Efficient algorithms for a scheduling problem and its applications to illicit drug market crackdowns

Pu Cai* Jin-yi Cai† Ashish V. Naik‡

Abstract

We give polynomial time algorithms for a job scheduling problem. By duality we transform a special case of the drug market crackdown scheduling problem to the above job scheduling problem and thus derive polynomial time algorithms to the second problem. Finally, using the algorithm for the special case, we develop a quasipolynomial time approximation algorithm for the general case of the drug market crackdown scheduling problem with monomial cost functions.

key words: job scheduling problem, drug market crackdown scheduling problem, duality, polynomial time algorithm, quasipolynomial time approximation algorithm.

1 Introduction

Let us first consider the following *Job Scheduling* problem. Suppose we are given two ordered sets of jobs to be processed: $\mathcal{A} = \{A_1, A_2, \ldots, A_L\}$ and $\mathcal{B} = \{B_1, B_2, \ldots, B_M\}$. The following constraints are imposed. Each job has a processing cost which is dependent on the starting time t. The cost function for each job as a function of starting time t is monotonically increasing, the rate of which also depends on the specific job. This increase in cost can reflect a variety of causes, e.g., job utility or processor deterioration. Each job A_i requires a time units to finish and each job B_j requires b time units to finish. All jobs have to be processed sequentially starting at time 0, i.e., only one job can be processed at a given moment. Furthermore, jobs in each set $\mathcal{A}$ and $\mathcal{B}$ should be processed in the given order. The objective is to schedule all the jobs in $\mathcal{A} \cup \mathcal{B}$ so that the total cost is minimized.

In this paper we develop a polynomial time algorithm to solve this problem optimally. Later we will generalize the problem in a number of ways, and provide efficient algorithms for the generalized problems.

*Computer Science Department, State University of New York at Buffalo, Buffalo, NY 14260. Research supported in part by NSF grants CCR 9057486 and CCR 9319093. email: pucai@cs.buffalo.edu

†Computer Science Department, State University of New York at Buffalo, Buffalo, NY 14260. Research supported in part by NSF grants CCR 9057486 and CCR 9319093, and an Alfred P. Sloan Fellowship. email: cai@cs.buffalo.edu

‡Computer Science Department, University of Chicago, Chicago, IL 60637. Research supported in part by NSF grants CCR 92-53582. email: naik@cs.uchicago.edu

There are many potential applications of our algorithms to real world situations. In particular we consider its application to the problem of how to schedule crackdowns on illicit drug markets, which motivated this study.

Drug abuse can cause significant damage on any society, perhaps most evident from the rising concern in the United States over the increase in drug related crime and violence.

The problem of drug abuse is a complex problem. Traditionally this problem has mostly been explored from a sociological and public policy point of view. In recent times, there has been a growing interest in the operations research community to explore a more quantitative approach to the problem. While the nature of the problems related to drug abuse are national or international in scope, there is a growing view among researchers that some of the most effective counter measures should be implemented locally and specific to the needs of a locality. In recent years U.S. drug control policy has given increasing attention to local drug enforcement. A currently favored local enforcement strategy—*crackdowns*, increases police activity in an area to increase the threat of apprehension.

Most cities, however, have more than one street corner drug market. In this case, theoretical models [4, 2] and practical experience [6] have indicated that a successful crackdown operation on a drug market needs to be followed by a maintenance phase to prevent the drug market from "springing back." The resource requirement for the maintenance phase is typically much less than that of the initial crackdown phase.

Taking these factors into consideration, Naik *et al.* [10] developed an analytical framework for specifying the sequence in which crackdowns should be initiated on several drug markets in a city to minimize the total cost. They formulated the above problem as the *Crackdown Scheduling* problem, which can be described as follows.

There are n drug markets, $D_1, D_2, \ldots, D_n$, all of which need to be brought down to a negligible level of drug activity. Let E represent the total level of resources allocated for this crackdown program. Each market is eliminated by a two step process: a crackdown phase and a maintenance phase. The crackdown phase utilizes all the *available* resources until the market is brought down to the desired level. This is followed by a maintenance phase which uses a significantly smaller amount of resources to maintain the market at this desired level. In general, each drug market D_i has:

1. A response function in terms of the amount of available resource utilized in the crackdown phase. This function denotes the time it takes to crackdown. If X is the amount of *unavailable* resource due to maintenance cost of previously scheduled crackdowns, then the *available* resource is $E - X$. For technical reasons, we will consider this response function as a function of unavailable resource X, denoted by $f_i(X)$. Since an increase in the amount of unavailable resource will necessarily result in an increase in the time needed to bring down the market, f_i is monotonically increasing in X.

2. A maintenance level w_i, which denotes the amount of resource needed to *maintain* market D_i at a negligible level of activity, once it is brought down. We assume no particular relationship between w_i and f_i.

The objective is to find an order of the crackdown schedule Π which minimizes the total time spent in collapsing all n drug markets. Let $\Pi(i)$ denote the ith market scheduled for crackdown, the cost of the schedule Π, denoted by $C(\Pi)$, is given by the following expression

$$C(\Pi) = \sum_{j=1}^{n} f_{\Pi(j)}(\textstyle\sum_{l=1}^{j-1} w_{\Pi(l)}).$$

We want to find a schedule Π of crackdowns such that $C(\Pi)$ is minimized.

Naik *et al.* [10] showed that the crackdown scheduling problem is NP-complete. They also showed that some strongly restricted versions of this problem can be solved in polynomial time. The NP-hardness of the problem assumes that the response function of each market is an arbitrary monotonically increasing function. Lack of reliable data [11] makes estimation of the exact nature of the response function very difficult. However, practical evidence [2, 7] indicates that most real response functions are convex functions, and in fact a simple monomial of the form $f_i(x) = m_i \cdot x^{\beta} + c_i$, with appropriate m_i, c_i, β, provides a good approximation to a convex response function. These might lead to efficient strategies for solving the otherwise intractable crackdown scheduling problem. Hence, it is perhaps more reasonable to consider a restricted version of crackdown scheduling where the response functions are monomials. For the rest of this paper, we will assume that the response function for each market is given by a monomial of the form above.

In addition, it is typically the case in practice that each market is associated with one of a fixed number of levels of maintenance cost (e.g., high or low). For simplicity let's assume there are just two levels of maintenance, costing a and b. Call these two lists of markets $\mathcal{A}'$ and $\mathcal{B}'$ respectively. The ith market in $\mathcal{A}'$ has cost function f_i and the jth market in $\mathcal{B}'$ has cost function g_j.

Now we claim there is a duality between the problem of drug market crackdowns and the job scheduling problem discussed at the beginning of this paper, where the roles of cost and time are reversed.

We can first order all markets in $\mathcal{A}'$ and $\mathcal{B}'$ separately in decreasing order of the coefficients m_i. Then a simple switching argument in Section 2 will show that the optimal ordering of a crackdown schedule will assign each market in $\mathcal{A}'$ in the order given in $\mathcal{A}'$, and similarly for each market in $\mathcal{B}'$. The duality can now be stated as follows: If we identify the unavailable resource X with the time spent in the *Job Scheduling* problem, and identify the total accumulative time spent in cracking down all drug markets with the total cost in the same *Job Scheduling* problem, then they become mathematically identical problems.

2 Description of results

We will first present a polynomial time algorithm for the *Job Scheduling* problem. The design of the algorithm is based on the principles of *dynamic programming*. We will give a careful analysis of the cost of the algorithm.

We then generalize the problem to the case where there are more than two distinct time units a_i, and each job takes exactly one of the a_i to process. Suppose there are k different categories of jobs in terms of the time units. We present an algorithm that runs in time $O(n^k)$, which finds the optimal schedule for n jobs. This is a polynomial time algorithm for a fixed k.

In terms of the drug market *Crackdown Scheduling* problem, we need the following lemma.

Lemma 1 *Let $\mathcal{C} = \{C_1, C_2, \ldots, C_L\}$ and $\mathcal{D} = \{D_1, D_2, \ldots, D_M\}$ be two ordered sets of drug markets, so that each C_i (respectively D_j) costs c (respectively d) to maintain after crackdown. Let $f_i(x)$ (respectively $g_j(x)$) be the response function for markets C_i (respectively D_j). We assume that $\forall i < j$, and $\forall x$,*

$$\frac{df_j(x)}{dx} < \frac{df_i(x)}{dx}$$

and similarly

$$\frac{dg_j(x)}{dx} < \frac{dg_i(x)}{dx}.$$

Then, any optimal scheduling Π of $\mathcal{C} \cup \mathcal{D}$ induces the same orders as given within $\mathcal{C}$ and $\mathcal{D}$.

Proof: The proof is a simple exchange argument. We observe that in any scheduling of $\mathcal{C} \cup \mathcal{D}$, if we exchange a pair of jobs within each list $\mathcal{C}$ or $\mathcal{D}$, since the two jobs within the same list have the same maintenance cost, the amount of available resource for each scheduled job is unchanged, and the only difference in the total time taken is the time associated with the exchanged jobs. Let $C(\Pi)$ be the cost of the optimal scheduling, where, say, we are to crackdown C_i *after* C_j, and $i < j$. Let Π' be the modified scheduling obtained from Π by switching C_i and C_j. Let s (respectively t) be the unavailable resource when market C_i (respectively C_j) is scheduled in Π. Then $s > t$ and

$$\begin{aligned} C(\Pi') - C(\Pi) &= [f_i(t) + f_j(s)] - [f_i(s) + f_j(t)] \\ &= [f_j - f_i]_t^s \\ &= \left.\frac{d[f_j - f_i]}{dt}\right|_\theta \\ &< 0 \end{aligned}$$

for some θ, $t \leq \theta \leq s$. This contradicts the hypothesis that Π is an optimal scheduling.

An identical argument applies to the induced order on the markets in $\mathcal{D}$. $\Box$

Corollary 1 *The lemma applies if the response functions are $f_i(X) = m_i X^\beta + c_i$, for some fixed $\beta \geq 0$ and $m_1 > m_2 > \cdots > m_n$.*

From now on we will assume that the response functions are such that our lemma applies, i.e., that the induced order on the individual list by any optimal scheduling is known (and indexed as such.)

By the duality discussed at the end of Section 1, a polynomial time algorithm to find an optimal scheduling for the *Job Scheduling* problem gives us immediately a polynomial time algorithm to find an optimal solution to the drug market *Crackdown Scheduling* problem with two lists of drug markets. A similar proof can be given for the case of *Crackdown Scheduling* problem where there are more than two maintenance levels. It runs in time $O(n^k)$ for n markets, if there are k maintenance levels.

However, in the case where every drug market has its own maintenance levels, the algorithm indicated above would be of little use. We use the polynomial time algorithm given above to develop a quasipolynomial-time approximation scheme for the general problem. The only requirement is that the maintenance level of each market is less than $p(n)$ for some polynomial p, and n is the number of markets. We show that for all $\epsilon > 0$, there exists an algorithm that runs in $n^{O\left(\frac{\beta}{\log_e(1+\epsilon)} \log n\right)}$-time and outputs a solution that is within $1+\epsilon$ of the optimal solution. To the best of our knowledge, this is the first subexponential-time approximation scheme for the general case of the *Crackdown Scheduling* problem.

In Section 3, we describe our algorithm for the *Job Scheduling* problem with two ordered sets of jobs. In Section 4, we extend the algorithm to any fixed number of ordered sets. In Section 5, we give our approximation algorithm for the general case.

3 The *Job Scheduling* problem with two ordered sets

Let $\mathcal{A} = \{A_1, A_2, \ldots, A_L\}$ and $\mathcal{B} = \{B_1, B_2, \ldots, B_M\}$ be two ordered sets of jobs to be processed. Let A_i (respectively B_j) have cost function f_i (respectively g_j); they are both monotonically increasing. For any schedule Π, if the starting time for job A_i is t then the cost for A_i is $f_i(t)$. Similarly, for job B_j the cost is $g_j(t)$ if it is started at t. Each job A_i (respectively B_j) requires a (respectively b) time units to finish. The objective is to find a sequential schedule of all the jobs in $\mathcal{A} \cup \mathcal{B}$ which merges the two lists, with minimum total cost.

Since all cost functions f_i and g_j are monotonically increasing, any optimal scheduling will not leave the processor idle at any time until all jobs are processed. Thus, to schedule the next job after i jobs from the $\mathcal{A}$ list and j jobs from the $\mathcal{B}$ list are processed, the starting time is $ia + jb$ and cost will be either $f_{i+1}(ia+jb)$ if the next job is from $\mathcal{A}$, or $g_{j+1}(ia+jb)$ if the next job is from $\mathcal{B}$.

Let $F_{i,j}$ be the cost of an optimal scheduling for the partial lists $\{A_1, A_2, \ldots, A_i\}$ and $\{B_1, B_2, \ldots, B_j\}$. Let $N = i + j$, the number of jobs scheduled so far. Then our objective is to find the value $F_{L,M}$. In addition, we can also compute the actual optimal scheduling that gives the value $F_{L,M}$. (This will be clear from the algorithm; for simplicity of its presentation, we will just focus on computing the value $F_{L,M}$ of the optimal scheduling rather than the optimal scheduling itself.)

As mentioned earlier the algorithm is based on the principles of *dynamic programming*. To begin with, we have $F_{00} = 0$, $F_{10} = f_1(0)$ and $F_{01} = g_1(0)$. Now suppose we have computed all the values $F_{i,j}$, with $i + j = k - 1$. We can compute all the values $F_{i,j}$ with

$i+j=k$ as follows: Assume $k \le L+M$. If $k \le L$ then $F_{k,0} = F_{k-1,0} + f_k((k-1)a)$. If $k \le M$ then $F_{0,k} = F_{0,k-1} + g_k((k-1)b)$. For $0 < i \le L$ and $0 < j \le M$ and $i+j=k$, let

$$F_{i,j} = \min \begin{cases} F_{i-1,j} + f_i((i-1)a + jb) \\ F_{i,j-1} + g_j(ia + (j-1)b). \end{cases}$$

This is because the last job for the partial lists $\{A_1, A_2, \ldots, A_i\}$ and $\{B_1, B_2, \ldots, B_j\}$ must be either the last job A_i of the first list or the last job B_j of the second list. In each case the optimal cost of scheduling the previous jobs are known and the starting time for the last job is also known. Thus the minimum of the two must be the optimal cost for scheduling the two partial lists $\{A_1, A_2, \ldots, A_i\}$ and $\{B_1, B_2, \ldots, B_j\}$ together. The algorithm terminates with $k = L+M$.

We now give estimates for the running time of the algorithm. Let $n = L+M$ be the total number of jobs in both lists. We assume that evaluation of the cost functions are easy and takes unit step to compute. There are $L+M$ many levels as k goes from 1 to $L+M$, and for each k there are at most $L+M$ many pairs (i,j) with $i+j=k$, and for each such pair, a constant amount of time is spent to compute the minimum value $F_{i,j}$. Thus the algorithm runs in time $O(n^2)$.

For practical implementation purposes there is a saving of at least a factor of two that can be achieved from the above estimate. This can be best visualized as follows. Consider a grid of lattice points in $\mathbf{Z} \times \mathbf{Z}$ where each point is labeled by (i,j), and $0 \le i \le L$, $0 \le j \le M$. Then it is not hard to see that every computation step in the above algorithm can be associated with a unique lattice point (i,j). Only lattice points within this rectangular grid region will need to be considered. We can precompute all the values $\{ia + jb\}$ for each lattice points (i,j), where $0 \le i \le L$ and $0 \le j \le M$. Then each lattice point on the lower or left boundaries only requires one evaluation of the cost function and one addition of its values. Every other lattice point not on the lower or left boundaries requires two evaluations and two additions of the values of the cost functions, and then one comparison. Thus, the number of values $F_{i,j}$ computed is the same as the number of lattice points $(L+1)(M+1)$. The number of evaluations and additions is the same as the number of edges in this rectangular grid, which is exactly $L(M+1) + (L+1)M$. The number of comparisons is LM. As $n = L+M$, $LM \le n^2/4$, thus we have a better estimate of at most $n^2/2$ for the total work of the algorithm. The more precise estimate $2LM + L + M$ is even better when the difference between L and M is large.

If in the above computation, we labeled the appropriate edge corresponding to the "winner" of the comparison for each lattice point that does not lie on the lower or left boundaries, and labeled the edge toward the origin $(0,0)$ for those that do lie on the lower or left boundaries, then, the unique path from (L,M) back to $(0,0)$ gives us the optimal scheduling itself.

4 The *Job Scheduling* problem with s ordered sets

The results in Section 3 can be readily generalized to the case with s different lists of jobs.

Let $\mathcal{A}_\ell = \{A_{\ell,1}, A_{\ell,2}, \ldots, A_{\ell,L_\ell}\}$, where $\ell = 1, 2, \ldots, s$, be s ordered lists of jobs to be processed. Suppose each job in $\mathcal{A}_\ell$ takes time a_ℓ. Job $A_{\ell,j}$ has monotonically increasing cost function $f_{\ell,j}$.

Let $F_{i_1,i_2,\ldots,i_s}$ be the cost of an optimal scheduling for the partial lists $\{A_{\ell,1}, A_{\ell,2}, \ldots, A_{\ell,i_\ell}\}$, where $0 \le i_\ell \le L_\ell$, and $\ell = 1, 2, \ldots, s$. Then we have $F_{0,0,\ldots,0} = 0$, and,

$$F_{i_1,i_2,\ldots,i_s} = \min \begin{cases} F_{i_1-1,i_2,\ldots,i_s} + f_{1,i_1}((i_1 - 1)a_1 + i_2 a_2 + \ldots, i_s a_s) \\ F_{i_1,i_2-1,\ldots,i_s} + f_{2,i_2}(i_1 a_1 + (i_2 - 1)a_2 + \ldots, i_s a_s) \\ \vdots \\ F_{i_1,i_2,\ldots,i_s-1} + f_{s,i_s}(i_1 a_1 + i_2 a_2 + \ldots, (i_s - 1)a_s). \end{cases}$$

Here, when an index becomes negative it is understood that that particular term disappears in the minimization.

In the algorithm, these values of $F_{i_1,i_2,\ldots,i_s}$ are computed in increasing order of $i_1 + i_2 + \cdots + i_s$. The algorithm terminates at $i_1 + i_2 + \cdots + i_s = L_1 + L_2 + \cdots + L_s$, obtaining $F_{L_1,L_2,\ldots,L_s}$. The correctness of the algorithm is justified exactly as before.

The time estimate of the algorithm can be obtained once again by visualizing each step of the computational process to be associated with a unique lattice point $(i_1, i_2, \ldots, i_s)$ in $\mathbf{Z}^s$, where $0 \le i_\ell \le L_\ell$, and $1 \le \ell \le s$. Thus, the number of values $F_{i_1,i_2,\ldots,i_s}$ computed is $(L_1 + 1) \cdot (L_2 + 1) \cdots (L_s + 1)$. The number of evaluations of the cost functions is exactly the number of edges in the s-dimensional lattice grid region bounded above. The number of comparisons and additions of the cost functions are similarly bounded. The number of edges in this s-dimensional lattice grid region is exactly

$$\left(\prod_{\ell=1}^{s} (L_\ell + 1) \right) \cdot \left[\sum_{\ell=1}^{s} \frac{L_\ell}{L_\ell + 1} \right],$$

which can be easily verified by induction. For $n = \sum_{\ell=1}^{s} L_\ell$, this gives us an estimate of $s \cdot \left(\frac{n}{s}\right)^s$. As before, the more precise estimate above is even better when the L_ℓ are not close to each other.

As mentioned earlier, by the duality between the *Job Scheduling* problem, and the *Crackdown Scheduling* problem, the same algorithm gives a polynomial time solution to the crackdown problem with any fixed number of maintenance levels for the drug markets.

Theorem 1 *The Job Scheduling problem can be solved in polynomial time. The optimal solution can be found in time $O(n^s)$ for n jobs and s ordered lists.*

Theorem 2 *The Crackdown Scheduling problem with a fixed number of maintenance levels for the drug markets can be solved in polynomial time. The optimal solution can be found in time $O(n^s)$ for n jobs and s different maintenance levels.*

5 A Quasipolynomial Approximation Scheme

Consider an instance of the *Crackdown Scheduling* problem where the i^{th} market has a response function $f_i(X) = m_i X^\beta + c_i$, where m_i and c_i depend on the job i, but $\beta > 0$ is a constant. Let p be a polynomial such that the maintenance level of each market is less than $p(n)$, where n is the number of markets. We assume that the maintenance level of each market is greater than or equal to 1. We now give an algorithm that solves the above instance of the *Crackdown Scheduling* problem arbitrarily close to the optimal solution in $n^{O(\log n)}$ time.

Theorem 3 *There exists a quasipolynomial time algorithm, such that, for any given constant $\epsilon > 0$, and any instance of the Crackdown Scheduling problem as above, the algorithm finds an approximate solution that is within $1+\epsilon$ of the optimal, in time $n^{O\left(\frac{\beta}{\log_e(1+\epsilon)}\cdot\log n\right)}$.*

Proof: Let $\epsilon' = \frac{1}{2\beta}\cdot\log(1+\epsilon)$. The main idea is to reduce the given instance I to an instance I' where the maintenance level of each market is one of $O(\frac{1}{\epsilon'}\cdot\log n)$ different values. Then, using the algorithm described in the previous section, this instance can be solved in $n^{O(\log n/\epsilon')}$ time. We then show that the schedule yielded by this algorithm is good enough to approximate within a factor of $1+\epsilon$ of the optimal.

Let $r = \lceil\log(p(n)) - 1\rceil$, and $a = \frac{p(n)}{2^{r+1}}$. (Observe that $a \leq 1$.) We first divide the interval $[1, p(n)]$ into $r+1$ many half-open intervals $[a, 3a], (3a, 7a], \cdots, ((2^r-1)a, p(n)]$. Next, we subdivide each of the intervals $((2^s-1)a, (2^{s+1}-1)a]$ into $1/\epsilon'$-many subintervals of equal size. So, each subinterval of the i^{th} interval is of size $\epsilon'\cdot 2^i a$.

Now, consider an instance I' of the problem where the maintenance level of each market is the smallest point in the subinterval it is in. (E.g if $w_1 \in (p,q]$, then $w'_1 = p$.)

This instance of crackdown scheduling has $(r+1)\cdot 1/\epsilon'$ many possible values of maintenance levels. Hence, by Theorem 2, I' can be solved optimally in $n^{O(r/\epsilon')}$ time. Let Π^* be the optimal schedule for I', $C(I', \Pi^*)$ be the cost of Π^* on instance I', and $C(I, \Pi^*)$ be the cost of Π^* on the original instance I. Let $Opt(I)$ denote the cost of an optimal solution of I.

We now show that $C(I, \Pi^*)$ is a good enough approximation of $Opt(I)$. We begin by proving the following claims.

Claim 1

$$C(I', \Pi^*) \leq Opt(I).$$

Proof Suppose not. Then let Π be an optimal solution of I such that $C(I, \Pi) < C(I', \Pi^*)$. Since for all schedules Π, it holds that $C(I, \Pi) \geq C(I', \Pi)$, it follows that $C(I', \Pi) \leq C(I, \Pi) < C(I', \Pi^*)$. This contradicts the fact that Π^* is an optimal solution of I'. $\square$

Claim 2

$$\frac{C(I, \Pi^*)}{Opt(I)} \leq \frac{C(I, \Pi^*)}{C(I', \Pi^*)} < (1+2\epsilon')^{\beta}.$$

Proof By renaming the markets if necessary, let $\Pi^* = 1, 2, \ldots, n$. For all i, let w'_i denote the maintenance level of market i in I'. We claim that for all i, $w_i < w'_i + 2\epsilon' w'_i = (1+2\epsilon')w'_i$.

To see this, recall our subdivision of the interval $[1, p(n)]$. For all s, the length of the s^{th} interval is $a \cdot 2^s$ and the smallest value in the s^{th} interval is $a \cdot (2^s - 1)$. Furthermore, each subinterval in the s^{th} interval is of length $a \cdot \epsilon' \cdot 2^s$. If w_i is in the interval $(w'_i, b]$, then the ratio $w_i / w'_i \leq b / w'_i$, which is maximized at the beginning the s^{th} interval. There the ratio b/w'_i is $\frac{a \cdot (2^s-1) + a \cdot \epsilon' \cdot 2^s}{a \cdot (2^s - 1)} < 1 + 2\epsilon'$. Hence $w_i < (1+2\epsilon')w'_i$. Therefore,

$$\begin{aligned}
\frac{C(I, \Pi^*)}{C(I', \Pi^*)} &= \frac{\sum_{i=2}^{n} \{m_i \cdot [\sum_{j=2}^{i-1} w_j]^\beta + c_i\}}{\sum_{i=2}^{n} \{m_i \cdot [\sum_{j=2}^{i-1} w'_j]^\beta + c_i\}} \\
&\leq \max_i \left\{1, \; \frac{m_i \cdot [\sum_{j=1}^{i-1} w_j]^\beta}{m_i \cdot [\sum_{j=1}^{i-1} w'_j]^\beta}\right\} \\
&< (1+2\epsilon')^\beta.
\end{aligned}$$

□

Now substituting the value of ϵ in place of ϵ', we have

$$\begin{aligned}
\frac{C(I, \Pi^*)}{Opt(I)} &< (1+2\epsilon')^\beta \\
&\leq e^{2\beta\epsilon'} \\
&\leq 1 + \epsilon.
\end{aligned}$$

Thus Π^* outputs a solution that is within $1+\epsilon$ of the optimal. The running time of this algorithm is bounded by $n^{O(\log n/\epsilon)}$. This completes the proof of this theorem. □

We end this section by summarizing the algorithm that is implicit in the above theorem. Given $\epsilon > 0$ and an instance I of the *Crackdown Scheduling* problem with n drug markets $\{1, 2, \ldots, n\}$, let $p(n)$ be a polynomial such that the maintenance level of every market is less than $p(n)$. Partition the interval $[1, p(n)]$ into $\log(p(n))$ many smaller intervals $[1, 3], (3, 7], (7, 15], \ldots (2^{\lceil \log(p(n)) \rceil - 1} - 1, p(n)]$ (hence the s^{th} interval is of length 2^s.) Next, partition each small interval into $\frac{1}{\epsilon'}$-many equal sized intervals, where $\epsilon' = \frac{1}{2\beta} \cdot \log(1+\epsilon)$. Next, construct an instance I' of the *Crackdown Scheduling* problem from the given instance I as follows.

(i) The markets in I' are denotes by $1', 2', \ldots, n'$;

(ii) For all i', the response function of i' is f_i; and

(iii) For all i' if w_i is contained in a subinterval $(t, v]$, then $w'_i = t$.

Next, run the algorithm described in the previous section on I'. If it outputs a schedule Π^*, then output Π^*. By Theorem 2, Π^* will give a solution that is within $1+\epsilon$ of the optimal solution for I.

Acknowledgements

We thank Alok Baveja for discussions on the drug market crackdown problem.

References

[1] Baker, K. R., *Introduction to Sequencing and Scheduling.* John Wiley and Sons Inc., New York, 1974.

[2] Baveja, A., Batta, R., Caulkins, J. P., Karwan, M. H., "Modeling the response of illicit drug markets to local enforcement", *Journal of Socio-Economic and Planning Sciences*, 27/2, 73–89, (1993).

[3] Baveja, A., Batta, R., Caulkins, J. P., Karwan, M. H., "Collapsing Street Markets for Illicit Drugs: The Benefits of Being Decisive", Working paper 93-39, Heinz School of Public Policy and Management, Carnegie Mellon University (1993).

[4] Caulkins, J. P., "The Distribution and Consumption of Illicit Drugs: Some Mathematical Models and their Policy Implications", *Ph.D. Thesis.* Massachusetts Institute of Technology (1990).

[5] Caulkins, J. P., "Thinking About Displacement in Drug Markets: Why Observing Change of Venue Isn't Enough", *Journal of Drug Issues*, 22/5, 17–30, (1991).

[6] Caulkins, J. P., Larson R. C., and Rich, T. F. "Geography's Impact on the Success of Focussed Local Drug Enforcement Operations," *Journal of Socio-Economic and Planning Sciences*, 27/1, 119–130, (1993).

[7] Caulkins, J. P., "Local Drug Markets' Response to Focussed Police Enforcement", *Operations Research* 41/5, 848-863, (1993).

[8] Garey, M., and Johnson, D., *Computers And Intractability: A Guide To The Theory of NP-Completeness*, W. H. Freeman, San Francisco 1979.

[9] M.H. Moore and M.A.R. Kleiman. The Police and Drugs. *Perspectives on Policing*, NCJ 117447. National Institute of Justice(1989).

[10] Naik, A., Baveja, A., Batta R., and Caulkins, J., *Scheduling Crackdowns on Illicit Drug Markets*, To appear in *European Journal of Operations Research*, 1994.

[11] Singer, M., "The Vitality of Mythical Numbers", *The Public Interest*, 23, 3–9, (Spring 1971).

[12] Worden, R. E., Bynum, T. S., and Frank, J., "Police Crackdowns", Draft for *Drugs and the Criminal Justice System*. Sage (Edited by Mackenzie and Uchida) (Forthcoming).