Stackelberg Scheduling Strategies*

Tim Roughgarden†

September 24, 2003

Abstract

We study the problem of optimizing the performance of a system shared by selfish, noncooperative users. We consider the concrete setting of scheduling small jobs on a set of shared machines possessing latency functions that specify the amount of time needed to complete a job, given the machine load. We measure system performance by the *total latency* of the system.

Assigning jobs according to the selfish interests of individual users, who wish to minimize only the latency that their own jobs experience, typically results in suboptimal system performance. However, in many systems of this type there is a mixture of "selfishly controlled" and "centrally controlled" jobs. The congestion due to centrally controlled jobs will influence the actions of selfish users, and we thus aspire to contain the degradation in system performance due to selfish behavior by scheduling the centrally controlled jobs in the best possible way.

We formulate this goal as an optimization problem via *Stackelberg games*, games in which one player acts a *leader* (here, the centralized authority interested in optimizing system performance) and the rest as *followers* (the selfish users). The problem is then to compute a strategy for the leader (a *Stackelberg strategy*) that induces the followers to react in a way that (approximately) minimizes the total latency in the system.

In this paper, we prove that it is NP-hard to compute an optimal Stackelberg strategy and present simple strategies with provably good performance guarantees. More precisely, we give a simple algorithm that computes a strategy inducing a job assignment with total latency no more than a constant times that of the optimal assignment of all of the jobs; in the absence of centrally controlled jobs and a Stackelberg strategy, no result of this type is possible. We also prove stronger performance guarantees in the special case where every machine latency function is linear in the machine load.

*A preliminary version of this paper appeared in the Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing, July 2001.

†This research was performed while the author was at the Department of Computer Science, Cornell University, and supported by an NSF Graduate Fellowship, a Cornell University Fellowship, and ONR grant N00014-98-1-0589. Present address: Computer Science Division, UC Berkeley, Soda Hall, Berkeley, CA 94720 USA. Email: `timr@cs.berkeley.edu`.

1 Introduction

Coping with selfishness

One of the most basic problems arising in the management of a set of resources is that of optimizing system performance. A concrete example of such a problem is as follows: given a large set of small jobs to be assigned to a set of machines, each with a latency function that specifies the amount of time needed to complete a job given the machine load, find the allocation of jobs to machines minimizing the *total latency* of the system.

In many such systems, including the Internet and other large-scale communication networks, there is no central authority controlling the allocation of shared resources; instead, system users are free to act in a selfish manner [7]. This observation has led many authors (e.g., [13, 26, 30, 37, 45, 48]) to model the behavior of users in such a system by a *noncooperative game* and to study the resulting *Nash equilibria*. (See [38] for an introduction to basic game-theoretic concepts.)

Motivated by the well-known fact that Nash equilibria may be *inefficient* [16] (i.e., they need not optimize system performance), researchers have proposed several different approaches for *coping with selfishness*—that is, for ensuring that selfish behavior results in a desirable outcome. Recent examples abound in the computer science literature: to name a few, Korilis et al. [27, 28] give methods for improving system performance by adding additional capacity to system resources, Cocchi et al. [8] and Cole et al. [9, 10] investigate the control of selfish users through various pricing policies, Shenker [48] demonstrates that an appropriate (centralized) protocol at a network switch induces selfish users to exhibit good flow control behavior, and Roughgarden [42] studies the problem of designing networks that exhibit good performance when used selfishly.

A related area of research concerned with controlling selfish behavior with efficient algorithms is that of *algorithmic mechanism design* [2, 19, 35, 36, 41], which in turn is inspired by classical mechanism design (see e.g. Mas-Colell et al. [33, Ch 23]). In this setting, an algorithm is designed to collect data from users and compute an outcome using this data. For example, the algorithm might compute the set of users that will receive some good, perhaps a movie multicast over the Internet, based on the bids of the users of the system. The difficulty of these problems stems from the assumptions that the algorithm is publicly known and that users are selfish and may report false data, if doing so improves their personal objective function. This problem is typically resolved via a *payment scheme*, where the algorithm distributes payments to users according to the outcome and the data collected so that all users have a strong incentive to report truthful data.

In this paper, we pursue a different approach. In many systems, there will be a mix of "selfishly controlled" and "centrally controlled" jobs—that is, the shared resource is used by both selfish individuals and some central authority. For example, clients of a system may be charged at two different prices: clients paying the higher price are given access to the system and the ability to schedule their own tasks (presumably in a way that accomplishes them as quickly as possible), while clients paying only the "bargain rate" can use the system but have no control over which resources their jobs consume (and thus these jobs qualify as centrally controlled). A more concrete example of such a system arises in networks that allow a large customer to set up a so-called *virtual private network* consisting of guaranteed and preassigned virtual paths for ongoing use [6, 17, 20, 22, 26]. The bandwidth needed for the virtual private network may be viewed as centrally controlled (its routes may be chosen by the network manager) while individual users of the network continue to behave in a selfish and independent fashion.

We investigate the following question: given a system with centrally and selfishly controlled jobs, *how should centrally controlled jobs be assigned to resources to induce "good" (albeit selfish) behavior from the noncooperative users?* This approach has several appealing aspects: no communication is required between the system users and an algorithm, no notion of currency is needed, no resources need to be added to or removed from the system, and the assignment of centrally controlled jobs is often easily modified as the amount of job traffic evolves over time.

Stackelberg games

We are thus led to consider a type of game in which the roles of different players are asymmetric. One player, responsible for assigning the centrally controlled jobs to resources and interested in optimizing social welfare, acts as a *leader*. The leader may hold its assignment (its *strategy*) fixed while all other agents (the *followers*) react independently and selfishly to the leader's strategy, reaching a Nash equilibrium relative to the leader's strategy. These types of games, called *Stackelberg games*, and the resulting *Stackelberg equilibria* have been well studied in the game theory literature (see, e.g., [3, §2.3] or [4, §3.6] for an introduction and [49] for their origin) and have been previously studied in the contexts of both competitive facility location [40] and networking [14, 15, 18, 26]. With the exception of [26], however, the leader/follower hierarchy has been used to model classes of selfish agents with different priority levels; this setting differs from ours in that no agent is interested in optimizing system performance.[1] The paper of Korilis et al. [26], while more similar in spirit to ours, focuses on deriving necessary and sufficient conditions (on the number of selfish users, the fraction of the job traffic that is centrally controlled, etc.) for the existence of a leader strategy inducing an optimal assignment of jobs to resources; moreover, only one type of latency function is considered. By contrast, we are interested in simple leader strategies that *always* induce optimal or near-optimal behavior from the system users for *any* set of latency functions.

Problems studied in this paper

We focus on the problem described at the beginning of the paper, of scheduling jobs on a set of machines with load-dependent latencies in order to minimize the total latency. In addition to being one of the most commonly studied models [11, 12, 18, 26, 29, 30, 34, 35, 36, 41], occasionally in the equivalent formulation of routing on a set of parallel links, the simple setting of scheduling jobs on machines permits a study of the effects of different leader strategies in Stackelberg games without any additional complications, such as the issue of path-selection in a complicated network. We also focus on a scenario in which there is a large

[1] Typically, Stackelberg games model *selfish* agents with asymmetric roles; our use of them is somewhat unconventional.

number of jobs, each of very small size. We note that this differs from nearly all scheduling research in the theoretical computer science and discrete optimization communities (surveyed in, for example, Hall [23]). This assumption is, however, consistent with a large body of existing literature on Nash equilibria in congested systems—see [5, 13, 44, 47] and the references therein—and greatly aids our analysis. For example, this assumption ensures the existence and essential uniqueness of the equilibrium reached by selfish users relative to any Stackelberg strategy. While it is clearly desirable to study more general networks as well as jobs of non-negligible size, we nonetheless feel that the simple model considered in this paper is sufficient to illustrate the main issues that arise in centrally scheduling traffic in the presence of selfish users.

We may now restate our central questions quantitatively:

(1) among all leader strategies for a given set of machines and jobs, can we characterize and/or compute the strategy inducing the *Stackelberg equilibrium*—i.e., the equilibrium of minimum total latency?

(2) what is the worst-case ratio between the total latency of the Stackelberg equilibrium and that of the optimal assignment of jobs to machines?

Our results

We give a simple polynomial-time algorithm for computing a leader strategy that induces a job assignment with total latency no more than $\frac{1}{\alpha}$ times that of the optimal assignment of jobs to machines, where α denotes the fraction of jobs that are centrally controlled. We also exhibit, for each value of α, an instance in which *no* strategy can achieve a better performance guarantee. This result stands in sharp contrast to known results about *Nash* equilibria in this model; in particular, the total latency of the Nash equilibrium may be arbitrarily larger than that of the optimal assignment of jobs to machines, even in the special case of two machines [45].

In the well-studied special case where every machine possesses a latency function linear in the congestion, we give a simple $O(m^2)$ algorithm for computing a strategy inducing a job assignment with total latency no more than $\frac{4}{3+\alpha}$ times that of the optimal assignment, where α is the fraction of centrally controlled jobs, and m is the number of machines. We again give instances in which no strategy can provide a stronger guarantee.

Finally, we consider the optimization problem of computing the strategy inducing the Stackelberg equilibrium and show that it is NP-hard, even in the special case where every latency function is linear.

We note that our results give a (sharp) trade-off between the optimal assignment and the Nash equilibrium, as a function of the fraction of centrally controlled jobs, in the following sense. Roughgarden and Tardos [45], motivated by a paper of Koutsoupias and Papadimitriou [30], showed that in general the Nash equilibrium can be arbitrarily more costly than the optimal assignment, but if every machine latency function is linear then the total latency of the Nash equilibrium is no more than $\frac{4}{3}$ times that of the optimal assignment. Thus, our results reduce to those of [45] when $\alpha = 0$, give the trivial result that the Stackelberg equilibrium for $\alpha = 1$ is the optimal assignment, and quantify the worst possible ratio between the cost of the Stackelberg equilibrium—in some sense, a "mixture" of the Nash equilibrium and the optimal assignment—and the cost of the optimal assignment for all intermediate values of α.

Our approach also adds an algorithmic dimension to the existing studies comparing Nash equilibria and optimal solutions [1, 12, 11, 24, 29, 30, 34, 43, 45, 46], in that one aspect of our analysis of Stackelberg equilibria is the design of algorithms for efficiently computing good Stackelberg strategies. Further, while Nash and optimal assignments in our model can be characterized and computed efficiently via convex programming [13, 45]—a fact that is crucial for existing comparisons of optima and equilibria [43, 45, 46]—our hardness result implies that no such characterization of Stackelberg equilibria is possible. With the central approach of earlier works ruled out, new techniques are required for our results.

Organization

In Section 2 we formalize our model and state several preliminary lemmas. In Section 3 we introduce three simple algorithms for computing Stackelberg strategies. In Sections 4 and 5, we prove that our third algorithm achieves the best-possible worst-case performance guarantee for instances with general and linear latency functions, respectively. In Section 6, we prove that computing the optimal strategy is NP-hard, even when every latency function is linear. Section 7 concludes with directions for future work.

2 Preliminaries

2.1 The Model

We consider a set M of m machines $1, 2, \ldots, m$, where machine i is endowed with a *latency function* $\ell_i(\cdot)$ that measures the load-dependent time required to complete a job. We require that each latency function be nonnegative, continuous, and nondecreasing in its argument. For our algorithms to be implemented efficiently, we also require the weak condition that $x_i \cdot \ell_i(x_i)$ is a convex function for each machine i (where x_i denotes the machine load).[2] We assume a finite and positive *rate* r of job arrivals; an *assignment* of the jobs to the machines is an m-vector $x \in \mathcal{R}^m_+$ such that $\sum_{i=1}^m x_i = r$. When we are interested in the total load on a subset $M' \subseteq M$ of the machines, we write $x(M') = \sum_{i \in M'} x_i$. We measure system performance via the *cost* or *total latency* $C(x)$ of an assignment x, defined by $C(x) = \sum_{i=1}^m x_i \ell_i(x_i)$. We note that all jobs assigned to the same machine experience the same latency; again, this differs from much of the traditional scheduling literature but agrees with common models of equilibria in congested systems where a particular allocation of resources represents a "steady-state solution" with jobs arriving continuously over time.

We will consider instances with and without centrally controlled jobs. We denote an instance with machines M, rate r, and no centrally controlled jobs by (M, r). An instance with centrally controlled jobs (a *Stackelberg instance*) will be denoted by (M, r, α), where

[2]Thus, $\ell_i(x_i)$ may be any convex function, or $\log(1+x_i)$, etc. Continuous approximations of step functions, however, do not satisfy this condition.

the third parameter $\alpha \in (0,1)$ indicates the fraction of the overall traffic that is centrally controlled.

2.2 Nash Equilibria and Optimal Assignments

If jobs are generated and assigned to machines by noncooperative agents who wish to minimize the amount of time it takes for their work to complete, we expect the assignment to be "stable" or "at equilibrium" in the following sense: no job can strictly decrease the latency it experiences by changing machines. The following definition is motivated by this notion of a stable assignment by noncooperative agents.

Definition 2.1 An assignment x to M is *at Nash equilibrium* (or is a *Nash assignment*) if whenever $i, j \in M$ with $x_i > 0$, $\ell_i(x_i) \leq \ell_j(x_j)$.

In particular, all machines in use by an assignment at Nash equilibrium have equal latency. We may thus express the cost of a Nash assignment in the following simple form.

Lemma 2.2 *If x is an assignment at Nash equilibrium for (M, r) such that all machines in use have common latency L, then*

$$C(x) = rL.$$

Example 2.3 An assignment at Nash equilibrium does not in general optimize the system performance. To see this, consider a two-machine example, in which the first machine has constant latency function $\ell_1(x_1) = 1$ and the second has latency function $\ell_2(x_2) = x_2$. If we put $r = 1$, we see that the (optimal) assignment $(\frac{1}{2}, \frac{1}{2})$ has total latency $\frac{3}{4}$, whereas the (unique) assignment at Nash equilibrium assigns all work to the second machine, thereby incurring a total cost of 1.

We end our preliminary discussion of Nash assignments by noting that they exist and are essentially unique.

Lemma 2.4 ([5, 13, 45]) *Suppose M is a set of machines with continuous, nondecreasing latency functions. Then:*

(a) For any rate $r > 0$ of job traffic, there exists an assignment of jobs to M at Nash equilibrium.

(b) If x, x' are assignments at Nash equilibrium for (M, r), then $\ell_i(x_i) = \ell_i(x'_i)$ for each machine i.

In particular, Definition 2.1 and Lemma 2.4(b) imply that any two Nash assignments for an instance (M, r) have equal cost.

For our final preliminary result, we give an analogous characterization of optimal assignments that will be useful in Section 5. For a machine i with differentiable latency function ℓ_i, define ℓ_i^* as the marginal cost of increasing the load of machine i—formally, by $\ell_i^*(x_i) = [\frac{d}{dy}(y \cdot \ell_i(y))](x_i) = \ell_i(x_i) + x_i \cdot \ell'_i(x_i)$. We will call ℓ_i^* the *marginal cost function* of machine i. Then, the following lemma holds.

Lemma 2.5 ([5, 13, 45]) *Suppose M is a set of machines with differentiable latency functions ℓ, and that $x_i \cdot \ell_i(x_i)$ is a convex function for each machine i. Then an assignment x to M is optimal if and only if whenever $i, j \in M$ with $x_i > 0$, $\ell_i^*(x_i) \leq \ell_j^*(x_j)$. Moreover, the optimal assignment can be computed in polynomial time.*

Lemma 2.5 clearly gives a characterization of *locally* optimal assignments; in particular, if the condition fails, moving a few jobs from a machine with a large marginal cost to a machine with a small marginal cost yields a new assignment with smaller cost. That it also characterizes *globally* optimal solutions follows from the fact that the optimal assignment minimizes a convex function ($C(x)$) over a convex set (the polytope of assignments) and that the local and global minima of a convex function on a convex set coincide (see for example [39, Thm 2.3.4]). This observation also implies that the optimal solution can be computed in polynomial time via convex programming.

Definition 2.1 and Lemma 2.5 yield the following useful corollary.

Corollary 2.6 ([5, 13, 45]) *Suppose M is a set of machines with differentiable latency functions ℓ, and that $x_i \cdot \ell_i(x_i)$ is a convex function for each machine i. Then an assignment x to M is optimal if and only if x is a Nash assignment with respect to latency functions ℓ^*.*

Remark: We will typically denote the optimal assignment for an instance by x^*. The marginal cost functions are denoted by ℓ^* as they are "optimal latency functions" in a sense made precise by Corollary 2.6: the optimal assignment x^* arises as an assignment at Nash equilibrium with respect to latency functions ℓ^*.

2.3 Stackelberg Strategies and Induced Equilibria

In this subsection we define our notion of a Stackelberg game and consider two examples. Recall we desire a hierarchical game, where a leader assigns centrally controlled jobs to machines and, holding this strategy fixed, the selfish users of the system react in a noncooperative and selfish manner. This idea is formalized in the next two definitions.

Definition 2.7 A *(Stackelberg) strategy* for the Stackelberg instance (M, r, α) is an assignment feasible for $(M, \alpha r)$.

Definition 2.8 Let s be a strategy for Stackelberg instance (M, r, α) where machine $i \in M$ has latency function ℓ_i, and let $\tilde{\ell}_i(x_i) = \ell_i(s_i + x_i)$ for each $i \in M$. An *equilibrium induced by strategy s* is an assignment t at Nash equilibrium for the instance $(M, (1-\alpha)r)$ with respect to latency functions $\tilde{\ell}$. We then say that $s + t$ is an *assignment induced by s* for (M, r, α).

Existence and essential uniqueness of induced equilibria follow easily from Lemmas 2.2 and 2.4.

Lemma 2.9 *Let s be a strategy for a Stackelberg instance with continuous, nondecreasing latency functions. Then there exists an assignment induced by s, and any two such induced assignments have equal cost.*

The following simple observation will be useful in Sections 4 and 5.

Lemma 2.10 *Let s be a strategy for Stackelberg instance (M, r, α) inducing equilibrium t. Let M' denote the machines on which $t_i > 0$. Then $s + t$, restricted to M', is an assignment at Nash equilibrium for the instance $(M', s(M') + t(M'))$. In particular, all machines on which $t_i > 0$ have a common latency with respect to $s + t$.*

We next consider two examples that demonstrate both the usefulness and the limitations of Stackelberg strategies.

Example 2.11 Recall that in Example 2.3, with two machines with latency functions $\ell_1(x_1) = 1$ and $\ell_2(x_2) = x_2$, in the absence of centrally controlled jobs the assignment at Nash equilibrium incurs total latency $\frac{4}{3}$ times that of the optimal assignment. Suppose instead that half of the jobs are controlled by the system manager (i.e., that $\alpha = \frac{1}{2}$) and consider the strategy $s = (\frac{1}{2}, 0)$. Then, as all remaining jobs will be assigned to the second machine in the equilibrium induced by s, the assignment induced by s is precisely the optimal assignment of all of the jobs. Thus, in this particular instance, system performance can be optimized via a Stackelberg strategy.

Example 2.12 Now modify Example 2.11 by replacing the latency function of the second machine with the latency function $\ell_2(x_2) = 2x_2$. The assignment at Nash equilibrium puts half of the jobs on each machine (for a cost of 1) while the optimal assignment is $(\frac{3}{4}, \frac{1}{4})$ (with a cost of $\frac{7}{8}$). On the other hand, if we again allow the system manager to assign half of the jobs, we see that for *any* strategy s, the assignment induced by s is $(\frac{1}{2}, \frac{1}{2})$ and hence is not optimal. In this example, there is no available strategy by which the system manager can improve system performance.

3 Three Stackelberg Strategies

3.1 Two Natural Strategies

We begin our investigation of Stackelberg strategies by considering two natural approaches that provide suboptimal performance guarantees. To motivate our results in the simplest possible way, throughout this subsection we will consider examples in which all latency functions are linear and half of the jobs are centrally controlled ($\alpha = \frac{1}{2}$). We are thus hoping for strategies that always induce an assignment of cost at most $\frac{8}{7}$ times that of the optimal assignment (this is the best possible by the Example 2.12).

First consider the following strategy for an instance $(M, r, \frac{1}{2})$: if x^* is the optimal assignment for instance $(M, \frac{1}{2}r)$, put $s = x^*$. In words, we choose the strategy of minimum cost, ignoring the existence of jobs that are not centrally controlled. We call this the *Aloof* strategy since it refuses to acknowledge the rest of the jobs in the system. Example 2.3 shows that this strategy performs quite poorly: the strategy is $(0, \frac{1}{2})$ and the induced assignment is $(0, 1)$, an assignment that we have seen to incur total latency $\frac{4}{3}$ times that of the optimal assignment.

A second attempt for a good strategy might be as follows: if x^* is the optimal assignment for (M, r), put $s = \frac{1}{2}x^*$. We call this the *Scale* strategy, since it is simply the optimal assignment of all the jobs, suitably scaled. Unfortunately, a simple example shows that the

Scale strategy also fails to provide the performance guarantee of $\frac{8}{7}$ that we are looking for: in a two-machine example with latency functions $\ell_1(x_1) = 1$ and $\ell_2(x_2) = \frac{3}{2}x_2$ and rate 1, the optimal assignment is $(\frac{2}{3}, \frac{1}{3})$ (with total cost $\frac{5}{6}$) and thus the Scale strategy will be $(\frac{1}{3}, \frac{1}{6})$, which induces the assignment $(\frac{1}{3}, \frac{2}{3})$ having cost 1. Hence, the Scale strategy may result in an induced assignment with total latency $\frac{6}{5}$ times the cost of the optimal assignment.[3]

3.2 The Largest Latency First (LLF) Strategy

Intuitively, both the Aloof and Scale strategies suffer from a common flaw: both allocate jobs to machines that will subsequently be inundated in any induced equilibrium while assigning too little work to machines that selfish users are prone to ignore. This observation suggests that a good strategy should give priority to the machines that are least appealing to selfish users—machines with relatively high latency. With this intuition in mind, the following strategy for a Stackelberg instance (M, r, α), which we call the *Largest Latency First* or *LLF* strategy, should seem natural:

(1) Compute the optimal assignment x^* for (M, r)

(2) Index the machines of M so that $\ell_1(x_1^*) \leq \cdots \leq \ell_m(x_m^*)$

(3) Let $k \leq m$ be minimal with $\sum_{i=k+1}^{m} x_i^* \leq \alpha r$

(4) Put $s_i = x_i^*$ for $i > k$, $s_k = \alpha r - \sum_{i=k+1}^{m} x_i^*$, and $s_i = 0$ for $i < k$.

We will say that a machine i is *saturated* by a strategy s if $s_i = x_i^*$. The LLF strategy thus saturates machines one-by-one, in order from the largest latency with respect to x^* to the smallest, until there are no centrally controlled jobs remaining. Note that Lemma 2.5 implies that the LLF strategy can be computed in polynomial time (the bottleneck is step (1)); in Section 5 we will see that it can be computed in $O(m^2)$ time when every latency function is linear.

The next two sections are devoted to proving that the LLF strategy always induces an assignment with near-optimal total latency.

4 A $\frac{1}{\alpha}$ Performance Guarantee for Arbitrary Latency Functions

In this section we prove that the LLF strategy induces a near-optimal assignment for any set of latency functions and any number of machines. We note that *no* performance guarantee is possible in the absence of centrally controlled jobs: without additional restrictions on machine latency functions, the Nash assignment may incur arbitrarily more latency than the optimal assignment [45]. Thus, the benefit of a leader (and of a carefully chosen leader strategy) is particularly striking in this general setting.

[3] In addition, the Aloof and Scale strategies can perform arbitrarily badly for instances with general latency functions.

A simple variation on previous examples demonstrates the limits of Stackelberg strategies. In a two-machine instance with $\alpha = \frac{1}{2}$ and latency functions $\ell_1(x_1) = 1$ and $\ell_2(x_2) = 2^k x_2^k$ for $k \in \mathcal{Z}^+$, any Stackelberg strategy induces the assignment $(\frac{1}{2}, \frac{1}{2})$ (having total latency 1) while the optimal assignment is $(\frac{1}{2} + \delta_k, \frac{1}{2} - \delta_k)$ having cost $\frac{1}{2} + \epsilon_k$, where $\delta_k, \epsilon_k \to 0$ as $k \to \infty$. Thus the best induced assignment may be (arbitrarily close to) twice as costly as the optimal assignment. Similar examples show that for any $\alpha \in (0,1)$, the best induced assignment may be $\frac{1}{\alpha}$ times as costly as the optimal assignment.

The main result of this section is that the LLF strategy always induces an assignment of cost no more than $\frac{1}{\alpha}$ times that of the optimal assignment. A rough outline of the proof is as follows. Our goal is to exploit the iterative structure of the LLF strategy and proceed by induction on the number of machines. If the LLF strategy first saturates the mth machine, a natural idea is to apply the inductive hypothesis to the remainder of the LLF strategy on the first $m-1$ machines to derive a performance guarantee. This idea nearly succeeds, but there are two difficulties. First, it is possible that the LLF strategy fails to saturate any machines; we will see below that this case is easy to analyze and causes no trouble. Second, in order to obtain a clean application of the inductive hypothesis to the first $m-1$ machines, we require that the optimal and LLF-induced assignments place the same total amount of jobs on these machines — i.e., that the LLF-induced equilibrium eschews the mth machine.[4] We resolve this difficulty with the following lemma, which states that if the LLF strategy saturates the mth machine, then *some* induced equilibrium assigns all jobs to the first $m-1$ machines; this suffices for our purposes, since different induced assignments have equal cost.

Lemma 4.1 *Let (M, r, α) denote a Stackelberg instance with optimal assignment x^* and index the machines of M so that $\ell_m(x_m^*) \geq \ell_i(x_i^*)$ for all i. If s is a strategy with $s_m = x_m^*$, then there exists an induced equilibrium t with $t_m = 0$.*

Proof. Consider an arbitrary induced equilibrium t and suppose $t_m > 0$. Roughly speaking, the idea is to prove that this scenario only occurs when several latency functions (that of the mth machine, and others) are locally constant; then, jobs assigned to machine m in the induced equilibrium can be evacuated to other machines with locally constant latency functions to provide a new induced equilibrium.

Formally, let $L = \ell_m(s_m + t_m) = \ell_m(x_m^* + t_m)$ denote the common latency with respect to $s+t$ of every machine with $t_i > 0$ (see Lemma 2.10). We must have $\ell_m(x_m^*) \geq L$; otherwise $\ell_i(x_i^*) < L$ for all i yet $\ell_i(s_i + t_i) \geq L$ for all i, contradicting that x^* and $s+t$ are assignments at the same rate. Thus, since ℓ_m is nondecreasing, ℓ_m is locally constant: ℓ_m is equal to L in the interval $[x_m^*, x_m^* + t_m]$.

Next, let M' denote the machines on which $s_i + t_i < x_i^*$; since $s_m + t_m > x_m^*$, M' is non-empty. For each $i \in M'$, we know that $\ell_i(s_i + t_i) \geq L$, $\ell_i(x_i^*) \leq \ell_m(x_m^*) = L$, and ℓ_i is nondecreasing, so ℓ_i is equal to L on $[s_i + t_i, x_i^*]$. Since x^* and $s+t$ are assignments at the same rate, we must have $\sum_{i \in M'} [x_i^* - (s_i + t_i)] \geq t_m$. Finally, consider modifying t as follows: move all jobs previously assigned to machine m to machines in M', subject to the

[4] To see a trivial example in which this does not occur, put $r = 1, \alpha = \frac{1}{2}$ and consider two machines each with the constant latency function $\ell_i(x_i) = 1$. One particular optimal assignment is $(\frac{2}{3}, \frac{1}{3})$, and a corresponding LLF strategy is $(\frac{1}{6}, \frac{1}{3})$; one particular induced assignment is $(\frac{1}{3}, \frac{2}{3})$. Even though the LLF strategy saturated the second machine, the induced equilibrium uses it.

constraints $s_i + t_i \leq x_i^*$. We have already observed that there is sufficient "room" on machines in M' for this operation, and that all latency functions are constant in the domain of our modifications. We have thus exhibited a new induced equilibrium with no jobs assigned to machine m, completing the proof. ■

We are now prepared to prove the main result of this section.

Theorem 4.2 *Let $\mathcal{I} = (M, r, \alpha)$ denote a Stackelberg instance. If s is an LLF strategy for $\mathcal{I}$ inducing equilibrium t and x^* is an optimal assignment for the instance (M, r), then $C(s+t) \leq \frac{1}{\alpha} C(x^*)$.*

Proof. We proceed by induction on the number of machines m (for each fixed m, we will prove the theorem for arbitrary ℓ, r, and α). The case of one machine is trivial.

Fix a Stackelberg instance $\mathcal{I} = (M, r, \alpha)$ with at least two machines, and let x^* denote an optimal assignment to the instance (M, r) and s the corresponding LLF strategy. Index the machines so that $\ell_1(x_1^*) \leq \ell_2(x_2^*) \leq \cdots \leq \ell_m(x_m^*)$. By scaling, we may assume that $r = 1$ (use latency functions $\tilde{\ell}$ with $\tilde{\ell}_i(x_i) = \ell_i(rx_i)$). Let L denote the common latency with respect to $s+t$ of every machine with $t_i > 0$ (see Lemma 2.10).

Case 1: Suppose $t_k = 0$ for some machine k. Let M_1 denote the machines i for which $t_i = 0$ and M_2 the machines for which $t_i > 0$; both of these sets are non-empty. For $i = 1, 2$ let α_i denote the amount of centrally controlled jobs assigned to machines in M_i (so $\alpha_i = s(M_i)$) and C_i the cost incurred by $s+t$ on machines in M_i. By Lemma 2.10, $C_2 = (1-\alpha_1)L$ and $C_1 \geq \alpha_1 L$. Further, since x^* restricted to M_2 is an optimal assignment for $(M_2, 1-\alpha_1)$, s restricted to M_2 is an LLF strategy for the instance $\mathcal{I}_2 = (M_2, 1-\alpha_1, \alpha')$, where $\alpha' = \frac{\alpha_2}{1-\alpha_1}$.

Applying the inductive hypothesis to $\mathcal{I}_2$ and using the fact that $x_i^* \geq s_i = s_i + t_i$ for all $i \in M_1$, we obtain

$$C(x^*) \geq C_1 + \alpha' C_2.$$

Proving that $C(s+t) \leq \frac{1}{\alpha} C(x^*)$ thus reduces to showing

$$\alpha(C_1 + C_2) \leq C_1 + \alpha' C_2.$$

Since $\alpha \leq 1$ and $C_1 \geq \alpha_1 L$, it suffices to prove this inequality with C_1 replaced by $\alpha_1 L$. Writing $C_2 = (1-\alpha_1)L$ and $\alpha' = \frac{\alpha_2}{1-\alpha_1}$ and dividing through by L, we need only check that

$$\alpha(\alpha_1 + (1-\alpha_1)) \leq \alpha_1 + \frac{\alpha_2}{1-\alpha_1}(1-\alpha_1)$$

which clearly holds (both sides are equal to α).

Case 2: Suppose $t_i > 0$ for every machine i, so $C(s+t) = L$. We may assume that the LLF strategy failed to saturate machine m; otherwise, by Lemma 2.9, we can finish by applying the previous case to the better-behaved induced assignment guaranteed by Lemma 4.1. Thus, $\alpha < x_m^*$.

As in the proof of Lemma 4.1, we must have $\ell_m(x_m^*) \geq L$; otherwise, $\ell_i(x_i^*) < L$ for all machines i while $\ell_i(s_i + t_i) = L$ for all i, contradicting that x^* and $s+t$ are assignments at the same rate. Having established that machine m has large latency with respect to x^* and that x^*_m is fairly large, it is now a simple matter to lower bound $C(x^*)$:

$$C(x^*) \geq x^*_m \ell_m(x^*_m) \geq \alpha L = \alpha C(s+t).$$

■

5 A $\frac{4}{3+\alpha}$ Performance Guarantee for Linear Latency Functions

5.1 Properties of the Nash and Optimal Assignments

In this subsection we undertake a deeper study of the Nash and optimal assignments for instances with linear latency functions. The results of this subsection will be instrumental in proving a stronger performance guarantee for the LLF strategy for these instances.

Fix a set of machines M with latency functions $\ell_i(x_i) = a_i x_i + b_i$ for each i $(a_i, b_i \geq 0)$ and index them so that $b_1 \leq b_2 \leq \cdots \leq b_m$. We may assume that at most one machine has a constant latency function $(a_i = 0)$ since all but the fastest may be safely discarded; under this assumption, the Nash and optimal assignments are always unique. We may similarly assume that a machine with a constant latency function is the last machine.

Our first goal is to understand the structure of the Nash assignment $\bar{x}$ as a function of the rate r. It is useful to imagine r increasing from 0 to a large value, with the corresponding Nash assignment changing in a continuous fashion; an intuitive description of this process is as follows. Initially, when r is nearly zero, all jobs will be assigned to the machine having the smallest constant term. Once the first machine is sufficiently loaded, the second machine looks equally attractive. This occurs when $a_1\bar{x}_1 + b_1 = b_2$—when the load on machine 1 is $\frac{b_2 - b_1}{a_1}$. At this point, new jobs will be assigned to both of the first two machines, at rates proportional to $\frac{1}{a_1}$ and $\frac{1}{a_2}$ so that these two machines continue to have equal latency. Once $(b_3 - b_2)(\frac{1}{a_1} + \frac{1}{a_2})$ further units of work have arrived and been assigned to the first two machines, machine 3 will be equally attractive and new jobs will be spread out among the first three machines, and so on. We may thus envision the Nash assignment as being constructed in phases: within phase i jobs are assigned to the first i machines according to fixed relative proportions and at the end of the phase, after enough new jobs have been assigned, an additional machine is put into use.

We now formalize this intuitive description of the Nash assignment $\bar{x}$. For $i = 1, \ldots, m$, let v_i denote the m-vector $(\frac{1}{a_1}, \frac{1}{a_2}, \ldots, \frac{1}{a_i}, 0, 0, \ldots, 0) \in \mathcal{R}^m_+$; if $a_m = 0$ put $v_m = (0, 0, \ldots, 1)$. The vector v_i should be interpreted as a specification of the way jobs are assigned to the first i machines during the ith phase. Next, define δ_i for $i = 0, 1, \ldots, m-1$ inductively by $\delta_0 = 0$ and $\delta_i = \min\{(b_{i+1} - b_i)\|v_i\|_1, r - \sum_{j=0}^{i-1} \delta_i\} \geq 0$, where $\|\cdot\|_1$ denotes the L_1-norm. We also put $\delta_m = r - \sum_{j=0}^{m-1} \delta_i$. The scalar δ_i is the amount of jobs assigned in the ith phase. We can then describe $\bar{x}$ as follows.

Lemma 5.1 *Let $\mathcal{I}$ be an instance with linear latency functions, as above. Then the Nash assignment for $\mathcal{I}$ is given by*

$$\bar{x} = \sum_{i=1}^{m} \delta_i \frac{v_i}{\|v_i\|_1}.$$

Our characterization of optimal assignments (Lemma 2.5 and Corollary 2.6) yields an analogous result for computing them by an explicit formula. Note that when a latency function has the form $\ell_i(x_i) = a_i x_i + b_i$, the corresponding marginal cost function is $\ell_i^*(x_i) = 2a_i x_i + b_i$. Recalling from Corollary 2.6 that an optimal assignment is simply a Nash assignment with respect to latency functions ℓ^*, we see that the optimal assignment is created by the same process as the Nash assignment, except that new machines are incorporated at a more rapid pace so as to spread jobs over a wider range of machines and thus achieve a smaller total latency.

Formally, let v_i be as above and define δ_i^* inductively by $\delta_0^* = 0$, $\delta_i^* = \min\{\frac{1}{2}(b_{i+1} - b_i)\|v_i\|_1, r - \sum_{j=0}^{i-1} \delta_i^*\}$, and $\delta_m^* = r - \sum_{j=0}^{m-1} \delta_i^*$. Letting x^* denote the optimal assignment to (M, r), the analog of Lemma 5.1 is as follows.

Lemma 5.2 *Let $\mathcal{I}$ be an instance with linear latency functions, as above. Then the optimal assignment for $\mathcal{I}$ is given by*

$$x^* = \sum_{i=1}^{m} \delta_i^* \frac{v_i}{\|v_i\|_1}.$$

Lemmas 5.1 and 5.2 have several useful corollaries. We summarize them below.

Corollary 5.3 *Let M be a set of machines with latency functions $\{\ell_i(x_i) = a_i x_i + b_i\}_{i \in M}$ and at most one machine with a constant latency function. Let b_i be nondecreasing in i. Then:*

(a) If x^* and $\bar{x}$ denote the optimal and Nash assignments to (M, r), then $x_m^* \geq \bar{x}_m$.

(b) If x^* and $\bar{x}$ denote the optimal and Nash assignments to (M, r), then $x_1^* \leq \bar{x}_1 \leq 2x_1^*$.

(c) If x^* is the optimal assignment for (M, r_1) and y^* is the optimal assignment for (M, r_2) with $r_1 \geq r_2$, then $x_i^* \geq y_i^*$ for each i.

(d) For any rate r, the optimal and Nash assignments of (M, r) can be computed in $O(m^2)$ time.

Proof. Parts (c) and (d) are immediate from Lemmas 5.1 and 5.2. For the remaining parts, fix an instance (M, r) and define v_i, δ_i, and δ_i^* as in Lemmas 5.1 and 5.2. Our first observation is that, for any $i \in \{1, 2, \ldots, m\}$, the Nash assignment schedules at least as many jobs in the first i phases as the optimal assignment—formally, that $\sum_{k=1}^{i} \delta_k \geq \sum_{k=1}^{i} \delta_k^*$ for all i. Since $\sum_{k=1}^{m} \delta_k = \sum_{k=1}^{m} \delta_k^* = r$, we obtain $\delta_m^* \geq \delta_m$ and hence $x_m^* \geq x_m$, proving (a).

It remains to prove part (b) of the corollary. We may assume that $m > 1$ (otherwise $\bar{x}_1 = x_1^* = r$). Letting m' equal m if there is no machine with constant latency function and

$m-1$ otherwise, Lemmas 5.1 and 5.2 give

$$x_1^* = \frac{1}{a_1}\sum_{i=1}^{m'} \frac{\delta_i^*}{\|v_i\|_1}$$

and

$$\bar{x}_1 = \frac{1}{a_1}\sum_{i=1}^{m'} \frac{\delta_i}{\|v_i\|_1}.$$

By the definitions of δ_i and δ_i^*, we have $\delta_i \le 2\delta_i^*$ for $i = 1, 2, \ldots, m$ and hence $\bar{x}_1 \le 2x_1^*$. For the other inequality, we recall that $\sum_{k=1}^{i} \delta_k^* \le \sum_{k=1}^{i} \delta_k$ for each i and observe that $\|v_i\|_1$ is increasing in i (for $i \in \{1, 2, \ldots, m'\}$); it follows that $x_1^* \le \bar{x}_1$. ■

Corollary 5.3(d) implies that the LLF strategy can be computed in $O(m^2)$ time for instances with linear latency functions.

5.2 Proof of Performance Guarantee

In Subsection 2.3 we saw an example with linear latency functions and $\alpha = \frac{1}{2}$ in which *no* strategy can induce an assignment with cost less than $\frac{8}{7}$ times that of the optimal assignment. This example is easily modified to show that, for any $\alpha \in (0, 1)$, the minimum-cost induced assignment for a Stackelberg instance (M, r, α) with linear latency functions may be $\frac{4}{3+\alpha}$ times as costly as the optimal assignment for (M, r). The main result of this section is a matching upper bound for the LLF strategy.

Before proving this result, we give an alternative description of LLF that is more convenient for our analysis. This description is based on the following lemma.

Lemma 5.4 *Let x^* be an optimal assignment for (M, r) where machine i has latency function $\ell_i(x_i) = a_i x_i + b_i$. Then $\ell_i(x_i^*) \ge \ell_j(x_j^*)$ if and only if $b_i \ge b_j$.*

Proof. The lemma is clear when $x_i^* = x_j^* = 0$. Otherwise, we will make use of our characterization of optimal assignments via machine marginal cost functions (Lemma 2.5). If exactly one of x_i^*, x_j^* is 0 (say x_i^*), then by Lemma 2.5 we know that the marginal cost $\ell_i^*(x_i^*) = \ell_i^*(0) = b_i$ of machine i is at least the marginal cost $\ell_j^*(x_j^*) = 2a_j x_j^* + b_j$ of machine j. Thus we necessarily have both $\ell_i(x_i^*) \ge \ell_j(x_j^*)$ and $b_i \ge b_j$. Finally, if $x_i^*, x_j^* > 0$ then by Lemma 2.5 we have $2a_i x_i^* + b_i = 2a_j x_j^* + b_j = L^*$ for some L^*; thus $b_i \ge b_j$ if and only if $a_i x_i^* \le a_j x_j^*$. The lemma follows by writing $\ell_i(x_i^*) = L^* - a_i x_i^*$ and $\ell_j(x_j^*) = L^* - a_j x_j^*$. ■

Lemma 5.4 gives the following equivalent description of the LLF strategy: saturate machines one-by-one, in decreasing order of constant terms, until no centrally controlled jobs remain. It may seem surprising that the LLF strategy makes no use of the a_i-values in ordering the machines; however, this is consistent with our observation in Subsection 5.1 that the order in which machines are used by the optimal assignment, if we think of the rate as increasing from 0 to some large value, depends only on the constant terms of the machines' latency functions.

We are finally prepared to prove a $\frac{4}{3+\alpha}$ performance guarantee for the LLF strategy for instances with linear latency functions. The general approach is similar to that of Theorem 4.2 and is again by induction on the number of machines. However, new difficulties arise in proving a stronger performance guarantee. The case in which there is some machine k on which the induced equilibrium assigns no jobs (i.e., $t_k = 0$ for some k) is nearly identical to the first case of Theorem 4.2: the desired performance guarantee can easily be extracted from the inductive guarantee for the smaller instance of machines on which $t_i > 0$. The second case, in which the induced equilibrium assigns jobs to all machines, is substantially more complicated. In particular, the simple approach in the proof of Theorem 4.2 does *not* use any inductive guarantee in this case and is thus not strong enough to prove a guarantee better than $\frac{1}{\alpha}$. For this reason, much of the proof is devoted to defining an appropriate smaller instance that allows for clean application of the inductive hypothesis and to extending the inductive guarantee into one for the original instance.

Theorem 5.5 *Let $\mathcal{I} = (M, r, \alpha)$ denote a Stackelberg instance with linear latency functions. If s is an LLF strategy for $\mathcal{I}$ inducing equilibrium t and x^* is an optimal assignment for (M, r), then $C(s+t) \leq \frac{4}{3+\alpha} C(x^*)$.*

Proof. We proceed by induction on the number of machines m (for each fixed m, we will prove the theorem for arbitrary (linear) ℓ, r, and α). The case of one machine is trivial.

Fix a Stackelberg instance $\mathcal{I} = (M, r, \alpha)$ with at least two machines and let $\ell_i(x_i) = a_i x_i + b_i$ (with $a_i, b_i \geq 0$). Let x^* denote an optimal assignment to (M, r). We begin with several simplifying assumptions, each made with no loss of generality. As in Theorem 4.2, we may assume that $r = 1$. We assume as usual that there is at most one machine with a constant latency function. It will also be convenient to assume that some machine i has constant term 0. To enforce this assumption we may subtract $\min_i b_i$ from every latency function before applying our argument: assuming $r = 1$, this modification decreases the cost of every assignment by precisely $\min_i b_i$ and will only increase the ratio in costs between any two assignments. Finally, we assume that no machine has latency function $\ell_i(x_i) = 0$ (otherwise the instance is trivial).

Let s denote an LLF strategy for $\mathcal{I}$ and t the induced equilibrium. Let $L > 0$ denote the common latency of every machine used by t (see Lemma 2.10). Index the machines of M as in the second description of the LLF strategy (see Lemma 5.4), so that $0 = b_1 \leq b_2 \leq \cdots \leq b_m$ and $a_1 > 0$. We will need to apply the inductive hypothesis in two different ways, and our analysis breaks into two cases.

Case 1: Suppose $t_k = 0$ for some k. Our argument will be essentially identical to the first case in the proof of Theorem 4.2. Let M_1 denote the machines on which $t_i = 0$ and M_2 the machines on which $t_i > 0$. For $i = 1, 2$, let α_i denote the amount of centrally controlled jobs on machines in M_i. For $i = 1, 2$ let C_i denote the cost incurred by $s + t$ on machines in M_i. Observe that $C_1 \geq \alpha_1 L$ and $C_2 = (1 - \alpha_1)L$. Since x^* restricted to M_2 is an optimal assignment for $(M_2, 1 - \alpha_1)$, s restricted to M_2 is an LLF strategy for $\mathcal{I}_2 = (M_2, 1 - \alpha_1, \alpha')$, where $\alpha' = \frac{\alpha_2}{1-\alpha_1}$. The inductive hypothesis, applied to $\mathcal{I}_2$, and the fact that $x_i^* \geq s_i = s_i + t_i$ for all $i \in M_1$ imply that

$$C(x^*) \geq C_1 + \frac{3 + \alpha'}{4} C_2.$$

Proving that $C(s+t) \le \frac{4}{3+\alpha}C(x^*)$ thus reduces to showing

$$(3+\alpha)(C_1+C_2) \le 4C_1 + (3+\alpha')C_2.$$

Since $\alpha \le 1$ and $C_1 \ge \alpha_1 L$, it suffices to prove this inequality with C_1 replaced by $\alpha_1 L$. Writing $C_2 = (1-\alpha_1)L$, $\alpha' = \frac{\alpha_2}{1-\alpha_1}$, and dividing through by L verifies the result.

Case 2: Suppose $t_i > 0$ for all machines $i \in M$. This implies that $s+t$ is a Nash assignment for $(M,1)$; by Corollary 5.3(a) we have $s_m < s_m + t_m \le x^*_m$ (in particular, we must have $x^*_m > 0$). It follows that the LLF strategy s failed to saturate machine m, so $s_m = \alpha$ and $s_i = 0$ for $i < m$.

Our first goal is to show that s is an LLF strategy not only for $\mathcal{I}$ but also for $\mathcal{I}' = (M', 1-t_1, \frac{\alpha}{1-t_1})$, where $M' = M \setminus \{1\}$ (we may then apply the inductive hypothesis to s in this smaller instance). Toward this end, let y^* denote the optimal assignment to the instance $(M', 1-t_1)$. Since $s+t$ restricted to M' is a Nash assignment for $(M', 1-t_1)$, we must have $y^*_m \ge s_m + t_m$ (see Corollary 5.3(a)); since $\alpha = s_m < s_m + t_m \le y^*_m$, the LLF strategy for $\mathcal{I}'$ is precisely s (restricted to M').

Let C^*_1, C^*_2 denote the total latency incurred by x^* on machine 1 and in M', respectively. The next claim gives a lower bound on C^*_2, as a function of the amount of jobs assigned to machines in M' in the optimal assignment.

Claim: *If $r' \ge 1-t_1$, then the cost of the optimal assignment for (M', r') is at least*

$$\frac{3+\alpha'}{4}(1-t_1)L + (r'-1+t_1)L$$

where $\alpha' = \frac{\alpha}{1-t_1}$.

Proof. The claim is proved for $r' = 1-t_1$ by applying the inductive hypothesis to the instance $\mathcal{I}' = (M', 1-t_1, \alpha')$ and using the fact that s is an LLF strategy for M' inducing an assignment of cost $(1-t_1)L$. Suppose now that $r' > 1-t_1$. We again denote the optimal assignment for $(M', 1-t_1)$ by y^*. Since y^* and $s+t$ (restricted to M') are assignments at the same rate (namely, $1-t_1$) and the common latency of every machine with respect to $s+t$ is L, there is some machine i with $y^*_i > 0$ and $\ell_i(y^*_i) \ge L$. Since the marginal cost of a machine is at least its latency, Lemma 2.5 implies that the marginal cost of every machine in M' is at least L with respect to y^*. By Corollary 5.3(c) and the observation that marginal costs are nondecreasing functions of the machine load, extending y^* from an optimal assignment for $(M', 1-t_1)$ to an optimal assignment for (M', r') involves the assignment of $r' - (1-t_1)$ units of jobs, all assigned at a marginal cost of at least L. Thus, the overall cost of an optimal assignment to (M', r') must be at least $C(y^*) + (r'-1+t_1)L \ge \frac{3+\alpha'}{4}(1-t_1)L + (r'-1+t_1)L$. ■

Our goal is to prove that $(3+\alpha)C(s+t) \le 4C(x^*) = 4(C^*_1 + C^*_2)$; with the claim in hand, we have reduced the proof of the theorem to proving the inequality

$$(3+\alpha)L \le (3+\alpha')(1-t_1)L + 4(t_1 - x^*_1)L + 4a_1(x^*_1)^2$$

where $\alpha' = \frac{\alpha}{1-t_1}$ (recall that $\ell_1(x_1) = a_1 x_1$ and hence $C^*_1 = a_1(x^*_1)^2$). For any fixed value of t_1, $x^*_1 \in [\frac{1}{2}t_1, t_1]$ (see Corollary 5.3(b)). Using the identity $a_1 t_1 = L$ and differentiating, we find that the right-hand side is minimized by $x_1^* = \frac{1}{2}t_1$. Since the left-hand side is independent of x_1^*, it suffices to prove that

$$(3+\alpha)L \leq (3+\alpha')(1-t_1)L + 2t_1 L + a_1 t_1^2.$$

Substituting for α', using the identity $a_1 t_1 = L$, and dividing by L gives

$$3+\alpha \leq (3+\frac{\alpha}{1-t_1})(1-t_1) + 3t_1$$

which clearly holds, proving the theorem. ■

6 The Complexity of Computing Optimal Strategies

Thus far, we have measured the performance of a Stackelberg strategy by comparing the cost of the corresponding induced assignment to the cost of the optimal assignment of all of the jobs. Another natural approach for evaluating a strategy is to compare the cost of the induced assignment to that of the *least costly assignment induced by some Stackelberg strategy*, i.e., to the cost of the assignment induced by the *optimal* strategy. Motivated by the latter measure, in this section we study the optimization problem of computing the optimal Stackelberg strategy.

We have seen that the LLF strategy provides the best possible (worst-case) performance guarantee relative to the cost of the optimal assignment, and in particular that the algorithm of Subsection 3.2 may be viewed as a $\frac{1}{\alpha}$-approximation algorithm[5] for computing the optimal strategy, and a $\frac{4}{3+\alpha}$-approximation algorithm for instances with linear latency functions. However, simple examples show that the LLF strategy is *not* always the optimal strategy, and thus our algorithm fails to solve this optimization problem exactly.[6] Our main result in this section is strong evidence that no such polynomial-time algorithm exists.

Theorem 6.1 *The problem of computing the optimal Stackelberg strategy is NP-hard, even for instances with linear latency functions.*

Proof. We reduce from a problem we call $\frac{1}{3}$-$\frac{2}{3}$ PARTITION: given n positive integers $a_1, a_2, \ldots, a_n$, is there a subset $S \subseteq \{1, 2, \ldots, n\}$ satisfying $\sum_{i \in S} a_i = \frac{1}{3}\sum_{i=1}^{n} a_i$? The canonical reduction from the NP-complete problem SUBSET SUM to PARTITION is easily modified to show that $\frac{1}{3}$-$\frac{2}{3}$ PARTITION is NP-hard; see Garey and Johnson [21] for problem definitions and Karp [25] or Kozen [31, P.129] for the canonical reduction. We will show that deciding the problem $\frac{1}{3}$-$\frac{2}{3}$ PARTITION reduces to deciding whether or not a given Stackelberg scheduling instance with linear latency functions admits a Stackelberg strategy inducing an assignment with a given cost.

[5] A *c-approximation algorithm* for a minimization problem runs in polynomial time and returns a solution no more than c times as costly as an optimal solution. The value c is the *approximation ratio* or *performance guarantee* of the algorithm.

[6] For example, consider a three machine instance with latency functions $\ell_1(x_1) = x_1, \ell_2(x_2) = 1 + x_2, \ell_3(x_3) = 1 + x_3$, with $r = 1$ and $\alpha = \frac{1}{6}$. The optimal assignment is $(\frac{2}{3}, \frac{1}{6}, \frac{1}{6})$ and thus the LLF strategy is $(0, 0, \frac{1}{6})$ inducing the assignment $(\frac{5}{6}, 0, \frac{1}{6})$ with cost $\frac{8}{9}$. On the other hand, the strategy $(0, \frac{1}{12}, \frac{1}{12})$ induces the assignment $(\frac{5}{6}, \frac{1}{12}, \frac{1}{12})$ having cost $\frac{7}{8}$.

Given an arbitrary instance $\mathcal{I}$ of $\frac{1}{3}$-$\frac{2}{3}$ PARTITION specified by positive integers $a_1, \ldots, a_n$, put $A = \sum_{i=1}^{n} a_i$ and define a Stackelberg scheduling instance $\mathcal{I}' = (\{1, 2, \ldots, n+1\}, 2A, \frac{1}{4})$ with $n+1$ machines and with linear latency functions $\ell_i(x_i) = \frac{x_i}{a_i} + 4$ for $i = 1, \ldots, n$ and $\ell_{n+1}(x_{n+1}) = \frac{3x_{n+1}}{A}$. It is clear that $\mathcal{I}'$ can be constructed from $\mathcal{I}$ in polynomial time. We claim that $\mathcal{I}$ is a "yes instance" (that is, admits a $\frac{1}{3}$-$\frac{2}{3}$ partition) if and only if there is a Stackelberg strategy for instance $\mathcal{I}'$ inducing an assignment with cost at most $\frac{35}{4}A$.

First suppose $\mathcal{I}$ is a "yes instance" of $\frac{1}{3}$-$\frac{2}{3}$-PARTITION, with $S \subseteq \{1, 2, \ldots, n\}$ satisfying $\sum_{i \in S} a_i = \frac{2}{3}A$, and consider the strategy defined by $s_i = \frac{3}{4}a_i$ for $i \in S$ and $s_i = 0$ otherwise (since $\sum_{i=1}^{n+1} s_i = \frac{3}{4}\sum_{i \in S} a_i = \frac{3}{4}\frac{2}{3}A = \frac{1}{2}A$, this defines a Stackelberg strategy). The induced equilibrium is then $t_i = 0$ for $i \in S$, $t_i = \frac{1}{4}a_i$ for $i \in \{1, 2, \ldots, n\} \setminus S$, and $t_{n+1} = \frac{17}{12}A$. In the induced assignment $s+t$, the $A/2$ units of jobs on machines corresponding to S experience $\frac{19}{4}$ units of latency, while the other $3A/2$ units of jobs experience $\frac{17}{4}$ units of latency. The cost of $s+t$ is thus

$$\frac{A}{2}\frac{19}{4} + \frac{3A}{2}\frac{17}{4} = \frac{35}{4}A.$$

Now suppose that $\mathcal{I}$ is a "no instance" of $\frac{1}{3}$-$\frac{2}{3}$-PARTITION, and consider any Stackelberg strategy s for $\mathcal{I}'$, inducing equilibrium t. We need to show that $C(s+t) > \frac{35}{4}A$. Call machine i *heavy* if $t_i = 0$ and *light* otherwise. Our first observation is that machine $n+1$ must be light (even if all centrally controlled jobs are assigned to machine $n+1$, some selfishly controlled jobs are subsequently assigned to it). Next, we note that for $i, j \in \{1, 2, \ldots, n\}$, the marginal cost $\frac{2(s_i+t_i)}{a_i} + 4$ of machine i is at most the marginal cost $\frac{2(s_j+t_j)}{a_j} + 4$ of machine j if and only if the latency $\ell_i(s_i + t_i)$ of machine i is at most $\ell_j(s_j + t_j)$. We may assume that all heavy machines have the same marginal cost with respect to $s+t$, as reassigning some centrally controlled jobs from a heavy machine with large marginal cost to a heavy machine with small marginal cost does not affect the induced equilibrium and can only decrease the cost of the induced assignment. All heavy machines must then have equal latency with respect to $s+t$. Naturally, Lemma 2.10 implies that all light machines possess a common latency with respect to $s+t$. That all machines have one of two latencies will make the cost of $s+t$ easy to compute.

With the induced assignment $s+t$ still fixed, let $S \subseteq \{1, 2, \ldots, n\}$ denote the set of heavy machines. If $S = \emptyset$ then $s+t$ is the Nash assignment for $\mathcal{I}'$ with $s_i + t_i = \frac{a_i}{2}$ for $i \in \{1, 2, \ldots, n\}$ and $s_{n+1} + t_{n+1} = \frac{3}{2}A$, satisfying $C(s+t) = 9A > \frac{35}{4}A$. So suppose S is nonempty and define $\beta \in (0, 1]$ by the equation $\sum_{i \in S} a_i = \beta A$. Define $\gamma \in (0, \frac{1}{2}]$ by the equation $\sum_{i \in S} s_i = \gamma A$. Our aim is to lower bound the cost of $s+t$ as a function of the parameters β and γ.

Since all heavy machines have equal latency, for i heavy we must have $s_i + t_i = s_i = \frac{\gamma}{\beta}a_i$ with $\ell_i(s_i + t_i) = 4 + \frac{\gamma}{\beta}$. Since all light machines have equal latency, we must have

$$s_i + t_i = a_i\frac{2 - 3\gamma}{4 - 3\beta}$$

with

$$\ell_i(s_i + t_i) = 4 + \frac{2 - 3\gamma}{4 - 3\beta}$$

for $i \in \{1, 2, \ldots, n\} \setminus S$ and

$$s_{n+1} + t_{n+1} = A\left(\frac{4}{3} + \frac{2-3\gamma}{12-9\beta}\right)$$

with

$$\ell_{n+1}(s_{n+1} + t_{n+1}) = 4 + \frac{2-3\gamma}{4-3\beta}.$$

The total cost of this solution is

$$\begin{aligned} C(s+t) &= \gamma A\left(4 + \frac{\gamma}{\beta}\right) + (2-\gamma)A\left(4 + \frac{2-3\gamma}{4-3\beta}\right) \\ &= A\left(8 + \frac{(4-3\beta)\gamma^2 + \beta(2-\gamma)(2-3\gamma)}{\beta(4-3\beta)}\right). \end{aligned}$$

Holding β fixed and differentiating with respect to γ, we find that this expression has unique minimizer $\gamma = \beta$ when $\beta \leq \frac{1}{2}$ and $\gamma = \frac{1}{2}$ when $\beta \geq \frac{1}{2}$ (subject to the condition $\gamma \in (0, \frac{1}{2}]$). There are now two cases to analyze. First suppose that $\beta \leq \frac{1}{2}$. Setting $\gamma = \beta$ we obtain

$$C(s+t) = A\left(8 + \frac{4-4\beta}{4-3\beta}\right);$$

differentiating with respect to β, we see that the expression has a unique minimizer $\beta = \frac{1}{2}$ (subject to the condition $\beta \in (0, \frac{1}{2}]$) yielding cost $A(8 + \frac{4}{5}) > \frac{35}{4}A$. Finally, assume that $\beta \geq \frac{1}{2}$. Setting $\gamma = \frac{1}{2}$ we find that the cost of $s+t$ is given by

$$C(s+t) = A\left(8 + \frac{1}{\beta(4-3\beta)}\right);$$

differentiating with respect to β, we find that this expression has unique minimizer $\beta = \frac{2}{3}$, at which point the equation reads $C(s+t) = \frac{35}{4}A$. However, since $\mathcal{I}$ is a "no instance" of $\frac{1}{3}$-$\frac{2}{3}$ Partition, we must have $\beta \neq \frac{2}{3}$ and hence $C(s+t) > \frac{35}{4}A$. We have exhausted all possible cases, and the reduction is complete. ■

7 Directions for Future Work

The results of this paper suggest a number of problems deserving further study. An important and general open question is to what extent our machine-scheduling results carry over to the more complex domain of general networks. For example, given a directed graph G with a source vertex s and a sink t, a rate r of network traffic that wishes to travel from s to t, a latency function on each arc, and a parameter α specifying the fraction of traffic that is centrally controlled, how should the managed traffic be routed so as to induce the best possible equilibrium?[7] On the one hand, an example in a four-vertex network shows that

[7] The notions of equilibria used in this paper are equally easy to define in general networks; see, e.g., [45] for a formal treatment.

a worst-case performance guarantee as small as $1/\alpha$ is impossible to attain with general latency functions [44]. On the other hand, the example does not rule out the possibility of a guarantee that is a larger function of α. Is it always possible to induce an assignment with cost no more than a constant—depending on α, but *not* on the size of the network—times that of the optimal assignment of traffic to s-t paths? Is a performance guarantee of $\frac{4}{3} - \epsilon$ possible for the special case of linear latency functions?

There are also unexplored avenues in the machine-scheduling setting. In Section 6, we considered the optimization problem of computing the optimal Stackelberg strategy, and observed that the LLF strategy achieves the best approximation ratio possible using the cost of the optimal assignment as a lower bound. Can a better approximation ratio be proved for LLF via a better lower bound on the cost of the assignment induced by the optimal strategy?

Another natural question is whether more sophisticated approximation algorithms can achieve a better approximation ratio. This question has been resolved in the affirmative by Kumar and Marathe [32], who have given a fully polynomial-time approximation scheme[8] for the problem under mild conditions on the machine latency functions.

Acknowledgements

We thank Éva Tardos for several helpful discussions and for comments on an earlier draft of this paper. We also thank Anupam Gupta for useful discussions, Ben Atkin and Jon Kleinberg for pointing out relevant references, and the anonymous referees for their helpful comments.

References

[1] E. Anshelevich, A. Dasgupta, É. Tardos, and T. Wexler. Near-optimal network design with selfish agents. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 511–520, 2003.

[2] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 482–491, 2001.

[3] A. Bagchi. *Stackelberg Differential Games in Economic Models.* Springer-Verlag, 1984.

[4] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* SIAM, 1999.

[5] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation.* Yale University Press, 1956.

[8] A *fully polynomial-time approximation scheme* for a minimization problem is an algorithm A with the following property for some polynomial $p(\cdot,\cdot)$: given error parameter $\epsilon > 0$ and problem instance $\mathcal{I}$ with size $|\mathcal{I}|$, A returns a solution to $\mathcal{I}$ with objective function value at most $1+\epsilon$ times that of optimal in time at most $p(|\mathcal{I}|, \epsilon^{-1})$.

[6] K. P. Birman. *Building Secure and Reliable Network Applications*. Manning, 1996.

[7] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. J. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the Internet. Network Working Group Request for Comments 2309, April 1998.

[8] R. Cocchi, S. J. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: Motivation, formulation, and example. *IEEE/ACM Transactions on Networking*, 1(6):614–627, 1993.

[9] R. Cole, Y. Dodis, and T. Roughgarden. How much can taxes help selfish routing? In *Proceedings of the Fourth Annual ACM Conference on Electronic Commerce*, pages 98–107, 2003.

[10] R. Cole, Y. Dodis, and T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 521–530, 2003.

[11] A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 287–296, 2002.

[12] A. Czumaj and B. Voecking. Tight bounds for worst-case equilibria. In *Proceedings of the 13th Annual Symposium on Discrete Algorithms*, pages 413–420, 2002.

[13] S. C. Dafermos and F. T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards, Series B*, 73B(2):91–118, 1969.

[14] C. Douligeris and R. Mazumdar. Multilevel flow control of queues. In *Proceedings of the Johns Hopkins Conference on Information Sciences and Systems*, page 21, 1989.

[15] C. Douligeris and R. Mazumdar. A game theoretic perspective to flow control in telecommunication networks. *Journal of the Franklin Institute*, 329:383–402, 1992.

[16] P. Dubey. Inefficiency of Nash equilibria. *Mathematics of Operations Research*, 11(1):1–8, 1986.

[17] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K.K. Ramakrishnan, and J. E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of ACM SIGCOMM, Computer Communication Review*, volume 29, pages 95–108, 1999.

[18] A. A. Economides and J. A. Silvester. Priority load sharing: An approach using Stackelberg games. In *Proceedings of the 28th Annual Allerton Conference on Communications, Control, and Computing*, pages 674–683, 1990.

[19] J. Feigenbaum, C. H. Papadimitriou, and S. J. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001. Preliminary version in *STOC '00*.

[20] J. A. Fingerhut, S. Suri, and J. S. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.

[21] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[22] A. Gupta, A. Kumar, J. Kleinberg, R. Rastogi, and B. Yener. Provisioning a Virtual Private Network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 389–398, 2001.

[23] L. A. Hall. Approximation algorithms for scheduling. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 1, pages 1–45. PWS Publishing Company, 1997.

[24] R. Johari and J. N. Tsitsiklis. Network resource allocation and a congestion game. Manuscript, 2003.

[25] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[26] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.

[27] Y. A. Korilis, A. A. Lazar, and A. Orda. Capacity allocation under noncooperative routing. *IEEE Transactions on Automatic Control*, 42(3):309–325, 1997.

[28] Y. A. Korilis, A. A. Lazar, and A. Orda. Avoiding the Braess paradox in noncooperative networks. *Journal of Applied Probability*, 36(1):211–222, 1999.

[29] E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate equilibria and ball fusion. In *Proceedings of the 9th Annual International Colloquium on Structural Information and Communication Complexity*, 2002.

[30] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[31] D. C. Kozen. *Design and Analysis of Algorithms.* Springer-Verlag, 1992.

[32] V. S. A. Kumar and M. V. Marathe. Improved results for Stackelberg scheduling strategies. In *29th International Colloquium on Automota, Languages, and Programming*, pages 776–787, 2002.

[33] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford University Press, 1995.

[34] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 510–519, 2001.

[35] N. Nisan. Algorithms for selfish agents: Mechanism design for distributed computation. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–15, 1999.

[36] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1/2):166–196, 2001. Preliminary version in *STOC '99*.

[37] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multi-user communication networks. *IEEE/ACM Transactions on Networking*, 1:510–521, 1993.

[38] G. Owen. *Game Theory*. Academic Press, 1995. Third Edition.

[39] A. L. Peressini, F. E. Sullivan, and J. J. Uhl. *The Mathematics of Nonlinear Programming*. Springer-Verlag, 1988.

[40] C. ReVelle and D. Serra. The maximum capture problem including relocation. *INFOR*, 29:130–138, 1991.

[41] A. Ronen. *Solving Optimization Problems among Selfish Agents*. PhD thesis, Hebrew University of Jerusalem, 2000.

[42] T. Roughgarden. Designing networks for selfish users is hard. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 472–481, 2001.

[43] T. Roughgarden. The price of anarchy is independent of the network topology. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 428–437, 2002. Full version to appear in *Journal of Computer and System Sciences*.

[44] T. Roughgarden. *Selfish Routing*. PhD thesis, Cornell University, 2002.

[45] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002. Preliminary version in *FOCS '00*.

[46] A. S. Schulz and N. Stier Moses. On the performance of user equilibria in traffic networks. In *Proceedings of the 14th Annual Symposium on Discrete Algorithms*, pages 86–87, 2003.

[47] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, 1985.

[48] S. J. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEE/ACM Transactions on Networking*, 3(6):819–831, 1995.

[49] H. von Stackelberg. *Marktform und Gleichgewicht*. Springer-Verlag, 1934. English translation, entitled *The Theory of the Market Economy*, published in 1952 by Oxford University Press.