Fairness Measures for Resource Allocation

Amit Kumar *　　Jon Kleinberg †

Abstract

In many optimization problems, one seeks to allocate a limited set of resources to a set of individuals with demands. Thus, such allocations can naturally be viewed as vectors, with one coordinate representing each individual. Motivated by work in network routing and bandwidth assignment, we consider the problem of producing solutions that simultaneously approximate *all* feasible allocations in a coordinate-wise sense.

This is a very strong type of "global" approximation guarantee, and we explore its consequences in a range of discrete optimization problems, including facility location, scheduling, and bandwidth assignment in networks. A fundamental issue — one not encountered in the traditional design of approximation algorithms — is that good approximations in this global sense need not exist for every problem instance; there is no *a priori* reason why there should be an allocation that simultaneously approximates all others. As a result, the existential questions concerning such good allocations lead to a new perspective on a number of basic problems in resource allocation, and on the structure of their feasible solutions.

1 Introduction

Fair Allocations. In many optimization problems, one seeks to allocate a limited set of resources to a set of individuals with demands. For example, given n users of a network, each seeking to transmit data between some pair of nodes, we must decide how to allocate a portion of the available bandwidth to each. A common approach to such resource allocation problems has sought allocations that optimize a particular aggregate figure of merit. In the case of network connections, for example, we could seek a solution that maximizes the minimum amount of bandwidth allocated to any request, or the total bandwidth allocated to all requests.

But these types of problems genuinely involve n distinct objectives — the demands of each individual — and a rich set of issues emerges when we adopt this perspective, treating the feasible solutions as vectors with a separate coordinate to represent the allocation to

*Department of Computer Science, Cornell University, Ithaca NY 14853. Email: amitk@cs.cornell.edu. Supported in part by the ONR Young Investigator Award of the second author.

†Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu. Supported in part by a David and Lucile Packard Foundation Fellowship, an ONR Young Investigator Award, and NSF Faculty Early Career Development Award CCR-9701399.

each individual. In the networking literature, this approach underlies the notion of *max-min fairness* [1, 11, 12], a widely used mechanism which produces an allocation for a fixed set of routes that is *lexicographically maximum*: the minimum bandwidth assigned to any connection is maximized; subject to this, the minimum bandwidth assigned to any of the remaining connections is maximized; and so forth. Such a vector is often called the *max-min fair vector*. The same notion was formulated in the context of network flow problems by Megiddo [17].

There are cases in which it is computationally intractable to determine the max-min fair vector of allocations. In such settings, Kleinberg, Rabani, and Tardos proposed the notion of a *coordinate-wise approximation*: we say that a vector X is a coordinate-wise c-approximation to a vector X' if for each j, the j^{th} largest coordinate of X is at least $1/c$ times the value of the j^{th} largest coordinate of X' [13]. One can then look for efficiently computable vectors X that form good approximations in this sense to the max-min fair vector. In recent work concerned with on-line path selection and bandwidth allocation, Goel, Meyerson, and Plotkin [7] developed algorithms to satisfy a much stronger goal: computing a vector X that is a coordinate-wise c-approximation to *every* allocation vector. Such a vector satisfies the following natural and strong type of guarantee: for any value of m, if there is any allocation in which m users each get bandwidth at least b, then at least m users get bandwidth at least b/c under allocation X. We will therefore refer to X as a *global c-approximation* for this instance. A simple example from [7] shows that the max-min fair vector can be a very poor solution under this measure of quality.

The Present Work. In this work, we develop algorithms that produce allocations with this stronger type of guarantee: global approximation to all possible allocations. We consider a range of problems that can be cast in the framework of resource allocation, with an emphasis on three particular domains.

(i) *Bandwidth allocation.* We consider a version of the setting discussed above, in which there are n given paths in a network, and we must assign bandwidth to each one in a way that respects capacity constraints. Such an assignment induces an *allocation vector* of length n, whose i^{th} coordinate is the amount of bandwidth assigned to path i.

(ii) *Scheduling.* Given a set of n jobs that need to be processed, a schedule for these jobs induces a *completion time vector*, whose i^{th} coordinate is the completion time of job i. Using the above notions, we can compare different completion time vectors in a coordinate-wise sense.

(iii) *Facility location.* Given a set of n demand points in a metric space, opening k facilities to serve these points induces a *distance vector* of length n: the i^{th} coordinate is the distance from the i^{th} demand point to its nearest facility. Again, we can compare the distance vectors induced by different sets of k facilities, so as to find the "fairest" solution.

What we find particularly interesting is that the notion of approximation actually plays a fundamentally different role in this type of allocation problem than it does in standard approximation algorithms for NP-complete problems. In the traditional design of approximation algorithms, the existence of an optimum (or of optima) is not in question; rather, we lack the computational power to find these optima. The definition of global approximation, on the other hand, comes with the following crucial point: there is no *a priori* guarantee that a good global approximation exists for a particular instance of a problem. Indeed, global 1-approximations — allocations that are as good as every allocation in every coordinate of a sorted order — turn out to be quite rare, and often indicate something striking in the structure of the instance.

Thus, our study of this notion has two related components: an existential aspect, in which we try to determine whether good global approximations exist for all instances of a particular problem; and an algorithmic aspect, in which we look for efficient ways to compute global approximations. Each is important for our understanding of this issue, with existential results illustrating the kinds of fair allocations that are achievable, even when we do not yet know of corresponding efficient algorithms that produce them.

Formulating the Problem. We now introduce some precise notation for expressing the basic questions and results. All of the three types of problems above have the property that a solution to the problem induces a vector of length n; thus, for each instance $\mathcal{I}$ of such a problem, we have a set $V(\mathcal{I})$ consisting of all vectors that are induced by some feasible solution to the instance. Now, for a vector $X \in V(\mathcal{I})$, we define $\overrightarrow{X}$ to be the vector in which the coordinates of X have been reordered so they are in ascending order, and $\overleftarrow{X}$ to be the vector in which the coordinates of X have been reordered so they are in descending order. (Note that we allow multiple coordinates in X may be the same; we simply require $\overrightarrow{X}$ and $\overleftarrow{X}$ to be in non-decreasing and non-increasing order, respectively.) For two vectors $X, Y \in V(\mathcal{I})$, we write $X \prec_c Y$ if $X_i \leq Y_i$ for all i. Note that for purposes of coordinate-wise comparison, we have $\overrightarrow{X} \prec_c \overrightarrow{Y}$ if and only if $\overleftarrow{X} \prec_c \overleftarrow{Y}$ — it doesn't matter for the comparison whether we write the small values at the left or the right.

As is common in discussing approximation, our terminology will be slightly different depending on whether we are discussing a minimization problem or a maximization problem. (Note that both arise in our list of problems above). For an allocation vector X in a minimization problem, we define $c(X)$ to be the infimum of α such that $\overrightarrow{X} \prec_c \alpha\overrightarrow{Y}$ for all $Y \in V(\mathcal{I})$. For an allocation vector X in a maximization problem, we define $c(X)$ to be the infimum of α such that $\overrightarrow{Y} \prec_c \alpha\overrightarrow{X}$ for all $Y \in V(\mathcal{I})$. In both cases, this can be informally viewed as the *global approximation ratio of* X: it is the smallest α for which $\overrightarrow{X}$ is an α-approximation, in the natural coordinate-wise sense, to every vector $Y \in V(\mathcal{I})$. The best global approximation ratio achievable on the instance $\mathcal{I}$ is then defined as

$$c^*(\mathcal{I}) = \inf_{X \in V(\mathcal{I})} c(X).$$

For several types of problems, we will find that good global approximations do not necessarily exist; but in many of these cases there will always exist solutions satisfying a weaker global guarantee, in which we compare the *prefix sums* of one allocation to the prefix sums of other allocations, as in [13]. Thus, for a vector $X \in V(\mathcal{I})$, we define $\sigma(X)$ to be the vector whose i^{th} coordinate is $\sum_{j \leq i} X_j$ We say that $X \prec_s Y$ if $\sigma(X) \prec_c \sigma(Y)$. Now for a minimization (resp. maximization) problem, we define $s(X)$ to be the infimum of α such that $\overleftarrow{X} \prec_s \alpha\overleftarrow{Y}$ (resp. $\overrightarrow{Y} \prec_s \alpha\overrightarrow{X}$) for all $Y \in V(\mathcal{I})$. In this case, we say that X is a *global α-approximation under prefix sums.* In a bandwidth allocation problem, for example, we can view such a vector X as conveying the following guarantee: for any m, if there is an allocation in which m users receive an *average bandwidth* of at least b, then at least m users receive an average allocation of at least b/α under allocation X. Note, crucially, that in a minimization (resp. maximization) problem, we begin computing prefix sums with the largest (resp. smallest) coordinates. Finally, for an instance $\mathcal{I}$, we define the best possible global approximation ratio under prefix sums as

$$s^*(\mathcal{I}) = \inf_{X \in V(\mathcal{I})} s(X).$$

Note that since $s(X) \leq c(X)$ for any vector X, we have $s^*(\mathcal{I}) \leq c^*(\mathcal{I})$ for any instance $\mathcal{I}$.

Sometimes we will have an instance $\mathcal{I}$, and want to evaluate the quality of an allocation $X \notin V(\mathcal{I})$. Thus, we allow the definitions of $c(X)$ and $s(X)$, exactly as written above, to apply to vectors $X \notin V(\mathcal{I})$; there is no difficulty in doing this, and the meaning will be clear from context.

An Illustrative Example. Before discussing our results, we consider a very simple example to illustrate how the optimal allocations can differ qualitatively under the c^* and s^* measures. Suppose we have n users of a network, each trying to transmit data across a single shared link of unit capacity. (Thus the underlying network consists of two nodes joined by a single edge.)

First, we claim that the bandwidth allocation vector X in which each coordinate is equal to $1/n$ satisfies $s(X) = 1$. Indeed, in any other vector Y, and for any $j \leq n$, the j smallest coordinates of Y must add up to at most j/n, while in X they add up to exactly j/n.

However, the uniform allocation vector X has a very large global approximation ratio under coordinate-wise comparison; specifically, $c(X) = n$. To see this, consider the vector Y_1 in which the first coordinate is 1 and all others are 0. A much better global approximation ratio is achieved by the vector X', whose j^{th} coordinate is equal to $1/jH_n$, where $H_n = \sum_{i=1}^{n} 1/i$. We claim that $c(X') = H_n$; for in any other vector Y, the j^{th} largest coordinate is at most $1/j$. In fact, $c^*(\mathcal{I}) = H_n$ for this simple instance $\mathcal{I}$: Any vector Y must have some coordinate, say the j^{th} largest, that is at most the value of corresponding coordinate in X'; and hence this coordinate is H_n times smaller than the j^{th} largest coordinate of the vector assigning $1/j$ to j of the coordinates, and 0 to all others.

In some sense, of course, the optimal allocation X under the s^* measure has a more intuitive meaning — equal division of resources among all individuals — than the optimal allocation X' under the c^* measure. However, it is interesting that the allocation X' also corresponds to a well-studied distribution of wealth: it is an instance of the *Pareto distribution* [18], one of the simplest types of allocations to follow an inverse power law.

2 Overview of Results

Bandwidth Allocation. We consider the problem that underlies the Hayden/Jaffe model of *max-min fairness* [1, 11, 12]. We are given a graph $G = (V, E)$, and paths $P_1, \ldots, P_n$ in this graph, representing users transmitting data. We must assign a *rate* x_i to each path P_i so that the following *capacity condition* is satisfied: if $P_{i_1}, \ldots, P_{i_t}$ all pass through a common edge, then $\sum_{j=1}^{t} x_{i_j} \leq 1$. Thus, a solution to an instance of this problem is a vector X of rates. We will refer to this problem as the *bandwidth allocation problem.* Note that our capacity condition implies that all links have the same capacity (set to 1); it is possible to obtain bounds for the more general case where edges have different capacities, but we do not explore this in detail here.

For an instance $\mathcal{I}$, let U denote the *congestion* of the set of paths: the maximum number of paths that pass through any single edge. Note that $U \leq n$. Although Goel, Meyerson, and Plotkin [7] considered a problem in which both the paths and the rates had to be chosen, their on-line algorithm can be applied to this setting with fixed paths as well; it shows that $c^*(\mathcal{I}) = (\log^2 |V| \log U)$ for every instance $\mathcal{I}$.

Here, we are not restricted to using on-line algorithms. Nevertheless, there are instances where a constant-factor global approximation does not exist, even in the sense of prefix sums. We show that for arbitrarily large U, there exists instances $\mathcal{I}$ of the bandwidth allocation problem with congestion U for which $s^*(\mathcal{I}) = \Omega\left(\frac{\log U}{\log \log U}\right)$. Since $c^*(\mathcal{I}) \geq s^*(\mathcal{I})$ for every instance, this is a lower bound for c^* as well.

We can prove upper bounds that nearly match this lower bound. Specifically, $s^*(\mathcal{I}) = O(\log U)$ for every instance $\mathcal{I}$ of the bandwidth allocation problem. We can produce a vector of rates X achieving this bound via a polynomial-time algorithm. More strongly, we show $c^*(\mathcal{I}) = O(\log U)$ for every instance $\mathcal{I}$ of the bandwidth allocation problem, but here we do not have a polynomial-time algorithm achieving this bound.

One can obtain strong bounds on s^* for certain related problems. For the problem of single-source fractional flow to each of n sinks — where we consider the vector of flow values to each sink — Megiddo showed that there is a maximum flow that induces the max-min fair vector of flows to the sinks [17]. By adapting some of the analysis from his proof of this, one can in fact show that $s^*(\mathcal{I}) = 1$ for all instances of this problem. For the uniform load balancing problem considered by Kleinberg, Rabani, and Tardos, one can extend the analysis of [13] to show that $s^*(\mathcal{I}) = 1$ for all instances $\mathcal{I}$.

Scheduling Problems. There are a large number of possible scheduling problems, but we can begin with some general ideas. First, in all the scheduling problems we will consider, a solution S induces a vector X_S of completion times. Two traditional objective functions are the *makespan* — the first coordinate in $\overleftarrow{X_S}$ — and the sum of all completion times. There has been a sequence of papers beginning with Stein and Wein [21] that showed there always exist solutions to many types of scheduling problems that are good approximations to both objectives simultaneously. The algorithm of [21] does not provide a simultaneous approximation to all prefix sums of $\overleftarrow{X_S}$, however.

We follow the standard notation for scheduling problems proposed by Graham et al. [9]. Under this notation, a scheduling problem is usually denoted by three parameters $\alpha|\beta|\gamma$, where α designates the number of machines (and whether they are identical or different); β generally designates special conditions such as precedence constraints, release dates, or fixed processing times; and γ denotes an objective function such as the makespan or the sum of completion times. In our formulations, γ will be set equal to the symbol *all*, to denote the fact that we are considering the full vector of completion times, rather than an aggregate objective function.

There are approximation algorithms for scheduling that perform coordinate-by-coordinate comparison with respect to the completion time vector of the optimal schedule. If this analysis implicitly holds with respect to *any* schedule, rather than just the optimal one, then one is fact giving a bound for c^*. The most basic example of this is Smith's rule for 1-processor scheduling, in which jobs are scheduled in ascending order of processing time [20]. The optimality proof for this schedule S in fact shows that $c(X_S) = 1$. Also, for 1-processor scheduling with release dates $(1|r_j|all)$, the analysis of Phillips, Stein, and Wein [19] shows that their schedule S achieves $c(X_S) \leq 2$; for the case of $m > 1$ identical parallel machines with release dates $(P|r_j|all)$, their schedule achieves $c(X_S) \leq 3$.

Here we focus on problems with precedence constraints: we have a set J of n jobs, with a partial order $\prec_J$ representing dependencies, and we must produce a schedule in which each job must end before the jobs that depend on it can begin. We prove that $c^*(\mathcal{I}) \leq 4$ for all instances $\mathcal{I}$ of precedence-constrained scheduling with release dates on m identical parallel machines $(P|prec, r_j|all)$, for any $m \geq 1$.

Our construction of the schedule S that achieves $c(X_S) \leq 4$ does not yield a polynomial-time algorithm even in the 1-processor case, and we show that it may be a difficult problem to obtain such a polynomial-time algorithm. Specifically, we prove that any polynomial-time algorithm producing a schedule S that satisfies $c(X_S) = O(1)$ for instances of precedence-constrained 1-processor scheduling $(1|prec|all)$ implies a polynomial-time constant-factor approximation algorithm for the *densest subgraph problem*: given a graph G and a number k, find an induced k-node subgraph with the maximum possible number of edges. For this problem, the current best approximation ratios have the form $O(k^c)$, for a constant $0 < c < 1$ [14]. It is also interesting to note that for many types of precedence-constrained scheduling problems, Hall, Schulz, Shmoys, and Wein [10] produced a coordinate-wise $O(1)$-approximation to an optimal vector of *fractional completion times*, corresponding to a linear programming relaxation. However, even for precedence-constrained 1-processor scheduling $(1|prec|all)$, there exist instances in which this optimal vector X of fractional completion times has a value of $c(X)$ that is arbitrarily large.

We consider the special case of precedence-constrained scheduling in which there are no release dates, and all jobs have unit processing times $(P|p_j = 1, prec|all)$. Our main result here is that for the case of two machines, $c^*(\mathcal{I}) = 1$ for every instance $\mathcal{I}$. It is not difficult to show that there are instances with $m \geq 3$ machines for which $c^*(\mathcal{I}) > 1$; and a simple greedy algorithm can be used to establish that $c^*(\mathcal{I}) \leq 2$ for every instance with $m \geq 3$ machines. This dichotomy between $m = 2$ and $m \geq 3$ forms an interesting parallel to the fact that the 2-processor case is known to be polynomially solvable [4, 6], while the complexity of the m-processor case for fixed $m \geq 3$ is an open question.

As a final note, one can view traveling salesman problems within this perspective on scheduling, since tours yield vectors of *arrival times* at each city. In this context, the approximation of the minimum latency problem by k-trees [2, 8] shows that there is a constant upper bound on $c^*(\mathcal{I})$ for all instances of the traveling salesman problem.

Facility Location Problems. Finally, we consider facility location problems. Suppose we have a metric space M, a set $P = \{p_1, \ldots, p_n\}$ of *demand points* in M, and a set F of n' *candidate facility points*. The *k-center problem* and the *k-median problem* are two classical facility location problems. The first asks: Choose k facilities so that the maximum distance from any point to its nearest facility is minimized; the second asks: Choose k facilities so that the total distance from all points to their nearest facility is minimized.

Guided by the present framework, we say that opening facilities at locations $F' \subseteq F$ induces a *distance vector* $X_{F'}$ whose i^{th} coordinate is $d(p_i, F')$, the distance from p_i to its nearest facility in F'. Thus, when we speak of a *k-facility instance* $\mathcal{I}$, we mean a metric space M, with sets P and F; and we will compare solutions to the set $V(\mathcal{I})$ of all vectors $X_{F'}$ such that $|F'| = k$. Now, the k-center problem seeks to minimize the first coordinate of $\overleftarrow{X_{F'}}$, while the k-median problem seeks to minimize the sum of all coordinates in $X_{F'}$. What can be said about global approximations for arbitrary k-facility instances?

We first consider the case $k = 1$, and show that $s^*(\mathcal{I}) \leq 4$ for every instance $\mathcal{I}$ of the 1-facility location problem. The c^* measure is trickier. Consider an instance with a set P of five points in the plane — two at the ends of a very short line segment, and three at the corners of a moderately sized equilateral triangle, arbitrarily far away. If $F = P$, it is easy to see that such an instance can be constructed in which c^* is arbitrarily large. By extending this idea, we can show that for any number α, there are 1-facility instances $\mathcal{I}$ for which no set F' of fewer than $\log n$ facilities achieves $c(X_{F'}) \leq \alpha$. At the same time, this lower bound is essentially tight: For every 1-facility instance $\mathcal{I}$, there is a set F' of $O(\log n)$ facilities for which $c(X_{F'}) \leq 3$. Note that here we are comparing the distance vector $X_{F'}$, for a set F' of $\leq O(\log n)$ facilities, to all possible distance vectors obtained by opening a *single* facility. We also note our two upper bounds here are achieved by polynomial-time algorithms.

Now consider a larger value of k. If there were a constant upper bound on $s^*(\mathcal{I})$ for every k-facility instance $\mathcal{I}$, it would imply that in every instance there is a set of k facilities that provides a simultaneous constant-factor approximation for the k-center and k-median objective functions. But this is not true: consider an instance with a set P of $2m + 1$ points in the plane: m are clustered in a small circle, m more are clustered in a nearby small circle, and one is very far away from all others. If $F = P$ and $k = 2$, then we see that any good solution for the k-median problem must put one facility in each circle, while any good solution for the k-center problem must put one facility on the far-away point.

However, we can prove the following upper bound: For every k-facility instance $\mathcal{I}$, there is a set F' of $O(k \log n + \varepsilon^{-1})$ facilities for which $s(X_{F'}) \leq 3 + \varepsilon$. Moreover, we can compute an F'' of this size in polynomial time for which $s(X_{F''})) \leq 9 + \varepsilon$. We have a lower bound showing this is almost tight: For every α, there are k-facility instances where no set F' of fewer than $\Omega\left(k \dfrac{\log(n/k)}{\log \alpha}\right)$ facilities can achieve $s(F') \leq \alpha$. For the c^* measure, we show that there are essentially no non-trivial upper bounds when $k > 1$: for every α, there are 2-facility instances for which no proper subset $F' \subseteq F$ can achieve $c(X_{F'}) \leq \alpha$.

There has been work in discrete location theory on defining notions of "fairness" in facility location [5, 16]. Most of the measures to arise from this previous work are concerned with making the distance vector approximately uniform relative to *a priori* weights on the points, rather than optimizing the types of approximation measures we are considering here.

3 Bandwidth Allocation

Recall that in our bandwidth allocation problem from Section 1, we have a graph $G = (V, E)$, and a set $\mathcal{P}$ of paths $P_1, \ldots, P_n$ in this graph. We wish to assign a vector X of rates to these paths, respecting the (unit) capacities in the graph. For a set of paths $\mathcal{P}' \subseteq \mathcal{P}$, we use $\nu(\mathcal{P}')$ to denote their congestion; we write $U = \nu(\mathcal{P})$.

We first show

Theorem 3.1 *$s^*(\mathcal{I}) = O(\log U)$ for every instance $\mathcal{I}$ of the bandwidth allocation problem.*

Proof. We describe an algorithm that constructs a vector X of rates so that $s(X) = O(\log U)$. We call a set of paths $\mathcal{P}'$ a *block* if they all contain a common edge e; we call e a *blocking edge* of $\mathcal{P}'$. We divide our algorithm into $1 + \lceil \log U \rceil$ phases. In phase $i = 0, 1, \ldots, \lceil \log U \rceil$, we find a maximal collection of disjoint blocks, each of which has size greater than $\frac{U}{2^{i+1}}$. For every path P in one of these blocks $\mathcal{P}'$ we allocate it a rate of $\frac{1}{|\mathcal{P}'|}$. We then delete all these paths, but — at this point in the construction — do not reduce the capacity of any edges.

We observe that the following invariant is maintained by this algorithm: at the end of phase $i \leq \lceil \log U \rceil$, the overall congestion is at most $\frac{U}{2^{i+1}}$. Thus after phase $1 + \lceil \log U \rceil$, all paths have been assigned a rate.

Let X_0 denote the resulting vector of rates. Although X_0 does not respect the capacities, it is useful to compare it to feasible rate vectors. Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be the set of blocks found by the above algorithm in descending order of size. Consider any feasible rate vector Y. For any i, let $X_0|\mathcal{P}_i$ and $Y|\mathcal{P}_i$ denote the vectors X_0 and Y restricted to coordinates corresponding to paths in $\mathcal{P}_i$. We claim that $\overrightarrow{Y|\mathcal{P}_i} \prec_s \overrightarrow{X_0|\mathcal{P}_i}$. Indeed, the rates assigned by Y to paths in $\mathcal{P}_i$ must sum to at most one — since they all pass through the blocking edge e — and so the j smallest coordinates in $\overrightarrow{Y|\mathcal{P}_i}$ must sum to at most $j/|\mathcal{P}_i|$. X_0 assigns rates to the paths in $\mathcal{P}_i$ so that the sum of the j smallest coordinates is exactly $j/|\mathcal{P}_i|$.

Now, $\overrightarrow{X_0}$ consists of vectors $\overrightarrow{X_0|\mathcal{P}_1}, \overrightarrow{X_0|\mathcal{P}_2}, \ldots, \overrightarrow{X_0|\mathcal{P}_r}$ concatenated in this order. Consider the vector Y with its coordinates arranged in this same order. Each prefix sum of $\overrightarrow{X_0}$ is at least as large as the corresponding prefix sum of Y in this order; and if we now re-arrange the coordinates of Y so that they are in ascending order, the prefix sums of Y cannot increase. Thus $\overrightarrow{Y} \prec_s \overrightarrow{X_0}$.

Now we claim that the rate vector X obtained by dividing each coordinate of X_0 by $2\lceil \log U \rceil + 2$ respects all the capacity conditions; this will complete the proof. Indeed, to prove this, we need only show that the total of all rates assigned to paths using edge e in a given phase i is at most 2. At the start of phase i, the congestion is at most $\frac{U}{2^i}$. Each set of paths $\mathcal{P}'$ which is selected in this phase has congestion at least $\frac{U}{2^{i+1}}$ – so, all paths that get assigned a rate in this phase get a rate of at most $\frac{2^{i+1}}{U}$. Hence at most 2 extra units of rate are assigned in this phase to all paths using e. ∎

We now prove the following stronger statement. It too is based on a construction in which we allocate bandwidth according to an exponentially decaying sequence; however, for this construction, we do not know how to run each iteration efficiently, since it involves the solution of an independent set problem.

Theorem 3.2 *$c^*(\mathcal{I}) = O(\log U)$ for every instance $\mathcal{I}$ of the bandwidth allocation problem.*

Proof. We give a procedure for assigning rates to paths. Initially, let $\mathcal{P}_0 = \mathcal{P}$. Let $\mathcal{Q}_0 \subseteq \mathcal{P}_0$ be a maximum cardinality subset of paths of congestion 1. Each path in $\mathcal{Q}_0$ is assigned a rate of 1, and we define $\mathcal{P}_1 = \mathcal{P}_0 \setminus \mathcal{Q}_0$. Now suppose we have a set of paths $\mathcal{P}_i$, $i \geq 1$. Define $\mathcal{Q}_i \subseteq \mathcal{P}_i$ to be a maximum cardinality subset of congestion at most 2^i. Each path in $\mathcal{Q}_i$ is assigned a rate of $\frac{1}{2^i}$, and we define $\mathcal{P}_{i+1} = \mathcal{P}_i \setminus \mathcal{Q}_i$. We define these set of paths for all i, $0 \leq i \leq \lceil \log U \rceil$ – it is clear that after the end of this process, every path gets assigned a rate. Let X^* denote the rate vector produced (note that it does not necessarily satisfy the capacity constraints), and let n_i denote $|\mathcal{Q}_i|$.

Let Y be any vector of rates that satisfies the capacity constraints. We claim that $\overleftarrow{Y} \prec_c 2\overleftarrow{X^*}$. To prove this, we fix an i, $0 \leq i \leq \log U$. Let $\mathcal{Q}'$ denote the set of paths which get rate at least $\frac{1}{2^i}$ in Y. Define $n' = |\mathcal{Q}'|$. Clearly, the congestion of $\mathcal{Q}'$ is at most 2^i. We claim that $n_0 + \ldots + n_i \geq n'$. Indeed, while selecting $\mathcal{Q}_i$, we must have considered the paths in $\mathcal{Q}' - (\mathcal{Q}_1 \cup \ldots \cup \mathcal{Q}_{i-1})$. Since the $(n_0 + \ldots + n_i)^{\text{th}}$ coordinate of $\overleftarrow{X^*}$ is 2^{-i}, it is at least as large as the $(n_0 + \ldots + n_i + 1)^{\text{st}}$ coordinate of $\overleftarrow{Y}$. Now consider a coordinate j. If $j \leq n_0$, then $\overleftarrow{X^*}_j \geq \overleftarrow{Y}_j$. Otherwise, there is some i so that $n_0 + n_1 + \ldots + n_i + 1 \leq j \leq n_0 + n_1 + \ldots + n_{i+1}$, and $\overleftarrow{X^*}_j = \frac{1}{2^{i+1}}$. But $\overleftarrow{Y}_j \leq \overleftarrow{Y}_{n_0+\ldots+n_i+1} \leq \overleftarrow{X^*}_{n_0+\ldots+n_i} = \frac{1}{2^i}$. Thus, $\overleftarrow{Y}_j \leq 2\overleftarrow{X^*}_j$; since this applies to an arbitrary coordinate j, we have $\overleftarrow{Y} \prec_c 2\overleftarrow{X^*}$.

We also claim that for any edge e, the total of all rates assigned in X^* to paths containing e is at most $1 + \lceil \log U \rceil$. To see this, note that in each phase is at most one additional unit of rate can be assigned to paths passing through a single edge e.

Thus, if we define the rate vector X by dividing each coordinate of X^* by $1 + \lceil \log U \rceil$, we obtain the theorem. ■

We now show that the above results are almost optimal.

Theorem 3.3 *For arbitrarily large U, there exists instances $\mathcal{I}$ of the bandwidth allocation problem with congestion U for which $s^*(\mathcal{I}) = \Omega\left(\frac{\log U}{\log\log U}\right)$.*

Proof. The construction involves a number of parameters defined in terms of U; the relationships among these parameters will be specified as needed. We define an instance $\mathcal{I}$ as follows. Let k be a constant to be specified later in terms of U. We define constants $b = k+1$; $d = (k+1)^2$; and $a_i = d^{2i}$ for $i = 0, 1, \ldots, k$. The underlying graph G is a simple path on $n = a_k$ vertices. The set of paths $\mathcal{P}$ is partitioned into $k+1$ sets of paths, which we refer to as *level* 0 through *level* k. For each r, the paths in level r are grouped into a_r edge-disjoint *bundles* of b^{k-r} identical paths, each having length n/a_r; the union of these bundles covers the entire graph G. Each path in level $r+1$ is entirely contained in some path in level r. Note that

$$U = b^k + b^{k-1} + \cdots + b + 1 = \frac{(k+1)^{k+1} - 1}{k},$$

and hence $k = \Theta\left(\frac{\log U}{\log\log U}\right)$.

Let X be an assignment of rates to paths. We define $\tilde{X}$ to be the vector obtained from X by arranging the coordinates as follows. The coordinates for paths in the same bundle occur contiguously in $\tilde{X}$, beginning with level 0, then all the bundles in level 1, then all the bundles in level 2, and so on. Moreover, the bundles in a fixed level r are arranged so that if bundle β occurs before bundle β', then the sum of the rates of paths in β is at most the sum of the rates of paths in β'.

We set $c = k/3$, and suppose that $s(X) \leq c$. Since $\sigma(\overrightarrow{X}) \prec_c \sigma(\overrightarrow{\tilde{X}})$, it follows that for any assignment Y of rates to paths in $\mathcal{P}$, we have $\sigma(\overrightarrow{Y}) \prec_c c\sigma(\tilde{X})$. Now consider $k+1$ different assignment of rates to the paths. Y_0 assigns $(b^k + b^{k-1} + \cdots + 1)^{-1}$ to all paths. Similarly, Y_r, for $0 < r \leq k$, assigns rate $(b^{k-r} + \cdots + 1)^{-1}$ to all paths in levels $r, r+1, \ldots, k$ and 0 to paths in levels $0, \ldots, r-1$.

The fact that $\sigma(\overrightarrow{Y}) \prec_c c\sigma(\tilde{X})$ for all vectors Y implies that $\tilde{X}$ assigns each path in level 0 a rate of at least $c^{-1}(b^{k-r} + \cdots + 1)^{-1}$. Now consider the vector Y_r; let p_r denote the total number of paths in levels 0 through $r-1$, and define a constant $\ell_r = d^{2r-1}$. The coordinates of Y_r for the bundles in levels 0 to $r-1$ are 0. The sum of the next $\ell_r b^{k-r}$ coordinates in $\overrightarrow{Y_r}$ is equal to $\ell_r b^{k-r}(b^{k-r} + \ldots + 1)^{-1}$. Thus the sum of first $p_r + \ell_r b^{k-r}$ coordinates in $\overrightarrow{Y_r}$ is $\ell_r b^{k-r}(b^{k-r} + \ldots + 1)^{-1}$. In $\tilde{X}$, the sum of the coordinates for paths at level j can be at most a_j, because the sum of the rate in any one bundle can be at most 1. So the sum of the first p_r coordinates in $\tilde{X}$ is at most $a_0 + a_1 + \ldots + a_{r-1}$. Therefore, the sum of the next ℓ_r bundles — each of which is from level r — must be at least

$$\frac{\ell_r b^{k-r}}{c(b^{k-r} + \ldots + 1)} - (a_0 + \ldots + a_{r-1}).$$

Hence one of these ℓ_r bundles must have the property that the sum of the rates assigned to paths in it in $\tilde{X}$ is at least

$$\frac{b^{k-r}}{c(b^{k-r} + \ldots + 1)} - \frac{a_0 + \ldots + a_{r-1}}{\ell_r}.$$

Now, the bundles in $\tilde{X}$ are arranged in ascending order of their total sums, and so we have the following fact: For at least $a_r - \ell_r$ bundles β in level r, the sum of the rates assigned to β in $\tilde{X}$ is at least

$$\frac{b^{k-r}}{c(b^{k-r} + \ldots + 1)} - \frac{a_0 + \ldots + a_{r-1}}{\ell_r}. \tag{1}$$

Call a bundle in level $r > 0$ *bad* if it doesn't satisfy the bound in (1); hence there are at most ℓ_r bad bundles in level r. The total number of edges covered by bad bundles of all levels is at most

$$\sum_{r=1}^{k} \frac{\ell_r a_k}{a_r} = \sum_{r=1}^{k} d^{2k-1} < d^{2k} = n,$$

and so there is at least one edge e of G that is not contained in any bad bundle.

Applying the capacity constraint to this edge e, we have

$$1 \quad \geq \quad \sum_{r=0}^{k} \left(\frac{b^{k-r}}{c(b^{k-r} + \ldots + 1)} - \frac{a_0 + \ldots + a_{r-1}}{\ell_r} \right)$$

```
(1) Initialize:     J_0 ← J.      t ← 0.
    (J_0 denotes the jobs that haven't been processed yet; t denotes current
time).
(2) While J_0 ≠ ∅ do
    (i) Let j ∈ J_0 be a minimal job under ≺_{J_0} such that p({j}, t) is minimum.
    (ii) If p({j}, t) > t,
          (a) then define J' = {j}.
       Else
          (b) Find a subset J' ⊆ J_0 such that J' is independent under ≺_{J−J_0},
                and p(J', t) ≤ t and |J'| is maximum.
                (Note that there will always exist such a J' because
                 {j} satisfies these conditions.)
     (iii) Schedule jobs in J'  starting at time t such that
             they finish processing by time t + p(J', t).
     (iv) J_0 ← J_0 − J'.      t ← t + p(J', t).
     Endwhile
```

Figure 1: Algorithm which shows $c^* \leq 4$ for $P|prec, r_j|all$.

$$
\begin{aligned}
&\geq \sum_{r=0}^{k}\left(\frac{b-1}{cb}-\frac{ra_{r-1}}{\ell_r}\right)=\sum_{r=0}^{k}\left(\frac{b-1}{cb}-\frac{r}{d}\right)\\
&\geq \frac{(k+1)(b-1)}{cb}-\frac{(k+1)^2}{d}\\
&\geq \frac{(k+1)k}{\frac{k}{3}(k+1)}-\frac{(k+1)^2}{(k+1)^2}=3-1=2.
\end{aligned}
$$

This contradiction completes the proof. ■

4 Scheduling

We first consider precedence-constrained scheduling with release dates on m identical parallel machines ($P|prec, r_j|all$). Thus, we are given a set of jobs $J = \{j_1, \ldots, j_n\}$, and a partial order $\prec_J$ on the jobs indicating precedence constraints. Job j_i has processing time p_i and release date r_i. We say that j_i "precedes" j_ℓ if $j_i \prec_J j_\ell$, and we say j_i is *minimal* if no job precedes it. For any subset $J' \subseteq J$, we can restrict the partial order to just the jobs in J'; we denote this by $\prec_{J'}$. We say that a set of jobs $K \subseteq J'$ is *independent* under $\prec_{J'}$ if there is no element in $J' \setminus K$ that precedes an element in K in the ordering $\prec_{J'}$.

Theorem 4.1 $c^*(\mathcal{I}) \leq 4$ *for all instances* $\mathcal{I}$ *of precedence-constrained scheduling with release dates on* m *identical parallel machines* ($P|prec, r_j|all$), *for any* $m \geq 1$.

Proof. Given a subset J' of jobs and a time t, define $p(J', t)$ as the minimum amount of time needed to process all jobs in J' on the m parallel machines (assuming there are no other jobs being processed on any machine) starting at time t and under precedence constraints given by $\prec_{J'}$.

We give the scheduling algorithm in Figure 1. Let S denote the schedule it produces, and X the vector of completion times of jobs under S. We write $\overrightarrow{X} = (C_1, C_2, \ldots, C_n)$. Suppose the `While` loop runs for r iterations. Let A_i denote the set J' selected in step 2 of iteration i, and let $n_i = |A_i|$. We define t_i as the time at which the jobs in A_i begin processing; we define t_{r+1} as the earliest time by which all jobs have finished processing.

Now consider some other schedule S' for this instance, let X' denote the vector of completion times of jobs under S', and write $\overrightarrow{X'} = (C'_1, C'_2, \ldots, C'_n)$. We focus on a particular coordinate k, and let B' denote the first k jobs to finish under the schedule S'.

We wish to show that $C_k \leq 4C'_k$. First observe that $n_1 = 1$, and A_1 consists of the minimal job j for which $p(\{j\}, 0)$ is minimum. Thus, $C_1 \leq C'_1$. For $k > 1$, we choose i so that $n_1 + \cdots + n_i < k \leq n_1 + \cdots + n_{i+1}$. Since $n_1 + \cdots + n_{i+1} \geq k$ jobs finish by time t_{i+2}, we have $C_k \leq t_{i+2}$; thus, it is enough to show that $t_{i+2} \leq 4C'_k$.

If $t_{i+2} \leq C'_k$ we are done. So let us assume that there exists an $s < i + 2$ such that $t_s \leq C'_k < t_{s+1}$. Recall that A_s is scheduled between time t_s and t_{s+1} in schedule S.

We first claim that $t_{s+1} \leq 2C'_k$. We prove this by considering two possible cases. First, suppose that A_s was chosen in case 2(ii)(a); so $A_s = \{j_s\}$. Now, since $n_1 + \cdots + n_{s-1} \leq n_1 + \cdots + n_i < k$, the set $B'' = B' \setminus (A_1 \cup \cdots \cup A_{s-1})$ is non-empty; consider any job j in this set that is minimal under $\prec_{B''}$. Note that $p(\{j\}, t_s) \leq p(\{j\}, 0)$, and $p(\{j\}, 0) \leq C'_k$, since $j \in B'$. By our choice of j_s, we have $p(\{j_s\}, t_s) \leq p(\{j\}, t_s) \leq C'_k$, and hence $t_{s+1} = t_s + p(\{j_s\}, t_s) \leq 2C'_k$. Alternately, suppose that the set A_s was chosen in case 2(ii)(b); then $p(A_s, t_s) \leq t_s$. Hence $t_{s+1} = t_s + p(A_s, t_s) \leq 2t_s \leq 2C'_k$. This completes the proof that $t_{s+1} \leq 2C'_k$.

Now, if $s = i + 1$, then $t_{i+2} \leq 2C'_k$ and we are done. So assume that $s < i + 1$. Define $J' = B' \setminus (A_1 \cup \cdots \cup A_s)$; note that J' is non-empty since $s \leq i$, and $n_1 + \cdots + n_i < k$. Now, $p(J', t_{s+1}) \leq p(J', 0) \leq p(B', 0) \leq C'_k \leq t_{s+1}$.

Hence the set A_{s+1} must have been selected in case 2(ii)(b), since for any minimal $j \in J'$, we have $p(\{j\}, t_{s+1}) \leq C'_k \leq t_{s+1}$. Moreover, J' is independent under $\prec_{J \setminus (A_1 \cup \cdots A_s)}$, and so the set A_{s+1} that is selected satisfies $|A_{s+1}| \geq |J'|$. Consequently, $n_1 + n_2 + \cdots + n_{s+1} \geq k$, and so we have $s \geq i$. But we already assumed $s \leq i$, and so $s = i$. Since A_{s+1} was selected in case 2(ii)(b), we have

$$t_{i+2} = t_{s+2} = t_{s+1} + p(A_{s+1}, t_{s+1}) \leq 2t_{s+1} \leq 4C'_k,$$

and the theorem follows. ■

The construction in the proof of Theorem 4.1 does not yield a polynomial-time algorithm, since computing J' in step 2(ii)(b) is an NP-hard problem. In fact, we can show that efficiently producing a schedule S satisfying $c(X_S) = O(1)$ for this problem would yield a greatly improved approximation algorithm for the *densest subgraph problem* described in the introduction: given a graph G and a number k, find an induced k-node subgraph with the maximum possible number of edges.

Theorem 4.2 *Suppose there exists a polynomial-time algorithm $\mathcal{A}$ that, for any instance of precedence-constrained 1-processor scheduling ($1|prec|all$), produces a schedule S satisfying $c(X_S) \leq \alpha$. Then there exists a polynomial-time $8\alpha^2$-approximation algorithm for the densest-subgraph problem.*

We now consider the special case of precedence-constrained scheduling in which there are two identical machines, and all jobs have unit size ($2|p_j = 1, prec|all$).

Theorem 4.3 *$c^*(\mathcal{I}) = 1$ for every instance $\mathcal{I}$ of precedence-constrained two-processor scheduling with unit jobs ($2|p_j = 1, prec|all$).*

Proof. For a schedule S, define $X(S)$ to be its vector of completion times. We will only consider schedules that place at least one job in each time slot until all jobs are completed. Define $Z(S)$ to be its vector of *gaps*, in ascending order; these are the time slots in which only one job is scheduled.

Here is a basic property of the vectors of gaps.

(1) *Let S and S' be two schedules. $\overrightarrow{X(S)} \not\prec_c \overrightarrow{X(S')}$ if and only if at least one of the following two conditions holds: (i) $Z(S')$ has fewer coordinates than $Z(S)$; or (ii) there is an index i so that $Z(S')_i > Z(S)_i$.*

We define the following two orderings on vectors.

(2) *Suppose $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_m)$ are two vectors. We say that A is lexicographically greater than B if either of the following hold:*
(i) $n < m$ and $a_1 = b_1, \ldots, a_n = b_n$.
(ii) There is an index i such that $a_1 = b_1, \ldots, a_{i-1} = b_{i-1}$ and $a_i > b_i$.

(3) *Let A and B be vectors defined as above. We say that A is greater than B in the colex ordering if either of the following hold:*
(i) $n < m$
(ii) $n = m$ and there exists an index i such that $a_m = b_m, \ldots, a_{i+1} = b_{i+1}$ and $a_i > b_i$.

Let S be a solution to $\mathcal{I}$ with lexicographically maximum vector $Z(S)$ (i.e., the first gap occurs as late as possible as possible, subject to this, the second gap occurs as late as possible, and so on). We show that $\overrightarrow{X(S)} \prec_c \overrightarrow{X(S')}$ for every schedule S'.

Suppose this is not true, and let Ω denote the set of S' for which $\overrightarrow{X(S)} \not\prec_c \overrightarrow{X(S')}$. Thus, every schedule $S' \in \Omega$ satisfies one of conditions (i) or (ii) in (1) .

Let S' be the schedule in Ω whose vector $Z(S')$ is maximum in the colex ordering. We write $Z(S) = (g_1, \ldots, g_m)$ and $Z(S') = (g'_1, \ldots, g'_{m'})$. We know that $Z(S)$ is greater than $Z(S')$ in lexicographic order; but if the first condition in (2) were to hold, this would contradict (1) . Thus there is an index i so that $g_1 = g'_1, \ldots, g_{i-1} = g'_{i-1}$ and $g_i > g'_i$.

Based on the precedence relation, we define the following undirected graph G_f on the set of jobs: (j, j') is an edge if and only if neither of j or j' precedes the other. The following result is due to Fujii et al. [6]

(4) *Suppose G_f has a perfect matching. Then all jobs can be scheduled with no gaps.*

For our analysis, we create $i-1$ fictitious jobs $k_1, \ldots, k_{i-1}$, which are involved in no precedence relations. We consider the schedules that result from augmenting S and S' with the jobs $k_1, \ldots, k_{i-1}$ placed in gap positions $g_1 = g'_1, \ldots, g_{i-1} = g'_{i-1}$ respectively.

Define A to be the set of jobs which start before time g_i in S. Define B to be the set of jobs which start at time g'_i or before in S'. Let x denote the job in B which is run at time g'_i. In both cases, we include the fictitious jobs, so A and $B \setminus \{x\}$ are each scheduled with no gaps. Note that $|A| > |B|$.

Now, there exists a matching M in G_f covering the set A, and there exists a matching M' in G_f covering the set $B \setminus \{x\}$. Let G'_f denote the subgraph of G_f on nodes $A \cup B$ and edges $M \cup M'$. All vertices of G'_f have degree at most 2, and x has degree 0 or 1 depending on whether it belongs to A. Thus, the component of G'_f containing x is a path P; note that P may be a single-node path if $x \notin A$. Let P consist of nodes $x = v_0, \ldots, v_r$ in order, with $e_i = (v_i, v_{i+1})$.

We now consider two possible cases.

(a) Suppose that P has an odd number of nodes, so that v_r is not matched in M. Hence $v_r \in B \setminus A$. Consider the set of jobs in $A \setminus B$, and let y be a minimal element of this set. We claim that (v_r, y) is an edge of G_f. Indeed, y occurs before v_r in the schedule S, and v_r occurs before y in the schedule S'. We define the matching M'' to consist of $M' \setminus E(P)$, together with the edges $(v_0, v_1), (v_2, v_3), \ldots, (v_r, y)$. Note that M'' covers $B \cup \{y\}$. Also, there cannot be a pair of jobs j, j' so that $j \notin B \cup \{y\}$, $j' \in B \cup \{y\}$, and j precedes j'. For if there were, then $j' = y$ — since B was an initial set of jobs in S' — and this contradicts the minimality of y among the set of jobs in $A \setminus B$.

(b) Suppose that P has an even number of nodes, so that v_r is matched in M but not in M'; thus, $v_r \in A \setminus B$. Among all jobs in $A \setminus B$ that precede v_r, including v_r, choose a minimal job y. (Thus we may have $y = v_r$.) We claim that (v_{r-1}, y) is an edge of G_f. For if v_{r-1} preceded y, then it would precede v_r, which is not the case; and we know that y cannot precede v_{r-1} since v_{r-1} precedes y in S'. We define the matching M'' to consist of $M' \setminus E(P)$, together with the edges $(v_0, v_1), (v_2, v_3), \ldots, (v_{r-1}, y)$. M'' covers $B \cup \{y\}$, and as in (a) there cannot a pair of jobs j, j' so that $j \notin B \cup \{y\}$, $j' \in B \cup \{y\}$, and j precedes j'.

So in both cases, we have a job $y \in A \setminus B$ so that no job outside $B \cup \{y\}$ precedes any job in $B \cup \{y\}$, and there is a matching in G_f that covers $B \cup \{y\}$. We construct a new schedule S'' as follows. Using (4) , we schedule all the jobs in $B \cup \{y\}$ initially with no gaps. We then schedule the rest of the jobs as in S', replacing fictitious jobs by gaps, and introducing a gap where y was in S'. (If this results in a time slot having two gaps, we slide all later jobs forward.)

We write $Z(S'') = (g''_1, \ldots, g''_{m''})$. We can view S'' as having been constructed from S' as follows: we first re-arrange the order of jobs in B, within the first g'_i time slots (eliminating fictitious jobs); we then move y forward; we then eliminate any resulting time slots with two gaps. The first of these steps only affects the first i gaps; the second step pushes the i^{th} gap later; and the third step does not push any gap earlier (and may lead to fewer gaps). Thus we have

(5) *$m'' \leq m'$, $g''_i > g'_i$, $g''_j \geq g'_j$ for $i+1 \leq j \leq m''$, and hence $Z(S'')$ is greater than $Z(S')$ in the colex ordering.*

Now, since $S' \in \Omega$, we know that $Z(S')$ is shorter than $Z(S)$, or there is a j so that $g'_j > g_j$. In the former case, we see that $Z(S'')$ is also shorter than $Z(S)$. In the latter case, we know that $j > i$, since $g_1 = g'_1, \ldots, g_{i-1} = g'_{i-1}$ and $g_i > g'_i$. Now, either $j > m''$, in which case $Z(S'')$ is again shorter than $Z(S)$, or else by (5) it follows that $g''_j \geq g'_j > g_j$. Thus $Z(S'')$ satisfies one of conditions (i) or (ii) in (1), and so we have

(6) $S'' \in \Omega$.

But (5) and (6) contradict our choice of S', which establishes that Ω must be empty. We have therefore proved the theorem. ■

The situation changes for precedence-constrained scheduling with unit jobs ($P|p_j = 1, prec|all$) when we have $m \geq 3$ machines. First, let $m = 3$ and consider an instance $\mathcal{I}$ with eight jobs and the following precedence relations: $j_1 \prec j_2 \prec j_3$, and $j_a \prec j_b$ for $a \in \{4, 5, 6\}$ and $b \in \{7, 8\}$. Then the only schedule with makespan 3 leaves a machine idle in the second time slot; while there is a schedule with makespan 4 that completes six jobs in the first two time slots. Thus $c^*(\mathcal{I}) = 4/3$ for this instance. Similar examples can be constructed for larger numbers of machines.

However, it is easy to show that the following greedy algorithm provides a global 2-approximation for every instance: Iteratively, in each time slot, schedule as many currently minimal jobs as possible (up to m). Thus we have

Theorem 4.4 *$c^*(\mathcal{I}) \leq 2$ for every instance of precedence-constrained scheduling with unit jobs ($P|p_j = 1, prec|all$).*

5 Facility Location

One Facility. Recall that in the facility location problems we consider, we have a metric space M, a set $P = \{p_1, \ldots, p_n\}$ of *demand points* in M, and a set F of n' *candidate facility points*. First, we consider the case of 1-facility instances. For each point $f \in F$, let X_f denote the distance vector that arises when the facility is placed at f.

Theorem 5.1 *For every 1-facility instance, there is a point $f^* \in F$ such that $s(X_{f^*}) \leq 4$.*

Proof. For simplicity, we assume n is even. We define the radius of $P' \subseteq P$ to be $\mathrm{rad}(P') = \min_{f \in F} \max_{p' \in P} d(f, p')$. We choose a set $P' \subseteq P$ of $n/2$ points of minimum radius r, and define $f^* \in F$ to be a point for which $\max_{p' \in P} d(f^*, p') = \mathrm{rad}(P')$.

Let $\overleftarrow{X_{f^*}} = (d_1, d_2, \ldots, d_n)$. Now, consider any facility $f' \in F$; let $\overleftarrow{X_{f'}} = (d'_1, d'_2, \ldots, d'_n)$. We must show that for each i, we have $d_1 + \cdots + d_i \leq 4(d'_1 + \cdots + d'_i)$.

We observe that $d'_{n/2} \geq r$ since P' has the minimum radius r of any set of $n/2$ points, and f^* achieves this radius. Also, consider the set Q' consisting of the $i - 1$ points at maximum distance from f', and the set Q^* consisting of the i points at maximum distance from f^*. There is at least one point $q \in Q^* \setminus Q'$, and we have

$$d_i \leq d(f^*, q) \leq d(f^*, f') + d(f', q) \leq d(f^*, f') + d'_i,$$

where the first inequality follows from the fact that $q \in Q^*$, and the third inequality follows since $q \notin Q'$.

If $p' \in P'$, then

$$r + d(p', f') \geq d(f^*, p') + d(p', f') \geq d(f^*, f'),$$

and so $d(p, f') \geq d(f^*, f') - r$. Since there are at least $n/2$ such points, it follows that for any index $i \leq n/2$, we have $d'_i \geq d(f^*, f') - r$. Hence $d(f^*, f') \leq d'_i + r \leq 2d'_i$, and so $d_i \leq d'_i + d(f', f^*) \leq 3d'_i$. For an index $i > n/2$, we have

$$\begin{aligned} d_1 + \cdots d_i &\leq d_1 + \cdots + d_{n/2} + (i - n/2)r \\ &\leq 3(d'_1 + \cdots + d'_{n/2}) + (d'_1 + \cdots + d'_{n/2}) \\ &\leq 4(d'_1 + \cdots + d'_i). \qquad \blacksquare \end{aligned}$$

For the measure c^*, we have the following complementary pair of results for 1-facility instances. For the first, we construct a set of $O(\log n)$ facilities by iteratively identifying sets of size $1, 2, 4, 8, \ldots$ of minimum radius, and including one facility for each of these sets that achieves this radius.

Theorem 5.2 *Let $\mathcal{I}$ be an arbitrary 1-facility instance. There exists a set $F^* \subseteq F$ of size at most $O(\log n)$ so that $c(X_{F^*}) \leq 3$.*

Proof. We construct the set F^* as follows. Initialize the set S_0 to the set of all demand points P. We inductively define sets S_i, $i > 0$, and sets A_i, $i \geq 0$, as follows : A_i is a set of 2^i points from S_i of minimum radius. Let f_i be the facility which achieves the radius of A_i. Define S_i as $S_{i-1} \setminus A_{i-1}$.

We stop this process when some set $S_i = \phi$. This defines $m = O(\log n)$ sets A_i total. Define $F^* = \{f_0, f_1, \ldots, f_m\}$. Note that the last set A_m may have less than 2^m points. Also, note that in the distance vector X_{F^*}, points in A_i are not necessarily assigned to f_i, but to the closest facility in F^*; this observation is important in the proof.

Now, consider any facility point $f \in F$, and let us label the points in P so that $d(f, p_1) \leq d(f, p_2) \leq \cdots \leq d(f, p_n)$. We fix an index t and consider the distance $d = d(f, p_t)$; we must show that at least t coordinates of X_{F^*} are each at most $3d$.

Let $P' = \{p_1, \ldots, p_t\}$; note that since $d(f, p_i) \leq d$ for all $p_i \in P'$, the radius of P' is at most d. We define s so that $2^s \leq t < 2^{s+1}$, and consider the following two cases.

First, suppose $P' \cap A_i = \phi$ for each i, $0 \leq i \leq s$. Now, A_i has minimum radius of all subsets of size 2^i in S_i; since we have $P' \subseteq S_i$ and $2^i \leq t$, it follows that A_i has radius at most d. Thus, all the coordinates in X_{F^*} corresponding to points in A_i have value at most d. Hence at least $|A_0| + \ldots + |A_s| = 2^{s+1} - 1 \geq t$ coordinates of X_{F^*} have value at most d.

Now, suppose instead that there is an index $i \leq s$ for which $P' \cap A_i \neq \phi$; in this case, choose the smallest such i. Let p' be a point in $P' \cap A_i$. Since P' does not intersect $A_0 \cup \cdots \cup A_{i-1}$, it is a subset of S_i; so the radius of A_i is at most the radius of P', and hence at most d. Now, consider the vector X_{f_i}; since $f_i \in F^*$, we have $X_{F^*} \prec_c X_{f_i}$, and so it is enough to show that X_{f_i} contains t coordinates of value at most $3d$. But for any $p'' \in P'$, we have

$$d(p'', f_i) \leq d(p'', f) + d(f, p') + d(p', f_i) \leq 3d. \qquad \blacksquare$$

We have an essentially matching lower bound, which has a structure paralleling the construction that proves the previous theorem; essentially we build an instance consisting of widely separated clusters of size $1, 2, 4, 8, \ldots$; by defining distances appropriately in each, we can argue that each cluster must contain a facility.

Theorem 5.3 *For any number α, there are 1-facility instances $\mathcal{I}$ for which no set F' of fewer than $\log n$ facilities achieves $c(X_{F'}) \le \alpha$.*

Multiple Facilities. Now we consider a k-facility instance $\mathcal{I}$, where $k > 1$ is arbitrary.

Theorem 5.4 *Fix numbers k and ε such that $\varepsilon \le 1$. There exists a set $F^* \subseteq F$ of size $O(k \log n + \varepsilon^{-1})$ so that $\overleftarrow{X_{F^*}} \prec_s (3+\varepsilon)\overleftarrow{X_{F'}}$ for every set F' of at most k facilities.*

Proof. We first describe the construction of the set F^* that achieves this bound. For a point p and a set of points Q, we define $d(p, Q) = \min_{q \in Q} d(p, q)$. We define the *$k$-radius* of a set of demand points P' as follows:

$$\mathrm{rad}_k(P') = \min_{|F'|=k} \max_{p \in P'} d(p, F').$$

We choose a sufficiently large constant $\alpha = \Theta(\varepsilon^{-1})$, and perform the following construction. Define $x_0 = n$. Let $P_1 = P$ be the initial set of demand points, $n_1 = |P_1|$, $x_1 = \lceil n_1/2 \rceil$, and C_1 a subset of x_1 points of P_1 of minimum k-radius. In general, having defined P_i, n_i, x_i, C_i, we set $P_{i+1} = P_i \setminus C_i$, $n_{i+1} = |P_{i+1}|$, $x_{i+1} = \lceil n_{i+1}/2 \rceil$, and C_{i+1} a subset of x_{i+1} points of P_{i+1} of minimum k-radius. Let r_{i+1} denote the k-radius of C_{i+1}.

We stop this process at the first s for which $x_s \le \alpha$. Let $R \subseteq P$ denote the set of demand points not contained in $C_1 \cup \cdots \cup C_s$; note that $|R| \le \alpha$. We define our set F^* of facilities as follows. For each set C_i, we include a set of k facilities that achieves the k-radius of C_i. For each point $p \in R$, we include a facility that is closest to p.

Note that $s = O(\log n)$, and hence $|F^*| = O(k \log n + \alpha)$. Now, consider any set F' of k facilities. We write $\overleftarrow{X_{F^*}} = (d_1, d_2, \ldots, d_n)$ and $\overleftarrow{X_{F'}} = (d'_1, d'_2, \ldots, d'_n)$.

The remainder of the proof consists of showing that for each j, we have $d_1 + \cdots + d_j \le (3+\varepsilon)(d'_1 + \cdots + d'_j)$. We do this by a sequence of claims, beginning with

(1) *For each i, $1 \le i \le s$, we have $d'_{x_i} \ge r_i$.*

For suppose this is not true for some i. Then there exists a set Q of at least $n - x_i + 1$ demand points of k-radius less than r_i. Since

$$\begin{aligned} &|Q \setminus (C_1 \cup \cdots \cup C_{i-1})| \\ &\quad \ge (n - x_i + 1) - (x_1 + \cdots + x_{i-1}) \\ &\quad = n_i - x_i + 1 \ge x_i, \end{aligned}$$

P_i contains a set Q' of at least x_i points from Q. But the k-radius of Q' is less than r_i, which contradicts our choice of C_i. This proves (1).

Now, consider an index i, For any $j < i$, we have $x_j \ge x_i$, and so $d'_{x_i} \ge d'_{x_j} \ge r_j$. Thus we have

(2) $d'_{x_i} \geq \max(r_1, \ldots, r_i)$.

Using this, we show

(3) *For each $j \leq x_s$, we have $d_j \leq d'_j$.*

We consider two cases in proving (3). First, if $d_j \leq \max(r_1, \ldots, r_s)$, then $d_j \leq d'_j$, since (2) implies that $d'_j \geq d'_{x_s} \geq \max(r_1, \ldots, r_s)$. But if $d_j > \max(r_1, \ldots, r_s)$, then each of $d_1, \ldots, d_j > \max(r_1, \ldots, r_s)$, and hence the points achieving these distances must belong to R. For each of these points, their coordinate in $X_{F'}$ is at least as large as their coordinate in X_{F^*}. It follows that the vector $\overleftarrow{X_{F'}}$ contains at least j coordinates of value at least d_j, and hence $d_j \leq d'_j$.

To handle boundary cases, we define $x_0 = n$, $x_{s+1} = 0$, $C_{s+1} = \phi$, and $n_{s+1} = n_s - x_s$. We now claim the following.

(4) *Suppose j is an integer satisfying $x_i < j \leq x_{i-1}$, where $1 \leq i \leq s$. Then $d_j \leq d'_{x_i}$.*

To prove this, we observe that

$$\begin{aligned} &|C_{i+1}| + |C_{i+2}| + \cdots + |C_s| + |R| \\ &\quad = x_{i+1} + \cdots x_s + \alpha \\ &\quad = n_{i+1} = n_i - x_i \leq x_i. \end{aligned}$$

Now, $j > x_i$, so among the points with the j largest coordinates in X_{F^*}, there is at least one point that does not belong to $C_{i+1} \cup \cdots C_s \cup R$. Thus, $d_j \leq \max(r_1, \ldots, r_i)$. But by (2), $d'_{x_i} \geq \max(r_1, \ldots, r_i)$, and so $d_j \leq d'_{x_i}$, as desired.

We now complete the proof. First, if α is a sufficiently large multiple of ε^{-1}, then $x_{i-1} - x_i \leq (2 + \varepsilon)(x_i - x_{i+1})$ for each i, $1 \leq i \leq s$. Now, for each such i, we have

$$\begin{aligned} \sum_{x_i < \ell \leq x_{i-1}} d_\ell &\leq (x_{i-1} - x_i) d'_{x_i} \leq (2 + \varepsilon)(x_i - x_{i+1}) d'_{x_i} \\ &\leq (2 + \varepsilon) \sum_{x_{i+1} < \ell \leq x_i} d'_\ell. \end{aligned}$$

Now, let j be any index between 1 and n. For $j \leq x_s$, we have $d_1 + \cdots d_j \leq d'_1 + \cdots + d'_j$ by (3). Otherwise, there is an i so that $x_i < j \leq x_{i-1}$.

$$\begin{aligned} &d_1 + \cdots + d_j \\ &\quad = \sum_{1 \leq \ell \leq x_s} d_\ell + \sum_{x_s < \ell \leq x_{s-1}} d_\ell + \cdots \\ &\qquad\qquad + \sum_{x_{i+1} < \ell \leq x_i} d_\ell + \sum_{x_i < \ell \leq j} d_\ell \\ &\quad \leq \sum_{1 \leq \ell \leq x_s} d'_\ell + (2 + \varepsilon) \left(\sum_{1 \leq \ell \leq x_s} d'_\ell + \cdots + \sum_{x_{i+1} < \ell \leq x_i} d'_\ell \right) \\ &\quad \leq (3 + \varepsilon)(d'_1 + \cdots + d'_{x_i}) \\ &\quad \leq (3 + \varepsilon)(d'_1 + \cdots + d'_j). \qquad \blacksquare \end{aligned}$$

Note that the construction described in the proof does not yield a polynomial-time algorithm since it involves computing the k-radius of subsets of points. However, a recent result of Charikar, Khuller, Mount, and Narasimhan [3] provides a polynomial-time 3-approximation algorithm for the following problem: given a set P and a parameter x, find a subset C of size x of minimum k-radius. We can use this algorithm to construct the sets C_i in the proof; as a result we lose a factor of 3 in each of claims (1)-(4) above, and hence obtain a set F^* of size $O(k \log n + \varepsilon^{-1})$ for which $s(F^*) \leq 9 + \varepsilon$.

We have a lower bound that nearly matches the result of Theorem 5.4;

Theorem 5.5 *For any number α, there are k-facility instances $\mathcal{I}$ for which no set F' of fewer than $\Omega\left(k\frac{\log(n/k)}{\log \alpha}\right)$ facilities achieves $s(X_{F'}) \leq \alpha$.*

For the c^* measure, we show that in general the lower bound is essentially as bad as possible when $k > 1$.

Theorem 5.6 *For any number α, there are 2-facility instances $\mathcal{I}$ for which no proper subset $F' \subseteq F$ achieves $c(X_{F'}) \leq \alpha$.*

References

[1] D. Bertsekas, R. Gallager, *Data Networks*, Prentice-Hall, 1987.

[2] A. Blum, P. Chalasani, D. Coppersmith, W. Pulleyback, P. Raghavan, M. Sudan, "The minimum latency problem," *Proc. 26th ACM STOC*, 1994.

[3] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan "Algorithms for facility location problems with outliers," submitted for publication, 2000.

[4] E.G. Coffman, R.L. Graham, "Optimal scheduling for two-processor systems," *Acta Informatica* 1(1972), pp. 200-213.

[5] H. Eiselt, G. Laporte, "Objectives in location problems," in *Facility Location: A Survey of Applications and Methods*, Z. Drezner, Ed., Springer, 1995.

[6] M. Fujii, T. Kasami, K. Ninomiya, "Optimal sequencing of two equivalent processors," *SIAM J. Appl. Math.* 17(1969), pp. 784-789.

[7] A. Goel, A. Meyerson, S. Plotkin, "Combining fairness with throughput: Online routing with multiple objectives," *Proc. 32nd ACM STOC*, 2000.

[8] M. Goemans, J. Kleinberg, "An improved approximation ratio for the minimum latency problem," *Proc. SIAM-ACM SODA*, 1996.

[9] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, "Optimization and approximation in determinsitic sequencing and scheduling: A survey," *Ann. Discrete Math.* 5(1979), pp. 287–326.

[10] L. A. Hall, A. Schulz, D. B. Shmoys, J. Wein, "Scheduling to minimize average completion time: off-line and on-line algorithms," *Mathematics of Operations Research* 22(1997), pp. 513-544.

[11] H. Hayden, *Voice Flow Control in Integrated Packet Networks*, Master's thesis, MIT Dept. of EECS, Cambridge MA, 1981.

[12] J. Jaffe, "Bottleneck flow control," *IEEE Trans. Communication*, 29(1981), p.p 954–962.

[13] J. Kleinberg, Y. Rabani, E. Tardos, "Fairness in routing and load balancing," *Proc. 40th IEEE FOCS*, 1999.

[14] G. Kortsarz, D. Peleg. "On choosing a dense subgraph," *Proc. 34th IEEE FOCS*, 1993.

[15] E. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D. Shmoys, "Sequencing and scheduling: Algorithms and complexity," in *Handbooks in Operations Research and Management Science, vol. 4*, North-Holland, 1993.

[16] M. Marsh, D. Schilling, "Equity measurement in facility location analysis: A framework and analysis," *European J. Oper. Res.* 74(1994), pp. 1-17.

[17] N. Megiddo, "Optimal flows in networks with sources and sinks," *Math Programming* 7(1974).

[18] V. Pareto, *Cours d'Economie Politique* (Lausanne, Switzerland) 1897.

[19] C. Phillips, C. Stein, J. Wein, "Scheduling jobs that arrive over time," *Proceedings of the Fourth Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science 955*, Springer, Berlin, 86-97, 1995.

[20] W. Smith, "Various optimizers for single-stage production," *Naval Res. Logist. Quart.* 3(1956).

[21] C. Stein, J. Wein, "On the existence of schedules that are near-optimal for both makespan and total weighted completion time," *Operations Research Letters* 21(1997).