains) demonstrating the submodularity and supermodularity of objective values of flow problems based on network parameters. Much of this concentrates on the parameter of arc capacities. The excess vector can be modelled using arc capacities by including arcs from a super source to each node with excess and arcs from each node with demand to a super sink. Interpreted in our framework, Shapley [17, 18] establishes submodularity of flow value in matchings, and relations among just two source demands. Megiddo gives a short proof for submodularity of maximum flow value in general graphs using the maximum-flow, minimum-cut theorem [14]. Supermodularity for minimum cost flows in standard networks is demonstrated by Gale and Politof [6]. Their work is extended further in [10]. Gautier and Granot [7] extend much of the work in [10] to flows with losses and gains. One of our contributions is to give an explicit, shorter, and different proof of sub- and supermodularity in generalized networks that relies only on linear programming duality. We also sketch an alternate proof for submodularity based on highest gain augmenting path algorithm for maximum generalized flows [8, 9, 15, 21].

2 Minimum Cost Generalized Flow is Supermodular

In this section we show that the objective functions of the minimum cost generalized flow and maximum generalized flow problems, as a function of the set of sources with excess, are supermodular and submodular functions, respectively.

Let $G = (V, E)$ be a directed graph with $|V| = n$. Arc (v, w) has capacity $u_{vw} \geq 0$, gain $\gamma_{vw} > 0$, and per-unit-flow cost $c_{vw} \geq 0$. Node v has excess b_v . If $b_v > 0$, v is a *source*; if $b_v < 0$ then v is a *sink*. Let V^+ be the set of source nodes and V^- be the set of sink nodes.

Let f_{vw} be the flow on arc (v, w) . The minimum cost generalized flow is an optimal solution to the following LP:

$$\begin{aligned} & \text{minimize} && \sum_{vw} c_{vw} f_{vw} \\ & \text{subject to} && \sum_w f_{vw} - \gamma_{vw} f_{wv} \leq b_v \quad \forall v \in V \\ & && f_{vw} \leq u_{vw} \quad \forall (v, w) \in E \\ & && f_{vw} \geq 0 \quad \forall (v, w) \in E \end{aligned}$$

The first constraint asks that the flow out of each source is at most its excess, and the flow into each sink is at

least its demand $-b_v > 0$. Since the objective is to minimize nonnegative costs, any optimal solution will satisfy each sink constraint exactly.

For $S \subset V$, let $\psi(S)$ denote the minimum cost flow with nonnegative excess just at the set of nodes S . That is, it is the minimum cost flow with excess vector b^S defined as $b_v^S = b_v$ for $v \in S \cup V^-$ and $b_w^S = 0$ for $w \in V \setminus (S \cup V^-)$.

Theorem 2.1 *ψ is a supermodular function. That is, that for any choice of $S \subset V^+$ and $\{j, k\} \in V^+ \setminus S$, the following inequality holds:*

$$\psi(S) - \psi(S + j) \geq \psi(S + k) - \psi(S + j + k).$$

This theorem can be expanded to include changes in sink demands as follows. Let $\pi(b)$ denote the minimum cost flow with excess vector $b \in \mathbb{R}^n$. Let e_i be the i^{th} elementary vector: $e_i(i) = 1$ and $e_i(j) = 0$ for $j \neq i$.

Theorem 2.2 *For any $b \in \mathbb{R}^n$, $\{j, k\} \in V$, and scalars δ_j and δ_k the following inequality holds:*

$$(2.1) \quad \pi(b) - \pi(b + \delta_j e_j) \geq \pi(b + \delta_k e_k) - \pi(b + \delta_j e_j + \delta_k e_k).$$

Setting $b = b^S$, $\delta_j = b_j$, and $\delta_k = b_k$ in Theorem 2.2 yields Theorem 2.1.

Proof of Theorem 2.2. To prove (2.1), we'll show below in Lemma 2.3 that for two excess vectors b^* and \hat{b} with $\hat{b}_i = b_i^*$ for all $i \neq k$ and $\hat{b}_k = b_k^* + \delta$ for $\delta > 0$, the corresponding minimum cost flows f^* and \hat{f} have marginal value (with respect to the objective function of minimizing flow cost) of sending more flow from j (or less flow to j if $b_j^* < 0$) is smaller in \hat{f} than in f^* . Since this is true for all values of b_j^* and b_k^* , we have that

$$\begin{aligned} \pi(b) - \pi(b + \delta_j e_j) &= \int_{\delta_j}^0 \frac{d\pi(b + x e_j)}{dx} dx \\ &\geq \int_{\delta_j}^0 \frac{d\pi(b + x e_j + \delta_k e_k)}{dx} dx \\ &= \pi(b + \delta_k e_k) - \pi(b + \delta_j e_j + \delta_k e_k) \end{aligned}$$

Linear programming duality theory says that the marginal value of the right hand side of a constraint equals to the value of the corresponding dual variable in an optimal dual solution.¹ In other words, if we change the right hand side value of constraint i by ϵ for small enough ϵ , then the objective value changes by ϵ times

¹This is often termed the ‘‘economic interpretation of dual variables’’. More information and details can be obtained from any standard text on linear optimization, for example Theorem 5.5 in [4], Section 4.4 in [2], or Section 10.4 in [16].

the value of the corresponding dual variable. The dual to the minimum cost generalized flow LP is

$$\begin{aligned} &\text{maximize} \quad \sum_v b_v q_v + \sum_{vw} u_{vw} y_{vw} \\ &\text{subject to} \quad q_v - \gamma_{vw} q_w + y_{vw} \leq c_{vw} \quad \forall (v, w), \\ &\quad \quad \quad q, y \leq 0 \end{aligned}$$

where q_v is the dual variable corresponding to the flow volume constraint for node v . Substituting $r_v = -q_v$ and $z_{vw} = -y_{vw}$, this problem is equivalent to

$$(2.2) \quad \begin{aligned} &\text{minimize} \quad \sum_v b_v r_v + \sum_{vw} u_{vw} z_{vw} \\ &\quad \quad \quad r_v - \gamma_{vw} r_w + z_{vw} \geq -c_{vw} \quad \forall (v, w) \\ &\quad \quad \quad r, z \geq 0 \end{aligned}$$

We consider the dual parametrized by the vector b and refer to the dual problem for b as $\mathbf{D}(b)$. Once r is given, the z minimizing the dual objective function is uniquely defined as $z_{vw} = [-c_{vw} + \gamma_{vw} r_w - r_v]^+$, where $[a]^+ := \max\{0, a\}$. Thus we can refer to dual optimal solutions in terms of r . To finish our proof of supermodularity of π , it remains to show that as we increase b , r decreases, and therefore q becomes less negative.

Lemma 2.3 *Let r^* and \hat{r} be the respective optimal solutions to $\mathbf{D}(b^*)$ and $\mathbf{D}(\hat{b})$. If $\hat{b} \geq b^*$ then $\hat{r} \leq r^*$.*

□

Proof of Lemma 2.3 We prove that if $\hat{b}_l = b_l^*$ for all $l \neq i$ and $\hat{b}_i > b_i^*$ then $\hat{r} \leq r^*$. The lemma then follows by induction. For r^* and \hat{r} , the corresponding z -vectors are z^* and \hat{z} .

If r^* is not optimal for the problem with \hat{b} , we have

$$\begin{aligned} \hat{b}^T \hat{r} + u^T \hat{z} &< \hat{b}^T r^* + u^T z^* \\ (b^*)^T r^* + u^T z^* &\leq (b^*)^T \hat{r} + u^T \hat{z} \end{aligned}$$

Summing these two inequalities, canceling like terms, and rearranging, we have that $(\hat{b}_i - b_i^*)(r_i^* - \hat{r}_i) > 0$, which implies that $\hat{r}_i < r_i^*$.

Suppose that the lemma is not true, and there is an index l with $\hat{r}_l > r_l^*$. Define \tilde{r} to be the vector defined by $\tilde{r}_l = [\hat{r}_l - r_l^*]^+$. Note that $\tilde{r}_i = 0$. Define \tilde{z} so that $\hat{z} - \tilde{z}$ is the z -vector corresponding to $\hat{r} - \tilde{r}$. Thus, $\tilde{z}_{vw} = \min\{\hat{z}_{vw}, (\hat{r}_v - \tilde{r}_v) - \gamma_{vw}(\hat{r}_w - \tilde{r}_w) + \hat{z}_{vw} + c_{vw}\}$. By definition $(\hat{r} - \tilde{r}, \hat{z} - \tilde{z})$ is feasible for the dual. Since it is not optimal, we have that

$$\hat{b}^T \hat{r} + u^T \hat{z} < \hat{b}^T (\hat{r} - \tilde{r}) + u^T (\hat{z} - \tilde{z}),$$

and hence $(b^*)^T \tilde{r} + u^T \tilde{z} < 0$. Since $\tilde{r}_i = 0$, this implies that $(r^* + \tilde{r}, z^* + \tilde{z})$ has strictly better objective value for $\mathbf{D}(b^*)$ than does (r^*, z^*) . This would contradict the optimality of r^* if $(r^* + \tilde{r}, z^* + \tilde{z})$ is feasible for the dual, and thus prove the lemma. So we check feasibility.

Let $V^+ := \{v \in V \mid \hat{r}_v > r_v^*\}$. Let $V^- = V \setminus V^+$. Note that for $v \in V^+$, $r_v^* = \hat{r}_v - \tilde{r}_v$, so that for $\{v, w\} \subseteq V^+$, $z_{vw}^* = \hat{z}_{vw} - \tilde{z}_{vw}$. Thus it suffices to check feasibility for pairs $\{v, w\}$ that have one endpoint in V^+ and one in V^- . Below, we show that $(r^* + \tilde{r}, z^* + \tilde{z})$ satisfies (2.2).

Case 1: $v \in V^-, w \in V^+$. Then, $\hat{r}_v - \tilde{r}_v = \hat{r}_v$, $\hat{r}_w - \tilde{r}_w = r_w^*$, $r_v^* + \tilde{r}_v = r_v^* \geq \hat{r}_v$, and $r_w^* + \tilde{r}_w = \hat{r}_w$. Thus $\tilde{z}_{vw} = \min\{\hat{z}_{vw}, \hat{r}_v - \gamma_{vw}r_w^* + \hat{z}_{vw} + c_{vw}\}$. If $\tilde{z}_{vw} = \hat{z}_{vw}$, checking (2.2) for $(r^* + \tilde{r}, z^* + \tilde{z})$, we have that $r_v^* - \gamma_{vw}\hat{r}_w + z_{vw}^* + \tilde{z}_{vw} \geq \hat{r}_v - \gamma_{vw}\hat{r}_w + \hat{z}_{vw} \geq -c_{vw}$, where the last inequality follows by feasibility of (\hat{r}, \hat{z}) . Otherwise,

$$\begin{aligned} r_v^* - \gamma_{vw}\hat{r}_w + z_{vw}^* + \tilde{z}_{vw} &= r_v^* - \gamma_{vw}\hat{r}_w + z_{vw}^* + \hat{r}_v - \gamma_{vw}r_w^* + \hat{z}_{vw} + c_{vw} \\ &= (r_v^* - \gamma_{vw}r_w^* + z_{vw}^*) + (\hat{r}_v - \gamma_{vw}\hat{r}_w + \hat{z}_{vw}) + c_{vw} \\ &\geq -c_{vw}, \end{aligned}$$

where the inequality follows from the feasibility of (r^*, z^*) and (\hat{r}, \hat{z}) .

Case 2: $v \in V^+, w \in V^-$. Then, $\hat{r}_v - \tilde{r}_v = r_v^*$, $\hat{r}_w - \tilde{r}_w = \hat{r}_w$, $r_v^* + \tilde{r}_v = \hat{r}_v$, and $r_w^* + \tilde{r}_w = r_w^*$. Thus $\tilde{z}_{vw} = \min\{\hat{z}_{vw}, r_v^* - \gamma_{vw}\hat{r}_w + \hat{z}_{vw} + c_{vw}\}$. For $(r^* + \tilde{r}, z^* + \tilde{z})$, to satisfy (2.2), $\hat{r}_v - \gamma_{vw}r_w^* + \hat{z}_{vw} + \tilde{z}_{vw}$ should be at least $-c_{vw}$. The sum of the first three terms is greater than the sum of the terms in the constraint for (\hat{r}, \hat{z}) , so the only problem could occur when \tilde{z}_{vw} is negative. In this case,

$$\begin{aligned} \hat{r}_v - \gamma_{vw}r_w^* + z_{vw}^* + \tilde{z}_{vw} &= \hat{r}_v - \gamma_{vw}r_w^* + z_{vw}^* + r_v^* - \gamma_{vw}\hat{r}_w + \hat{z}_{vw} + c_{vw} \\ &= (r_v^* - \gamma_{vw}r_w^* + z_{vw}^*) + (\hat{r}_v - \gamma_{vw}\hat{r}_w + \hat{z}_{vw}) + c_{vw} \\ &\geq -c_{vw}, \end{aligned}$$

where the inequality follows from the feasibility of (r^*, z^*) and (\hat{r}, \hat{z}) . \square

Maximum generalized flow is submodular.

In the maximum generalized flow problem, there is a special node t called the *sink*. For this node, $b_t = 0$. For all other nodes v , $b_v \geq 0$. With these parameters, the set of constraints of the maximum generalized flow problem is the same as for the minimum cost generalized flow problem. The objective for the maximum generalized flow problem is

$$\text{maximize } \sum_v \gamma_{vt} f_{vt},$$

or equivalently,

$$\text{minimize } \sum_v -\gamma_{vt} f_{vt},$$

This objective, having negative coefficients, is the negative of a possible objective function for the minimum cost flow problem. Hence the value of the objective as a function of the demand vector b , or more specifically, as a function of the set of nodes S with positive excess, say $\kappa(S)$, is a submodular function.

Theorem 2.4 κ is a submodular function. That is, that for any choice of $S \subset V - t$ and $\{j, k\} \in V \setminus (S + t)$, the following inequality holds:

$$\kappa(S + j) - \kappa(S) \geq \kappa(S + j + k) - \kappa(S + k).$$

\square

An alternate proof of Theorem 2.4 could be obtained using properties of the highest gain augmenting path algorithm for maximum generalized flows [8, 9, 15, 21]. The highest gain augmenting path algorithm first cancels gainy cycles and then maintains node labels that are the reciprocals of the gain of maximum gain path in a relabeled residual graph from each node to the sink. The gain of this path for node u is the marginal benefit of increasing b_u . Augmenting on maximum gain paths and updating node labels maintains the property that the node labels are only increasing throughout the algorithm, and thus the marginals are only decreasing.

3 The Greedy Algorithm for Supermodular Packing Problems

In this section, we present the supermodular packing problem, and prove some approximation guarantees for the solution produced by the greedy algorithm for it.

Let f be a monotone (nondecreasing) submodular set function on ground set V and c a vector of costs for items in V , so that $c(S) := \sum_{i \in S} c_i$ is a modular function on V . For any bound F , the problem

$$Z := \min_{S \subseteq V} \{c(S) : f(S) \geq F\}$$

is a *submodular covering problem*.

Let g be a monotone (nonincreasing) supermodular set function on ground set V and c the a vector of costs as above. For any bound G , the problem

$$Y := \min_{S \subseteq V} \{c(S) : g(S) \leq G\}$$

is a *supermodular packing problem*.

Input: ground set V ,
 $f : 2^V \rightarrow \mathbb{R}$ monotone submodular,
 $c : V \rightarrow \mathbb{R}^+$, $F \in \mathbb{R}$.

Initialize $S^0 = \emptyset$, $t = 0$.
while $f(S^t) < F$ and $|S^t| < |V|$
 $t \leftarrow t + 1$,
 $\theta^t \leftarrow \min_{j \in V \setminus S^{t-1}} \frac{c_j}{\rho_j(S^{t-1})}$,
 $j_t \leftarrow \operatorname{argmin}_{j \in V \setminus S^{t-1}} \frac{c_j}{\rho_j(S^{t-1})}$,
 $S^t \leftarrow S^{t-1} \cup \{j_t\}$
end while
 $\tau \leftarrow t$
return S^τ .

Figure 1: Greedy algorithm for submodular covering.

Input: ground set V ,
 $g : 2^V \rightarrow \mathbb{R}$ monotone supermodular,
 $c : V \rightarrow \mathbb{R}^+$, $G \in \mathbb{R}$.

Initialize $R^0 = \emptyset$, $t = 0$.
while $f(R^t) > G$ and $|R^t| < |V|$
 $t \leftarrow t + 1$,
 $\theta^t \leftarrow \min_{j \in V \setminus R^{t-1}} \frac{c_j}{\mu_j(R^{t-1})}$,
 $j_t \leftarrow \operatorname{argmin}_{j \in V \setminus R^{t-1}} \frac{c_j}{\mu_j(R^{t-1})}$,
 $R^t \leftarrow R^{t-1} \cup \{j_t\}$
end while
 $\tau \leftarrow t$
return R^τ .

Figure 2: Greedy algorithm for supermodular packing.

For each of these problems, we can define a corresponding greedy algorithm. Let $\rho_j(S) := f(S \cup j) - f(S) \geq 0$ and $\mu_j(S) := g(S) - g(S \cup j) \geq 0$.

Let $Z^G := \sum_{i \in S^\tau} c_i$ and $Y^G := \sum_{i \in R^\tau} c_i$ be the cost of the greedy solutions returned by each algorithm. The greedy algorithm for the submodular covering problem is analyzed by Wolsey in [22], in which he gives several bounds on the approximation guarantee. Of relevance for this work, he proves the following.

Theorem 3.1 (Wolsey [22]) *If Z is the optimal value of the submodular covering problem and Z^G is the solution value returned by the greedy algorithm, then*

$$(a) \frac{Z^G}{Z} \leq 1 + \ln(\max_{j,t} \{ \frac{\rho_j(S^0)}{\rho_j(S^t)} : \rho_j(S^t) > 0 \}).$$

$$(b) \frac{Z^G}{Z} \leq 1 + \ln(\frac{\theta^\tau}{\theta^1}).$$

$$(c) \frac{Z^G}{Z} \leq 1 + \ln(\frac{F - f(\emptyset)}{F - f(S^{\tau-1})}).$$

For completeness, we show here that the same arguments yield a symmetric claim for the supermodular covering problem.

Theorem 3.2 *If Y is the optimal value of the supermodular packing problem and Y^G is the solution value returned by the greedy algorithm, then*

$$(a) \frac{Y^G}{Y} \leq 1 + \ln(\max_{j,r} \{ \frac{\mu_j(R^0)}{\mu_j(R^r)} : \mu_j(R^r) > 0 \}),$$

$$(b) \frac{Y^G}{Y} \leq 1 + \ln(\frac{\theta^\tau}{\theta^1}).$$

$$(c) \frac{Y^G}{Y} \leq 1 + \ln(\frac{g(\emptyset) - G}{g(R^{\tau-1}) - G}).$$

To prove this, we employ the following lemma of Wolsey, and some additional facts about supermodular functions.

Lemma 3.3 (Wolsey [22]) *Let $0 < u_1 \leq u_2 \leq \dots \leq u_n$, and $x_1 \geq x_2 \geq \dots \geq x_n > 0$. If $S = \sum_{i=1}^{n-1} u_i(x_i - x_{i+1}) + u_n x_n = u_1 x_1 + \sum (u_{i+1} - u_i) x_{i+1}$, then*

$$S \leq (\max_i u_i x_i) \left[1 + \ln \left(\min \left\{ \frac{x_1}{x_n}, \frac{u_n}{u_1} \right\} \right) \right].$$

The following lemma is easily checked.

Lemma 3.4 *A set function g is supermodular and nonincreasing if and only if either*

- $\mu_j(S) \geq \mu_j(R) \geq 0$, $\forall S \subseteq T \subseteq V$, or
- $g(S) \leq g(T) + \sum_{j \in T \setminus S} \mu_j(S)$.

Consider the integer program \mathbf{Q}^I

$$Y^I := \min \sum_{j \in V} c_j y_j$$

$$\sum_{j \in V} \mu_j(R) y_j \geq g(R) - G, \quad \forall R \subseteq V$$

$$y_j \in \{0, 1\} \quad j \in V$$

Lemma 3.5 *W is a feasible supermodular packing if and only if its characteristic vector y^W is feasible in \mathbf{Q}^I .*

The proof of Lemma 3.5 is symmetric to the proof of Proposition 2 in [22], and is omitted here.

Proof of Theorem 3.2. Let $k_3 := \frac{g(\emptyset) - G}{g(R^{\tau-1}) - G}$, $k_1 := \max_{j,t} \left\{ \frac{\mu_j(R^0)}{\mu_j(R^t)} : \mu_j(R^t) > 0 \right\}$, and $k_2 := \frac{\theta^\tau}{\theta^1}$.

To obtain a lower bound on Y , we use a linear program relaxation of the supermodular packing problem, called \mathbf{Q}^L .

$$Y^L := \min \begin{array}{l} \sum_{j \in V} c_j y_j \\ \sum_{j \in V} \mu_j(R^t) y_j \geq g(R^t) - G, \quad t \leq \tau - 1 \\ y_j \geq 0, \quad j \in V \end{array}$$

By linear programming duality, a feasible solution to the LP dual of \mathbf{Q}^L provides a lower bound on Y^L , and hence on Y . The LP dual is

$$\begin{array}{ll} \max & \sum_{t=0}^{\tau-1} [g(R^t) - G] w_t \\ \text{s.t.} & \sum_{t=0}^{\tau-1} \mu_j(R^t) w_t \leq c_j \quad j \in V \\ & w_t \geq 0, \quad t = 0, \dots, \tau - 1 \end{array}$$

(a) & (b): Let $\xi^* := (\theta^1, \theta^2 - \theta^1, \dots, \theta^\tau - \theta^{\tau-1})$. For each j , there exists an $r \leq \tau$ such that $\mu_j(R^{r-1}) > 0$ and $\mu_j(R^r) = 0$. Applying Lemma 3.3 to $0 < \theta^1 \leq \dots \leq \theta^r$ and $\mu_j(R^0) \geq \dots \geq \mu_j(R^{r-1}) > 0$ yields

$$\begin{aligned} & \theta^1 \mu_j(R^0) + (\theta^2 - \theta^1) \mu_j(R^1) + \dots \\ & \quad + (\theta^r - \theta^{r-1}) \mu_j(R^{r-1}) \\ & \leq \max_{t=1, \dots, r} \{ \theta^t \mu_j(R^{t-1}) \} \times \\ & \quad \left[1 + \ln \left(\min \left\{ \frac{\mu_j(R^{r-1})}{\mu_j(R^0)}, \frac{\theta^r}{\theta^1} \right\} \right) \right] \\ & \leq c_j [1 + \ln(\min\{k_1, k_2\})], \end{aligned}$$

where the last inequality follows from the definition of θ in the greedy algorithm. Thus $(1 + \ln \min\{k_1, k_2\})^{-1} \xi$ is feasible for the dual. Hence, by weak LP duality,

$$\begin{aligned} & (1 + \ln \min\{k_1, k_2\})^{-1} \times \\ & \quad [\theta^1(g(R^0) - G) + \sum_{t=2}^{\tau} (\theta^t - \theta^{t-1})(g(R^{t-1}) - G)] \\ & \leq Y^L \leq Y \end{aligned}$$

The greedy solution has value

$$\begin{aligned} Y^G &= \sum_{t=1}^{\tau} \theta^t (g(R^{t-1}) - g(R^t)) \\ &= \theta^1 (g(R^0) - G) + \sum_{t=2}^{\tau} (\theta^t - \theta^{t-1})(g(R^{t-1}) - G). \end{aligned}$$

Hence $Y^G \leq Y[1 + \ln(\min\{k_1, k_2\})]$.

(c): Let $u_i^t = \theta^t$ if $i = t$, $u_i^t = 0$ otherwise. Then $(u^t)^T (\mu_j(R^0), \dots, \mu_j(R^{\tau-1})) = \theta^t \mu_j(R^{t-1}) \leq c_j$, and hence u^t is dual feasible for $t = 1, \dots, \tau$. Thus,

$$\begin{aligned} & \max_{t=1, \dots, \tau} (u^t)^T (g(R^0) - G, \dots, g(R^{\tau-1}) - G) \\ &= \max_{t=1, \dots, \tau} \theta^t (g(R^{t-1}) - G) \leq Y^L \leq Y. \end{aligned}$$

Applying Lemma 3.3, with $0 < \theta^1 \leq \dots \leq \theta^\tau$ and $g(R^0) - G \geq g(R^1) - G \geq \dots \geq g(R^{\tau-1}) - G$ gives

$$\begin{aligned} Y^G &= \sum_{t=1}^{\tau} \theta^t (g(R^{t-1}) - g(R^t)) + \theta^\tau (g(R^{\tau-1}) - G) \\ &\leq \max_t \{ \theta^t (g(R^{t-1}) - G) \} \left[1 + \ln \frac{g(R^0) - G}{g(R^{\tau-1}) - G} \right] \\ &\leq Y(1 + \ln k_3). \end{aligned}$$

□

The next lemma describes a property of the greedy algorithm applied to feasible supermodular packing problems that is used in the next section.

Lemma 3.6 *Let R^* be the set of centers opened in the optimal solution. For all $t = 0, \dots, \tau - 1$, $\theta^t \leq \frac{c(R^*) - c(R^{t-1})}{g(R^{t-1}) - G}$.*

Proof. By definition of θ^t , we have that

$$[g(R^{t-1}) - g(R^{t-1} \cup \{j\})] \theta^t \leq c_j.$$

Summing this over all $j \in R^* \setminus R^{t-1}$ yields

$$\theta^t \sum_{j \in R^* \setminus R^{t-1}} \mu_j(R^{t-1}) \leq c(R^* \setminus R^{t-1}).$$

Using Lemma 3.4(b) and rearranging finishes the proof:

$$\theta^t \leq \frac{c(R^*) - c(R^{t-1})}{g(R^{t-1}) - g(R^*)} \leq \frac{c(R^*) - c(R^{t-1})}{g(R^{t-1}) - G}.$$

□

4 Data Center Scheduling

In this section, we give polynomial-time approximation algorithms for the data-center scheduling problems (DCM), (JDC), (DCC), and (ADCM) defined in the introduction, and prove bicriteria approximation guarantees for each of them.

Data center selection problems are a version of unrelated machine scheduling problems. These are

commonly modeled as generalized flow problems [13, 19]. For (DCM), given a fixed set of open data centers, we can determine if it is feasible to fractionally assign all jobs to centers by solving a maximum generalized flow problem. There are nodes for each data center and each job. An arc from data center i to job j has infinite capacity and loss $\frac{1}{p_{ij}}$. The open centers have supply T , and the jobs have demand 1. This can be extended to a single sink problem by adding an arc from each job node to a super sink t with capacity 1 and gain 1. The goal is to select a minimum cost subset of the data centers to open so that the maximum integral flow value at t is n .

For (ADCM), there is an additional per-unit flow-cost of $\frac{c_{ij}}{p_{ij}}$ on the arc from data center i to job j . The goal is to select a set of data centers to open so that every sink demand is saturated, and the sum of opening costs and assignment (flow) costs is minimum. In order to remove the issue of flow feasibility when choosing data centers to open in the greedy algorithm, we introduce a data center η of cost 0 that is open at the start. We assume η can process all jobs by the given time T (by introducing an arc to each job with gain n/T) at sufficiently high cost per job M so that no dummy arc or center is used in an optimal solution. If $C = \max_{ij} c_{ij}$, then $M = 2nC$ is sufficient, since the maximum cost (simple) augmenting path restricted to original arcs is nC : at most cost of C for each forward arc in an augmenting path. For a fixed set of centers, the minimum cost generalized flow finds a feasible flow with minimum flow cost.

Integral generalized flow problems are NP-hard. However, the fractional versions (both maximum flow and minimum cost flow) are solvable in polynomial time exactly via linear programming, or via combinatorial techniques [1, 9, 20]. All of these techniques can start with a partial solution and obtain a new solution for a problem with one additional source or sink much more efficiently than starting from scratch. In addition, there are numerous polynomial time approximation schemes that can be used to speed up the run time [5, 20].

The fractional solution to the generalized flow problem for (DCP) and (ADCM) yields a fractional assignment of jobs to data centers. The fraction of job j assigned to data center i is equal to the amount of flow arriving at the node for job j from the node for data center i , or equivalently, $\frac{f_{ij}}{p_{ij}}$.

To prevent the fractional assignment of a job to a center that would be infeasible by time T in any integral assignment (and thus in the optimal assignment), we perform a preprocessing step before solving the fractional generalized flow problem that sets $p_{ij} = \infty$ if $p_{ij} > T$, and we do not include these arcs in the gener-

alized flow graph.

4.1 The Minimum Opening-Cost Problem

Above, we describe how, given a fixed set of open data centers, we can determine if it is feasible to fractionally assign all jobs to centers by solving a maximum generalized flow problem. Let κ_T be the maximum flow value for time bound T as a function of the set of open data centers. By Theorem 2.4, the κ_T is a submodular function. Thus, we can use the greedy algorithm for the submodular covering problem. We do this with inputs $f = \kappa_T, c, F = n, D$.

If at termination, $\kappa_T(S^\tau)$ is less the number of jobs, then the problem is infeasible for the given time bound T . Otherwise, we consider the first time σ when $\kappa(S^\sigma) \geq n - \epsilon$, for a chosen approximation parameter $\epsilon > 0$. By Theorem 3.1(c), the cost of the set of centers S^σ is at most $(1 + \ln \frac{n - \kappa(\emptyset)}{n - \kappa(S^{\sigma-1})}) \text{OPT}_{\text{DCM}} = (1 + \ln \frac{n}{\epsilon}) \text{OPT}_{\text{DCM}}$.

To get an integral schedule, we temporarily reassign the processing time of a job j to be the fraction of the processing time that completes by time T in the fractional schedule, so that with the new processing times, all jobs complete by time T . We next apply the rounding technique of Lenstra, Shmoys, and Tardos [13] to this fractional schedule, to get an assignment that completes by time $2T$ with the the modified processing times, in which each job is assigned to exactly one machine. We next return each job its original processing time. Since in total ϵ of the jobs were not scheduled in the fractional schedule, the total change in processing times is at most $\epsilon P \leq \epsilon T$, where $P := \max_{ij} \{p_{ij} : p_{ij} < \infty\}$. Hence, with the original processing times, all jobs complete by time $(2 + \epsilon)T$.

Theorem 4.1 *The greedy algorithm is a $(1 + \log \frac{n}{\epsilon}, 2 + \epsilon)$ bicriteria approximation algorithm for the minimum cost data center scheduling problem.*

The greedy algorithm improves the approximation guarantee of the randomized rounding-based $(O((1 + \frac{1}{\epsilon}) \log n), 2 + \epsilon)$ bicriteria algorithm analyzed by Khuller, Li, and Saha [12].

Job profits. We can also accomodate job profits into this model. Suppose each job has profit ϕ_j . We seek a schedule that completes by time T that minimizes opening costs of centers plus profits of jobs not scheduled. At each step of the greedy algorithm, we can pick either a center to open, or a job to exclude from the schedule. We run the submodular covering algorithm as above, but with the function f equal to the number of jobs excluded from the schedule plus $\psi_T(S)$, and with $F = n$. This f is a submodular function since

excluding a job j is equivalent to changing b_j from -1 to 0 , and the proof of supermodularity/submodularity in Section 2 of the negative function allows positive changes to any b value.

The above analysis implies that the fractional assignment found by the greedy algorithm can be converted into an integral schedule that completes by time $T(2 + \epsilon)$ and has total cost plus lost profit at most $(1 + \log \frac{n}{\epsilon})$ times the cost plus lost profit of the optimal solution.

Theorem 4.2 *The greedy algorithm is a $(1 + \log \frac{n}{\epsilon}, 2 + \epsilon)$ bicriteria approximation algorithm for (JDC).*

4.2 The Average Completion Time Objective

In the average completion time version (DCC), we are given a bound H on the total completion time of all the jobs. We want to pick a minimum cost set of centers so that there is a nonpreemptive schedule on these centers with sum of completion times at most H . Given an unrelated machine scheduling with a fixed set of centers, the minimum total completion time can be computed with a single minimum cost flow computation [3, 11]. The graph for this computation has a node for every job and n nodes for every machine. The arc between machine node i_k and job j has cost kp_{ij} and capacity 1. If there is flow on this arc, that represents that j is the k^{th} -to-last job scheduled on machine i . We add to this graph a node for every machine i . Node i is connected to i_k for all $1 \leq k \leq n$ with an arc of unit capacity and 0 cost. Node i can have excess n if i is selected as a data center.

To solve the minimum cost data center problem for the bound on the completion time, we first create a dummy center that has $p_{ij} = H$ for all jobs j and then run the greedy algorithm with R^0 equal to the dummy center, and $G = H$. We have $g(R^0) = nH$ and if all processing times are integer, we have $g(R^{\tau-1}) - G \geq 1$. Plugging these bounds into the expression in Theorem 3.2(c) implies the following approximation guarantee.

Theorem 4.3 *There is a polynomial time approximation algorithm for the minimum cost data center problem under a total completion time constraint with approximation guarantee $(1 + \ln(nH))\text{OPT}_{\text{DCC}}$.*

A trivial upper bound on H is $n^2P/2$ where P is the maximum processing time, so we could replace H in the approximation guarantee with this value.

The supermodular packing algorithm can be applied to other data center problems with similar results. For example, if the processing time of job j on machine i is l_i , and H is a bound on the objective $\sum_j w_j(h_j(S))$

where $h_j(S)$ is the completion time of job j in schedule S and $w_j(t)$ is a nonnegative, monotone function in t . Then a construction similar to the previous analysis yields a schedule S satisfying $\sum_j w_j(h_j(S)) \leq H$ and an opening cost that is at most logarithmic approximation to the optimal opening cost for bound H .

4.3 The Minimum Assignment-Cost Problem

In the beginning of Section 4, we describe how, given a fixed set of data centers, we can find the minimum cost fractional assignment of jobs to the open data centers by solving a minimum cost generalized flow problem. If completion by time T is infeasible, then the solution will utilize the dummy arcs, at per-unit flow cost of $M = 2nC$.

In order to get good bounds on the cost of the solution found by the greedy algorithm, we do some additional preprocessing. Let C_O be the opening cost of the set of data centers in the optimal solution to (ADCM). (We can find $C_O \leq \bar{C} \leq 2C_O$ by binary search in the interval $[\min_{ij} c_{ij}, \bar{C}]$.) We initially open all data centers i with $c_i < \frac{\bar{C}}{n}$. Let D^0 be these set of jobs. There could be more than n such centers if $m > n$, but in the rounding step, we will close all but at most n of them, so that the total contribution to the opening cost of these centers will be at most C_O .

In this graph, let ψ_T be the minimum cost flow value for time bound T as a function of the set of open data centers. By Theorem 2.1, ψ_T is a supermodular function. Thus, we can use the greedy algorithm for the supermodular packing problem. We do this with $g = \psi_T$ and G equal to the minimum fractional assignment cost if we open all centers.

Theorem 4.4 *A solution to the minimum assignment cost data center scheduling problem with opening and assignment cost at most $(4 + 2 \log \frac{2n}{\epsilon})\text{OPT}_{\text{ADCM}}$ that completes by time $(2 + \epsilon)T$ can be found in polynomial time.*

We start with a couple of helpful lemmas.

Lemma 4.5 *Let $r = \max\{t : \theta^t < 1\}$. The sets R^t found by the greedy algorithm for supermodular packing satisfy*

- i) $c(R^0) + g(R^0) \geq c(R^1) + g(R^1) \geq \dots \geq c(R^r) + g(R^r) \leq \dots \leq c(R^r) + g(R^r)$, and
- ii) $c(\emptyset) + g(R^0) \geq c(R^1 \setminus R^0) + g(R^1) \geq \dots \geq c(R^r \setminus R^0) + g(R^r) \leq \dots \leq c(R^r \setminus R^0) + g(R^r)$.

Proof. i) By definition, θ^t is the ratio of the additional contribution to the opening cost to the additional savings to the flow cost. Thus, while $\theta^t < 1$, the sum

of these costs is decreasing, and then when $\theta^t > 1$, the sum of these costs is increasing. For ii), since c is modular, we can subtract $c(R^0)$ from all terms and the relations still hold. \square

The next lemma is helpful to bound the cost of our final solution. It concerns the cost of the fractional assignment solutions found by the greedy algorithm, when feasibility is not an issue.

Lemma 4.6 *Consider a data center problem that starts with a set R^0 of open centers such that all jobs can be completed by time T if assigned to R^0 at assignment cost $\leq 2nC$, and such that there exists a set of R^* additional centers with $\psi_T(R^0 \cup R^*) + c(R^*) = \text{OPT}$. The greedy algorithm finds a set R satisfying $c(R \setminus R^0) \leq (1 + \ln(8n))\text{OPT}$ and $\psi_T(R) \leq 2\text{OPT}$.*

Proof. Let C_O be the cost of the open data centers in this minimum cost solution and A_O be the corresponding assignment cost. Let $\tilde{A} = \max\{A_O, \alpha C_O\}$, and suppose we run the greedy algorithm for supermodular covering with $G = \beta \tilde{A}$, for $\beta > 1$. Let ζ be the final iteration. Then, by Lemma 3.6, $\theta^\zeta < \frac{C_O}{(\beta-1)\alpha C_O} \leq \frac{1}{\alpha(\beta-1)}$. Using Theorem 3.2(b), along with a lower bound of $\frac{C_O}{n}$ on the opening cost of the first facility and the trivial upper bound on $g(\emptyset) - g(R^1)$ of $2nC$, this yields a bound on the opening cost of the greedy solution of $(1 + \ln \frac{2n}{\alpha(\beta-1)})C_O$. The bound on the assignment cost is $\beta \tilde{A} \leq \beta \max\{F_O, \alpha C_O\}$. In the worst case for the approximation guarantee on opening cost, $F_O \ll C_O$, and choosing $\alpha = \frac{1}{4}$ and $\beta = 2$ yields assignment plus opening cost of the greedy solution to be at most $(3 + \ln(8n))C_O$.

By Lemma 4.5, the cheapest greedy solution, is not the final one, but the set R^r for r defined as in Lemma 4.5, as its cost can only be less than R^ζ . Thus, the values of C_O and A_O not necessary to find a solution bounded by this cost: we can run the greedy algorithm instead with the value of G equal to the minimum cost assignment if all data centers are open, and then take the cheapest intermediate solution found by greedy. \square

Proof of Theorem 4.4 We run the greedy algorithm for supermodular packing with $V = D \setminus D^0$, $R^0 = \eta \cup D^0$, $f = \psi_T$, and $G = \psi_T(D)$. Let

$$s = \min\{t \in \{0, \dots, \tau\} : \text{at least } n - \epsilon \text{ jobs are assigned to } R^t \setminus \eta, \text{ and } \theta^{t+1} > 1\}.$$

We first bound $c(R^s \setminus R^0) + \psi_T(R^s)$. If $s = r =$

$\max\{t : \theta^t < 1\}$, then by Lemma 4.6, the combined cost is at most $(3 + \ln(8n))\text{OPT}_{\text{ADCM}}$. Else, if $r = \max\{t : \theta^t < 1\} < s \leq \zeta$, then $c(R^s \setminus R^0) + \psi_T(R^s) \leq c(R^\zeta \setminus R^0) + \psi_T(R^\zeta)$ and Lemma 4.6 again implies that this is at most $(3 + \ln(8n))\text{OPT}_{\text{ADCM}}$.

Else, s equals $\min\{t \in \{0, \dots, \tau\} : R^t \text{ is } (1 - \epsilon)\text{-feasible}\} > \zeta$. In this case, it suffices to get a bound on the opening cost, as the assignment cost is decreasing so that by Lemma 4.6, $\psi_T(R^s) \leq \psi_T(R^\zeta) \leq 2\text{OPT}_{\text{ADCM}}$.

By Lemma 3.6, $\theta^s \leq \frac{c(R^s)}{\psi_T(R^{s-1}) - G}$. Since R^{s-1} does not serve at least ϵ of the jobs, the reduction in assignment cost from moving these ϵ jobs from the dummy center to a true center is at least $\epsilon(M - nC) = \epsilon M/2$. Thus $\psi_T(R^{s-1}) - G \geq \epsilon M/2$, and $\theta^s \leq \frac{2C_O}{\epsilon M}$. We bound the opening cost of centers in $R^s \setminus R^0$ by considering the greedy algorithm run with $G = \psi_T(R^s)$. By property of the greedy algorithm, sets R^0 through R^s are the same in this run as in the run with the smaller value of G . Thus, we can use Theorem 3.2(b). To bound θ^1 , we have that $\theta^1 \geq \frac{c(R^1)}{\psi_T(R^0) - g(R^1)} \geq \frac{C_O/n}{nM}$ where the last inequality follows from our preprocessing step that includes all low-cost centers and the fact that the cost of the assignment of all jobs to the dummy center is nM . All together, this implies that

$$\begin{aligned} \frac{\psi_T(R^s)}{C_O} &\leq 1 + \ln \frac{\theta^s}{\theta^1} \\ &\leq 1 + \ln\left(\left(\frac{2C_O}{M\epsilon}\right)\left(\frac{n^2 M}{C_O}\right)\right) \\ &\leq 1 + 2 \ln \frac{2n}{\epsilon}. \end{aligned}$$

Thus $c(R^s \setminus R^0) + \psi_T(R^s) \leq (3 + 2 \ln \frac{2n}{\epsilon})\text{OPT}_{\text{ADCM}}$.

We next run the rounding algorithm of Shmoys and Tardos [19] restricted to the fractional jobs assigned to centers in $R^s \setminus \eta$. This leads to an assignment in which each job is assigned to exactly one center in $R^s \setminus \eta$, completes by time $2T$, and all but ϵ of the jobs are scheduled. This new assignment has cost at most $\psi_T(R^s)$ minus the cost of jobs assigned to η . By increasing the allowed time to $(2 + \epsilon)T$, all n jobs can be completely scheduled on machines in $R^s \setminus \eta$ by this time, as explained in Section 4.1. Moving jobs from η to $R^s \setminus \eta$ can only decrease their assignment cost. Finally, we close all centers that are not assigned a job in this assignment.

The final cost is then at most $(3 + 2 \ln \frac{2n}{\epsilon})\text{OPT}_{\text{ADCM}}$ for the assignment cost plus the opening cost of $R^s \setminus R^0$, plus at most $2C_O$ for the cost of centers in R^0 , for a total of at most $(5 + 2 \ln \frac{2n}{\epsilon})\text{OPT}_{\text{ADCM}}$ \square

While we did not try to optimize the constants with

regard to the approximation to cost above, this result improves on the approximation guarantee of the randomized rounding-based ($O((\log(n+m)), 3+\epsilon)$) bicriteria algorithm analyzed by Khuller, Li, and Saha [12], in particular in the guarantee on the approximation to the time bound.

Acknowledgment

I am grateful to Jian Li and Samir Khuller for inspiring this work by posing questions regarding the applicability of the greedy algorithm for data center problems.

References

- [1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.
- [2] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
- [3] J. Bruno, E. G. Coffman, and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17:382387, 1974.
- [4] V. Chvatal. *Linear programming*. W. H. Freeman, New York, 1983.
- [5] L. K. Fleischer and K. D. Wayne. Fast and simple approximation schemes for generalized flow. *Math. Programming*, 91:215–238, 2002.
- [6] D. Gale and T. Politof. Substitutes and complements in network flow problems. *Discrete Applied Mathematics*, 3(3):175–186, 1981.
- [7] A. Gautier and F. Granot. Ripples, complements, and substitutes in generalized networks. *Naval Research Logistics*, 43(1):1 – 21, 1996.
- [8] F. Glover and D. Klingman. On the equivalence of some generalized network flow problems to pure network problems. *Mathematical Programming*, 4:269–278, 1973.
- [9] D. Goldfarb, Z. Jin, and J. B. Orlin. Polynomial-time highest gain augmenting path algorithms for the generalized circulation problem. *Mathematics of Operations Research*, 22:793–802, 1997.
- [10] F. Granot and A. F. Veinott. Substitutes, complements and ripples in network flows. *Mathematics of Operations Research*, 10(3):471–497, 1985.
- [11] W. A. Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21:846847, 1973.
- [12] S. Khuller, J. Li, and B. Saha. Energy efficient scheduling for data centers via partial shutdown. In *Proc. Symp. Discrete Algorithms (SODA)*. ACM/SIAM, January 2010.
- [13] J.K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
- [14] N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7:97–107, 1974.
- [15] K. Onaga. Dynamic programming of optimum flows in lossy communication nets. *IEEE Trans. Circuit Theory*, 13:308–327, 1966.
- [16] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.
- [17] L. S. Shapley. On network flow functions. *Naval Research Logistics Quarterly*, 8(2):151 – 158, 1961.
- [18] L. S. Shapley. Complements and substitutes in the optimal assignment problem. *Naval Research Logistics Quarterly*, 9:45–48, 1962.
- [19] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(3):461–474, 1993.
- [20] É. Tardos and K. D. Wayne. Simple generalized maximum flow algorithms. In *6th International Integer Programming and Combinatorial Optimization Conference*, pages 310–324, 1998.
- [21] K. D. Wayne. *Generalized Maximum Flow Algorithms*. PhD thesis, Department of Operations Research and Industrial Engineering, Cornell University, 1999.
- [22] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, December 1982.