Solving LP Relaxations of Large-Scale Precedence Constrained Problems

Conference Paper · June 2010

DOI: 10.1007/978-3-642-13036-6_1 · Source: DBLP

2 authors, including:

Daniel Bienstock
Columbia University
147 PUBLICATIONS **4,961** CITATIONS

SEE PROFILE

Solving LP relaxations of large-scale precedence constrained problems

Daniel Bienstock* Mark Zuckerberg†

October, 2009

Abstract

We describe new algorithms for solving linear programming relaxations of very large precedence constrained production scheduling problems. We present theory that motivates a new set of algorithmic ideas that can be employed on a wide range of problems; on data sets arising in the mining industry our algorithms prove effective on problems with many millions of variables and constraints, obtaining provably optimal solutions in a few minutes of computation.

1 Introduction

We consider problems involving the scheduling of jobs over several periods subject to precedence constraints among the jobs as well as side-constraints. We must choose the subset of jobs to be performed, and, for each of these jobs, how to perform it, choosing from among a given set of options (representing facilities or modes of operation). Finally, there are side-constraints to be satisfied, including period-wise, per-facility processing capacity constraints, among others. There are standard representations of these problems as (mixed) integer programs.

Our data sets originate in the mining industry, where problems typically have a small number of side constraints - often well under one hundred – but may contain millions of jobs and tens of millions of precedences, as well as spanning multiple planning periods. Appropriate formulations often achieve small integrality gap in practice; unfortunately, the linear programming relaxations are far beyond the practical reach of commercial software.

We present a new iterative algorithm for solving the LP relaxation of this problem. The algorithm incorporates, at a low level, ideas from Lagrangian relaxation and column generation, but is however based on fundamental observations on the underlying combinatorial structure of precedence constrained, capacitated optimization problems. Rather than updating dual information, the algorithm uses primal structure gleaned from the solution of subproblems in order to accelerate convergence. The general version of our ideas should be applicable to a wide class of problems. The algorithm can be proved to converge to optimality; in practice we have found that even for problems with millions of variables and tens of millions of constraints, convergence to *proved* optimality is usually obtained in under twenty iterations, with each iteration requiring only a few seconds on current computer hardware.

*Department of Industrial Engineering and Operations Research, Columbia University.
Research partially funded by a gift from BHP Billiton Ltd., and ONR Award N000140910327.
†Resource and Business Optimization Group Function, BHP Billiton Ltd.

2 Definitions and Preliminaries

2.1 The Precedence Constrained Production Scheduling Problem

Definition 1 *We are given a directed graph $G = (\mathcal{N}, \mathcal{A})$, where the elements of $\mathcal{N}$ represent jobs, and the arcs $\mathcal{A}$ represent precedence relationships among the jobs: for each $(i, j) \in \mathcal{A}$, j can be performed no later than job i. Denote by F, the number of facilities, and T, the number of scheduling periods.*

Let $y_{j,t} \in \{0, 1\}$ represent the choice to process job j in period t, and $x_{j,t,f} \in [0, 1]$ represent the proportion of job j performed in period t, and processed according to processing option, or "facility", f.

Let $c^T x$ be an objective function, and let $Dx \leq d$ be a collection of arbitrary "side" constraints.

The linear programming relaxation of the resulting problem, which we will refer to as PCPSP, is as follows:

$$\textbf{(PCPSP):} \qquad \max \quad c^T x \tag{1}$$

$$\textit{Subject to:} \qquad \sum_{\tau=1}^{t} y_{i,\tau} \leq \sum_{\tau=1}^{t} y_{j,\tau}, \ \forall (i,j) \in \mathcal{A}, \ 1 \leq t \leq T \tag{2}$$

$$Dx \leq d \tag{3}$$

$$y_{j,t} \ = \ \sum_{f=1}^{F} x_{j,t,f}, \ \forall j \in \mathcal{N}, \ 1 \leq t \leq T \tag{4}$$

$$\sum_{t=1}^{T} y_{j,t} \leq 1, \ \forall j \in \mathcal{N} \tag{5}$$

$$x \geq 0. \tag{6}$$

For precedence constrained production scheduling problems that occur in the mining industry some typical numbers are as follows:

- 1 million – 10 million jobs, and 1 million – 100 million precedences,
- 20 – 200 side-constraints, 10 – 20 periods, and 2 – 3 facilities.

These numbers indicate that the number of constraints of the form (2), (4) and (5) can be expected to be very large.

2.2 Background

2.2.1 The Open Pit Mine Scheduling Problem

The practical motivating problem behind our study is the open pit mine scheduling problem. We are given a three-dimensional region representing a mine to be exploited; this region is divided into "blocks" (jobs, from a scheduling perspective) corresponding to units of earth ("cubes") that can

be extracted in one step. In order for a block to be extracted, the set of blocks located (broadly speaking) in a cone above it must be extracted first. This gives rise to a set of precedences, i.e. to a directed graph whose vertices are the blocks, and whose arcs represent the precedences. Finally, the extraction of a block entails a certain (net) profit or cost.

The problem of selecting which blocks to extract so as to maximize profit can be stated as follows:

$$\max \left\{ c^T x \,:\, x_i \leq x_j \;\forall\, (i,j) \in \mathcal{A},\; x_j \in \{0,1\} \;\forall j \right\},$$

where as before $\mathcal{A}$ indicates the set of precedences. This is the so-called *maximum weight closure problem* – in a directed graph, a closure is a set S of vertices such that there exist no arcs (i,j) with $i \in S$ and $j \notin S$. It can be solved as a minimum $s-t$ cut problem in a related graph of roughly the same size. See [P76], and also [J68], [Bal70] and [R70]. Further discussion can be found in [HC00], where the authors note (at the end of Section 3.4) that it can be shown by reduction from max clique that adding a single cardinality constraint to a max closure problem is enough to make it NP-hard. For additional related material see [F06], [LG65], [CH03], and references therein.

The problem we are concerned with here, by contrast, also incorporates production scheduling. When a block is extracted it will be processed at one of several facilities with different operating capabilities. The processing of a given block i at a given facility f consumes a certain amount of processing capacity v_{if} and generates a certain net profit p_{if}. This overall planning problem spans several time periods; in each period we will have one or more knapsack (capacity) constraints for each facility. We usually will also have additional, ad-hoc, non-knapsack constraints. In this version the precedence constraints apply across periods as per (2): if $(i,j) \in \mathcal{A}$ then j can only be extracted in the same or in a later period than i.

Typically, we need to produce schedules spanning 10 to 20 periods. Additionally, we may have tens of thousands (or many more) blocks; this can easily make for an optimization problem with millions of variables and tens of millions of precedence constraints, but with (say) on the order of one hundred or fewer processing capacity constraints (since the total number of processing facilities is typically small).

2.2.2 Previous work

A great deal of research has been directed toward algorithms for the maximum weight closure problems, starting with [LG65] and culminating in the very efficient method described in [H08] (also see [CH09]). A "nested shells" heuristic for the capacitated, multiperiod problem, based on the work in [LG65], is applicable to problems with a single capacity constraint, among other simplifications. As commercial integer programming software has improved, mine scheduling software packages have recently emerged that aggregate blocks in order to yield a mixed integer program of tractable size. The required degree of aggregation can however be enormous; this can can severely compromise the validity and the usefulness of the solution. For an overview of other heuristic approaches that have appeared in the open mine planning literature [HC00] and [F06].

Recently (and independent of our work) there has been some new work relevant to the solution of the LP relaxation of the open pit mine scheduling problem. [BDFG09] have suggested a new approach in which blocks are aggregated only with respect to the digging decisions but not with respect to the processing decisions, i.e. all original blocks in an aggregate must be extracted in a common period, but the individual blocks comprising an aggregate can be processed in different ways. This problem is referred to by the authors as the "Optimal Binning Problem". As long as there is more than one processing option this approach still maintains variables for each block and period and is therefore still very large, but the authors propose an algorithm for the LP relaxation

of this problem that is only required to solve a sequence of linear programs with a number of variables on the order of the number of aggregates (times the number of periods) in order to come to a solution of the large LP. Thus if the number of aggregates is small the LP can be solved quickly.

Another development that has come to our attention recently is an algorithm by [CEGMR09] which can solve the LP relaxation of even very large instances of the open pit mine scheduling problem very efficiently. This algorithm is only applicable however to problems for which there is a single processing option and for which the only constraints are knapsacks and there is a single such constraint in each scheduling period. The authors note however that more general problems can be relaxed to have this form in order to yield an upper bound on the solution value.

3 Our results

Empirically, it can be observed that formulation (1-6) frequently has small integrality gap. We present a new algorithm for solving the continuous relaxation of this formulation and generalizations. Our algorithm is applicable to problems with an arbitrary number of process options and arbitrary side constraints, and it requires no aggregation.[1] On very large, real-world instances our algorithm proves very efficient.

Our algorithmic developments hinge on three ideas. In order to describe these ideas, we will first recast PCPSP as a special case of a more general problem, to which these results (and our solution techniques) apply.

Definition 2 *Given a directed graph $G = (\mathcal{N}, \mathcal{A})$ with n vertices, and a system $Dx \leq d$ of d constraints on n variables, the General Precedence Constrained Problem is the following linear program:*

$$\textbf{(GPCP):} \qquad \max \quad c^T x \tag{7}$$

$$Dx \leq d \tag{8}$$

$$x_i - x_j \leq 0, \quad \forall\, (i,j) \in \mathcal{A}, \tag{9}$$

$$0 \leq x_j \leq 1, \quad \forall\, j \in \mathcal{N}. \tag{10}$$

This problem is more general than PCPSP:

Lemma 3 *Any instance of PCPSP can be reduced to an equivalent instance of GPCP with the same number of variables and of constraints.*

Proof. Consider an instance of PCPSP on $G = (\mathcal{N}, \mathcal{A})$, with T time periods, F facilities and side constraints $Dx \leq d$. Note that the y variables can be eliminated. Consider the following system of inequalities on variables $z_{j,t,f}$ ($j \in \mathcal{N}$, $1 \leq t \leq T$, $1 \leq f \leq F$):

$$z_{j,t,f} - z_{j,t,f+1} \leq 0, \quad \forall\, j \in \mathcal{N}, 1 \leq t \leq T, 1 \leq f < F, \tag{11}$$

$$z_{j,t,F} - z_{j,t+1,1} \leq 0, \quad \forall\, j \in \mathcal{N}, 1 \leq t < T, \tag{12}$$

[1] Though our work was conducted independently of that of [BDFG09] and of [CEGMR09] and our results are more broadly applicable, there are some themes in common. Specifically, the use of simplifying assumptions related in some way to the duals in order to shrink the linear program is a theme in common between our work and that of [BDFG09]. In the full version of this paper we describe what our algorithm would look like when applied to the aggregated problem treated by [BDFG09]. As we note there, the resulting special case of our algorithm is broadly similar to their algorithm. The relationship between the max closure problem and the LP is a theme in common with the work of [CEGMR09].

$$
\begin{aligned}
z_{j,T,F} &\leq 1, \quad j \in \mathcal{N}, && (13)\\
z_{i,t,F} - z_{j,t,F} &\leq 0, \quad \forall\, (i,j) \in \mathcal{A},\ 1 \leq t \leq T, && (14)\\
z &\geq 0. && (15)
\end{aligned}
$$

Given a solution (x, y) to PCPSP, we obtain a solution z to (11)-(15) by setting, for all j, t and f:

$$z_{j,t,f} = \sum_{\tau=1}^{t-1} \sum_{f'=1}^{F} x_{j,\tau,f'} + \sum_{f'=1}^{f} x_{j,t,f'},$$

and conversely. Thus, for an appropriate system $\bar{D}z \leq \bar{d}$ (with the same number of rows as $Dx \leq d$) and objective $\bar{c}^T z$, PCPSP is equivalent to the linear program:

$$\min\{\bar{c}^T z \ :\ \bar{D}z \leq \bar{d}, \text{ and constraints (11)-(15)}\}. \blacksquare$$

Note: In Lemma 3 the number of precedences in the instance of $GPCP$ is larger than in the original instance of PCPSP; nevertheless we stress that the number of constraints (and variables) is indeed the same in both instances.

We will now describe ideas that apply to $GPCP$. First, we have the following remark.

Observation 4 *Consider an instance of problem GPCP, and let $\pi \geq 0$ be a given vector of dual variables for the side-constraints (8). Then the Lagrangian obtained by dualizing (8) using π,*

$$
\begin{aligned}
\max \quad & c^T x + \pi^T (d - Dx) && (16)\\
\textit{Subject to:} \quad & x_i - x_j \leq 0, \ \forall\ (i,j) \in \mathcal{A} && (17)\\
& 0 \leq x_j \leq 1, \quad \forall\, j \in \mathcal{N}. && (18)
\end{aligned}
$$

is a maximum closure problem with $|\mathcal{A}|$ precedences.

Note: There is a stronger version of Observation 4 in the specific case of problem PCPSP; namely, the x variables can be *eliminated* from the Lagrangian (details: full paper, also see [BZ09]).

Observation 4 suggests that a Lagrangian relaxation algorithm for solving problem $GPCP$ – that is to say, an algorithm that iterates by solving problems of the form (16-18) for various vectors π – would enjoy fast individual iterations. This is correct, as our experiments confirm that even extremely large max closure instances can be solved quite fast using the appropriate algorithm (details, below). However, in our experiments we also observed that traditional Lagrangian relaxation methods (such as subgradient optimization), applied to $GPCP$, performed quite poorly, requiring vast numbers of iterations and not quite converging to solutions with desirable accuracy.

Our approach, instead, relies on leveraging combinatorial structure that optimal solutions to $GPCP$ must satisfy. Lemmas 5 and 6 are critical in suggesting such structure.

Lemma 5 *Let $P = \{x \in R^n \ :\ Ax \leq b,\ Dx \leq d\}$, where A, D, b, d are matrices and vectors of appropriate dimensions. Let $\hat{x}$ be an extreme point of P. Let $\bar{A}x = \bar{b}$, $\bar{D}x = \bar{d}$ be the set of binding constraints at $\hat{x}$. Assume $\bar{D}$ has q linearly independent rows, and let $N^{\hat{x}}$ be the null space of $\bar{A}$. Then $dim(N^{\hat{x}}) \leq q$.*

Proof: $\bar{A}$ must have at least $n - q$ linearly independent rows and thus its null space must have dimension $\leq q$. $\blacksquare$

Lemma 6 *Let P be the feasible space of a GPCP with q side constraints. Denote by $Ax \leq b$ the subset of constraints containing the precedence constraints and the constraints $0 \leq x \leq 1$, and let $Dx \leq d$ denote the side constraints. Let $\hat{x}$ be an extreme point of P, and the entries of $\hat{x}$ attain k distinct fractional values $\{\alpha_1, \ldots, \alpha_k\}$. For $1 \leq r \leq k$, let $\theta^r \in \{0,1\}^n$ be defined by:*

$$\text{for } 1 \leq j \leq n, \quad \theta^r_j = \begin{cases} 1, & \text{if } \hat{x}_j = \alpha_r, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\bar{A}$ be the submatrix of A containing the binding constraints at $\hat{x}$. Then the vectors θ^r are linearly independent and belong to the null space of $\bar{A}$. As a consequence, $k \leq q$.

Proof: First we prove that $\bar{A}\theta^r = 0$. Given a precedence constraint $x_i - x_j \leq 0$, if the constraint is binding then $\hat{x}_i = \hat{x}_j$. Thus if $\hat{x}_i = \alpha_r$, so that $\theta^r_i = 1$, then $\hat{x}_j = \alpha_r$ also, and so $\theta^r_j = 1$ as well, and so $\theta^r_i - \theta^r_j = 0$. By the same token if $\hat{x}_i \neq \alpha_r$ then $\hat{x}_j \neq \alpha_r$ and again $\theta^r_i - \theta^r_j = 0$. If a constraint $x_i \geq 0$ or $x_i \leq 1$ is binding at $\hat{x}$ then naturally $\theta^r_i = 0$ for all r as $\hat{x}_i$ is not fractional. The supports of the θ^r vectors are disjoint, yielding linear independence. Finally, $k \leq q$ follows from Lemma 5. ■

Observation 4 implies that an optimal solution x^* to an instance of $GPCP$ can be written as a weighted sum of incidence vectors of closures, i.e.,

$$x^* = \sum_{q=1}^{Q} \mu_q v^q, \tag{19}$$

where $\mu \geq 0$, and, for each q, $v^q \in \{0,1\}^n$ is the incidence vector of a closure $S^q \subset \mathcal{N}$. [In fact, the S^q can be assumed to be nested]. So for any $i, j \in \mathcal{N}$, $x^*_j = x^*_i$ if i and j belong to precisely the same family of sets S^q. Also, Lemma 6 states that the number of distinct values that x^*_j can take is *small*, if the number of side constraints is small. Therefore it can be shown that when the number of side constraints is small the number of closures (terms) in (19) must also be small. In the full paper we flesh out this connection further, showing that a rich relationship exists between the max closures produced by lagrangian problems and the optimal dual and primal solutions to $GPCP$. Next, we will develop an algorithm that solves $GPCP$ by attempting to "guess" the correct representation (19).

First, we present a result that partially generalizes Lemma 6.

Theorem 7 *Let P, A, $\bar{A}$, D, q, $\hat{x}$ and $N^{\hat{x}}$ be as in Lemma 5, and assume additionally that A is totally unimodular and that b is integral. Define*

$$I^{\hat{x}} = \{y \in R^n : y_i = 0, \ \forall i \text{ s.t. } \hat{x}_i \text{ is integer}\}. \tag{20}$$

Then there exists an integral vector $x^i \in R^n$, and vectors $\theta^h \in R^n$, $1 \leq h \leq q$, such that:

(a) $Ax^i \leq b$,

(b) $\bar{A}x^i = \bar{b}$,

(c) $x^i_j = \hat{x}_j$, $\forall j$ *s.t.* $\hat{x}_j$ *is integer*,

(d) $\hat{x} = x^i + \sum_{r=1}^{q} \alpha_r \theta^r$, *for some* $\alpha \in R^q$,

(e) The set $\{\theta^1, \ldots, \theta^q\}$ *spans* $N^{\hat{x}} \cap I^{\hat{x}}$,

(f) $|\theta^h_j| \leq \text{rank}(\bar{A})$, *for all* $1 \leq h \leq q$ *and* $1 \leq j \leq n$,

In the special case of the GPCP, we can choose x^i *satisfying the additional condition:*

(g) $x^i_j = 0$, *for all* j *such that* $\hat{x}_j$ *is fractional.*

Proof sketch: Let us refer to the integer coordinates of x as x_I and to the corresponding columns of A as A^I, and to the fractional coordinates of x as x_F, and to the corresponding columns of A as A^F. Let h be the number of columns in A^F. Note that $b - A^I x_I$ is integer, and so by total unimodularity there exists integer $y \in R^h$ satisfying $A^F y \leq b - A^I x_I$, $\bar{A}^F y = \bar{b} - \bar{A}^I x_I$. Defining now $x^i = (x_I, y)$ then x^i is integer; it is equal to x everywhere that x is integer, and it satisfies $Ax^i \leq b$ and $\bar{A}x^i = \bar{b}$. Clearly $x - x^i$ belongs to I^x, and moreover $\bar{A}(x - x^i) = 0$ so that it belongs to N^x as well, and so it can be decomposed as

$$x - x^i = \sum_{r=1}^{q} \alpha_r \theta^r. \tag{21}$$

For the special case of $GPCP$ we have already described a decomposition for which x^i equals x everywhere that x is integer and is zero elsewhere. See the full paper for other details. ■

Comment: Note that $\text{rank}(\bar{A})$ can be high and thus condition (d) is not quite as strong as Lemma 6; nevertheless q is small in any case and so we obtain a decomposition of $\hat{x}$ into "few" terms when the number of side-constraints is "small". Theorem 7 can be strengthened for specific families of totally unimodular matrices. For example, when A is the node-arc incidence matrix of a digraph, the θ vectors are incidence vectors of cycles, which yields the following corollary.

Corollary 8 *Let* P *be the feasible set for a minimum cost network flow problem with integer data and side constraints. Let* $\hat{x}$ *be an extreme point of* P, *and let* q *be the number of linearly independent side constraints that are binding at* $\hat{x}$. *Let* $\zeta = \{j : \hat{x}_j \text{ integral}\}$. *Then* $\hat{x}$ *can be decomposed into the sum of an integer vector* v *satisfying all network flow (but not necessarily side) constraints, and with* $v_j = \hat{x}_j \ \forall j \in \zeta$, *and a sum of no more than* q *fractional cycle flows, over a set of cycles disjoint from* ζ.

4 A General Algorithmic Template

Now we return to the generic algorithm for $GPCP$ that attempts to guess the right representation of an optimal solution as a weighted sum of incidence vectors of "few" closures. To motivate our approach, we first consider a more general situation. We are given a linear program:

$$\begin{aligned} (P_1): \quad & \max \ c^T x \\ \text{s.t.} \quad & Ax \ \leq \ b \\ & Dx \ \leq \ d. \end{aligned} \tag{22}$$

Denote $L(P_1, \mu)$ be the Lagrangian relaxation in which constraints (22) are dualized with penalties μ, i.e. the problem $\max\{c^T x + \mu^T(d - Dx) : Ax \leq b\}$.

One can approach problem (P_1) by means of Lagrangian relaxation, i.e. an algorithm that iterates by solving multiple problems $L(P_1, \mu)$ for different choices of μ; the multipliers μ are updated according to some procedure. A starting point for our work concerns the fact that traditional

Lagrangian relaxation schemes (such as subgradient optimization) can prove frustratingly slow to achieve convergence, often requiring seemingly instance-dependent choices of algorithmic parameters, and frequently failing to deliver a sufficiently accurate solution. However, as observed in [B02] (and also see [BA00]) Lagrangian relaxation schems *can* discover useful "structure."

For example, Lagrangian relaxation can provide early information on which constraints from among (22) are likely to be tight, and on which variables x are likely to be nonzero, even if the actual numerical values for primal or dual variables computed by the relaxation are inaccurate. The question then is how to use such structure in order to accelerate convergence and to obtain higher accuracy. In [B02] the following approach was used:

- Periodically, interrupt the Lagrangian relaxation scheme to solve a *restricted* linear program consisting of (P_1) with some additional constraints used to impose the desired structure. Then use the duals for constraints (22) obtained in the solution to the restricted LP to restart the Lagrangian procedure.

The restricted linear program includes all constraints, and thus could (potentially) still be very hard – the idea is that the structure we have imposed renders the LP much easier. Further, the LP includes all constraints, and thus the solution we obtain is fully feasible for (P_1), thus proving a lower bound. Moreover, if our guess as to "structure" is correct, we also obtain a high-quality dual feasible vector, and our use of this vector so as to restart the Lagrangian scheme should result in accelerated convergence (as well as proving an upper bound on (P_1)). In [B02] these observations were experimentally verified in the context of several problem classes.

We now seek to extend, and generalize, these ideas. In the following template, at each iteration k we employ a linear system $H^k x = h^k$ which can be interpreted as an educated guess for conditions that an optimal solution to P_1 should satisfy. This is problem-specific; we will indicate later how this structure is discovered in the context of $GPCP$.

Algorithm Template

1. Set $\mu_0 = 0$ and set $k = 1$.

2. Solve $L(P_1, \mu_{k-1})$. Let w^k be an optimal solution.

If $k > 1$ and $H^{k-1} w^k = h^{k-1}$, **STOP**.

3. Let $H^k x = h^k$ be a linear system of equations that is satisfied by w^k.

4. Define the **restricted problem**:

$$
\begin{aligned}
(P_2^k): \quad & \max \ c^T x & \\
\text{s.t.} \quad & Ax \ \leq \ b & (23)\\
& Dx \ \leq \ d & (24)\\
& H^k x = h^k & (25)
\end{aligned}
$$

5. Solve P_2^k to obtain x^k, an optimal primal vector (with value z^k) and μ_k, an optimal dual vector corresponding to constraints (24). If $\mu_k = \mu_{k-1}$, **STOP**.

6. Set $k = k + 1$ and goto Step **2**.

Notes:
1. Ideally, imposing $H^k x = h^k$ in Step 4 should result in an *easier* linear program.
2. For simplicity, in what follows we will assume that P_2^k is always feasible; though this is a requirement that can be easily circumvented in practice (full paper).

Theorem 9 *(a) If the algorithm stops at iteration k in Step 2, then x^{k-1} is optimal for P_1. (b) If it stops in Step 5 then x^k is optimal for P_1.*

Proof: (a) We have

$$z^{k-1} \;=\; \max\{c^T x \,+\, \mu_{k-1}^T(d - Dx) \,:\, Ax \le b,\; H^{k-1}x = h^{k-1}\} \;=\; c^T w^k \,+\, \mu_{k-1}^T(d - Dw^k),$$

where the first equality follows by duality and the second by definition of w^k in Step 2 since $H^{k-1}w^k = h^{k-1}$. Also, clearly $z^{k-1} \le z^*$, and so in summary

$$z^* \le c^T w^k \,+\, \mu_{k-1}^T(d - Dw^k) = z^{k-1} \le z^*. \tag{26}$$

(b) $\mu_k = \mu_{k-1}$ implies that w^k optimally solves $L(P_1, \mu_k)$, so that we could choose $w^{k+1} = w^k$ and so $H^k w^{k+1} = h^k$, obtaining case (a) again. ■

4.1 Applying the template

We will now apply the above template to a case where P_1 is an instance of GPCP, where, as before, we denote by $\mathcal{N}$ the set of jobs; and we use $Ax \le b$ to describe the precedences and the box constraints $0 \le x_j \le 1$ ($\forall j \in \mathcal{N}$), and $Dx \le d$ denotes the side-constraints.

Thus, in Step 2 of the template, $L(P_1, \mu_{k-1})$ can be solved as a max closure problem (Observation 4); therefore its solution can be described by a 0/1-vector which we will denote by y^k. At iteration k, we will construct a partition $\mathcal{C}^k = \{C_1^k, \ldots, C_{r_k}^k\}$ of $\mathcal{N}$. Our basic application of the template is as follows:

GPCP Algorithm

1. Initializations: Set $\mu_0 = 0$. Set $r_0 = 1$, $C_1^0 = \mathcal{N}$, $\mathcal{C}^0 = \{C_1^0\}$, $z^0 = -\infty$, and $k = 1$.

2. Let y^k be an optimal solution to $L(P_1, \mu_{k-1})$, and define

$$I^k = \{j \in \mathcal{N} : y_j^k = 1\} \tag{27}$$

   and define

$$O^k = \{j \in \mathcal{N} : y_j^k = 0\}. \tag{28}$$

   If $k > 1$, and, for $1 \le h \le r_{k-1}$, either $C_h^{k-1} \cap I^k = \emptyset$ or $C_h^{k-1} \cap O^k = \emptyset$, then **STOP**.

3. Let $\mathcal{C}^k = \{C_1^k, \ldots, C_{r_k}^k\}$ consist of all *nonempty* sets in the collection

$$\left\{I^k \cap C_h^{k-1} \,:\, 1 \le h \le r_{k-1}\right\} \cup \left\{O^k \cap C_h^{k-1} \,:\, 1 \le h \le r_{k-1}\right\}.$$

   Let $H^k x = h^k$ consist of the constraints:

$$x_i = x_j, \;\text{ for } 1 \le h \le r_k, \text{ and each pair } i, j \in C_h^k.$$

4. Let P_2^k consist of P_1, plus the additional constraints $H^k x = h^k$.

5. Solve P_2^k, with optimal solution x^k, and let μ_k denote the optimal duals corresponding to the side-constraints $Dx \leq d$. If $\mu_k = \mu_{k-1}$, **STOP**.

6. Set $k = k + 1$ and goto Step **2**.

We have:

Lemma 10 *(a) For each k, problem P_2^k is an instance of GPCP with r_k variables and the same number of side-constraints as in $Dx \leq d$. (b) If P_2^1 is feasible, the above algorithm terminates finitely with an optimal solution.*

Proof: full paper. ■

Comments: The above algorithm is a basic application of the template. The partition $\mathcal{C}^k$ can either be refined, or coarsened. In particular, x^k (Step 5) may attain *fewer* than r_k distinct values (it should attain "few" values, by Lemma 10 (a) and Lemma 6) and we can correspondingly merge different sets C_j^k. The feasibility assumption in (b) of Lemma 10 can be bypassed. Details will be provided in the full paper.

5 Computational Experiments

In this section we present results from some of our experiments. A more complete set of results will be presented in the full paper. All these tests were conducted using a single core of a dual quad-core 3.2 GHz Xeon machine with 64 GB of memory. The LP solver we used was Cplex, version 12 and the min cut solver we used was our implementation of Hochbaum's pseudoflow algorithm ([H08]).

The tests reported on in Tables 1 and 2 are based on three real-world examples provided by BHP Billiton[2], to which we refer as 'Mine1', 'Mine2' and 'Mine3' and a synthetic but realistic model called 'Marvin' which is included with Gemcom's Whittle [W] mine planning software. 'Mine1B' is a modification of Mine1 with a denser precedence graph. Mine3 comes in two versions to which we refer as 'big' and 'small'. Using Mine1, we also obtained smaller and larger problems by modifying the data in a number of realistic ways. Some of the row entries in these tables are self-explanatory; the others have the following meaning:

- **Problem arcs**. The number of arcs in the graph that the algorithm creates to represent the scheduling problem (i.e., the size of the min cut problems we solve).

- **Iterations, time to 10^{-5} optimality**. The number of iterations (resp., the CPU time) taken by the algorithm until it obtained a solution it could certify as having $\leq 10^{-5}$ **relative** optimality error.

- **Iterations, time to combinatorial optimality**. The number of iterations (resp., the CPU time) taken by the algorithm to obtain a solution it could certify as *optimal* as per the stopping criteria in Steps 2 or 5. Notice that this implies that the solution is optimal as per the numerical tolerances of Cplex.

Finally, an entry of "—" indicates that Cplex was unable to terminate after 100000 seconds of CPU time. More detailed analyses will appear in the full paper.

[2]Data was masked.

	Marvin	Mine1B	Mine2	Mine3,s	Mine3,b
Jobs	9400	29277	96821	2975	177843
Precedences	145640	1271207	1053105	1748	2762864
Periods	14	14	25	8	8
Facilities	2	2	2	8	8
Variables	199626	571144	3782250	18970	3503095
Constraints	2048388	17826203	26424496	9593	19935500
Problem arcs	2229186	18338765	30013104	24789	23152350
Side-constraints	28	28	50	120	132
Binding side-constr. at optimum	14	11	23	33	44
Cplex time (sec)	55544	—	—	5	—
Algorithm Performance					
Iterations to 10^{-5} optimality	8	8	9	13	30
Time to 10^{-5} optimality (sec)	10	60	344	1	1076
Iterations to comb. optimality	11	12	16	15	39
Time to comb. optimality (sec)	15	96	649	1	1583

Table 1: ***Sample runs, 1***

References

[Bal70] M.L. Balinsky, On a selection problem, *Management Science* **17** (1970), 230–231.

[BA00] F. Barahona and R. Anbil, The Volume Algorithm: producing primal solutions with a subgradient method, *Math. Programming* **87** (2000), 385 – 399.

[B02] D. Bienstock,*Potential Function Methods for Approximately Solving Linear Programming Problems, Theory and Practice*, ISBN 1-4020-7173-6. Kluwer Academic Publishers, Boston (2002).

[BZ09] D. Bienstock and M. Zuckerberg, A new LP algorithm for precedence constrained production scheduling, posted on Optimization Online, 08/2009.

[BDFG09] N. Boland, I. Dumitrescu, G. Froyland and A.M. Gleixner, LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity, *Computers and Operations Research* **36** (2009), 1064 – 1089.

[CH03] L. Caccetta and S.P. Hill, An application of branch and cut to open pit mine scheduling, *Journal of Global Optimization* **27** (2003), 349–365.

	Mine1 very small	Mine1 medium	Mine1 large	Mine1 full	Mine1,3 weekly
Jobs	755	7636	15003	29277	87831
Precedences	222	22671	113703	985011	2955033
Periods	12	12	12	12	100
Facilities	2	2	2	2	2
Variables	14282	160944	292800	489552	12238800
Constraints	8834	327628	1457684	11849433	295591331
Problem arcs	22232	477632	1727565	12280407	307654269
Side-constraints	24	24	24	24	200
Binding side-constr. at optimum	12	11	11	11	151
Cplex time (sec)	1	12424	—	—	—
Algorithm Performance					
Iterations to 10^{-5} optimality	6	6	8	7	10
Time to 10^{-5} optimality (sec)	0	1	7	45	2875
Iterations to comb. optimality	7	7	11	9	20
Time to comb. optimality (sec)	0	2	10	61	6633

Table 2: ***Sample runs, 2***

[CH09] B. Chandran and D. Hochbaum, A Computational Study of the Pseudoflow and Push-Relabel Algorithms for the Maximum Flow Problem, *Operations Research* **57** (2009), 358–376.

[CEGMR09] R. Chicoisne, D. Espinoza, M. Goycoolea, E. Morena, E. Rubio A New Algorithm for the Open-Pit Mine Scheduling Problem. Submitted for publication, available at http://mgoycool.uai.cl/

[F06] C. Fricke, Applications of integer programming in open pit mine planning, PhD thesis, Department of Mathematics and Statistics, The University of Melbourne, 2006.

[H08] D. Hochbaum, The pseudoflow algorithm: a new algorithm for the maximum flow problem, *Operations Research* **58** (2008), 992–1009.

[HC00] D. Hochbaum and A. Chen, Improved planning for the open - pit mining problem, *Operations Research* **48** (2000), 894–914.

[J68] T.B. Johnson, Optimum open pit mine production scheduling, PhD thesis, Operations Research Department, University of California, Berkeley, 1968.

[LG65] H. Lerchs, and I.F. Grossman, Optimum design of open-pit mines, *Transactions C.I.M.* **68** (1965), 17 – 24.

[P76] J.C. Picard, Maximal Closure of a graph and applications to combinatorial problems,, *Management Science* **22** (1976), 1268 – 1272.

[R70] J.M.W. Rhys, A selection problem of shared fixed costs and network flows, *Management Science* **17** (1970), 200–207.

[W] Gemcom Software International, Vancouver, BC, Canada.