# A New LP Algorithm for Precedence Constrained Production Scheduling

Daniel Bienstock[*] Mark Zuckerberg[†‡]

May, 2015

Wed.May.27.1549AEST.2015@IGSM488074

**Abstract**

The precedence constrained production scheduling problem is the problem of scheduling the performance of jobs over a number of scheduling periods subject to precedence constraints among the jobs. The jobs can each be performed in a number of ways, and it also needs to be determined which processing option (or options) is to be chosen for each job. There can also be arbitrary side constraints among these variables. The side constraints typically represent either period capacity constraints, or profile constraints on the aggregate product produced in each period.

These problems, as they occur in the mining industry, typically have a small number of side constraints - often well under 100, but may contain millions of jobs and tens of millions of precedences. Thus despite the fact that the integrality gap is often small in practice, the LP itself is beyond the practical reach of commercial software.

We present a new iterative lagrangian-based algorithm for solving the LP relaxation of this problem. This algorithm can be proven to converge to optimality and 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 20 iterations, with each iteration requiring only a few seconds to solve with current computer hardware.

## 1 Introduction

### 1.1 Background

#### 1.1.1 Problem Definition

The production scheduling problems with which we will concern ourselves are those in which, given a collection of "jobs" and a number of "scheduling periods", we need to decide which jobs should be processed in which scheduling period(s); processing a job consumes resources that may be constrained in each period and affects the profile of the products produced in each period, which may also be constrained. Additionally there will be two further modeling details:

1. Precedence relationships may exist among the jobs.
2. There may be more than one way of processing any given job.

#### 1.1.2 The Open Pit Mine Scheduling Problem

The practical motivating problem behind our study is the open pit mine scheduling problem. The open pit mine scheduling problem seeks to determine the optimal schedule for the extraction of

[*]Department of Industrial Engineering and Operations Research, Columbia University
[†]Technology, Geoscience and Engineering Group Function, BHP Billiton Ltd.
[‡]Department of Mathematics and Statistics, University of Melbourne

mineralized earth from an open pit mine. The principal structural constraint associated with open pit mining – known as the "slope constraint" – is that for any point $x$ in the orebody, for reasons of structural stability a schedule may not extract material at $x$ before it extracts an upward facing (generally) circular cone that sits above $x$, i.e. that the slope of the resulting pit after $x$ is extracted must not exceed some given angle. (See [F06] for a more thorough description of the problem.)

In this model a "job" is the extraction of a unit of material, and the slope constraint is modeled by precedence relationships between each unit and the units in its upward facing cone. There are, moreover, a number of options as to what to do with the unit of material once it has been extracted. It may be sent to waste (this can happen even if the material has positive value and can be blended to produce a saleable product, in order to conserve valuable capacity in the processing plant), or it may be processed in one of several possible ways.

The distinguishing feature of the mine scheduling problem is that the vast majority of the constraints are those that model the slope constraint as described. The number of "production planning" constraints (i.e. resource capacity, product profile and similar constraints) is generally on the order of the number of scheduling periods, which for strategic problems is usually small. Thus there may be under 100 production planning constraints in a typical problem.[1]

To obtain a reasonably accurate model of the slope constraint however, it is necessary that the scheduling units (the "jobs," or discrete units of work to be scheduled) not be overly large (typically no larger than 30 meters in any dimension), and in any case it is desired to keep the unit sizes small in order to maintain selectivity. Moreover to approximate a circular slope constraint via unit precedences typically requires a substantial number of precedence constraints per unit (often 20 or more)[2]

Considering that a typical orebody can have a size that measures into the cubic kilometers, there can be many orebodies that need to be simultaneously scheduled, and a schedule needs to be produced for 10 to 20 periods or more, this can easily make for a problem with millions of variables and tens of millions of precedence constraints. Thus despite the fact that in practice the integrality gap between the integer programming solution and the linear programming relaxation is often small,[3] the linear programming problem itself is well beyond the reach of commercial linear programming solvers.

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

- Typical number of periods : $10 - 20$.
- Typical number of destinations : $2 - 5$.
- Typical number of side constraints : $20 - 200$.
- Typical number of scheduling units: $10,000 - 10$ million.
- Typical number of precedences : 1 million – 4 billion.

### 1.1.3 Previous Work

The traditional approach to the open pit mine planning problem, popularized in Gemcom's Whittle [W] mine planning software, is known as the "nested shells" method. This method, which is based on the work or Lerchs and Grossman ([LG65]), is applicable to problems for which the side constraints are comprised of a single capacity constraint for each period and the processing method is fixed for each unit. It assumes that any block of ore is defined by a single value $v$, and that the objective

[1]This is true of the open pit mine planning problems that occur in the academic literature. In actual practice one can also find problems with a larger number of side constraints, though the number is still usually well under 1000. We will have more to say about such problems later on.

[2]The reader can verify that if, for example, units are cubes and the maximum slope angle is 45°, then the naive model in which for each cube $c$ there are five precedence cubes in the shape of a plus sign above $c$ would yield a square precedence cone rather than a circular precedence cone.

[3]Again, this is true of most problems in the academic literature, though in actual practice the integrality gaps are not always small.

value of that block in any period $t$ is the present value in period 1 of $v$ obtained in period $t$ subject to some discount rate.

In this method the problem is first converted to one in which there is only one scheduling period, the capacity constraints are ignored and the only decision is whether or not each unit of earth is to be extracted. The resulting problem is known as a maximum weight closure problem, or "Max Closure"[4]. Formally,

**Definition 1 : Max Closure Problem.**
*Given a directed graph $G$ whose vertices are weighted, find a maximum-weight subset of vertics $C$ with the property that there exist no arcs $(i, j)$ with $i \in C$ and $j \notin C$.*

A number of algorithms have been proposed for Max Closure, beginning with that of Lerchs and Grossman, and culminating in the very efficient method described in [H08] (also see [CH09]).

The solution to the max closure problem yields a collection $C^1$ of units. If the coefficient values of the units in the objective are monotonically increased then it can be shown that there exists an optimal solution $C^2$ to the new problem, for which $C^2 \supseteq C^1$. The standard method is to penalize every unit by some constant $\lambda$ times the capacity consumption of the unit (for some chosen capacity), and then to increase $\lambda$ parametrically. Repeating this procedure yields a sequence of nested sets

$$C^n \supseteq C^{n-1} \cdots C^2 \supseteq C^1 \tag{1}$$

for which it can be shown that the inner sets have a higher average value per unit of capacity consumption than the outer sets. For the purposes of present value optimization, these shells thus give a rough guide indicating that the inner shells should be extracted before the outer shells. The entire parametric analysis can be carried out efficiently with a parametric max flow algorithm ([HC00]).

This is obviously a very rough approach though, and it can be fairly useless when there are constraints other than capacity constraints, or when there are multiple capacity constraints, or multiple processing options. Moreover it is often the case that there is a huge jump in size between some set and the next set in the sequence, and the algorithm can give no guidance as to how to break up the difference between the sets into manageable chunks.

More recently, as commercial integer programming software has become more powerful, mine scheduling software packages have emerged that aggregate units of earth into a small number of large scheduling blocks in order to yield a mixed integer programming problem of tractable size. Nevertheless the degree of aggregation that is required in order to obtain a solvable problem can be enormous – combining thousands of units in the original formulation into a single aggregated unit – which can compromise the validity and the usefulness of the solution.

Other heuristic approaches have appeared in the open mine planning literature (see [HC00], [F06] and more recently, [NRCWE10] for overviews). Regarding exact approaches, Caccetta and Hill [CH03] announce a branch-and-cut based algorithm designed for large scale problems, but due to commercial confidentiality considerations no details of their method are provided.

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 [CEGMR12] which can solve the LP relaxation of even very large instances of the open pit mine scheduling

[4]In the mining industry this problem is known as the "ultimate pit problem".

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.

From a broad perspective, the method we give below uses dual information in order to effectively reduce the size of the linear program, which is a feature in common with [BDFG09], and in fact when our method is applied to the "Optimal Binning Problem" (which can be shown to be a special case of the precedence constrained production scheduling problem), the result is an algorithm broadly similar to theirs. The relationship between the max closure problem and the LP is a theme in common with the work of [CEGMR12].

## 1.2 Overview

In this paper we will present a new algorithm for provably solving the linear programming relaxation of the precedence constrained production scheduling problem. Our algorithm is applicable to problems with an arbitrary number of process options and arbitrary side constraints, and it requires no aggregation. We will show that the precedence constrained production scheduling problem can be reformulated as a problem in which all constraints needed to model the slope constraint and the multiple processing options are of the form

$$x_i \leq x_j. \tag{2}$$

A problem in which *all* constraints (other than the box constraints) are of this form is in fact a max closure problem, and can be solved as a minimum $s$–$t$ cut problem ([P76] and others). Thus the mine scheduling problem can be thought of as a minimum $s$–$t$ cut problem with a small number of side constraints. This structure makes the problem amenable to lagrangian based approaches, since by dualizing the side constraints we are left with a min cut problem, for which the aforementioned sizes are well within the reach of published algorithms.

The algorithm we will present iteratively solves a lagrangian relaxation, which it uses to generate a "small" LP (with a number of variables and constraints on the order of the number of side constraints, depending on the exact implementation[5]), whose solution is in turn used to update the lagrangian relaxation. The algorithm will be proven to converge, solving both the lagrangian and the original LP.

On very large, real-world instances our algorithm proves very efficient. In practice convergence is often in ten iterations or less, and thus the algorithm is suited to be embedded inside of a branch and bound algorithm to solve the original IP.

## 1.3 Roadmap

In Section 2 we will formally describe the precedence constrained production scheduling problem, and we show that it can be reformulated as a min cut problem with the same number of side constraints. We will refer to this problem as the "General Precedence Constrained Problem" (GPCP). This observation motivates the attempt to tackle our problem with some form of lagrangian formulation. In Section 3 we develop a general algorithmic framework in which the structure of an optimal primal solution is "guessed", and then an LP incorporating the guess is solved, supplying a candidate set of duals to be supplied to the lagrangian. If the guess is wrong, then the lagrangian solution with the given duals can be shown to detect this and to imply a modification of the guess, leading to a new – and strictly better – LP. The guess is thus a lever that connects the lagrangian and the primal problems, enabling the progress in the one to improve the other. The result is an algorithm in which lagrangian steps are iteratively coupled with LP steps to jointly optimize both the lagrangian and the primal.

[5]The number of constraints in the small LP could conceivably be quadratic in the number of side constraints, but it is typically of the same order as the number of variables in the small LP.

The algorithmic framework that we introduce – which we will refer to as an "Algorithm Template" – expands earlier work ([B02]) in which structure was algorithmically extracted from lagrangian solutions in the effort to obtain LP solutions. We will also indicate several other ways to characterize the template, including a geometric interpretation as a method to approximate the lagrangian epigraph at its minimizer. In the appendix we develop this approach more fully, proving thereby that the tempate converges pseudologorithmically in the case where only one constraint was dualized. We also show that one of the ways in which to implement the template is equivalent to column generation. In general the template is strictly monotonic, and under mild conditions it converges finitely.

In Section 3.4 we show that the structure of the GPCP affords a natural "combinatorial" implementation of the Algorithm Template. We define the algorithm and prove convergence. Along the way we also prove that given an extreme point solution $x$ to a min cut problem with $k$ side constraints, $x$ contains no more than $k$ distinct fractional values (we prove something stronger, actually), a result that plays an important role in the algorithm. In the appendix we also show that a similar but more general result holds for all problems whose constraint sets are totally unimodular matrices with the addition of side constraints, which may open the possibility that a similar algorithm can be developed for min cost network flow problems plus side constraints.

In Section 4 we describe a variation of the algorithm in which the solution iterates (i.e. the sequence of feasible points discovered by the algorithm) are all extreme points. We also describe a method by which the solution returned by the algorithm in its standard formulation can be converted to an extreme point solution by way of solving a single, typically small LP (i.e. of size on the order of the number of side constraints), or by solving a number of guaranteed small LP's.

In Section 5 we consider a variation of the algorithm in which rather than to obtain a single lagrangian solution at each iteration, we obtain *every* lagrangian solution. Using a result of Picard and Queyranne ([PQ80]) characterizing the full set of max closures of a graph, we show that it is often possible to do this fairly efficiently. This results in a nice graph theoretical characterization of the relationship between the full diversity of primal and lagrangian solutions, associated with the algebra of max closures. We have not however, managed to obtain clear practical benefits from this approach, and we discuss why this may be the case.

The underlying idea however, is that it may be possible to extract more information from the lagrangian solutions than is obtained by the vanilla implementation of the algorithm. This idea is one that we have been able to use profitably. In Section 6 we begin by describing a specialized version of the $PCPSP$ to which we refer as the "Parcel Assignment Problem" (which is essentially equivalent to the "Optimal Binning Problem" of [BDFG09]). In this problem the decisions as to *when* the jobs are to be performed are aggregated to a small number of "super" decisions, but the decisions as to *how* they are to be performed are left as is. This problem is instructive as it has a particularly natural implemenatation of the algorithm template (one which is, incidentally, quite similar to the method employed by [BDFG09] for this problem) insofar as in this case the shared structure between the lagrangian and the LP is quite obvious. It turns out however, that the information that can be extracted from the lagrangian in the PAP case, can also be extracted in the more general $PCPSP$ case, and thus the two items of information can be combined into a richer implementation of the algorithm. We then describe several other approaches along these lines by which more information can be discovered early, thus expediting convergence.

In section 7 we present the results of computational experiments, and finally in Section 8 we discuss directions for further research.

## 2 Precedence Constrained Production Scheduling and Maximum Closure

**Definition 2 : Precedence Constrained Production Scheduling Problem**.
*In this problem we are given a set of jobs to be processed over $T$ time periods, using a set of $F$ processing options (or "destinations"), while respecting a set of precedence relationships among the jobs as well as additional "side"-constraints, and at minimum cost. Each job must be processed in a*

*single period,*[6] *but its processing can be split among the various options. Formally,*

*(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 ($(i, j) \in \mathcal{A}$ means that job $j$ must be completed no later than job $i$).*

*(2) For each job $j$, time period $t$ and processing option $k$, the cost of processing job $j$ at time $t$ using option $k$ is denoted $c_{j,t,k}$.*

*(3) We denote by $x_{j,t,k} \in [0, 1]$ the fraction of job $j$ processed in period $t$ using option $k$, and by $y_{j,t} \in \{0, 1\}$ the decision to process job $j$ in period $t$; we assume the side-constraints can be stated in the form $Dx = d$, where the rows of $D$ are linearly independent.*

*Using this choice of decision variables the LP relaxation of the problem, which we will refer to as PCPSP, is as follows:*

$$\textbf{(PCPSP):} \qquad \max \; c^T x \tag{3a}$$

$$\textit{Subject to:} \quad \sum_{\tau=1}^{t} y_{i,\tau} \le \sum_{\tau=1}^{t} y_{j,\tau}, \quad \forall (i, j) \in \mathcal{A},\; 1 \le t \le T \tag{3b}$$

$$Dx = d \tag{3c}$$

$$y_{j,t} = \sum_{f=1}^{F} x_{j,t,f}, \quad \forall j \in \mathcal{N},\; 1 \le t \le T \tag{3d}$$

$$\sum_{t=1}^{T} y_{j,t} \le 1, \quad \forall j \in \mathcal{N} \tag{3e}$$

$$x \ge 0. \tag{3f}$$

Alternatively, we may represent the side-constraints as inequalities. In practice this is usually the more natural and convenient model, and in fact the algorithm will usually perform better this way for reasons we will point out later. Nevertheless, the analysis is simplified considerably by assuming that these are equalities and that they are linearly independent, and so we will follow this assumption throughout. Given a subset of vertices $S$ in a directed graph we denote by $\delta^+(S)$ the set of arcs $(i, j)$ of $G$ with $i \in S$ and $j \notin S$.

**Definition 3 : General Precedence Constrained Problem.**
*Given a directed graph $G = (\mathcal{N}, \mathcal{A})$ with $n$ vertices, a weight $c_j$ for each $j \in \mathcal{N}$, and a (possibly empty) system $Dx = d$ of linear constraints on $n$ variables, we consider the following linear program:*

$$\textbf{(GPCP):} \qquad \max \; c^T x \tag{4a}$$

$$Dx = d \tag{4b}$$

$$x_i - x_j \le 0, \quad \forall\, (i, j) \in \mathcal{A}, \tag{4c}$$

$$0 \le x_j \le 1, \quad \forall\, j \in \mathcal{N}. \tag{4d}$$

*If the system $Dx = d$ is empty then this problem is equivalent to the max closure problem on $G$ with weight vector $c$; see Definition 1.*

This problem is more general than PCPSP:

[6] This requirement makes the problem easier to describe, but it is relaxed in the LP, and it is not generally strictly required by the IP either.

**Lemma 4** *Any instance of PCPSP can be reduced to an equivalent instance of GPCP with the same number of $x$ variables and the same total number of constraints.*

*Proof.* Consider an instance of PCPSP on $G = (\mathcal{N}, \mathcal{A})$, with $T$ time periods, $F$ destinations and side constraints $Dx = d$. Using (3d) the $y$ variables can be eliminated. Consider the following system of inequalities on variables $z_{j,t,f}$ ($j \in \mathcal{N}$, $1 \le t \le T$, $1 \le f \le F$):

$$\begin{aligned}
z_{j,t,f} - z_{j,t,f+1} &\le 0, \quad \forall\, j \in \mathcal{N},\ 1 \le t \le T,\ 1 \le f < F, && \text{(5a)}\\
z_{j,t,F} - z_{j,t+1,1} &\le 0, \quad \forall\, j \in \mathcal{N},\ 1 \le t < T, && \text{(5b)}\\
z_{j,T,F} &\le 1, \quad j \in \mathcal{N}, && \text{(5c)}\\
z_{i,t,F} - z_{j,t,F} &\le 0, \quad \forall\, (i,j) \in \mathcal{A},\ 1 \le t \le T, && \text{(5d)}\\
z_{j,1,1} &\ge 0. && \text{(5e)}
\end{aligned}$$

Given a solution $(x, y)$ to PCPSP, we obtain a solution $z$ to (5) by setting, for all $j$, $t$ and $f$

$$z_{j,t,f} = \sum_{\tau=1}^{t-1} \sum_{f'=1}^{F} x_{j,\tau,f'} + \sum_{f'=1}^{f} x_{j,t,f'}. \tag{6}$$

Conversely given $z$ that satisfies (5) we can obtain a feasible solution $(x, y)$ to PCPSP by using (6) to write $x$ as a function of $z$. Thus, for an appropriate system $\bar{D}z = \bar{d}$ (with the same number of rows as $Dx = d$) and objective $\bar{c}^T z$, PCPSP is equivalent to the linear program:

$$\min\{\bar{c}^T z \ :\ \bar{D}z = \bar{d}, \text{ and constraints (5)}\}. \ \blacksquare$$

**Note:** The idea behind the transformation in Lemma 4 is that the variable $z_{j,t,f}$, which corresponds to a node $(j, t, f)$ in a GPCP, represents the proportion of job $j$ performed either in a period before $t$, or in period $t$ at a destination $h \le f$. This is enforced by drawing arcs from each node $(j, t, f)$ to the node $(j, t, f + 1)$, and from each node $(j, t, F)$ to the node $(j, t + 1, 1)$. The precedence constraints in period $t$ between jobs $i \to j$ in PCPSP are enforced by drawing arcs from node $(i, t, F)$ to $(j, t, F)$, indicating that the proportion of job $i$ performed by the end of period $t$ is less or equal to the proportion of job $j$ performed by the end of period $t$. Note also that though 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.

**Observation 5** *Given an instance of problem GPCP let $\mu \ge 0$ be a vector of dual variables for the side-constraints (4b). Then the Lagrangian problem obtained by dualizing (4b) using $\mu$,*

$$\begin{aligned}
\max \quad & c^T x + \mu^T(d - Dx) && \text{(7a)}\\
\textit{Subject to:} \quad & x_i - x_j \le 0, \ \forall\ (i,j) \in \mathcal{A} && \text{(7b)}\\
& 0 \le x_j \le 1, \ \ \forall\, j \in \mathcal{N}. && \text{(7c)}
\end{aligned}$$

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

**Observation 6** *Given an instance of PCPSP consider the Lagrangian problem obtained using a given set of penalties $\mu$ corresponding to the side constraints. As per Lemma 4 and Observation 5 this Lagrangian problem is a max closure problem with node weights $c - D^T\mu$, i.e. penalized values $c$. It is easy to see from the transformation in Lemma 4 that in this max closure problem all of the nodes corresponding to triples $(j, t, k)$ arising from a given job $j$ and period $t$ can be replaced by a single node corresponding to a destination $h$ that maximizes $\sum_{k \ge h}[c - D^T\mu]_{j,t,k}$.*

**Observation 7** *The feasible space of the linear relaxation of the max closure problem, $\max w^T x$ : $x_i - x_j \le 0,\ \forall (i,j) \in A,\ 0 \le x \le 1$ is defined by a totally unimodular matrix with integer right-hand side and integer bounds and is therefore an integral polytope.*

The following result, together with Observation 7, shows that GPCP with no side constraints (and thus the Lagrangian problem obtained by dualizing the side-constraints of GPCP) can be solved with a min-cut algorithm.

**Theorem 8** *[P76] Given a graph $G$ with weights $w_n$ associated with each node in the graph, define the graph $G'$ as follows: Add a node $s$ to the graph and an arc $(s, n)$ with capacity $w_n$ for each node $n$ for which $w_n \geq 0$, and add a node $t$ to the graph and an arc $(n, t)$ with capacity $-w_n$ for each node $n$ for which $w_n < 0$. The $s$ side of a minimum $s$–$t$ cut in $G'$ constitutes a maximum value closure in $G$ w.r.t. $w$.*

Other constructions that demonstrate the reducibility of max closure to max-flow or min-cut can be found in [J68], [Bal70] and [R70]. Further discussion can be found in [HC00], where the authors note (at the end of Section 3.4) that max closure with an added cardinality constraint is an NP-hard problem.

## 3 Solving GPCP

### 3.1 LP solutions, Lagrangian Solutions and Decomposition Theorems

Observation 5 together with Theorem 8 suggests that a Lagrangian relaxation algorithm for solving GPCP – i.e., an algorithm that iterates by solving problems of the form (7) – would enjoy fast individual iterations. This is correct: it has been observed in the literature that extremely large max closure instances can be solved quite fast using the appropriate min-cut algorithms and our experiments confirm this fact. However, in our experiments we also observed that the application of traditional Lagrangian relaxation methods (such as subgradient optimization) to GPCP performs quite poorly, requiring vast numbers of iterations and frequently not converging to solutions with desirable accuracy.

Our approach, instead, relies on leveraging combinatorial structure that optimal solutions to GPCP must satisfy. We do this however within a more general algorithmic framework. The preliminary idea that we will develop in this section is that for any primal feasible solution, it is possible to associate a *family* of "Lagrangian equivalent" solutions (which may include primal infeasible solutions). These are defined by a decomposition which we will characterize here, and which will play a fundamental role in the analysis that follows.

**Notation 9** *Let $Dx = d$ be a fixed linear system with $D \in \mathbb{R}^{q \times n}$ of full row rank. Given a polyhedron*

$$\mathcal{P} \ = \ \mathcal{P}(A, b) \ \doteq \ \{x \in \mathbb{R}^n : Ax \leq b, \ Dx = d\}, \tag{8}$$

*(1) We write*

$$\mathbf{LP}(\mathcal{P}) : \quad \max\{c^T x : x \in \mathcal{P}\}.$$

*(2) For each vector $\mu \in \mathbb{R}^q$, the Lagrangian function defined by dualizing $Dx = d$ with penalties $\mu$ is the affine function*

$$L_\mu(x) \ \doteq \ (c^T - \mu^T D)x \, + \, \mu^T d.$$

*We will also write $\mathbf{L}(\mu)$ to denote the optimization problem*

$$\max\{L_\mu(x) \, : \, Ax \leq b\}, \tag{9}$$

*and denote the value of $\mathbf{L}(\mu)$ by $L^*_\mu$.*

*(3) We will say that a vector $\mu$ of duals for $Dx = d$ is optimal for the dual of $\mathbf{LP}(\mathcal{P})$ if there exists a vector $\lambda$ of duals for $Ax \leq b$ such that the pair $(\lambda, \mu)$ is an optimal vector for the dual of $\mathbf{LP}(\mathcal{P})$ (and likewise with $\lambda$).*

*(4) For any given $x \in \mathbb{R}^n$, let $I_x^=$ denote the set of rows $i$ of $A$ such that $a_i^T x = b_i$ (where $a_i^T$ denotes $i^{th}$ row of $A$) and write $A_x^=$ and $b_x^=$ to denote the corresponding submatrix of $A$ and subvector of $b$, respectively. We will drop the superscript when the dependence is clear. The null space of $A_x^=$ will be denoted $N_A^x$.*

*(5) Given a vector $\mu$ of duals for the constraints $Dx = d$, and a vector $y \in \mathbb{R}^n$, the penalized value of $y$ (with respect to $\mu$) is $(c^T - \mu^T D)y$.*

In Lemmas 10 - 14 we consider a fixed polyhedron $\mathcal{P} = \mathcal{P}(A, b)$.

**Lemma 10** *Let $\hat{x}$ be an extreme point of $\mathcal{P}$. Then $dim(N_A^{\hat{x}}) \leq q$.*

**Proof:** We must have $\text{rank}(A_{\hat{x}}^=) \geq n - q$ from which the result follows. ■

**Lemma 11** *Let $\hat{x} \in \mathcal{P}$, let $x^i \in \mathbb{R}^n$ satisfy $A_{\hat{x}}^= x^i = b_{\hat{x}}^=$, and let $\Theta$ be a matrix whose columns span $N_A^{\hat{x}}$. Denote by $s$ the number of columns of $\Theta$. Then*
*(1) There exists $\alpha \in \mathbb{R}^s$ such that*

$$\hat{x} = x^i + \Theta\alpha. \tag{10}$$

*(2) If the columns of $\Theta$ are linearly independent and $\hat{x}$ is an extreme point of $\mathcal{P}$, then $s \leq q$.*

**Proof:** The first statement results from the fact that $\bar{A}^{\hat{x}}(\hat{x} - x^i) = 0$, and the second statement follows from Lemma 10. ■

We will see later that in the case of GPCP particular structure of the $A$ matrix gives rise to a combinatorial interpretation of Lemma 11 (additional discussions in the Appendix).

**Lemma 12** *Let $\breve{x} \in \mathbb{R}^n$ satisfy $D\breve{x} = d$. Suppose we can write $\breve{x} = x^i + \Theta\alpha$ for some $x^i \in \mathbb{R}^n$, $s \geq 1$, $\Theta \in \mathbb{R}^{n \times s}$ and $\alpha \in \mathbb{R}^s$. Suppose $\mu \in \mathbb{R}^q$ satisfies $c^T\Theta = \mu^T D\Theta$. Then*

$$c^T\breve{x} = L_\mu(x^i). \tag{11}$$

**Proof:**

$$c^T\breve{x} = c^T x^i + c^T\Theta\alpha = c^T x^i + \mu^T D\Theta\alpha = c^T x^i + \mu^T D(\breve{x} - x^i) = c^T x^i - \mu^T(Dx^i - d) \tag{12}$$

$$= L_\mu(x^i). \tag{13}$$

■

**Remark:** In other words, for any dual vector $\mu$ such that the penalized value of each column of $\Theta$ (w.r.t. $\mu$) is zero, the objective value of $\breve{x}$ equals $L_\mu(x^i)$.

By linear programming duality if $\breve{x} \in \mathcal{P}$ and $\mu$ is a vector of duals corresponding to the constraints $Dx = d$, then $\breve{x}$ and $\mu$ are optimal (for $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ and its dual, respectively) if and only if $L_\mu(\breve{x}) = L_\mu^*$. The following results generalize this fact to the context of the decomposition in Lemma 11.

**Lemma 13** *Let $\hat{x} \in \mathcal{P}$, let $\Theta$ be a matrix each of whose columns is in $N_A^{\hat{x}}$, and suppose $\hat{\mu} \in \mathbb{R}^q$ is a vector of duals for $Dx = d$. If $\hat{x}$ and $\hat{\mu}$ are optimal for $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ and its dual, respectively, then $\hat{\mu}^T D\Theta = c^T\Theta$.*

**Proof:** If $\hat{\lambda}$ is any optimal vector of duals for constraints $Ax \leq b$, we have $c^T = \hat{\mu}^T D + \hat{\lambda}^T A$, and so

$$c^T\Theta = \hat{\mu}^T D\Theta + \hat{\lambda}^T A\Theta = \hat{\mu}^T D\Theta, \tag{14}$$

as desired . ■

**Lemma 14** *Let $\breve{x}$, $x^i$, $\alpha$ and $\Theta$ be as in Lemma 11, and suppose $\mu \in \mathbb{R}^q$ is a vector of duals for $Dx = d$. Then $\breve{x}$ and $\mu$ are optimal for $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ and its dual, respectively, if and only if $\mu^T D\Theta = c^T\Theta$ and $L_\mu(x^i) = L_\mu^*$.*

**Proof:** Suppose that $\breve{x}$ and $\mu$ are optimal for $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ and its dual. By Lemma 13

$$c^T\Theta = \mu^T D\Theta, \tag{15}$$

and by Lemma 12 $L_\mu(x^i) = c^T\breve{x} = L^*_\mu$. Conversely if $L_\mu(x^i) = L^*_\mu$ and $\mu^T D\Theta = c^T\Theta$, then by Lemma 12 $c^T\breve{x} = L_\mu(x^i)$, which by assumption equals $L^*_\mu$, establishing that $(\breve{x}, \mu)$ is an optimal pair. ■

Informally, the preceding results show that the columns of $\Theta$, i.e. the vectors that span $N^{\breve{x}}_A$, are interesting for a number of reasons:

(N.1) They preserve the set of binding constraints for the Lagrangian problem, i.e. for all $x$ and $\alpha$, $A^=_{\breve{x}}x = A^=_{\breve{x}}(x + \alpha\Theta)$.

(N.2) They preserve the Lagrangian objective value, i.e. where $\mu$ is such that $c^T\Theta = \mu^T D\Theta$, $L_\mu(x) = L_\mu(x + \Theta\alpha)$.

(N.3) If $D\Theta$ is invertible, there is a a unique $\mu$ with $c^T\Theta = \mu^T D\Theta$, namely $\mu^T = c^T\Theta(D\Theta)^{-1}$ .

Items (N.1) – (N.3) state that if $D\Theta$ is invertible, then $\Theta$ determines $\mu$, and the family of vectors $\{\breve{x} + \Theta\alpha\}$ are "equivalent'' to $\breve{x}$ in the sense that they have a common Lagrangian objective value and attain constant $A^=_{\breve{x}}x$ value (and so are also feasible for $L_\mu$ for all $\alpha$ within a neighborhood of 0). For this reason we term the columns of $\Theta$ **Lagrangian neutral vectors**. Note that if $D\Theta$ is invertible, then for any vector $x'$, we can find $\alpha'$ so that $\tilde{x} = x' + \Theta\alpha'$ satisfies $D\tilde{x} = d$, $A^=_{\breve{x}}x' = A^=_{\breve{x}}\tilde{x}$ and $L_\mu(x') = L_\mu(\tilde{x})$. In other words, if $D\Theta$ is invertible then it is always possible to move from any point $x'$ along lagrangian neutral vectors (thus maintaining lagrangian value and satisfaction of $A^=_{\breve{x}}$ constraints) to a new point $\tilde{x}$ that will also satisfy the $D$ constraints. We will see this more formally in the following section.

## 3.2 An Algorithmic Template

We begin with the following algorithmic idea used to address a problem $\mathbf{LP}(\boldsymbol{\mathcal{P}})$. Suppose we have made a guess as to some valid property that an optimal solution to $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ will satisfy, and we can express this property as $Hx = h$. We could then solve the more constrained system

$$P_2(H, h): \qquad \max\{cx \;:\; Ax \le b,\; Hx = h,\; Dx = d\}.$$

This approach can prove useful if problem $P_2(H, h)$ is significantly easier than $\mathbf{LP}(\boldsymbol{\mathcal{P}})$, and if the process of guessing does not entail a heavy computational load.

In our implementation we use an approximate guess, so that the system $Hx = h$ may not be truly valid (i.e. not satisfied by an optimal solution to $\mathbf{LP}(\boldsymbol{\mathcal{P}})$), and then correct the guess to make it "more" valid. To perform this correction we can use an optimal primal and dual solution to problem $P_2 = P_2(H, h)$. Thus, let $\hat{x}$ be optimal for $P_2$, and let $\hat{\mu}$ be optimal dual variables for $Dx = d$ (in $P_2$). If the guess is incorrect then $L^*_{\hat{\mu}} > c^T\hat{x}$, (or else, by weak linear programming duality, $\hat{x}$ would be optimal for $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ and the guess would have been correct). We will show soon that under mild conditions the converse also holds (i.e. if the guess is correct then $L^*_{\hat{\mu}} = c^T\hat{x}$). When the guess is incorrect we can still evaluate its merits and improve upon it by relying on the Lagrangian function; we will assume that the Lagrangian relaxations (9) are "easy" to solve.

Thus, assume $L^*_{\hat{\mu}} > c^T\hat{x}$ and let $\bar{x}$ be any optimal solution to $\mathbf{L}(\hat{\mu})$. Observe that $\bar{x}$ cannot satisfy $H\bar{x} = h$, because if it did we would have

$$\max\{(c^T - \hat{\mu}^T D)x + \hat{\mu}^T d \;:\; Ax \le b,\; Hx = h\} \;\ge\; (c^T - \hat{\mu}^T D)\bar{x} + \hat{\mu}^T d \;=\; L^*_{\hat{\mu}} \;>\; c^T\hat{x},$$

while on the other hand since $(\hat{x}, \hat{\mu})$ is an optimal pair for $P_2$ we also have

$$c^T\hat{x} \;=\; \max\{(c^T - \hat{\mu}^T D)x + \hat{\mu}^T d \;:\; Ax \le b,\; Hx = h\},$$

a contradiction.

Intuitively, the fact that $\bar{x}$ does not satisfy $H\bar{x} = h$ exposes a flaw in the guess, as the whole idea of the $H$ constraints is a guess as to constraints that hold at an optimal solution, and we see here that they do indeed cut off the optimal solution $\bar{x}$ for the lagrangian with the optimal duals that result from the guess. The natural correction then is to modify $H$ so as not to cut off $\bar{x}$. While admittedly this assessment is based on the duals that emerged from an overly restrictive guess, they still may provide useful, if approximate, information as to how the guess ought to be modified. We will in fact show below in Theorem 17 that, under mild conditions, relaxing the constraints $Hx = h$ into a system $\bar{H}x = \bar{h}$ such that $\bar{H}\bar{x} = \bar{h}$ always leads to a feasible improving direction for $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ at $\hat{x}$.

First we present some auxiliary results, assuming a given polyhedron $\mathcal{P} = \mathcal{P}(A, b)$ of the form (8).

**Lemma 15** *Let $y \in \mathcal{P}$ and $\Theta$ be a matrix with columns in $N_A^y$. Suppose $D\Theta$ is nonsingular, and set $\mu^T = c^T\Theta(D\Theta)^{-1}$. Finally, let $x_\mu \in \mathbb{R}^n$ be such that $Ax_\mu \le b$. Then there exists a vector $\breve{x}$ of the form $\breve{x} = x_\mu + \Theta\alpha$ (for some $\alpha$) satisfying $A_y^= \breve{x} \le b_y^=$, $D\breve{x} = d$. Thus $\breve{x}$ satisfies all the constraints for* $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ *that are binding at $y$. Moreover, $c^T\breve{x} = L_\mu(\breve{x}) = L_\mu(x_\mu)$.*

**Proof:** Let $\alpha$ be such that $Dx_\mu + D\Theta\alpha = d$ and let $\breve{x} = x_\mu + \Theta\alpha$. Then

$$A_y^= \breve{x} = A_y^= x_\mu + A_y^= \Theta\alpha = A_y^= x_\mu \le b_y^=, \quad \text{and by construction}$$

$$D\breve{x} = d.$$

By Lemma 12, $L_\mu(x_\mu) = c^T\breve{x}$, which equals $L_\mu(\breve{x})$ since $D\breve{x} = d$. ■

**Remark 16** *Under the conditions of Lemma 15, there exists a scalar $\gamma > 0$ such that for all $0 < \gamma' \le \gamma$, $y + \gamma'(\breve{x} - y)$ is feasible for* $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ *and has objective (and Lagrangian) value $c^T y + \gamma'(L_\mu(x_\mu) - c^T y)$. Thus, if $L_\mu(x_\mu) > c^T y$, there is a feasible improving direction along which, additionally, we have $L_\mu(x) = c^T x$ (as $Dx = d$ holds for all $x$ along the direction).*

We will now prove the theorem.

**Theorem 17** *Let the system $Hx = h$ be given, let $\hat{x}$ be an optimal solution of $P_2(H, h)$, let $\Theta^H$ be a matrix with columns in the null space of $\binom{A_{\hat{x}}^=}{H}$, and let the system $\bar{H}x = \bar{h}$ be such that*

$$\{x \in \mathbb{R}^n : \bar{H}x = \bar{h}\} \supseteq \{x \in \mathbb{R}^n : Hx = h\}. \tag{16}$$

*Assume $D\Theta^H$ is nonsingular, and let $\mu^T \doteq c^T\Theta^H(D\Theta^H)^{-1}$ and $x_\mu \in \operatorname{argmax}\{L_\mu : Ax \le b\}$. We assume:*

*(a) $\bar{H}x_\mu = \bar{h}$.*

*Then:*
**(1)** *Suppose $L_\mu(x_\mu) > c^T\hat{x}$. Then, at $\hat{x}$,*

*(i) There is a strictly improving direction for $c^T x$ which is feasible for $P_2(\bar{H}, \bar{h})$ and*

*(ii) Along this direction $L_\mu(x) = c^T x$ holds.*

**(2)** *Suppose $\hat{x}$ is optimal for* $\mathbf{LP}(\boldsymbol{\mathcal{P}})$ *as well. Then $L_\mu(x_\mu) = c^T\hat{x}$ and thus $\mu$ is an optimal dual vector for* $\mathbf{LP}(\boldsymbol{\mathcal{P}})$.

**Proof:** **(1)** Note first that the null space of $H$ is a subset of the null space of $\bar{H}$. The proof is as follows: Assume there exists a vector $\theta$ with $H\theta = 0$ but $\bar{H}\theta \ne 0$. Considering that $\{x : Hx = h\} \ne \emptyset$ (it contains $\hat{x}$), let $y$ belong to this set, and so $y$ also belongs to $\{x : \bar{H}x = \bar{h}\}$ by (16). Thus $y + \theta$ belongs to $\{x : Hx = h\}$ but not to $\{x : \bar{H}x = \bar{h}\}$, which contradicts (16). It follows therefore that $\bar{H}\Theta^H = 0$. Observe now that $\hat{x} \in \mathcal{P}$, that the columns of $\Theta^H$ all belong to $N_A^{\hat{x}}$, and that $Ax_\mu \le b$.

Letting $\Theta = \Theta^H$, we can therefore apply Lemma 15 with the $x_\mu$ of Lemma 15 chosen the same as $x_\mu$ here, to obtain the vector $\breve{x} = x_\mu + \Theta^H \alpha$ as defined there, and $L_\mu(\breve{x}) = L_\mu(x_\mu)$. Consider the direction $\delta$ from $\hat{x}$ to $\breve{x}$. By Remark 16, $\delta$ is strictly improving for $c^T x$ and all points $x$ along $\delta$ satisfy $L_\mu(x) = c^T x$. To complete the proof we will show that all $x$ along $\delta$ satisfy all constraints for $P_2(\bar{H}, \bar{h})$ that are binding at $\hat{x}$, namely $A^=_{\hat{x}} x \le b^=_{\hat{x}}$, $Dx = d$ and $\bar{H}x = \bar{h}$. The former two follow directly from Lemma 15. To prove the latter we must show that the opposite endpoint of $\delta$, namely $\breve{x}$, also satisfies $\bar{H}\breve{x} = \bar{h}$. This is shown as follows:

$$\bar{H}\breve{x} = \bar{H}x_\mu + \bar{H}\Theta^H\alpha = \bar{H}x_\mu = \bar{h}. \tag{17}$$

**(2)** If $L_\mu(x_\mu) > c^T\hat{x}$ then by **(1)** $\hat{x}$ would not be optimal for $\mathcal{P}$. ■

**Remarks:**
**1**. Since $L_\mu(x_\mu)$ is an upper bound on $\mathbf{LP(\mathcal{P})}$, Theorem 17 and Corollary 16 show that $P_2(\bar{H}, \bar{h})$ contains points that are at least the fraction $\gamma$ of the way from the value at $\hat{x}$ to optimality.
**2**. Using (ii), the improving direction is also improving for the Lagrangian value, and by the same amount.

Informally, the procedure suggested by Theorem 17 for solving $\mathbf{LP(\mathcal{P})}$ is as follows. Given constraints $Hx = h$ (these are the constraints we currently "guess" to hold at optimality),

- Solve the problem $P_2(H, h)$ to get optimal primal $\hat{x}$, and optimal dual $\bar{\mu}$.
- Get $x_{\bar{\mu}}$ maximizing the lagrangian $L(\bar{\mu})$.
- Check if $x_{\bar{\mu}}$ satisfies $L_\mu(x_\mu) = c^T\hat{x}$. If so, then we are done, otherwise $x_{\bar{\mu}}$ violates $Hx = h$. Relax $Hx = h$ to $\bar{H}x = \bar{h}$, where $(\bar{H}, \bar{h})$ is chosen so that $\bar{H}x_{\bar{\mu}} = \bar{h}$. Replace $(H, h)$ with $(\bar{H}, \bar{h})$, and repeat.

Theorem 17 implies that if at each step of this procedure, $\Theta^H$ is a matrix with columns in the null space of $\binom{A^=_{\hat{x}}}{H}$, and $D\Theta^H$ is nonsingular, then this procedure will yield a strictly monotonically improving sequence of solutions to $\mathbf{LP(\mathcal{P})}$. The next set of results will show that with minor changes to the problem, $D\Theta^H$ can in fact be assumed to be nonsingular. We remind the reader that $q$ denotes the number of rows of $D$.

**Definition 18 : Random model.** *Let $\epsilon > 0$ be small and $\bar{d} \in \mathbb{R}^q$. A polyhedron $\mathcal{P} = \mathcal{P}(A, b)$ (see eq. (8)) where $d$ is chosen* **randomly***, with uniform probability, from within the ball*

$$B(\bar{d}, \epsilon) \doteq \{d \in \mathbb{R}^q \,:\, ||d - \bar{d}|| < \epsilon\}$$

*will be called a random instance.*

**Lemma 19** *Let $y \in \mathcal{P}$, let $\breve{A}$ be a submatrix of $A^=_y$ (with corresponding subset of rows $\breve{I}$ and subvector $\breve{b}$ of $b$) such that $\binom{\breve{A}}{D}$ is nonsingular, and let $\breve{\Theta}$ be a matrix each of whose columns is contained in the null space of $\breve{A}$. With $d$ chosen under the random model, the probability that $a_i^T\breve{\Theta} \neq 0$ for any $i \in I^=_y - \breve{I}$ is zero.*

*Proof.* Choose any $i \notin \breve{I}$. Then there exist unique vectors $\lambda_1, \lambda_2$ such that $\lambda_1^T\breve{A} + \lambda_2^T D = a_i^T$. If, additionally, $a_i^T\breve{\Theta} \neq 0$ then $\lambda_2 \neq 0$. If furthermore $i \in I^=_y$ then $\lambda_1^T\breve{b} + \lambda_2^T d = b_i$. This completes the proof since, with $\lambda_2 \neq 0$, under the random model the measure of the set $\{d \in B(\bar{d}, \epsilon) \,:\, \lambda_1^T\breve{b} + \lambda_2^T d = b_i\}$ is zero. ■

**Lemma 20** *Let $\breve{A}$ be a submatrix of $A$ such that $\binom{\breve{A}}{D}$ is $n \times n$ and nonsingular, and let $\breve{\Theta} \in \mathbb{R}^{n \times s}$ be a matrix whose columns are linearly independent and such that $\breve{A}\breve{\Theta} = 0$. Then (1) $s \le q$ and (2) the columns of $D\breve{\Theta}$ are linearly independent, and so if $s = q$ then $D\breve{\Theta}$ is $q \times q$ and invertible.*

**Proof:** Since $D$ has $q$ rows it follows that $\breve{A}$ has $n - q$ rows, which by assumption are linearly independent, and so $s \leq q$. Next we prove that the columns of $D\breve{\Theta}$ are linearly independent. Thus, suppose $\pi \in \mathbb{R}^s$ is such that $D\breve{\Theta}\pi = 0$. Then $\binom{\breve{A}}{D}\breve{\Theta}\pi = 0$, and since $\binom{\breve{A}}{D}$ is nonsingular, we have $\breve{\Theta}\pi = 0$, which implies $\pi = 0$. ■

**Lemma 21** *Let $\hat{x} \in \mathcal{P}$ be an extreme point and let $\Theta$ be a matrix whose columns form a basis for the null space of $A^{=}_{\hat{x}}$. Then under the random model, $D\Theta$ is invertible with probability* 1.

**Proof:** Since $\hat{x}$ is an extreme point, there exists a submatrix $\breve{A}$ of $A^{=}_{\hat{x}}$ such that $\binom{\breve{A}}{D}$ is $n \times n$ and invertible; thus every column of $\Theta$ belongs to the null space of $\breve{A}$. If we can prove that $\Theta$ is a basis for the null space of $\breve{A}$ then it will follow (since $\breve{A}$ has $n-q$ rows) that $\Theta$ has $q$ columns, and so $D\Theta$ is invertible by Lemma 20. Thus suppose $\theta$ is in the null space of $\breve{A}$. Applying Lemma 19 with $y = \hat{x}$, and $\breve{\Theta}$ the matrix with the single column $\theta$, we have that under the random model $A^{=}_{\hat{x}}\theta = 0$ with probability 1. But in that case by assumption $\theta$ is indeed spanned by the columns of $\Theta$, as desired. ■

**Lemma 22** *If* **LP($\mathcal{P}$)** *has on optimal solution, then under the random model, with probability 1 there is a unique optimal dual $\hat{\mu}$ corresponding to $Dx = d$.*

**Proof:** Let $\hat{x}$ be an extreme point optimal solution to **LP($\mathcal{P}$)** and let $\Theta$ be a matrix whose columns form a basis for the null space of $A^{=}_{\hat{x}}$. By Lemma 13, if $\hat{\mu}$ is dual optimal for the constraints $Dx = d$,

$$\hat{\mu}^T D\Theta \;=\; c^T\Theta.$$

This completes the proof, since by Lemma 21, $D\Theta$ is invertible with probability 1 under the random model. ■

Note now that each problem $P_2(H, h)$ is also of the form **LP($\mathcal{Q}$)**, with $\mathcal{Q}$ defined as

$$\mathcal{Q} \;\doteq\; \{x \,:\, Ax \leq b,\; Hx = h,\; Dx = d\} \;=\; \mathcal{P}(\hat{A}, \hat{b}), \quad \text{where} \tag{18}$$
$$\hat{A} = \begin{pmatrix} A \\ H \\ -H \end{pmatrix} \quad \text{and} \quad \hat{b} = \begin{pmatrix} b \\ h \\ -h \end{pmatrix}.$$

Thus applying Lemma 21 to **LP($\mathcal{Q}$)**, we conclude that under the random model, $D\Theta^H$ in Theorem 17 can be assumed to be nonsingular,[7] so long as the method used for solving the problems $P_2(H, h)$ returns an extreme point solution, and so long as we assume that the pool $\mathcal{H}$ from which the $H$ constraints are drawn is fixed and finite. The reason we must assume the latter is that the $H$ constraints belong to the category of the $A$ constraints of $P$, and therefore if the selection of the $H$ constraints is solely opportunistic, then the probability argument will fail.

The next theorem shows that the usefulness of having $D\Theta$ nonsingular goes beyond the fact that it ensures monotonicity by Theorem 17.

**Theorem 23** *Let $\hat{x}$ be an optimal solution to* **LP($\mathcal{P}$)***, and assume that $\hat{x}$ satisfies some constraints $Hx = h$. Let $\Theta$ be a matrix whose columns form a basis of $N^{\hat{x}}_A$, and let $\Theta^H$ be a matrix whose columns form a basis of the null space of $\binom{A^{=}_{\hat{x}}}{H}$.*

*(1) If $D\Theta^H$ is nonsingular, then $\mu = c^T\Theta^H(D\Theta^H)^{-1}$ is the unique optimal dual solution to* **LP($\mathcal{P}$)***.*

[7]While in principle one can usually perturb the rhs microscopically without affecting the usefulness of the result, there is in practice some price to doing so, as due to the finite precision of computers, the perturbation cannot be arbitrarily small.

*(2) If the constraints $(H, h)$ belong to a fixed finite pool $\mathcal{H}$, and $\hat{x}$ is an extreme point solution to* $\mathbf{LP}(\mathcal{P})$*, then under the random model, with probability* 1*, $D\Theta$ and $D\Theta^H$ are nonsingular, the column space of $\Theta$ is the same as that of $\Theta^H$, and $\tilde{\mu} = c^T\Theta(D\Theta)^{-1} = c^T\Theta^H(D\Theta^H)^{-1}$ is the unique optimal dual solution to both* $\mathbf{LP}(\mathcal{P})$ *and $P_2(H, h)$.*

**Proof:** (1) Note first that $\hat{x}$ is obviously also an optimal solution to $P_2(H, h)$, in other words an optimal solution to $\mathbf{LP}(\mathcal{Q})$, with $\mathcal{Q}$ defined as in (18). Applying Lemma 13 to $\mathbf{LP}(\mathcal{Q})$ implies that $\mu$ is the unique dual-optimal for the constraints $Dx = d$ for $P_2(H, h)$. Now let $\bar{\mu}$ be a vector of duals for $Dx = d$ that is optimal for $\mathbf{LP}(\mathcal{P})$. Then $\bar{\mu}$ must be optimal for $P_2(H, h)$ as well, since

$$\begin{aligned} c^T\hat{x} &\leq \max\{(c^T - \bar{\mu}^T D)x + \bar{\mu}^T d : Ax \leq b, Hx = h\} \\ &\leq \max\{(c^T - \bar{\mu}^T D)x + \bar{\mu}^T d : Ax \leq b\} = L^*_{\bar{\mu}} = c^T\hat{x}. \end{aligned}$$

But since the unique optimal dual for $P_2(H, h)$ is $\mu$, we conclude that $\bar{\mu} = \mu$.
(2) Noting that $\hat{x}$ is an extreme point solution of $P_2(H, h)$ as well, it follows from Lemma 21 that with probability 1, both $D\Theta$ and $D\Theta^H$ are nonsingular. This implies that both $\Theta$ and $\Theta^H$ have the same number of columns, and thus that the null spaces of $N_A^{\hat{x}}$ and $\binom{A_{\hat{x}}^{=}}{H}$ are the same. By Lemma 13, $c^T\Theta(D\Theta)^{-1}$ is the unique optimal dual solution to $\mathbf{LP}(\mathcal{P})$, and $c^T\Theta^H(D\Theta^H)^{-1}$ is the unique optimal dual solution to $P_2(H, h)$, and since we have already shown that any dual solution of the former is a dual solution of the latter, these must be the same. ■

The significance of Theorem 23 is that it shows that the dual solution obtained at any iteration of the procedure outlined above is actually also the unique optimal dual solution of *every* relaxation of $P_2(H, h)$ (restricting $\mathbf{LP}(\mathcal{P})$) for which the optimal solution value does not increase. The second statement of the theorem shows that after solving a problem of the form $P_2(H, h)$ to obtain a solution $\hat{x}$, we can tighten the formulation of $H$ to anything that does not cut off $\hat{x}$ without altering the unique optimal dual solution. Essentially this indicates that the optimal dual is very robust. We will also use this fact to strengthen the procedure outlined above to allow such tightenings to take place before the third step, thus eliminating the requirement in the third step that the $\bar{H}$ constraints must be a relaxation of the $H$ constraints. The robustness of the dual, particularly in the face of these tightenings, can have great significance for the performance of the algorithm.[8]

It is convenient for us to prove one more result here, which we will need shortly.

**Lemma 24** *Assume* $\mathbf{LP}(\mathcal{P})$ *has an optimal solution. Let $x$ be a feasible solution of* $\mathbf{LP}(\mathcal{P})$ *and let $\Theta$ be a matrix whose columns form a basis for the null space of $A_x^{=}$. Then the columns of $D\Theta$ are linearly independent if and only if $x$ is an extreme point solution.*

**Proof:** Suppose $x$ is an extreme point solution, and let $\breve{A}$ be as in Lemma 19. Note that whatever is in the null space of $A_x^{=}$ is also in the null space of $\breve{A}$, so we can always choose $\breve{\Theta}$ in Lemma 19 to

[8]This is plain to see in the case where the restricted problem's primal solution is optimal for the unrestricted problem, as in this case this property will guarantee dual optimality as well (since the primal solution remains, in this case, an optimal solution for the original problem, it guarantees that the restricted problem's dual solution is the unique dual optimal of the original problem). More generally though, it indicates that the addition of $H$ constraints created no excess ***ambiguity*** in the choice of the dual. By this we refer to the fact that the definition of the optimal dual, by Lemma 14, is $\mu$ such that $\mu D\Theta = c\Theta$ and such that $x^i$ maximizes $L(\mu)$, where the primal optimal solution $x$ is decomposed as $x = x^i + \Theta\alpha$. These conditions are weakened by appending $H$ constraints, and this creation of ambiguity can be expected to cause problems. But where $D\Theta$ is nonsingular, the optimal dual space remains unaltered by $H$ constraints that do not reduce the optimal objective, and no excess ambiguity is introduced.

Moreover it may be the case that if the optimal dual space is a polyhedron rather than a single point, then extreme point solutions of this polyhedron may very well take advantage of balancing that exists within the restricted problem to move far out to a corner of the polyhedron, while these balances may not exist or may be slightly different in the unrestricted problem, and thus these solutions may be highly suboptimal for the unrestricted lagrangian. Note also that if $D\Theta$ is nonsingular then $\mu = (D\Theta)^{-1}c\Theta$, and so $\mu$ is defined by problem data and is insulated from being drawn out to extremes.

In particular, a large dual solution obtained due to ambiguity resulting from the fact that the restricted problem is not identical to the original problem is highly likely to be very far from the Lagrangian function's minimizer. In practice, for the GPCP, we have found that it is often the case that when $D\Theta^H$ is singular, then the duals that result are very large, particularly after tightening are performed, and convergence is seriously degraded.

have $\Theta$ as a submatrix. Thus since the columns of $D\breve{\Theta}$ are linearly independent by Lemma 20, it follows that the columns of $D\Theta$ are also. Suppose now that $x$ is not an extreme point. Then there must exist some vector $h \neq 0$ such that $A_x^= h = 0$ and such that $Dh = 0$. But if $A_x^= h = 0$, then $h = \Theta\pi$ for some $\pi$, and since $h \neq 0$ it follows that $\pi \neq 0$ either. But then $0 = Dh = D\Theta\pi$, implies that the columns of $D\Theta$ are not linearly independent. ■

Before we attempt to apply these ideas to GPCP in particular, we first crystallize them in the form of an "algorithm template".

**Algorithm Template**

```
4. Given an LP:

        $(P_1):$   $\max\ c^T x$
        s.t.      $Ax \le b,\quad Dx = d.$

1. Set $\mu_0 = 0$ and set $k = 1$, and randomly perturb the vector $d$.

2. Solve $L(P_1, \mu_{k-1})$. Let $w^k$ be an optimal solution.
   If $k > 1$ and $L(P_1, \mu_{k-1}) = z^{k-1}$ or $H^{k-1} w^k = h^{k-1}$, STOP.

3. Let $H^k x = h^k$ be a linear system of equations that is satisfied by $w^k$ and
   by $x^{k-1}$ (if $k > 1$).

4. Define the restricted problem:

        $(P_2^k):$   $\max\ c^T x$
        s.t.        $Ax \le b,\quad Dx = d,\quad H^k x = h^k.$

5. Solve $P_2^k$ to obtain $x^k$, a primal optimal vertex (with value $z^k$) and $\mu_k$, an optimal
   dual vector corresponding to constraints $Dx = d$. 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 ease of exposition, we have assumed that $P_2^k$ is always feasible and finite; though this is a requirement that can be easily circumvented in practice (Theorem 26).
**3.** The requirement in step 3 that $H^k x = h^k$ is satisfied by $x^{k-1}$ is less stringent than the requirement in the procedure outlined after Theorem 17, which would have demanded that $H^{k+1} x = h^{k+1}$ be a relaxation of $H^k x = h^k$. Strict monotonicity is still preserved though, as we will show in Theorem 26.

**Theorem 25** *(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) If $L(P_1, \mu_{k-1}) = z^{k-1}$, then $x^{k-1}, \mu_{k-1}$ form an optimal pair by duality. If $H^{k-1} w^k = h^{k-1}$, then 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

$$z^* \le c^T w^k \,+\, \mu_{k-1}^T (d - Dw^k) = z^{k-1} \le z^*. \tag{19}$$

(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. ■

**Theorem 26** *If the pool $\mathcal{H}$ from which all $H$ constraints are drawn is fixed and finite, then with probability* 1 *the algorithm template converges finitely and strictly monotonically.*

**Proof:** First we prove that if $P_1$ itself is feasible and finite then we are entitled to assume that each $P_2^k$ is feasible and finite, and if not that this can be detected. Note first that if any $P_2^k$ is unbounded, then so is $P_1$, and so the algorithm can stop if such a $P_2^k$ is encountered. Note next that if $P_2^1$ is feasible, then every $P_2^k$ is feasible. A feasible initial $P_2^1$ can always be obtained (or $P_1$ can be determined to be infeasible) by first applying the algorithm to a "linear programming Phase 1" type modified problem, in which $Dx = d$ is replaced by $Dx + v = d$, and each $Hx = h$ is replaced by $Hx + w = h$, where $w$ and $v$ are nonnegative artificial variables to be pushed by the objective function to 0. For this problem, the initial restricted LP is always feasible for $x = 0$ and an appropriate choice of $w$ and $v$ (if the signum of the artificial variables in the constraints was chosen appropriately). If the optimal solution has $w = 0$ and $v = 0$, then the terminal $(H, h)$ will define a feasible $P_2^1$ problem for "linear programming Phase 2" (and otherwise we will have determined that $P_1$ is not feasible).

Now to the proof of monotonicity and termination: For each iteration $k$, define the matrix $\Theta^k$ to have as its columns a basis for the null space of $\binom{A^=_{x^k}}{H^k}$. If the pool of $H$ constraints is fixed and finite, then by Lemma 21, with probability 1, each $D\Theta^k$ is nonsingular. Observe now that if constraints $H^k x = h^k$ are tightened to $H^k x = h^k,\ H^{k+1}x = h^{k+1}$, then $x^k$ remains an optimal primal vertex and thus Theorem 23 is applicable. Define $\Theta^{k,k+1}$ to be the matrix whose columns form a basis of the null space of $A^=_{x^k}$, $H^k$ and $H^{k+1}$. By Theorem 23, $D\Theta^{k,k+1}$ is nonsingular with probability 1. Since $\mu^k$ also remains an optimal dual solution (as in the proof of Theorem 23), and since the system $H^{k+1}x = h^{k+1}$ is a relaxation of the system $H^k x = h^k,\ H^{k+1}x = h^{k+1}$, Theorem 17 can be applied to $P_2\left(\begin{smallmatrix}H^k\\H^{k+1}\end{smallmatrix}, \begin{smallmatrix}h^k\\h^{k+1}\end{smallmatrix}\right)$, with the former system playing the role of $\bar{H}$ and the latter playing the role of $H$. By Theorem 17, if the algorithm does not stop, there is therefore a strictly improving feasible direction for $P_2^{k+1}$ from the point $x^k$, which establishes strict monotonicity. Since there are only finitely many choices of $H$ and strict monotonicity implies that there can be no repetition, the algorithm must terminate finitely. ■

## 3.3 Interpreting the Template

### 3.3.1 Exploiting Decompositions Versus Column Generation

Note that the weakest way in which to implement the template would be to take $H^k$ to be the constraint that the solution vector must be a linear combination of $w^k$ and $x^{k-1}$ themselves, or in a somewhat stronger form, a linear combination of $\{w^j, j \le k\}$. Note that in this form, if the points $w^j$ are always chosen to be extreme point solutions of $L(P_1, \mu_{j-1})$, then this will satisfy the condition of the finiteness of the pool $\mathcal{H}$ from which the $H$ constraints are drawn, as the number of extreme points of this set is finite.

This latter version, in which a single column, which is the solution to the $P_2$ problem, is added to the $P_1$ problem in each iteration, is reminiscent of column generation. In fact, if we would also require the solution to be a *convex* combination of the $\{w^j, j \le k\}$, then this becomes a Dantzig Wolfe decomposition exactly (as noted in [EGMM12], the lagrangian is the pricing problem for a decomposition in which the columns are the extreme points of $\{x : Ax \le b\}$ and the $D$ constraints are left in place). In this case the $A$ constraints could be left out of the restricted problems $P_2^k$, as they would be satisfied for free by virtue of the convexity constraint. So if the $A$ constraints alone and the $D$ constraints alone each make for an "easy" problem, then the lagrangians and the $P_2^k$ problems will also be easy. Dantzig Wolfe decomposition therefore turns out to be a special case of the Algorithm Template. Considering this fact, it may follow that randomly perturbing the right hand side vector of the constraints to be left in the master problem (the $D$ constraints, in our case) may be beneficial for general column generation approaches as well.

There is a natural interpretation of the template however, which will tend toward a different kind of implementation. Observe first that the smallest collection of constraints $H$ that are satisfied at optimality is just the set of constraints that define the optimal face of $\mathcal{P}$, i.e. they form a subset of $A$. This point can actually be made somewhat sharper, as the following theorem shows.

**Theorem 27** *Let $x$ be an extreme point solution of* $\mathbf{LP}(\mathcal{P})$*, assume $d$ has been randomly perturbed, and let $Hx = h$ be a collection of constraints satisfied at $x$ drawn from a fixed and finite pool $\mathcal{H}$. Then with probability 1, all $H$ constraints are dominated by $\bar{A}x = \bar{b}$.*

**Proof:** Let $\Theta$ be a matrix whose columns are a basis of $N_A^x$, let $\Theta^H$ be the matrix whose columns are a basis of the null space of $A_x^=$ and $H$, and note that $x$ remains an extreme point solution of $P_2(H, h)$. If any $H$ constraint is not in the span of $A_x^=$, then $\Theta^H$ must have fewer columns than $\Theta$, but by Lemma 21, both $\Theta$ and $\Theta^H$ have the same column space with probability 1. ■

Thus ultimately we ought to be guessing at $A_x^=$ for an optimal $x$ to constitute $H$, or equivalently, at the nature of the decomposition $x^i + \Theta\alpha$ of the optimal $x$. In practice, one would not choose $H$ that fixes a single extreme point, but rather only some $A$ constraints, or other constraints that are compatible with multiple sets of $A$ constraints, and then let the solution of $P_2(H, h)$ determine the best set of $A$ constraints compatible with $H$ to make tight in its solution, implicitly selecting $\Theta$. The guess would then ideally be updated by relaxing sufficiently many of the $A$ constraints that were required to be tight (i.e. which were in $H$) in order not to cut off the solution to the lagrangian that was obtained at that iteration. The extra tightening that could be imposed after finding a solution at an iteration, as in Theorem 23, would similarly be the imposition of equality upon those $A$ constraints that happen to be at equality in the current solution (i.e. including them into $H$).

This update can also be framed in terms of the decomposition. Given a solution $x^j$ at iteration $j$ and its decomposition $x^j = x^i + \Theta^j\alpha$, where $\Theta^j$ is the matrix whose columns form a basis of the null space of $\binom{A_{x^j}^=}{H^j}$, if the lagrangian solution $w^{j+1}$ at the next iteration does not satisfy the $H^j$ constraints, then there was no way to move along column vectors of $\Theta^j$ from $x^j$ to $w^{j+1}$. Dropping $H^k$ constraints violated by $w^{j+1}$ could be thought of as allowing $\Theta^j$ to be modified (as it no longer needs to be in the null space of the dropped constraints) in such a way as to to indeed make $w^{j+1}$ accessible. This suggests that we could use information we may know about the structure of what lagrangian neutral vectors $\Theta$ and decompositions $x^i + \Theta\alpha$ might look like to frame constraints compatible with what we think an optimal decomposition might look like, and then use the lagrangian solution to update these assumptions regarding the structure of the decomposition. We will see in Section 3.4 how we use precisely these ideas in framing an algorithm for GPCP using the Algorithm Template.

Returning to the issue of the desirability of choosing $A$ constraints to be $H$ constraints, observe that in solving the restricted LP at the $j$'th iteration, $P_2^j$, we obtain a solution $x^j$ which can be decomposed as $x^i + \Theta^j\alpha$, where, again, $\Theta^j$ is the matrix whose columns form a basis of the null space of $\binom{A_{x^j}^=}{H^j}$. Now note that each of these column vectors is also in $N_A^{x^j}$ (i.e. the null space of $A_{x^j}^=$), and thus could constitute a column of $\Theta$ (the matrix whose columns are a basis of $N_A^{x^j}$), but they may not span $N_A^{x^j}$. Specifically, they span $N_A^{x^j}$ iff the $H^j$ constraints are all dominated by $A_{x^j}^=$. So in this sense an arbitrary choice of $H$ amounts also to a matrix $\Theta$, but one which is missing some of its columns.

Observe further that if we have perturbed $d$, and this led to the desired result that $D\Theta^j$ is nonsingular, and if all $H^j$ constraints are dominated by $A_{x^j}^=$, then we can choose $\Theta = \Theta^j$, which will imply that $x^j$ is an extreme point solution of $P_1$ by Lemma 24. But if some $H^j$ constraints are not dominated by $A_{x^j}^=$, then $\Theta$ must have more columns than $\Theta^j$ and $D\Theta$ therefore must have more columns than rows, which implies that $x^j$ is not an extreme point solution of $P_1$ (also by Lemma 24). Moreover the dual $\mu^j$, returned in solving $P_2^j$, is $c\Theta^j(D\Theta^j)^{-1}$, but due to the extra columns in $\Theta$, this may not satisfy $\mu^j D\Theta = c\Theta$. The conclusion is that having $H$ constraints nondominated by $\bar{A}$ constraints at a solution $x^j$ implies that $x^j$ is not an extreme point solution, that the decomposition matrix $\Theta^j$ is missing some columns of $\Theta$, and that $\mu^j$ may not satisfy $\mu^j D\Theta = c\Theta$, which is prerequisite to optimality.

**Theorem 28** *Let $x^j$ be the solution returned by the j'th iteration of the algorithm template, let $\Theta^j$ be a matrix whose columns form a basis of the null space of $\binom{A^=_{x^j}}{H^j}$, and assume that $D\Theta^j$ is nonsingular.*

- *A matrix $\Theta$ whose columns form a basis of $N_A^{x^j}$ can be chosen to contain all the columns of $\Theta^j$.*
- *$x^j$ is an extreme point solution of $P_1$ iff $\Theta$ can be chosen to be $\Theta^j$ iff $H^j x = h$ is dominated by $A^=_{x^j} x = b^=_{x^j}$.*
- *If $\Theta = \Theta^j$, then $\mu^j = c\Theta(D\Theta)^{-1}$.*

In short, the presence of many nondominated $H$ constraints in the $P_2^k$ solutions implies a deficiency in the decomposition implied by $H$. Thus it would seem desirable that the $H$ constraints should either be $A$ constraints themselves, or constraints that are likely to be dominated by $A$ constraints in the $P_2$ solutions. At the least they ought to be constraints that lend themselves toward a characterization of a potentially optimal decomposition.

### 3.3.2 Virtuous Cycles: Using Lagrangian Relaxation to Expose Structure

The philosophy behind the template is that we would like the restricted problem to be an approximation of the original problem – and to this end we would try to choose $H|h$ in such a way as we believe would not cut off an optimal primal solution. The quality of the dual solution that emerges from solving the restricted LP is broadly reflective of how good an approximation it was.

The lagrangian step uses this information to identify flaws in the approximation and to improve it in the next iteration. How well the lagrangian will do will be affected by the quality of the duals, but it always corrects some false assumptions, and Theorem 17 shows that it also always makes concrete progress in the primal sense as well. The better approximation though now "should" again lead to better duals, which would imply a virtuous cycle effect.

**Observation 29 (Heuristic Observation)** *If the constraints $H^k(x) = h^k$ induced by the lagrangian solution $L(P_1, \mu^{k-1})$ describe the optimal LP solution with "increasing accuracy" as $\mu^{k-1}$ gets "closer" to the optimal $\mu$, then we can expect to see a "virtuous cycle" effect.*

In one sense we always have improving "accuracy" in $H^k(x) = h^k$ as we have strict monotonicity, though in practice we would try to use $H^k(x) = h^k$ constraints that say something about the structure of the solution, and which would be refined by the algorithmic template to make statements about the structure that are increasingly accurate. We will see that in the GPCP, we will choose $H^k(x) = h^k$ that say something about the graph defining the problem, and that the algorithm quite literally refines these statements to make them a more accurate characterization of the optimal solution.

Another related perspective on the algorithm emerges from general meta-observations regarding lagrangian relaxation. Traditional lagrangian relaxation schemes (such as subgradient optimization) can prove frustratingly slow to achieve convergence, often requiring seemingly instance-dependent choices of algorithmic parameters. They also do not typically yield optimal feasible primal solutions; in fact frequently failing to deliver a sufficiently accurate solutions (primal or dual). However, as observed in [B02] (and also see [BA00]) Lagrangian relaxation schemes *can* discover useful "structure."

For example, Lagrangian relaxation can provide early information on which constraints are likely to be tight (and as indicated above, this can be crucial in identifying an optimal $\Theta$), 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 in general 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 the dualized constraints 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.

The template described here can be interpreted in this context as an algorithm to systematically extract structure from the lagrangian and from restricted LP's symbiotically so as to solve the lagrangian and the primal LP simultaneously.

#### 3.3.3 Approximating the Lagrangian Graph

In the appendix we will describe yet another way to interpret the template in a geometric fashion as a method of approximating the lagrangian. The main idea, stated roughly, is to consider the lower surface of the convex solid defined by the epigraph of the lagrangian of $P_1$ (i.e. the $|D|+1$ dimensional "graph" of the lagrangian). Note first that the "graph" of the lagrangian of each $P_2^k$ is is an approximation of that of $P_1$ which is everywhere at or below this surface, and the degree of relevant "error" in the approximation is the amount by which the lowest point on the $P_2^k$ graph is smaller than that of the $P_1$ graph. In each iteration the algorithm can be shown to effectively pick this lowest point in the graph of $P_2^k$, and "pin" the graph at that point up to that of $P_1$ (in the later iterations). But by the convexity of these solids, doing so pulls the graph up not only at that point, but more generally as well.

One result that emerges from this analysis is that the algorithm converges in a pseudologorithmic number of iterations if there is only one $D$ constraint. On its own this result is not necessarily significant, as binary search on the lagrangian dual is also applicable in this case (though that would not directly yield a primal solution), but it is still instructive in supplying insight into how the algorithm works. More generally, this characterization also supports a "virtuous cycle" expectation, in that the better the approximation that is provided to the $P_1$ lagrangian graph by the $P_2^k$ lagrangian graph, the closer may one expect the minimizer (which will be be pinned up) of the one to be to that of the other. As above, this effect will likely be strongest if the approximations are based upon combinatorial structure.

## 3.4 Applying the Template to GPCP

The discussion in Section 3.3.1 indicates that the principal idea behind the template is to systematically improve the estimate of the decomposition $x^i + \Theta\alpha$ (as per Lemma 11) of an optimal solution. In the case of GPCP, $x^i$ and $\Theta$ have a particularly nice structure:

**Theorem 30** *Let* $\mathbf{LP}(\mathcal{P})$ *be defined by* $\max c^T x : Ax \leq b,\ Dx = d$*, where* $Ax \leq b$ *defines a closure problem for a graph* $G = (\mathcal{N}, \mathcal{A})$ *(i.e.* $x \in R^{\mathcal{N}},\ 0 \leq x \leq 1,\ x_i \leq x_j,\ \forall (i,j) \in \mathcal{A}$*). Let* $x$ *be any feasible solution, let* $A_x^=$ *be the submatrix of* $A$ *constraints that are binding at* $x$*, and let* $b_x^=$ *be the corresponding right hand side vector. Then the integer portion of* $x$*, which we denote* $x^i$*, satisfies* $A_x^= x^i = b_x^=$*. Let* $G(x)$ *be the graph* $G$ *with all nodes* $j$ *for which* $x_j$ *is integer removed, and containing only those arcs that are binding at* $x$*. The incidence vectors of the connected components of* $G(x)$ *are linearly independent, their span is the null space of* $A_x^=$*, and* $x$ *has a constant fractional value within each component. Thus where* $\Theta$ *is the matrix whose columns are the incidence vectors* $\{\theta\}$*,* $x = x^i + \Theta\alpha$ *is a valid decomposition of* $x$ *as per Lemma 11.*

**Proof:** First we will prove that the incidence vectors of the connected components of $G(x)$ form a basis of the null space of $A_x^=$. Let $\theta$ be an incidence vector of a nonempty component $C$ of $G(x)$.

Clearly the vectors $\theta$ are linearly independent as they are nonzero and their supports do not overlap each other in any coordinate. A binding constraint $(a^i)^T x \leq b_i$ from $A_x^= \leq b_x^=$ is either of the form $x_j \leq 1$, $-x_j \leq 0$ or $x_i - x_j \leq 0$. If $(a^i)^T x \leq b_i$ is of one of the first two forms, then node $j$ was removed in forming $G(x)$, and so $C$ cannot contain node $j$, and so $(a^i)^T\theta = 0$. Considering now constraints of the third type corresponding to an arc $(i,j)$, suppose $C$ contains $i$. Since $C$ is a connected component of $G(x)$ and binding arcs were not removed, it must also contain $j$, and similarly if it contains $j$ it must contain $i$, and so again we have $(a^i)^T\theta = 0$. Conversely, suppose that $\theta \neq 0$ is an arbitrary vector for which $A_x^=\theta = 0$. For any $j$ such that $\theta_j \neq 0$, the constraints $x_j \leq 1$ or $-x_j \leq 0$ cannot be binding or else their corresponding row $a^i$ in $A_x^=$ would satisfy $(a^i)^T\theta = \pm\theta_j \neq 0$, and so any such node $j$ has fractional $x$ value and belongs to $G(x)$. Observe now that for any connected component $C$ of $\bar{G}$ we must have $\theta_i$ constant for every $i \in C$. To see this, suppose there is an arc $(i,j)$ in $C$ for which $\theta_i \neq \theta_j$. But since $G(x)$ only includes the binding arcs, the arc constraint $x_i - x_j \leq 0$ belongs to the system $A^= x^i = b_x^=$, and its inner product with $\theta$ would then be nonzero. We conclude that $\theta$ is a linear combination of the incidence vectors of the components of $\bar{G}$. Finally, since the incidence vectors of the fractional components are all in the null space of $A_x^=$, we can subtract off the constant fractional value of $x$ in each such component to arrive at $x^i$ without altering the value of $A_x^= x$, justifying the statement that $A_x^= x^i = b_x^=$. ■

**Corollary 31** *Under the conditions of Theorem 30, if $x$ is an extreme point feasible solution, then $\bar{G}$ has $\leq q$ components, and therefore $x$ has $\leq q$ distinct fractional values, where $q$ is the number of side constraints.*

**Proof:** This follows directly from Theorem 30 and Lemma 11. ■

We conclude that for any solution $x$, its decomposition is defined by the incidence vector of its nodes at value 1, and the columns of $\Theta$, which are the incidence vectors of the connected components of the graph $\bar{G}$, each of which also has constant $x$ value. This affords a natural implementation for the algorithm template:

***The "assumption" that defines each iteration of the algorithm is the assumption that certain collections of nodes have common values,*** which is essentially a guess as to the optimal decomposition.[9]

Observe now that this simplifying assumption results again in a GPCP, but one with fewer nodes, i.e. an easier problem. The idea then is to solve this problem to obtain $x$ and the associated dual $\mu$, and check if $x$ maximizes $\mathbf{L}(\mu)$. If it does, then we're done, otherwise the vector $\bar{x}$ that does maximize $\mathbf{L}(\mu)$ must split some of the collections of nodes that we assumed had a common value (Theorem 25), and so we would update our assumption by allowing these collections to be split as per $\bar{x}$.

A critical observation is that if we solved the restricted LP by the simplex algorithm, then its solution was an extreme point, and therefore by Observation 31, it only contained a maximum of $q+2$ distinct values (where $q$ is the number of side constraints). Thus in order to ensure that the next iteration's assumption ($H^k x = h^k$) that the node set can be partitioned into some collection of parts each with a common value, is compatible both with the lagrangian solution $w^k$ and with the previous LP solution $x^{k-1}$, we never need a partition with more than $2(q+2)$ parts, and therefore the restricted LP to be solved never needs to have more than $O(q)$ variables.[10] The number of precedence constraints in the reduced problem is the number of arcs in the reduced graph. This can conceivably be quadratic in the number of collapsed nodes ($O(q^2)$)), though it is typically $O(q)$ as

[9]Note that for connected nodes in such a collection, this assumption is equivalent to the assumption that the arc constraints in the smallest subgraph connecting them are all tight, i.e. these $H$ constraints are actually $A$ constraints. Suppose such a collection contains a disconnected pair of nodes $i$ and $j$. Then if the solution assigns an integer value to the collection, then the assumption of a common value aligns either with the $A$ constraints $x_i = 0,\ x_j = 0$ or with the $A$ constraints $x_i = 1,\ x_j = 1$. If the solution assigns a fractional common value to the collection then that will not align with any $A$ constraint, and will indicate that the resulting solution is not an extreme point. Later we will consider an algorithm variant in which this cannot happen.

[10]In practice we will ensure that it refines the partition implied by the components of the graph $\bar{G}$, which also has $\leq q+2$ parts, though we will see later than in principle these should amount to the same thing, though in rare circumstances they may not.

well. Note that these observations would not have held had we implemented the template as column generation.

Before we describe the algorithm in detail, considering that the restricted problems will actually be modelled as GPCP problems themselves on contracted graphs, rather than as the original GPCP problem with additional constraints $Hx = h$, we will clarify the relationship between these two equivalent ways of formulating the problem. The proofs are all either straightforward, or utilize the same logic as used in the proof of Theorem 30.

**Observation 32** *Let* $\mathbf{LP(\mathcal{P})}$ *be defined by* $\max c^T x : Ax \leq b,\ Dx = d$*, where* $Ax \leq b$ *defines a closure problem for a graph* $G = (\mathcal{N}, \mathcal{A})$*. Let* $\mathcal{C} = \{C^1, \ldots, C^r\}$ *be a partition of* $\mathcal{A}$ *with the* $i'th$ *part in the partition described by the incidence vector* $\phi^i$*, and let* $\Phi$ *be the matrix with columns* $\{\phi^i\}$*.*

- *The restriction that all nodes within a part in the partition* $\mathcal{C}$ *must have a common solution value can be modelled by appending variables* $y_1, \ldots, y_r$ *corresponding to each part, with* $0 \leq y \leq 1$*, and with new constraints* $\Phi y = x$*.*

- *Replacing* $x$ *with* $\Phi y$*, the problem can be rewritten as* $\max c^T \Phi y : A\Phi y \leq b,\ D\Phi y = d$*. After removing the redundant rows from* $A\Phi y \leq b$ *we are left with the same closure problem on the graph* $\bar{G}$ *that results when all nodes in each part* $C^i$ *are contracted to a single node.*

- *Let* $(x, y)$ *be a solution to the restricted problem, as we have defined it. Let* $\bar{G}(y)$ *be the graph* $\bar{G}$ *after removing all nodes with integer* $y$ *value and all nonbinding arcs in* $y$ *and let* $G(x)$ *be the graph* $G$ *after removing all nodes with integer* $x$ *value and all nonbinding arcs in* $x$*. Then* $\bar{G}(y)$ *is the contraction of the graph* $G(x)$*.*

- *For all parts* $C^i \in \mathcal{C}$*, define the expanded incidence vector* $\hat{\phi}^i \in R^{|\mathcal{N}|+|\mathcal{C}|}$ *to be the same as* $\phi^i$ *in the* $\mathcal{N}$ *coordinates, and to have a 1 in the coordinate corresponding to* $C^i$*, and zeros elsewhere, and let* $\hat{\Phi}$ *be the matrix with columns* $\{\hat{\phi}^i\}$*. Let* $\{\theta^i \in R^{|\mathcal{C}|}\}$ *be the incidence vectors of the components of* $\bar{G}(y)$*, and let* $\Theta$ *be the matrix with columns* $\{\theta^i\}$*. The matrix* $\hat{\Phi}\Theta$ *has a column for each component of* $\bar{G}(y)$*, with its* $\mathcal{N}$ *coordinates constituting the incidence vector of this component with respect to* $G$*, and its* $\mathcal{C}$ *coordinates constituting the incidence vector of this component with respect to* $\bar{G}$*.*

- *The columns of* $\hat{\Phi}\Theta$ *form a basis of the null space of the binding* $A$ *constraints and the* $\Phi y = x$ *constraints, and thus* $\hat{\Phi}\Theta$ *is the matrix* $\Theta^H$ *of Theorem 17.*

- *If we append on zero-valued columns to* $D$ *corresponding to the* $y$ *variables to form the matrix* $\hat{D}$*, then the matrix* $D\Theta^H$ *of Theorem 17 is therefore*
$$\hat{D}\hat{\Phi}\Theta = D\Phi\Theta = (D\Phi)\Theta \tag{20}$$
*which is just the side constraint matrix for the contracted problem times the* $\Theta$ *matrix of Theorem 30 for the contracted problem.*

### 3.4.1 The Algorithm

Here is an informal description of the algorithm. A formal description follows.

- At every iteration, solve the lagrangian with penalties $\mu^{k-1}$ equal to the optimal duals for the restricted LP, $P_2^{k-1}$, solved in the previous iteration.

- If the max closure solution $w^k$ obtained to the lagrangian itself satisfies the partitioning constraint $H^{k-1}(x) = h^{k-1}$ then the solution $x^{k-1}$ to $P_2^{k-1}$ is optimal for the unrestricted problem. Otherwise update the partition by splitting each part into the elements that were included in the max closure $w^k$ and the elements that were excluded from the max closure $w^k$.

- After any iteration we may collapse the current partition into a new partition with no more than the number of side constraints plus 2 by replacing the current partition with the partition suggested by $x^k$.

**GPCP Algorithm**

1. Initializations: Set $\mu^0 = 0$. Set $\mathcal{C}^0 = \{\mathcal{N}\}$. Set $r_0 = 0$. Set $z^0 = -\infty$. Set $k = 1$. Designate the GPCP problem as $P_1$. Ensure that the rows of the side constraint matrix $Dx = d$ are linearly independent, and randomly perturb the right hand side $d$.

2. Use a max closure algorithm to obtain integer optimal solution $y^k$ to $L(P_1, \mu^{k-1})$, and define
$$I(y^k) = \{n \in \mathcal{N} : y^k_n = 1\} \tag{21}$$
and define
$$O(y^k) = \{n \in \mathcal{N} : y^k = 0\}. \tag{22}$$
If $k > 1$ and the partition $\mathcal{C}^{k-1}$ of $\mathcal{N}$ is represented as
$$\mathcal{C}^{k-1} = \{C^{k-1}_1, \ldots, C^{k-1}_{r_{k-1}}\} \tag{23}$$
and no set $C^{k-1}_j$ overlaps both $I(y^k)$ and $O(y^k)$ then STOP.

3. Let
$$\mathcal{L} = \{L_1, \ldots, L_{l_k}\} \tag{24}$$
be a partition of $\mathcal{N}$ refining $\{I(y^k), O(y^k)\}$ and define the partition
$$\{L_i \cap C^{k-1}_j : i = 1, \ldots, l_k,\ j = 1, \ldots, r_{k-1}\} \tag{25}$$
of $\mathcal{N}$. Denote the nonempty sets in this partition as $\mathcal{C}^k = \{C^k_1, \ldots, C^k_{r_k}\}$, and define corresponding incidence vectors $\phi^1, \ldots, \phi^{r_k}$.

4. Solve the new GPCP problem $P^k_2$ defined by collapsing all nodes in each $C^k_j$ into a single supernode and replacing the side constraints $Dx \le d$ by $D\Phi x \le d$, where $\Phi$ is the matrix whose columns are the vectors $\phi^j$. Use a method that returns an extreme point solution.

   Let $x^k$ be the optimal solution to $P^k_2$ represented in terms of the original graph, let $\mu^k$ be the optimal duals corresponding to the side constraints, and let $z^k$ be the solution value.

5. If $\mu^k = \mu^{k-1}$ STOP.

6. (Optional) Let $G(x^k)$ be the graph obtained from $G$ by removing all arcs that are nonbinding in $x^k$ and all nodes $i$ at which $x^k_i$ is integer. Let $\mathcal{C} = \{C^r\}$ be the partition of $\mathcal{N}$ which contains a part for the nodes with $x^k$ value 1, a part for the nodes $x^k$ value 0, and a part for each connected component of $G(x^k)$. (Note that the number of parts in $\mathcal{C}$ does not exceed 2 more than the number of side constraints.)[11]

   Update $\mathcal{C}^k := \mathcal{C}$, and $r_k$ to be the number of parts in $\mathcal{C}$.

7. Set k=k+1 and GOTO step 2.

**Notes:**
**1.** The perturbation performed at Step 1 is intended to ensure that $D\Theta^k$, where $\Theta^k$ is the matrix whose columns are the incidence vectors of the components of the graph $G(x^k)$ (where $x^k$ is represented with respect to the contracted graph), as defined in Theorem 30, is nonsingular at every iteration $k$. As shown in Observation 32, $D\Theta^k$ is equal to $D\Theta^H$ of Theorem 17, where $H$ represents the partitioning constraints, and so this is essential for convergence, as well as for other reasons, as outlined in Section 3.2. Often however, this matrix works out to be nonsingular even without any perturbation.

[11]In Section 7 we will describe a variant of this procedure which may be superior in cases where no perturbations are used.

**2.** The algorithm as written assumes that the side constraints are written as equalities. This simplifies our analysis here, but in practice if the constraints are inequalities in their native form it is best to leave them as such. This is because the number of components in each graph $G(x^k)$ is bounded by the number of **binding** side constraints at $x^k$, which is often far less than the overall number of side constraints. A perturbation of the right hand side of the side constraints is still sufficient to ensure that $\bar{D}\Theta^k$ (as in Note 1) is nonsingular at every iteration, where $\bar{D}$ is the matrix of rows of $D$ that are binding at the iteration's solution. All the results we will prove here will therefore continue to hold in this case as well. Note that where the side constraints are written as inequalities, it is often more convenient for the perturbations to be relaxations, to avoid a situation in which the perturbation makes the problem infeasible.
**3.** In Step 3 we pointedly described Ł as "*a*" partition refining $\{I(y^k), O(y^k)\}$ rather than as *the* partition $\{I(y^k), O(y^k)\}$. We will see later that it may be a better idea to use a finer partition than $\{I(y^k), O(y^k)\}$.
**4.** In Step 4 it is not strictly necessary to use a method that returns an extreme point solution. It can in fact be shown that the GPCP algorithm itself can be used to solve $P_2^k$ (i.e. it can be nested) so long as the innermost nested problem is solved by a method that returns an extreme point solution.
**5.** While in general the algorithm relies on Step 6 to ensure that the restricted problems $P_2^k$ never become too large, it is generally better not to perform this step as a matter of course, but rather only when the restricted LP's are becoming too large to solve quickly, as doing so can destroy some of the structure that has been uncovered in the prior iterations. Following on from this observation, and noting that the current iteration's partition $\mathcal{L}$ is itself the refinement of several previously defined partitions (e.g. previous $\{I(y^k), O(y^k)\}$ partitions, or previous partitions $\{C^r\}$ as defined in Step 6 (these can be defined regardless of whether or not a consolidation is performed)), it is possible to restore some of the precision lost in a consolidation by refining the next iteration's $\mathcal{L}$ with some of those previously defined partitions.
**6.** The algorithm obtains at each iteration both an upper and a lower bound on the LP solution value, and thus we could also terminate early if these bounds have converged within some tolerance.
**7.** As noted above, the fact that the number of distinct values (or fractional components) in an extreme point solution cannot exceed the number of side constraints is crucial in ensuring that the problems $P_2^k$ never need to become large. This fact can also presumably act as a contributing factor in reducing the number of *iterations* to optimality, insofar as it ensures that though the lagrangian solution at each iteration only implies a binary partition of the nodes, there do not need to be a great many parts in the partition that defines the optimal LP solution, and therefore the optimal solution can be obtained in a small number of steps. The exact extent to which it does in fact contribute is not clear to us.

**Theorem 33** *If $P_2^1$ is feasible, the GPCP Algorithm converges with probability 1 in a finite number of iterations to the optimal LP solution, even if Step 6 is performed at every iteration. If Step 6 is only performed if $z^k > z^{k-1}$, then the algorithm will converge finitely even without performing any perturbation.*

**Proof:** By construction, it is clear that the partitioning constraints (i.e. the requirement in each restricted problem that the nodes in each part of the associated partition have common value) at each iteration $k$ do not cut off the previous iteration's restricted LP solution $x^{k-1}$, nor the current iteration's lagrangian solution $y^k$. Since the number of possible partitions is finite, the first statement follows from Theorem 26. In the absence of perturbations however, there will be no guarantee of strict monotonicity and this argument will fail. If however, Step 6 is only performed if $z^k > z^{k-1}$ then convergence is still guaranteed to be finite, as the number of partitions is finite and no partition can repeat. ■

**Observation 34** *If the condition that $P_2^1$ is feasible in Theorem 33 is not met then we can first solve a Phase 1 problem with the condition that the artificial variables are not aggregated in the first iteration, so that the first iteration of that problem is always feasible. If the Phase 1 problem yields a feasible solution $x^*$ to the original problem, then the partition defined by the components of $G(x^*)$ (or any refinement thereof) will result in a feasible restricted LP for the original problem.*

The application of the Algorithm Template to GPCP casts a combinatorial perspective on a number of the algorithm's aspects. In particular, we saw in Theorem 30 that given a solution $x$, the columns of the matrix $\Theta$ that form a basis of the null space of the binding precedence and box constraints are just the incidence vectors of the components of the graph $G(x)$ defined there. Next we will show that the nonsingularity of the matrices $D\Theta$, which played a crucial role in the convergence and the dual behavior of the Algorithm Template, also has a combinatorial interpretation, which can act as a combinatorial test of the nonsingularity of this matrix.

**Lemma 35** *Let $x \in R^n$ be an extreme point solution to a GPCP associated with a graph $G$ on $n$ nodes, for which the constraint matrix without the side constraints is $Ax <= b$, and there are $q$ side constraints $Dx = d$. Let $A_x^=$ be the submatrix of binding constraints from $A$ at $x$. Let $G(x)$ be the graph with all nodes with integer $x$ value and all nonbinding arcs removed, and let $\Theta$ be the matrix whose columns are the incidence vectors of the components of $G(x)$. Then $D\Theta$ is nonsingular iff $G(x)$ contains $q$ components.*

**Proof:** By Lemma 24, the columns of $D\Theta$ are linearly independent, and so $D\Theta$ is nonsingular iff it is square, i.e. iff the number of columns of $\Theta$ – which is the number of components of $G(x)$ – is $q$. ■

It is easy to see that under the conditions of Lemma 35, if we collapse the nodes within each component of $G(x)$, as well as the nodes with $x$ value of 1 and the nodes with $x$ value of 0 respectively, the solution remains an optimal extreme point (just represented in a smaller space), and the number of components of the new "$G(x)$" is unaltered. Therefore if $G(x)$ had $q$ components, then the new "$D\Theta$" matrix represented with respect to the collapsed graph is also nonsingular. In fact the new $D\Theta$ matrix is actually exactly the same as the old one, which implies the following result.

**Corollary 36** *Under the conditions of Lemma 35, if $G(x)$ has $q$ components, the optimal dual solution for the dual variables corresponding to constraints $Dx = d$, which is unique, is not altered by the consolidation of nodes within components of $G(x)$, or by the consolidation of nodes with common integer $x$ values.*

This is a stronger result than that of Theorem 23, in which the addition of constraints compatible with the optimal solution is only guaranteed with probability 1 not to affect the optimal dual space.[12]

We will close off the section by noting some other aspects of the combinatorial interpretation of the algorithm in the GPCP context. By Lemma 14 and Theorem 30, the solution $(x^k, \mu^k)$ obtained by the algorithm at iteration $k$, when $x^k$ is expressed in terms of the contracted graph that defines $P_2^k$, is such that the set of nodes $C^1$ with value 1 in $x^k$ is a max closure with respect to penalties $\mu^k$, with value equal to the LP objective value of $x^k$. It is also such that the set of nodes in any one component $C^i$ of $G(x^k)$ (as defined in Theorem 30, again relative to the contracted graph) has penalized value of exactly 0 – i.e. the penalty balances the value exactly.

Lemma 14 states that the solution $x^k$, (when cast in the original space,) is optimal for $P_1$ also if and only if $C^1$ remains a max closure with respect to penalties $\mu^k$ even when recast in terms of the original uncontracted graph. It is easy to see from the properties of max closures that the solution is therefore optimal for $P_1$ if and only if there is no way peel off a subset of $C^1$ which is closed in the reverse graph and has negative value, or to append to $C^1$ a closed subset of its complement with a positive value (and similar statements can be made about each of the $C^i$ sets). It follows that if a solution $(x^k, \mu^k)$ is optimal for the restricted problem $P_2^k$ but not for the original problem $P_1$, then this means that subject to the node consolidation at iteration $k$, there was no way to peel off negative value reverse closed subsets off of the associated max closure, nor was there a way to append positive value closed subsets to the max closure. But the fact that without node consolidation the

[12]In principle we could have allowed a consolidation of all components of $G(x)$ with a common value without invalidating the current solution, and by Theorem 23, with probability 1, this also would not alter the optimal dual space. Note however, that if there is in fact more than 1 component in $G(x)$ with a common value, then combining them would eliminate a column from $\Theta$, making the new $D\Theta$ singular, and losing the guarantee that the optimal dual space is unaltered. Effectively this means that it is highly unlikely that $G(x)$ would contain multiple components with a common value. Nevertheless, due to finite precision computation this can happen, and it is therefore preferable to set the consolidation rule so as to only consolidate nodes within a component of $G(x)$.

closure is no longer a max closure means exactly that by splitting the aggregated nodes one can indeed obtain such sets. Thus this means that the implicit assumption in the definition of problem $P_2^k$ that it was acceptable to aggregate those nodes together was incorrect, as by splitting them one can improve the value of the lagrangian and therefore of the primal (by Theorem 17).

Note also that the duals $\mu^k$ at iteration $k$, as an estimate of dual prices, give an estimate of the scale of improvement to be obtained in modifying a solution. So splitting aggregated nodes so as to maximize penalized value constitutes the "best" possible way of splitting the nodes subject to this estimate. This also supports the notion that the more accurate $\mu^k$ is, the more effective the algorithm steps will be, which would imply the virtuous cycle behavior described in Section 3.3.2.

## 4 Extreme Point Solutions

If the right hand side $d$ of the side constraints is randomly perturbed, we will show that there is a neat combinatorial characterization of whether or not a solution $x$ to GPCP is an extreme point. But first we need to state a preliminary result.

**Lemma 37** *Let $x \in R^n$ be a solution to a GPCP associated with a graph $G$ on $n$ nodes, for which the constraint matrix without the side constraints is $Ax <= b$. Let $A_x^=$ be the submatrix of binding constraints from $A$ at $x$, and let $G(x)$ be the graph with all nodes with integer $x$ value and all nonbinding arcs removed, and assume that $G(x)$ has $r$ connected components. Define the submatrix $\bar{A}$ of the binding rows $A_x^=$ to contain only those rows corresponding to box constraints and to precedence constraints for arcs defining a spanning forest of $G(x)$. Then $\bar{A}$ contains $n-r$ rows, all of which are linearly independent, and these rows span all rows of $A_x^=$.*

**Proof:** Each component of $G(x)$ is spanned by a tree with one fewer arcs than nodes, and so if there are t nodes in $G(x)$, then these nodes contribute $t-r$ rows to $\bar{A}$. Each node in $G$ outside of $G(x)$ (i.e. each node with integer $x$ value) has exactly one associated row in $\bar{A}$. So there are indeed $n-r$ rows in $\bar{A}$. To see that the precedence constraints for a spanning tree are linearly independent, say that a tree has $h$ nodes, and say that we append a unit vector for one of the nodes $s$ in the tree to the set. Then we could obtain a linear combination of this unit vector $e^s$ and the arc vectors equalling the unit vector $e^v$ for any other node $v$ in the tree by summing the vectors along the tree path between $s$ and $v$. Thus these $h$ vectors span the $h$ unit vectors and we conclude that the $h-1$ arc vectors must be linearly independent. Since the components are all distinct from each other and from the integer valued nodes, we conclude that all $n-r$ rows are linearly independent. To establish that the span of these rows includes all rows of $A_x^=$, note that the additional rows of $A_x^=$ are all associated with arcs within a component. But each such row can be obtained by summing the rows corresponding to the tree path between that arc's endpoints, which completes the proof. ∎

**Theorem 38** *Let $x$ be a solution to a GPCP associated with a graph $G$ and with $q$ side constraints $Dx = d$, where $d$ is chosen under the random model of Definition 18. Define the graph $G(x)$ as in Lemma 37. With probability 1, $x$ is an extreme point iff the number of components of $G(x)$ is $q$.*

**Proof:** Letting $A_x^=$ be the binding precedence constraints at $x$, and letting $\Theta$ be the matrix whose columns are the incidence vectors of the components of $G(x)$, the columns of $\Theta$ form a basis of the null space of $A_x^=$ by Theorem 30. If $x$ is an extreme point, then by Lemma 21, $D\Theta$ is nonsingular with probability 1, and so $\Theta$ must have $q$ columns, so that $G(x)$ must have $q$ components. Conversely, assume that $G(x)$ has $q$ components, but that $x$ is not at extreme point. Define $\bar{A}$ as in Lemma 37, and let $\bar{b}$ be the corresponding right hand side vector. By Lemma 37, $\bar{A}$ is comprised of $n-q$ linearly independent rows. Since we have assumed that $D$ is comprised of $q$ rows, but that $x$ is not an extreme point, it follows that $\binom{\bar{A}}{D}$ is not of full row rank. Define $D(i)$ to be the matrix made up of the first $i$ rows of $D$, with corresponding right hand side vector $d(i)$, and let $i$ be minimal subject to the condition that $\binom{\bar{A}}{D(i)}$ is not of full row rank. Then if we denote the $i$'th row of $D$ as $D^i$, there are unique vectors $\lambda_1$ and $\lambda_2$ such that $D^i = \lambda_1^T \bar{A} + \lambda_2^T D(i-1)$. Moreover, since $D^i x = d_i$ is also binding at $x$, we must have $\lambda_1^T \bar{b} + \lambda_2^T d(i-1) = d_i$. This completes the proof since, under the random model, the measure of the set $\{d \in B(\bar{d}, \epsilon) : \lambda_1^T \bar{b} + \lambda_2^T d(i-1) = d_i\}$ is zero. ■

Theorem 38 implies an easy test for determining whether or not expanding nodes in a collapsed graph will preserve the status of a solution as an extreme point: $x$ will remain an extreme point with probability 1 iff expanding nodes does not disconnect any of the components of $G(x)$. The next lemma shows that this statement can actually be made absolutely, without the "with probability 1" caveat.

**Lemma 39** *Let $G = (N, E)$ be the graph associated with a GPCP for which the constraint matrix without the side constraints is $Ax <= b$, and let $G' = (N', E')$ be the graph $G$ after collapsing together some of the nodes. Let $x' \in R^{N'}$ be a solution w.r.t. the collapsed graph $G'$, and let $x \in R^N$ be $x'$ expressed w.r.t. the graph $G$. Let $G(x)$ be the graph $G$ with all nodes $j$ for which $x_j$ is integer, and all arcs that are nonbinding w.r.t. $x$ removed, and similarly for $G'(x')$. If $G(x)$ and $G'(x')$ have the same number of connected components, and $x'$ is an extreme point, then $x$ is an extreme point as well.*

**Proof:** This follows from Theorem 28, as for each component of $G'(x')$ to remain a single component even when uncollapsed, means that the binding $H$ constraints defining the collapsed problem are all $A$ constraints, as noted in footnote 9. ■

Thus if we solve a restricted problem $P_2^k$ with the simplex algorithm to obtain an extreme point solution $x'$, this solution will remain an extreme point even when deconsolidated so long as deconsolidating does not disconnect any components of $G'(x')$. We can guarantee that this will always be the case if we always construct the consolidated graphs $G'$ such that each of its elements represents a connected subgraph of $G$ (i.e. by breaking up each aggregated node in a nominal graph $G'$ into its connected components). This ensures that the $H$ constraints that define each restricted problem are all $A$ constraints, as they all correspond to arcs in the graph. This proves the following theorem:

**Theorem 40** *If the restricted LP's $\{P_2^k\}$ solved by the GPCP algorithm at each iteration are all constructed to have the property that each aggregated node represents a connected subgraph of the original graph, and we solve each restricted LP with the simplex method, then all solutions obtained will be extreme points of the unconsolidated problem.*

In principle there are two possible advantages to structuring the problem so that all $x^k$ are extreme points in the original space. One is that is might be supposed or hoped that this will aid in convergence. This may be particularly the case in line with our observations in Section 3.3.1 indicating that $H$ constraints should ideally be closely associated with $A$ constraints, as in this case all $H$ constraints would actually be $A$ constraints. The second is that each previous solution can be used to construct a starting basis for the next iteration's restricted LP. Regarding this latter possibility, it is noteworthy that even if an iteration's solution is not an extreme point w.r.t. the original fully unconsolidated graph $G$, it will be an extreme point w.r.t. the partially unconsolidated graph that will be in use in the following iteration if the refinement used to construct the next iteration's graph does not disconnect any components of $G'(x')$.

In practice however, we have not observed an unambiguous improvement in iterations to optimality due to structuring the problems $P_2^k$ in this manner. This is perhaps due to the fact that decomposing the units into their connected components can create a large number of scheduling units, which can make the restricted problems hard to solve.

Regarding the second issue, the improvement is also often marginal, as commercial LP codes already contain crossover algorithms to allow warmstarting from nonbasic solutions as well.

Of possibly far greater utility, however, is that we will show that these ideas can be used to convert a nonbasic solution yielded by the algorithm into a basic one. This makes the algorithm suitable for embedding into any IP algorithm which requires its LP subroutines to return basic solutions:

**Theorem 41** *Given a possibly non-basic solution $x$ to a GPCP defined on a graph $G$, let $G(x)$ be $G$ after removing the nodes that are integer in $x$ and the arcs that are nonbinding in $x$. Define the graph $G'$ as follows: Collapse all nodes in $G$ with $x$ value $0$ to a single node $Z$, all nodes with $x$ value $1$ to a single node $O$ and collapse the nodes in each component of $G(x)$ to a single node. Now for each node representing a component of $G(x)$, decompose the node into its connected components in $G$. (Do not do this for $Z$ and $O$.) Define a new GPCP on $G'$ with the same side constraints, and with additional constraints requiring node $Z$ to have value $0$ and node $O$ to have value $1$ (this is equivalent to just removing these nodes from the problem). An extreme point solution $x'$ to this problem in the consolidated space is also an extreme point solution in the original unconsolidated space of value $\geq$ that of $x$.*

**Proof:** The original solution $x$ remains describable in $G'$, and so clearly $x'$ must have value $\geq$ that of $x$. Let $G'(x')$ be defined analogously to $G(x)$. By Theorem 40 all we need to show is that the connected components of $G'(x')$ remain connected even after $G'$ is decollapsed to $G$. Observe first that for any collapsed node $B$ in $G'(x')$, by construction $B$ has fractional value, and therefore its constituent nodes are all in $G(x)$ (as the other nodes were all explicitly excluded from having fractional value in the restricted problem). But this implies, again by construction, that $B$ is a connected set in $G$. Suppose now that node $i$ in collapsed node $I$ and node $j$ in collapsed node $J$ are in a single component of $G'(x')$, and note that $i$ and $j$ must therefore have a common fractional $x'$ value. So there is an undirected path in $G'(x')$ connecting $I$ and $J$. But since a path in $G'(x')$ corresponds to a sequence of arcs between constituent members of collapsed nodes, there must also be a $G$ path connecting any constituent node in $I$ to any constituent node in $J$. Thus since all nodes on this path have common $x'$ value, it follows that $i$ and $j$ are in a common component in $G(x')$ as well. ■

The theorem shows that solving a single, typically small, restricted LP will convert an arbitrary solution to an extreme point solution of no smaller value. It should be noted that there is no guarantee that in decomposing the fractional components defined by $x$ into connected sets we will not generate a large number of units, but if $x$ was optimal or near-optimal then in most cases it is not likely that many independent unconnected nodes will happen to have the same fractional value in an optimal LP solution. This has in fact been our experience in practice. Alternatively, the following observation shows that the same result can be achieved by solving no more than $q \log n$ guaranteed small LP's (with no more than $q+1$ variables each), where $q$ is the number of side constraints, and $n$ is the number of components in the unconsolidated graph $G$.

**Observation 42** *Let $G$ be the graph associated with a GPCP with $q$ side constraints, let $G^k$ be the collapsed version of $G$ associated with the $k$'th iteration problem $P_2(k)$, let $x$ be an extreme point solution to $P_2(k)$, and let $G^k(x)$ be the graph $G^k$ after removing the nodes that are integer in $x$ and the arcs that are nonbinding in $x$. Note that there are $q$ components in $G^k(x)$, and assume that the $j$'th such component is comprised of $p > 1$ components w.r.t. the uncollapsed graph $G$. Define the graph $\tilde{G}$ having a single node $Z$ representing all nodes with $x$ value $0$, a single node $O$ representing all nodes with $x$ value $1$, and a single node for each component of $G(x)$ other than the $j$'th, and two nodes for the $j$'th, one containing $\lceil \frac{p}{2} \rceil$ original components, and the other containing the remaining $\lfloor \frac{p}{2} \rfloor$ original components. Define the GPCP associated with $G'$ with the additional constraints that $x(O) = 1,\ x(Z) = 0$, and note that there are $q+1$ remaining unfixed variables, and that the solution to this problem has value at least that of $P_2(k)$. Every extreme point solution to this problem will assign an integer value to at least one of these $q+1$ variables.*

Thus it is typically easy to get an extreme point solution. The following results show that there is also an easy "combinatorial" way to manually construct a basis representing an extreme point solution using the forest defined in Lemma 37.

**Lemma 43** *Let $x$ be an extreme point solution to a GPCP associated with a graph $G$ and with $q$ side constraints $Dx = d$, and let $G(x)$, $A^{=}_{x}$ and $\bar{A}$ be as in Lemma 37, with associated rhs $b^{=}_{x}$ and $\bar{b}$ respectively. Then the rows of $\bar{A}$ and of $D$ are linearly independent and uniquely define $x$.*

**Proof:** Since $x$ is an extreme point, it is the unique solution to $A_x^= = b_x^=$, $Dx = d$. By Lemma 37, $A_x^= = b_x^=$ can be replaced with $\bar{A}$, as $A_x^=$ and $\bar{A}$ have the same span. But since $\bar{A}$ has $n - q$ rows, and we need $n$ rows to define an extreme point, all $n$ rows in $\bar{A}$ and $D$ must be linearly independent. ■

Thus given an extreme point solution for which the number of components equals the number of side constraints (which is to be expected by Lemma 37), it is easy to construct a basis, as we know that the $n$ linearly independent constraints that define $x$ are the side constraints and the precedence constraints that constitute a spanning forest of the components.[13] Note that the actual number of nodes in the graph does not matter in this analysis. All we need to know is the $q$ components, and that $x$ is extreme. Even if we subsequently coarsify the graph and increase the dimension, so long as the components are not disconnected, we are still assured that $x$, even though it is now represented in a higher dimension, is extreme, and it is defined by the precedence constraints that constitute the spanning forest in the new graph together with the side constraints.[14]

## 5 The Algebra of Max Closures

Let $x$ be an optimal solution to a GPCP defined by a graph $G$ and side constraints $Dx = d$. The following lemma follows from Lemma 14 and the decomposition $x = x^i + \Theta\alpha$ defined in Theorem 30, where $x^i$ is the integer part of $x$, and the columns of $\Theta$ are the connected components of the graph $G(x)$ defined by removing the nodes with integer $x$ value and the nonbinding arcs:

**Lemma 44** *Let $x$ be an optimal solution to a GPCP associated with a graph $G$, and let $\mu$ be optimal duals corresponding to $Dx = d$. Then for any $\alpha \in (0, 1]$, $\{i \in G : x_i \geq \alpha\}$ is a max closure optimizing the lagrangian problem $\mathbf{L}(\mu)$ in which $Dx = d$ is dualized with penalties $\mu$.*

Note that since unions of max closures are max closures and intersections of max closures are max closures, there is a smallest max closure (which is contained in every max closure) and a largest max closure (which contains every max closure).

**Corollary 45** *Let $x$ be an optimal solution to a GPCP, and let $\mu$ be optimal duals corresponding to $Dx = d$. Then every node $i$ that is in the smallest max closure solving $\mathbf{L}(\mu)$ satisfies $x_i = 1$, and every node $i$ that is not in the largest max closure satisfies $x_i = 0$.*

This may lead to a suggestion for a strengthened implementation of the algorithm: Instead of getting a single max closure solving the lagrangian $L(\mu^k)$, and partitioning into the nodes that are in the closure and those that are not, we may consider getting the smallest and the largest max closures, and partition into those that are in the smallest, those that are not in the largest, and the rest.

Consider further that in one sense it may be considered a weakness of the GPCP algorithm that even if the algorithm were supplied an optimal dual vector $\mu$, it cannot in general use that to arrive at an optimal primal solution immediately, as any dual vector leads only to a partition of the node set into two parts, while an optimal primal solution can require a partition into as many parts as there are side constraints (plus two). With this in mind it may be sensible to look for a more

[13]Note that while this basis will uniquely define $x$, even if $x$ is optimal, this basis may not technically be optimal, as it may imply a dual infeasible vector (for the dual variables corresponding to the arcs). To attain dual feasibility one may need to use a different spanning forest. This could be obtained by performing degenerate pivots of the network simplex algorithm (the dual of max closure is min cost network flow).

[14]An alternative method which might work to obtain a basis in the absence of the of the condition that the number of components is equal to the number of side constraints, but in the presence of a known basis for a more consolidated graph, is as follows. Given a constrained closure problem with $n$ variables which was derived by collapsing some nodes in some prior graph, and an arbitrary basic feasible solution $x^*$ with $n$ associated binding constraints, consider the precedence constraints in this collection (i.e. those with nonbasic slack). Draw the graph generated by these arcs, and note that this graph is a forest $\mathcal{F}$, and now append all other arcs from the original problem to this graph except for those that connect trees in $\mathcal{F}$, to form a new graph $\tilde{G}$. If decollapsing nodes in $\tilde{G}$ does not disconnect any component of $\tilde{G}$, then $x^*$ (represented w.r.t. the deconsolidated space) remains an extreme point solution, and it is defined by any set of arcs that constitute a spanning forest of the decollpased $\tilde{G}$, the same set of $x = 0$ and $x = 1$ constraints as previously (if a node $i$ with $x_i = 0$ or 1 was such a defining constraint previously, and $i$ has been decollapsed, then we include the constraint for each node that had been collaped to $i$), and the same set of side constraints as previously.

powerful version in which more information is extracted from a dual vector, so that an optimal dual vector would lead to an optimal primal solution in one step.

A naive approach might then be to define a partition in which all nodes that are in the largest max closure but not in the smallest max closure are left unaggregated, as in this case we could certainly recover an optimal primal solution given an optimal dual. This is in general a bad idea, as there can be very many such nodes. A more subtle view however would be to note that there is more information in the dual vector than merely a single partition into two parts resulting from a single solution to the lagrangian max closure problem, and that perhaps we ought to be considering the set of all possible max closures solving the lagrangian problem.

**Definition 46** *A collection of sets that is closed under unions, intersections and complementations is called an algebra. A set $a$ in an algebra $A$ is called an "atom" if for every $h \in A$, either $a \cap h = a$ or $a \cap h = \emptyset$, i.e. $a$ is indivisible. Let $G$ be a graph with associated node weights. Denote the smallest max closure $C_m$ and the largest max closure $C_M$. Let $\mathcal{C}$ be the algebra generated by the max closures, where the universal set with respect to complementation is defined to be $C_M$, and let $\mathcal{A} = \{a_i\} \subset \mathcal{C}$ be the set of atoms of $\mathcal{C}$.*

**Lemma 47** *The atoms of an algebra generated from a finite collection of sets $A_1, \ldots, A_n$ are the nonempty sets*

$$\bigcap_{i \in S} A_{s_i} \cap \bigcap_{j \in S^c} A^c_{s_j} \tag{26}$$

*for all subsets $S \subseteq \{1, \ldots, n\}$, and the atoms partition the universal set.*

**Corollary 48** *$C_m \in \mathcal{A}$, and the weight of any atom $a \in \mathcal{A}$ other than $C_m$ is $0$.*

**Proof:** If expression (26) defining $a$ contains no terms $A^c$ then it is just the intersection of all max closures, namely $C_m$. Otherwise, note that the intersection of the $A$ terms is a max closure (if there aren't any $A$ terms, then their "intersection" is the universal set, which is also a max closure), and the intersection of the $A^c$ terms is the complement of a max closure, and so $a$ is of the form $B \cap C^c$, where $B$ and $C$ are both max closures. Now $B$ is the disjoint union of $B \cap C$ and $B \cap C^c$, where the former is also a max closure. So since the weight of $B$ and the weight of $B \cap C$ are the same, we conclude that the weight of $B \cap C^c$ is zero. ■

**Corollary 49**

- *Every closed set of atoms including $C_m$ is a max closure.*
- *If the nodes in each atom are collapsed to a single node, there are no directed cycles among the atoms in the resulting graph.*
- *If $C \in \mathcal{C}$ is the union of $C_m$ and the closure of an atom $a \neq C_m$, then $C - a$ is also closed. Thus every atom is the difference of two max closures.*
- *The collection $\mathcal{A}$ is the coarsest one such that every max closure can be written as a union thereof.*

**Proof:** The first statement follows directly from Corollary 48. The second follows from the fact that had a directed cycle existed, then no closure could separate any atom in the cycle from the others, but Lemma 47 states that every atom has a different profile of max closures to which it belongs. The third statement follows from the second, since had $C - a$ not been closed, then since $C$ is closed (as it is a union of closed sets), there must be an arc from some atom in $C - a$ to $a$. But as $C_m$ is itself a closure, this arc must emanate from the closure of $a$ itself, yielding a direct cycle. The fourth statement follows from the third. ■

**Theorem 50** *[PQ80] Given a graph $H$ with designated source node $s$ and sink node $t$, let $\tilde{H}$ be the residual graph of any maximum $s$-$t$ flow in $H$. Any closure in $\tilde{H}$ separating $s$ from $t$ is a minimum $s$-$t$ cut in $H$ and conversely.*

The following, also due to ([PQ80]) now follows directly from Theorem 8 (due to ([P76]).

**Corollary 51** *Let $G$ be a graph with associated node weight vector $w$. Define a graph $H$ on the same node set, but with an additional "source" node $s$ with an edge of capacity $w_i$ to each node $i$ for which $w_i > 0$, and an additional "sink" node $t$ with an edge of capacity $-w_i$ from every node $i$ for which $w_i < 0$, and an edge of an infinite capacity for each edge in $G$. Determine a max flow $x$ in $\tilde{G}$ from $s$ to $t$, and obtain the residual graph $\tilde{H}$. Collapse the closure of $s$ in $\tilde{H}$ to a single node, which we will call $C_m$ (as this set minus the node $s$ is the smallest max closure of $G$), collapse all strongly connected components to single nodes and delete the anticlosure of $t$ from $\tilde{H}$. The* closures *of the resulting graph $\tilde{G}$ which include $C_m$ are exactly the* max closures *of the original graph $G$ (when decollapsed). Note that the (decollapsed) node set of $\tilde{G}$ is the largest max closure.*

**Lemma 52** *Under the conditions of Corollary 51, the atoms of the algebra $\mathcal{A}$ defined in Definition 46 are exactly the nodes of $\tilde{G}$.*

**Proof:** Note that since we collapsed all directed cycles, and since $C_m$ is in no directed cycle, we can use the same argument as in Corollary 49 to show that each node $u$ in $\tilde{G}$ other than $C_m$ is the difference between two max closures, and therefore it must be a union of atoms. But had $u$ been a union of more than one atom, then since each atom is also the difference between two max closures, it would follow that the nodes of $\tilde{G}$ would not be able to distinguish between these two closures, contradicting Corollary 51. To see this, assume that $u$ contains two atoms $a_1$ and $a_2$, and that $a_1$ is the difference between closures $C_1$ and $C_2$, i.e. $C_2$ is the disjoint union of $C_1$ and $a_1$. Then had it been possible to write $C_2$ as a union of nodes from $\tilde{G}$, then these nodes must include $u$ in order to include $a_1$. This implies that $a_2$ is in $C_2$, as well, which in turn implies that $a_2$ is in $C_1$ (as the only difference between $C_1$ and $C_2$ is $a_1$). But then it would be impossible to describe $C_1$ as a union of nodes from $\tilde{G}$, as it must exclude $u$ in order to exclude $a_1$, but it cannot exlude $u$ without excluding $a_2$. ■

**Corollary 53** *The partition of the node set of $G$ defined by the graph $\tilde{G}$ of Corollary 51, along with $C_M^c$, is the coarsest partition that can describe every max closure.*

This then suggests a modification of the GPCP algorithm in which instead of using the partition into two parts yielded by a single solution to $\mathbf{L}(\mu^{\mathbf{k}})$, we use the partition yielded by the Picard-Queyranne procedure which captures all solutions to $\mathbf{L}(\mu^{\mathbf{k}})$. The following lemma follows directly from Lemma 51 and Corollary 53.

**Lemma 54** *If Step 3 of the GPCP algorithm is performed using the partition yielded by the Picard-Queyranne procedure, and if $\mu^k$ is optimal, then the solution to the next restricted problem $P_2^{k+1}$ will be an optimal solution to the original problem $P_1$.*

As noted at the beginning of the section (and elsewhere), an optimal primal solution $x$ can be decomposed as $x^i + \Theta\alpha$, where $x^i$ is the integer part of $x$ and the columns of $\Theta$ are the incidence vectors of the components of $G(x)$ (the graph after removing integer nodes and nonbinding arcs), and these components have lagrangian value 0 w.r.t. the optimal duals $\mu$. The results here reflect the fact that these components are actually unions of atoms, and thus by fully representing the diversity of the lagrangian solutions we can capture the full primal solution.

The first hesitation however, is that there may be many more atoms than components, and thus using the Picard-Queyranne procedure may generate large restricted LP's. It can be shown however that the atoms essentially represent the diversity of the primal solution, i.e. *every* primal solution, with its distinct set of components, can always have its components decomposed into the same atoms (and so no primal solution assigns multiple values to the elements of any one atom). Moreover the atoms represent this diversity efficiently, in the sense that under certain conditions, there are primal optimal solutions that assign different values to each atom within a component. This leads to the conclusion that if the optimal primal is unique, then we may very well be able to guarantee that the number of atoms is small.

We will not pursue these ideas in detail here though, as in practice we have not been successful in using this variation to obtain unambiguous algorithmic improvements. We believe that the reason for this is primarily that the advantages of being able to capture all of the lagrangian solutions manifest for the most part when the algorithm is near optimality anyway. When far from optimality the optimal lagrangian solutions tend to be unique (as the zero valued sets that are associated with multiple optimal solutions do not tend to appear otherwise), so that no extra information is yielded. But when close to optimality the existing algorithm can usually do a more targeted job of finding the relevant unions of atoms anyway. (Note that slight changes in the duals will push near-zero valued sets in or out of the lagrangian max closures).

Nevertheless the idea that we may be able to extract more information from the lagrangian solution than merely the binary partition described in the standard version of the GPCP algorithm is one that we have managed to use successfully. In the next section we will discuss this in some detail.

## 6 Parcel Assignment Problems, PCPSP and Rich Partitioning

In the mining industry, a "job" is the extraction of a particular unit of earth. It is often the case that for the purposes of defining the extraction schedule over periods of time on the order of a year or more, that it is acceptable for the units of earth considered to be somewhat coarse. Thus block sizes of $30 \times 30 \times 30\ m^3$ may be acceptable. But blocks of such a size can be highly heterogeneous, and thus for the purposes of the processing decisions it is desired to subdivide these large blocks into smaller units. These units are called "*parcels*" in the mining industry, and they can be thought of as "sub-jobs".

For the purposes of the model, the extraction of a portion of a unit of earth is treated as the extraction of that portion of each parcel within the unit, i.e. there is a single "dig'' decision specifying the proportion of the entire larger unit to be extracted in a period, and which is inherited by all of its parcels. The processing decisions however, are separate for each parcel. Allowing for parcels yields the following generalization of Definition 2.

**Definition 55** *The Precedence Constrained Production Scheduling Problem With Parcels is defined as follows:*

*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 job $j \in \mathcal{N}$, given a set of parcels (sub-jobs) $P(j)$*

*Given $R$ processing options for each parcel.*

*Given $T$ scheduling periods.*

*Let $y_{j,t} \in \{0, 1\}$ represent the choice to perform job $j$ in period $t$.*

*For each $J \in \mathcal{N}$ and $j \in P(J)$, let $x_{j,t,d} \in [0, 1]$ represent the proportion of parcel $j$ performed in period $t$, and processed according to processing option, or "destination", $d$*

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

*The LP relaxation of the problem, which we will refer to as PCPSP-Parcel, is as follows:*

*maximize $c^T x$ subject to:*

$$\sum_{\tau=1}^{t} y_{i,\tau} \leq \sum_{\tau=1}^{t} y_{j,\tau}, \ \forall (i,j) \in \mathcal{A}, \ t = 1, \ldots, T \tag{27}$$

$$Dx \le d \tag{28}$$

$$y_{J,t} = \sum_{d=1}^{R} x_{j,t,d}, \ \forall j \in P(J), \ J \in \mathcal{N}, \ t = 1, \dots, T \tag{29}$$

$$\sum_{t=1}^{T} y_{j,t} \le 1, \ \forall j \in \mathcal{N} \tag{30}$$

$$x \ge 0$$

Note that it is easy to see that the parcels of a single job can be modelled as separate jobs, but with a directed precedence cycle connecting them (e.g. if $P(J) = \{j_1, j_2, j_3\}$ then $\mathcal{A}$ would contain arcs $j_1 \to j_2 \to j_3 \to j_1$). Thus the addition of parcels is not truly a generalization but rather a notational convenience.

We consider a variety of the problem for which the number of parcels remains large but the number of *jobs* is small (so that only the number of constraints (29) will be large). The physical interpretation of such a problem as a mine scheduling problem would be that a planner has somehow aggregated together the earth in the orebody into a small number of very large units, and has determined that the extraction schedule can be assumed to be defined in terms of those units. The planner however has made no predetermination as to how the original component units and parcels within each of those units is to be processed. We will refer to problems of this variety as "*Parcel Assignment Problems*" to emphasize that the principal difficulty is the assignment of a processing option to each parcel (i.e. constraint 29).

**Definition 56** *A Precedence Constrained Production Scheduling Problem With Parcels for which the number of parcels is "large" but the number of jobs is "small" will be referred to as a Parcel Assignment Problem.*

We first consider what is perhaps the easiest version of such a problem.

## 6.1 Cutoffs

**Definition 57** *The Simple Parcel Assignment Problem is a parcel assignment problem for which:*

- *There are two processing options denoted as*
  1. *Process Plant*
  2. *Waste Dump*
- *There is one side constraint per period, which is a periodic capacity constraint on "process plant" (i.e. a nonnegative knapsack constraint with nonzero coefficients corresponding only to $x$ variables for "process plant" in the given period). Waste is uncapacitated.*
- *The value of a parcel sent to waste never exceeds the value when sent to the process plant.*

The Simple Parcel Assignment Problem is almost the same as the problem considered in [BDFG09] (in their problem there are two capacity constraints per period). In that paper an algorithm is described to solve the linear program, which works out to be basically the same algorithm that results from applying our Algorithm Template.

**Definition 58** *Let destination* 0 *be the processing plant and let destination* 1 *be waste. Define the "processing value" of a parcel $j$ in period $t$ as*

$$v_{j,t} = c_{j,t,0} - c_{j,t,1} \tag{31}$$

*that is, the extra value to be gained by sending the job for processing.*

*Additionally, define the contribution of parcel $j$ in period $t$ toward the capacity constraint for period $t$ as $w_{j,t}$.*

**Lemma 59** *In an optimal solution to a Simple Parcel Assignment Problem, suppose in period $t$, a parcel $j$ with processing value $v_{j,t}$ and process plant consumption $w_{j,t}$ and belonging to job $J$ which is (partially) carried out in period $t$, has some proportion sent to the processing plant. Then any parcel $j'$ belonging to any job $J'$ that is (partially) performed in period $t$ and such that*

$$v_{j',t}/w_{j',t} > v_{j,t}/w_{j,t}, \tag{32}$$

*will have all of its carried out proportion assigned to the processing plant.*

*Conversely, if parcel $j$ had some proportion sent to waste in period $t$ then any parcel $j'$ (partially) performed in period $t$ with $v_{j',t}/w_{j',t} < v_{j,t}/w_{j,t}$ will have all of its carried out proportion assigned to waste.*

*There therefore exists a* cutoff *value $V_t$ for each period $t$ such that every parcel $j$ performed (partially) in $t$ with $v_{j,t}/w_{j,t} > V_t$ will be sent to the process plant, and every parcel $j$ performed (partially) in $t$ with $v_{j,t}/w_{j,t} < V_t$ will be sent to waste.*

The lemma says that in any period there must be some value per unit process capacity such that all extracted parcels with better value per unit are sent to the process plant and all extracted parcels with worse value per unit are sent to waste. The proof is straightforward, and in any case follows from the proof of the following theorem.

**Theorem 60** *The optimal dual $\mu_t$ corresponding to the capacity constraint in period $t$ is a cutoff $V_t$, where "cutoffs" are as defined in Lemma 59.*

**Proof:** Let $\mu_t$ be the optimal dual variable associated with the capacity constraint for period $t$.
Let $\delta_{J,t}$ be the optimal dual variable associated with the constraint $y_{J,t} = \sum_{d=1}^{R} x_{j,t,d}$.
Let $\pi_{j,t,d}$ be the optimal dual variable associated with the nonnegativity constraint $x_{j,t,d} \geq 0$.
These are the only constraints in which the variable $x_{j,t,d}$ appears, and where $d$ is the waste destination it does not appear in any capacity constraint. Thus where destination 0 is the process plant and destination 1 is waste, the dual constraint associated with the variable is $x_{j,t,0}$ is

$$c_{j,t,0} = \mu_t w_{j,t,0} + \delta_{J,t} - \pi_{j,t,0} \tag{33}$$

and the dual constraint associated with the variable is $x_{j,t,1}$ is

$$c_{j,t,1} = \delta_{J,t} - \pi_{j,t,1}. \tag{34}$$

Subtracting the latter constraint from the former constraint yields:

$$v_{j,t} = \mu_t w_{j,t,0} - \pi_{j,t,0} + \pi_{j,t,1} \tag{35}$$

If $y_{J,t} > 0$, then $x_{j,t,0} + x_{j,t,1} > 0$ and by complementary slackness

$$x_{j,t,0} > 0 \Rightarrow v_{j,t}/w_{j,t,0} \geq \mu_t \tag{36}$$

$$x_{j,t,1} > 0 \Rightarrow v_{j,t}/w_{j,t,0} \leq \mu_t \quad \blacksquare \tag{37}$$

## 6.2 Generalized Cutoffs

Now we return to the general Parcel Assignment Problem for which we allow arbitrary destinations and arbitrary side constraints.

**Theorem 61** *Given a General Parcel Assignment Problem, let $\mu$ be the optimal dual values corresponding to the side constraints, and given a parcel $j$ in job $J$ and a period $t$ such that $J$ has been (partially) performed in period $t$, let $D^{j,t,d}$ be the column of the side constraints corresponding to the variable $x_{j,t,d}$, and let $\mathcal{D}$ be the set of destinations $d$ for which the expression*

$$c_{j,t,d} - \mu^T D^{j,t,d} \tag{38}$$

*is maximized. An optimal primal solution must process the entire proportion of parcel $j$ performed in period $t$ at destinations $d \in \mathcal{D}$.*

**Proof:** Let $x$ be an optimal primal solution, let $\mu$ be an optimal dual vector associated with the side constraints and let $\delta$ and $\pi$ be as in the proof of Theorem 60. Then for each variable $x_{j,t,d}$ the dual constraint is

$$c_{j,t,d} = \mu^T D^{j,t,d} + \delta_{J,t} - \pi_{j,t,d}. \tag{39}$$

By the nonnegativity of $\pi$ observe that for all destinations $d$

$$c_{j,t,d} - \mu^T D^{j,t,d} \le \delta_{J,t} \tag{40}$$

and by complementary slackness if $x_{j,t,d} > 0$ we must have

$$c_{j,t,d} - \mu^T D^{j,t,d} = \delta_{J,t}. \ \blacksquare \tag{41}$$

So the notion of "cutoffs" can be generalized for the General Parcel Assignment Problem, and the cutoffs are completely determined by the optimal duals of the side constraints.

**Observation 62** *Given optimal side-constraints duals $\mu$, and a variable $x_{j,t,d}$, with parcel $j$ in job $J$, if $d$ does not maximize (38) then $x_{j,t,d}$ can be replaced with 0, and if $d$ uniquely maximizes (38) then it can be replaced with $y_{J,t}$. Thus where the number of "ties" (i.e. instances in which the destination maximizing (38) is not unique) is small (and we have found this to typically be the case) and the number of jobs is small, the General Parcel Assignment Problem collapses to a small LP when the optimal cutoffs are known.*

**Observation 63** *In general, given any set of side-constraints duals, we never need more than one variable to describe all variables associated with any given block-period pair in the lagrangian problem, i.e. the $y_{J,t}$ variable. This is because any tie in the selection of the destination maximizing (38) can be broken arbitrarily for the lagrangian, as all choices satisfy the nondualized constraints, and thus the $x$ variables can be deleted from the problem altogether. This applies for the PCPSP as well, as indicated in Observation 6.*

## 6.3 Lessons of PAP

Based on the preceding analysis, the application of the Algorithm Template to PAP is quite obvious: The simplifying $H^k x = h^k$ constraints will be just the requirement that the cutoffs are as per the current iteration's dual vector $\mu^k$. We obtain this without even explicitly solving the lagrangian, and it results in a very small restricted problem $P_2^k$ which can be readily solved.

We have tested this implementation of the algorithm, and in our experience, even if one does not relax the $H$ constraints to ensure compatibility with the previous iteration's solution (Step 3 of the Algorithm Template), it usually very nearly attains the optimal solution within a small number $(5 - 10)$ of iterations.[15] The most straightforward way to maintain compatibility with the previous solution is rather than to replace all of the variables associated with a given job $j$ and period $t$ with a single variable representing the choice to process all parcels as per the current duals, to replace them instead with two variables, one representing the choice to process at destinations as per the previous solution, and one representing the choice to process at destinations as per the current duals. These two variables act essentially as two "destinations" for job $j$ in period $t$.

In any case, the key observation that is relevant to GPCP is the fact that the dual solution actually contains cutoff information in every period. This is masked if one looks only at the solution to the lagrangian. In that case we would see that in whichever period the lagrangian solution performed a job, it chose the destination as per the duals, but we would have missed the fact that optimal destination information is also provided for the periods in which the solution did not perform the job.

*The lesson is therefore that there is another partition hidden in the lagrangian solution besides the partition into nodes that are in the solution and nodes that are not.* ***This is the partition between the destinations that are best as per the duals, and those that are not.***

[15]Without fully carrying out Step 3 however, it does not usually converge. It should be noted though, that given a near optimal dual, and considering the fact that given an optimal dual one can obtain an optimal primal by solving a small LP, it is often possible to obtain convergence even without Step 3.

We will now briefly outline how the GPCP algorithm can be modified to take account of this partition. To simplify the presentation we will ignore the parcels, and treat this as a regular PCPSP, though the analysis will hold in the presence of parcels also. If we apply the transformation described in Lemma 4, the nodes of the GPCP are each (job, period, destination) triples, and recall that there is an arc from each such triple $(j, t, d-1)$ to $(j, t, d)$. Recall also that the solution value $x(j, t, d)$ is taken to mean the fraction of job $j$ performed either before period $t$ (and sent to any destination), or performed in period $t$ and sent to a destination $\leq d$. Equivalently, $x(j, t, d) - x(j, t, d-1)$ is the proportion of job $j$ performed in period $t$ and sent to destination $d$. Therefore the only way for a restricted problem to admit a solution in which $j$ was sent to destination $d$ in period $t$ is if the restricted problem separates between nodes $(j, t, d-1)$ and $(j, t, d)$.

**Definition 64** *Given a PCPSP modelled as a GPCP as in Lemma 4, and given a vector of penalties $\mu$, define the function $BD(j, t, \mu)$ to be the mapping, given $\mu$, from job and period to the most valuable destination in the lagrangian sense (i.e. that which maximizes (38), with ties broken arbitrarily). The "Best Destination Partition" associated with $\mu$ is the binary partition that separates all nodes $(j, t, d)$ for which $d \geq BD(j, t, \mu)$ from those for which $d < BD(j, t, \mu)$.*

**Lemma 65** *Under the conditions of Definition 64, if Step 3 of the GPCP algorithm is modified so that $\mathcal{C}$ refines the Best Destination Partition associated with $\mu^{k-1}$ as well, then if $P_2^k$ admits solutions in which a job $j$ may be performed in a period $t$, then it will admit solutions for which $j$ performed in $t$ can be processed at the destination $BD(j, t, \mu)$.*

## 6.4 Hidden Partitions and Apriori Partitioning

Another example of a partition that we may perceive to be implicit in the lagrangian solution of a PCPSP, is a non-binary partition of the jobs. Say that the lagrangian solution performed job $j$ in period $t$ and job $j'$ in period $t+1$, and let us ignore destinations to simplify the presentation. If we look only at the collection of nodes $(j, t)$ in the max closure, we could not distinguish between $(j, t+1)$ and $(j', t+1)$, as both belong to the max closure, though one belongs because it was performed in that period, and the other belongs because it was performed in the previous period. Thus if the LP solution were to try to perform $j$ in period $t+1$, it would have no scope to separate $j$ from $j'$. Here again we may wish to separate such elements from one another in line with the hidden partition implicit in the lagrangian solution.

**Definition 66** *Given a PCPSP modelled as a GPCP as in Lemma 4, and given a vector of penalties $\mu$, and a max closure $x$ solving $\mathbf{L}(\mu)$. If the set of jobs is denoted $B$, and there are $T$ periods, then $B$ can be partitioned into the collection $\{B_i,\ i = 1, \ldots, T+1\}$, where each $B_i$ is the collection of jobs performed in period $i$ (i.e. $j$ such that $x(j, i) = 1$ and either $i = 1$ or $x(j, i-1) = 0$), and $B_{T+1}$ is the collection of jobs never performed. The "Lagrangian Job Partition" is the partition of the nodes $(j, t)$ (ignoring destinations for ease of exposition) into collections $\{N_i, i = 1, \ldots, T+1\}$ where each $N_i = \{(j, t) : j \in B_i, 1 \leq t \leq T\}$.*

Another type of partition with which we may wish to refine $\mathcal{C}$ in Step 3 of the GPCP algorithm, is what we call an "*apriori*" partition. The motivating idea is that the algorithm essentially moves from the coarse to the fine, discovering along the way where nodes need to be separated from one another to maximize value and meet the constraints. We may happen to know up front that certain classes of nodes act independently of one another and are likely to need to have different values assigned. PCPSP in particular is a scheduling problem, and the side constraints are usually periodic (i.e. each side constraint usually only entails variables associated with a single period). This means that the variables associated with different periods are very likely to need to be separated in many cases, and an apriori partition by periods allows the algorithm to work in parallel discovering structure within each period individually.

It is also evident that to get the most out of the Best Destination Partition and the Lagrangian Job Partition, we ought to separate the nodes of different periods from one another, as the promise of those partitions is predicated on the ability to make different decisions about blocks or destinations in different periods. Note in particular that the algorithm described above for the Parcel Assignment Problem maintains separate variables for each period (and each job) in the restricted problems.[16]

**Definition 67** *Given a PCPSP with $T$ periods modelled as a GPCP as in Lemma 4, the "Period Partition" separates the nodes into sets $\{P_1, \ldots, P_T\}$ with each $P_i = \{(b,t,d) : t = i\}$.*

In some cases it may make sense to partition also by destination, particularly if many constraints only concern a single destination. If destinations however are defined abstractly then this may not help much. Still another apriori partition that might be considered is into the graph's connected components, as separate components may be expected to take different solution values in an optimal solution.

Our overall assessment based on our practical experience is that "hidden partitions" are broadly helpful, as they arise from the lagrangian itself and represent a genuine application of the algorithm template, while apriori partitioning should be used with care, depending on the particular problem characteristics. The balance ultimately that needs to be achieved is to get as much as possible out of each algorithm iteration without making the restricted LP's too large to solve quickly, and to prefereably avoid having to perform the consolidation step (Step 6) too frequently, as this can be destructive of information.

In this context we note however, that though utilizing the richer partitioning described will make the problems $\{P_2^k\}$ larger, it is possible (as noted in Section 3.4) to solve them with GPCP itself, rather than with simplex or another standard solver. For large problems (i.e. ones with far more nodes than side constraints) this is often the most effective strategy.

In the following section we will describe our implementation and outline our computational results.

## 7 Computational Experiments

In this section we present experimental results. All these tests (with one exception) were conducted using a single core of a dual six-core Xeon X5690 with respective CPU speeds of 3.47 GHz and 3.46 GHz, and with 192 GB of memory, running Windows 7. The LP solver we used for the subproblems was Cplex 12.5 and the min cut solver we used was our implementation of Goldberg and Tarjan's preflow push algorithm ([GT88]) (though the overall tenor does not change when using Hochbaum's pseudoflow algorithm ([H08])).

We cut off all optimizations at 50000 seconds, though to get some idea of how long it would take Cplex to actually solve these problems we tried to run the concurrent optimizer on the "zuck medium" problem in MineLib until optimality. This test was performed on a different computer than the others, a dual quad-core Xeon X5580 with respective CPU speeds of 3.2 GHz and 3.19 GHz, and with 96 GB of memory, running Windows Server 2008R2. The computer shut down unexpectedly after running the problem for over 16 days. Only the simplex log was shown (and in fact, judging from the CPU time and usage it seems that only primal simplex (applied to the dual problem) was running), and this showed an iteration value that was still about 6% away from optimality.

For all Cplex and Gurobi runs we attempted first to use the concurrent optimizer and all available threads, though for a number of runs there was insufficient memory to run the concurrent optimizer without resorting to virtual memory, in which case we used dual simplex. This is indicated by the placement of a * after the solution time in our tables. For some runs we were not able to even build the problem in memory for either Cplex or Gurobi, and this is indicated by a ** after the solution time. In two instances Gurobi encountered an error in constructing the problem (perhaps due to the size), and this is indicated by ***. The version of Gurobi used was 5.6.

The auxiliary linear programs solved at each iteration of our algorithm were solved with dual simplex when no warmstart information was available, and with primal simplex when either a starting basis or a starting solution was available. When the number of nodes in an auxiliary problem exceeded 8000 or the number of arcs exceeded 40000 then the algorithm was nested, i.e. we

[16] Applying the GPCP algorithm to PAP with apriori partitions into jobs and periods, and the Best Destination partition, in addition to the regular lagrangian partition, will yield the algorithm described in Section 6.3.

used our algorithm (warmstarted) to solve the auxiliary problem. When homogeneous constraints were present however, the bar was made higher, as these tended to be more numerically challenging for our algorithm. In these cases the algorithm was nested when the number of nodes exceeded 15000 or the number of arcs exceeded 75000, and if more than 50 homogeneous constraints were present then no nesting was performed at all unless perturbations were present as well.

The tests were performed on the publically available MineLib library ([EGMN12]) of mine planning problems (except for Mclaughlin Limit, as this is apparently just a truncated version of Mclaughlin, and P4HD, for which no PCPSP formulation is given). A basic preprocessing step was performed on all models with only knapsack side constraints, in which a single pass of max closure was performed. It can be shown that any block which is not selected by a max closure procedure (ignoring the side constraints) will not be selected in any period by the LP either, and so these blocks can be eliminated. The time to perform this preprocessing step is not recorded in the tables, though it never took more than a few seconds. We also did not count the time taken to generate the problems. In all cases but Mclaughlin this also took only a few seconds, though due the size of the Mclaughlin data set it took about 7 minutes to generate that problem.

Additionally we tested four additional real world problems provided by BHP Billiton[17], to which we refer as 'LargeProblem', 'Coal1', 'Coal2' and 'ManySC', which we felt were more challenging in various dimensions than the MineLib problems. 'LargeProblem' is constructed by taking three copies of the block set of a real mine which possesses a fairly dense precedence graph to begin with, and defining 100 (short) scheduling periods. This yields a problem with a very large number of variables and constraints.

The other three problems are distinguished primarily for having more, and more complex, side constraints and more destinations than most problems treated in the literature, while having on the other hand fewer decision variables and a less dense precedence structure. These essentially correspond to more advanced stages in the planning process, when the number of decisions is reduced, but the complexity of the questions is greater. The two coal problems have a large number of side constraints (between 3000 and 4000, about 400 per period), a large number of which are blend constraints (i.e. homogeneous side constraints), and therefore pose more difficulty for the algorithm than do the other problems.[18] 'ManySC' has even more side constraints (nearly 7000, about 1100 per period), but only 12 of them are homogeneous. Additionally, as indicated, these latter three have a relatively small number of variables and precedence constraints (as compared to the other problems), and so they are altogether a less natural fit for our algorithm.

In all problems, the side constraints were modelled as inequality constraints.

We present data for seven implementations of the algorithm:

1. The '"vanilla" version of the algorithm, in which $\mathcal{L}$ in Step 3 of the algorithm is defined by the binary partition $\{I(y^k), O(y^k)\}$ defined by the lagrangian solution, and in which no perturbation is performed.

2. The vanilla version with perturbation.

3. The vanilla version with gradual "constraint tightening" (we will explain what this is soon).

4. "Rich partitioning", in which $\mathcal{L}$ is defined as the refinement of the partitions $\{I(y^k), O(y^k)\}$, the "Best Destination Partition" defined in Definition 64, the "Lagrangian Job Partition" of Definition 66, and the "Period Partition" defined in Definition 67. No perturbation is performed.

5. Rich partitioning with perturbation.

6. Rich partitioning with perturbation and gradual constraint tightening.

[17] Data was masked.

[18] In practice, homogeneous constraints seem to pose more difficulty for the algorithm than nonhomogeneous constraints. It may be that the dual information associated with such constraints is more unstable and less meaningful due to its zero contribution to dual objective. This may particularly be the case when the left hand side is zero due to having no nonzero contributions.

7. "Tuned" settings.

As we noted when we described the algorithm, node consolidation (Algorithm Step 6), in general, increases the number of iterations to optimality, but decreases the restricted LP sizes. The basic principle of node consolidation is therefore to only consolidate when "necessary". This directive can, of course, be implemented in many different ways, with greater or lesser "intelligence", though for the purposes of this presentation we used a few simple rules.

- For the "vanilla" partitioning implementations, we consolidated whenever the number of nodes in the restricted problem exceeded 2500, but when consolidating we "kept" the previous six "solution partitions" (i.e. the $\{C^r\}$ partitions defined in Algorithm Step 6) and the six "best" (measured by lagrangian objective value) lagrangian partitions $\{I(y^k), O(y^k)\}$ encountered so far (even if by doing so we exceeded 2500 nodes).
- For the "rich" partitioning implementations, we consolidated whenever the number of nodes in the restricted problem exceeded 5000, but when consolidating we "kept" the previous four "solution partitions" (i.e. the $\{C^r\}$ partitions defined in Algorithm Step 6), the four "best" lagrangian partitions $\{I(y^k), O(y^k)\}$ encountered so far, and the four "best" "Best Destination Partition" (again measured according to lagrangian objective value).

Some more intelligence that could have been incorporated into the rules, but which was not included in any but the last implementation ("Tuned"), would be

- To enforce a minimum gap in the number of iterations between consolidations.
- To reduce the incidence or scope of consolidations if the algorithm is making slow progress.
- To associate thresholds with the number and types of partitions kept through consolidations.
- To link the thresholds with the number of binding side constraints.

Perturbations were implemented as relaxations. For $\leq$ knapsack constraints, the perturbation was $10^{-7}$ times the right hand side. For other constraints, we considered $10^{-6}$ times the maximum absolute coefficient, $10^{-5}$ times the median absolute coefficient, and $10^{-9}$ times the sum of the absolute coefficients. The relaxation was either the maximum of these three, or 100 times their minimum, whichever was less.

Considering that the constraints that give the most difficulty tend to be the homogeneous ones, in the absence of homogeneous constraints, the perturbations were reduced by a factor of ten. If at any iteration a dual solution was detected to have dimension $\geq 10$ (i.e. the number of fractional components in the iteration solution was at least 10 less than the number of linearly independent binding side constraints), this was taken as an indication that the perturbations were too small, and they were increased by a factor of 10. In order not to change the original problem too drastically, however, this was done a maximum of once in any solve. Note that in the presence of perturbations the dimension is meant to be zero, but numerical tolerances can change this. In particular the number of fractional components is highly dependent on the numerical tolerance of the definition of "equality".

After solving a problem with perturbations, we used the terminal partition and penalties to warmstart the problem with the perturbations removed, so as to obtain a solution to the non-perturbed problem. After removing perturbations the optimality gap was relaxed from 0 to $10^{-5}$, to reflect the fact that in the absence of any perturbations problems may become prohibitively difficult to solve to full optimality, though in most cases the terminal gaps were still quite close to 0. In practice this last step may not be necessary. Even if it is desired to remove the perturbations, when solving an IP the attempt to remove the perturbation may often be postponed to the terminal node.

The implementations with gradual constraint tightening begin with substantial relaxations of the homogeneous constraints, and milder relaxations of the other constraints, which are progressively tightened as the relaxed versions of the problem approach solution. The idea is to accelerate the algorithm by starting with simpler versions of the problem, so as to give the algorithm preliminary guidance in its choices of duals and partitions.

The reasoning here is that for hard problems that are far from optimality, the solutions to the restricted problems do not immediately begin to converge, and the resulting diversity in the iteration solutions increases the size of the restricted problems rapidly. But once the solutions begin to converge, this diversity decreases, and the growth is slowed. It is therefore to the algorithm's advantage that structure is recognized early, and needless diversity in solutions is avoided. Another benefit of this approach is that since the relaxations are fairly large, they also act as large perturbations, which reduces the risk of the perturbations falling under the "numerical radar" of the computer.

This approach is useful primarily for situations in which there is a large number of homegeneous side constraints, and so we only report results for this implementation for instances with homogeneous side constraints. Considering that there are any number of – basically similar – ways in which this may be implemented, we will not describe the details of our particular implementation. Our comprehensive test result file however, includes data that indicate the evolution of the relaxations as the algorithm progresses.

The final implementation, "Tuned", attempts to apply intelligence in selecting the various levers to push, as described above. It also segregates nodes whose values are fixed, though this is largely not relevant in any of these models. In general the results show that in most cases we haven't managed to squeeze that much more out of the lemon in this way than in using the simple implementations described above.

Comprehensive test results, showing iteration by iteration progress, for these as well as a number of other problem instances, and for these as well as some other implementations, can be obtained at http://www.columbia.edu/ dano/. Full data sets for all test cases may also be downloaded from this site.

We present here summary performance data for the problem instances. Some of the entries are self-explanatory, the others have the following meaning:

- **Blocks**. The units of earth whose extraction we wish to schedule. These are the jobs in Definition 2. If a block was fixed to be extracted in a particular period or not at all then it was not counted. For problems in which blocks contain multiple parcels, this is the number of parcels.

- **Variables**. The number of unfixed (block, period, destination) triples, or unfixed quadruples (block, parcel, period, destination) if a problem has multiple parcels per block. Note that this is typically less than the product of blocks, periods and destinations, as not every block may be available in every period and at every destination.

- **Problem arcs**. The number of arcs in the graph that the algorithm creates to represent the scheduling problem (as per Theorem 4 and Definition 3). Together with the side constraints, the $\geq 0$ bound constraints on each block in the first period and destination, and the $\leq 1$ bound constraints at the last period and destination, these correspond with the full set of constraints when the problem is recast as per Definition 3. Note that while Theorem 4 states that the restatement of the problem does not alter the number of variables or constraints, the number of problem arcs is usually larger than the reported number of constraints in the original formulation, as there is a problem arc corresponding to the $\geq 0$ bound constraint for each block at every intermediate period and destination, but the bound constraints are not traditionally included in the reported number of constraints.

- **Constraints**. As indicated, these refer to the number of constraints in the PCPSP formulation, and they do not include bound constraints.

- **Variables, Constraints Cpx presol**. Note that Cplex presolve in many cases formed the dual, so these are the statistics for the reduced dual problem.

- **Nonzero duals at optimality**. The number of side constraints with positive dual values in the final iteration of the "tuned'' implementation.

- **Iterations, time to $\mathbf{10^{-5}}$ 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. For runs with perturbations, this refers to the time at which this threshold was reached for the perturbed problem. For runs with gradual constraint tightening, it refers to the time at which this threshold was reached once the relaxation multipliers for the homogeneous constraints reached 10 (i.e. when their assigned relaxations were a factor of ten greater than what was chosen for the runs without gradual constraint tigtening), which is the final value they are assigned before being turned off.

- **Iterations, time to optimality**. The number of iterations (resp., the CPU time) taken by the algorithm until it terminated, obtaining a solution it could certify as *optimal* as per the algorithm's termination criteria, or for which the gap was $\leq 0$. Notice that this implies that the solution is optimal as per the numerical tolerances of Cplex. For perturbed problems this is the time until the problem with its perturbation reached optimality, i.e. the time at which all perturbations and relaxations were switched off. Note that for perturbed problems, if optimality was only certified at the following iteration's lagrangian (at which point the perturbations were removed), then the time recorded here does not include this certification time.

- **Iterations, time to termination**. This is only different than the previous item for perturbed implementations, or for problems that timed out. It records the total time and iterations including those performed after removing all perturbations, until termination at either optimality or time out.

- **Gap at termination**. Gap between upper and lower bounds at termination. This can be nonzero due to numerical tolerances. Also, as mentioned above, for perturbed problems the optimality tolerance is reduced to $10^{-5}$ after removing perturbations. Also, if a problem timed out prior to obtaining optimality, then this records the gap at termination.

- **Lagrangian time**. The total CPU time expended by the Lagrangian procedure until termination.

- **Subproblem LP time**. The total CPU time expended solving auxiliary linear programs until termination. This may exceed the time out of 50000 seconds if the cut off time was reached in the middle of an iteration.

Finally, an entry of **"—"** for an elapsed time means that the operation timed out after 50000 seconds of CPU time.

Some general observations regarding performance:

- For problems in which there were no homogeneous side constraints, perturbations made essentially no difference.

- While this information is not recorded here, a look at the detailed log will show that for the problems in the current data set with no homogeneous side constraints, the dimensionality of the dual solutions was always zero even with no perturbations. In our expanded data set there was one instance of a problem with no homogeneous side constraints which had positive dual dimension occasionally, but here too the dimensionality was usually well under 10. For other problems, in particular for the coal problems, the dimensionality in the absence of perturbations could reach the hundreds, but perturbations were usually sufficient to reduce it to zero, or at least to well under 10. In general, and this is borne out in our expanded data set as well, high dual dimensionality was associated with slow convergence and sometimes with greater numerical difficulties. Interestingly however, while 'ManySC' recorded high dual dimensionality without perturbation, and close to zero dimensionality with perturbations, solution times over all implementations were fairly comparable, showing only about a 25% improvement in speed for the perturbed implementations with rich partitioning.

- For problems with a large number of periods and slow lagrangian performance, rich partitioning made a dramatic difference. The reason for this is, as we will see in the next section, that vanilla partitioning is guaranteed to require at least as many iterations for most problems as the number of periods, and in fact the 100 period problem required between 150 and 200 iterations to solve for all vanilla partitioning implementations. The rich partitioning implementations however were within a $10^{-5}$ factor of optimality within 10 iterations and reached optimality within 20.

- The two coal problems encountered extreme numerical difficulty when no perturbations were added. Larger perturbations tended to help more, which probably contributed to the fact that the gradual tightening implementation worked best for these problems. In the absence of any perturbations individual iterations often took longer to solve than did the full unconsolidated problem (though the full problem was solved with concurrent Cplex, rather than a warmstarted primal simplex). In general, the presence of homogeneous constraints was often associated with greater numerical difficulties, and the presence (and magnitude) of perturbations with reduced numerical difficulties, and potentially dramatically faster solution times.

- Even with many thousands of side constraints and a configuration with a relatively low number of variables and constraints, the tuned algorithm was still competitive (or outperformed) concurrent twelve-threaded Cplex and Gurobi, and it outperformed their respective simplex algorithms by wide margins.

- Another implementation that we considered was one in which rather than to use solution partitions (i.e. the $\{C^r\}$ partitions defined in Algorithm Step 6) defined by the fractional components in the solution graph, we used a variant defined by the solution basis. In the standard solution partition, a part is defined for the nodes at zero value, and a part is defined for the nodes at 1 value. The remaining graph then has all nonbinding arcs removed, and then a part is descibed for each component. In this variant, a part is defined for nonbasic variables at zero, a part for nonbasic variables at 1 (and if an arc constraint connecting a nonbasic variable to a basic variable has nonbasic slack, then the basic variable is lumped together with the nonbasics). In the remaining graph we remove all arcs with basic slack, and then define a part for each component of the resulting graph.

  This procedure produces a partition that refines the standard solution partition, as it may break up some of its components (if a binding arc has degenerate basic slack), or may include some integer valued nodes in its component graph (if they are degenerate basic). It can be shown that the resulting graph always has as many components as there are nonbasic side constraints (and so the number of parts is always "small'').

  If there are perturbations then the two partitions should be identical, but due to numerical tolerances they may not be. The more interesting case however, is in the absence of perturbations, as the "basis partition'' we have described here can in that case be significantly finer than the standard solution partition, and therefore contain more information. It actually contains all the information associated with its dual, as it uniquely defines it, and in fact the algorithm in some cases ran considerably faster for this variant when no perturbations were used. For example, the vanilla algorithm with no perturbations in its original version ran for the full 50,000 seconds on Coal1, terminating with a 7% gap after 1581 iterations, while with this variant it obtained optimality after 13,385 seconds and 312 iterations. Note though that it still could not solve Coal2 within 50,000 seconds, and in general it still underperformed by far the implementation with perturbations.

  In the presence of perturbations this implementation also improved solution times in some cases, but the differences were more marginal. The biggest improvements were in Coal1 and Coal2, for which tuned solve times reduced to 1665 seconds and 581 seconds, respectively. Further details can be found in the comprehensive log file.

## 8 Further Work

From a theoretical standpoint we know that the GPCP algorithm converges finitely, and we also know that the algorithm template converges pseudologorithmically when there is only one side constraint. It is also easy to construct examples where the algorithm requires as many iterations to optimality as there are side constraints.

**Lemma 68** *Consider the problem* $\max x_1 + \cdots + x_k$ *subject to* $x_i \leq 1/(i+1),\ i = 1, \ldots, k$*. This is a GPCP with no arcs and* $k$ *side constraints. The GPCP algorithm solves this problem in* $k$ *iterations.*

A more interesting example is due to Goycoolea et al [EGMM12], as it applies to most of the classical open pit mine planning problems in the academic literature.

**Lemma 69** *Consider a PCPSP with* $T$ *timeperiods for which all side constraints are knapsacks of the form* $ax \leq b$*, and for which each individual constraint has nonzero coefficients only for variables* $x(b, t, d)$ *associated with a single period* $t$*. Assume further that for each period* $t$ *after the first, there is a "discount factor"* $\delta_t,\ 0 < \delta_t < 1$*, such that for every node* $(b, p, t)$*, the objective function contributions satisfy* $c(b, p, t) = \delta_t(c(b, p, t-1))$*. If the GPCP algorithm is applied with* $\mathcal{L}$ *in Step 3 defined in its "vanilla" version as* $\{I(y^k), O(y^k)\}$*, then the algorithm requires at least* $t$ *steps to obtain a solution in which any blocks are extracted in period* $t$*.*

We have seen in the computational experiments that when richer partitioning schemes are used convergence can be much faster in this example (the 100 period problem terminated in under 20 iterations), but the example is still instructive.

In principle, the algorithm needs to discover which variables need to be separated from one another. Each constraint individually could drive this in its own way, but if the constraints are not tight then they have zero dual and cannot drive anything. Thus the algorithm first needs to discover that a constraint is relevant, and we have now seen that in principle this can require at least as many iterations as there are side constraints. Then it needs to pin down which variables exactly should be separated - and this step even with a single constraint alone can be pseudologorithmic.

In practice the algorithm is very quick. Problems with millions of variables and tens of millions of constraints typically converge in under 20 iterations (for the rich partitioning implementations), even when there are hundreds of side constraints. As we have seen in the computational results, there are even problems with many thousands of side constraints that converge in under 20 iterations. Qualitatively this may perhaps be expected due to the virtuous cycle effect that the algorithm exploits, but quantitatively we still do not have any tight characterization of convergence properties.

Another area for further research is whether and how the results obtained for the general precedence constrained problem could be applied to other problems whose constraint matrix is an "easy problem" plus some extra side constraints. Minimum cost network flow problems in particular share the structure in which the lagrangian could be solved with a fast combinatorial algorithm, as well as the structure that the extreme point LP solutions are only a "little" fractional when the number of side constraints is small (as discussed in the Appendix).

Finally, the integer programming relevance of the result that the number of distinct fractional values in the extreme point LP solution cannot exceed the number of binding side constraints – as well as the parallel results for network flow and more general TUM problems – needs to be further investigated.

## A Decomposition for TUM Problems with Side Constraints

We saw in Section 3.4 that the decomposition $x = x^i + \Theta\alpha$ defined in Lemma 11, when applied to GPCP with graph $G$ and $q$ side constraints, amounts to the statement that an extreme point solution of the LP relaxation of GPCP can be decomposed into its integer portion plus a fractional combination of no more than $q$ indicator vectors of various subgraphs of $G$. We show here that a similar result holds for all problems defined by a totally unimodular constraint matrix with additional side constraints.

**Lemma 70** *Under the conditions of Lemma 11, assume further that $A$ is totally unimodular and that $b$ is integer. Then $x^i$ can be chosen to be integer and to satisfy:*

1. $Ax^i \leq b$
2. $x^i_j = x_j, \ \forall j : x_j$ *is integer*

**Proof:** 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 TUM 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 $\bar{A}x^i = \bar{b}$. ■

The lemma says that any extreme point LP solution $x$, can be decomposed as an integer solution satisfying the $A$ constraints and matching $x$ wherever $x$ is integer, plus a linear combination of no more than $q$ vectors from the null space of $\bar{A}$. In this sense, each side constraint from $D$ adds only a one "unit of fractionality" to the solution.

**Observation 71** *Under the conditions of Lemma 70, vectors $\{\theta^1, \ldots, \theta^s\}$, as defined in Lemma 11, exist for which all values for each $\theta^r$ are either $0$, $1$ or $-1$. Thus any extreme point solution $x$ can be decomposed as an integer vector (matching with $x$ wherever $x$ is integer) and a linear combination of no more than $q$ $(0, 1, -1)$ vectors.*

**Proof:** All we need to show is that the null space of $\bar{A}$ is spanned by $(0, 1, -1)$ vectors. Choose a maximal linearly independent subset of the rows of $\bar{A}$, and call this matrix $\breve{A}$. Note that removing linearly dependent rows does not affect the null space, so we need only show that the null space of $\breve{A}$ is spanned by $(0, 1, -1)$ vectors. Say that $\breve{A}$ is an $m \times n$ matrix, and choose $m$ linearly independent columns of $\breve{A}$ to form the basis $B$, and let the remaining $n - m$ columns be denoted by the matrix $N$. The null space of $\breve{A}$ is spanned by the vectors $\{(B^{-1}N^j_{\breve{A}}, -e^j)^T\}$, where $N^j_{\breve{A}}$ is the $j$'th column of $N$ and $e^j$ is the $j$'th unit vector in $R^{n-m}$. But since $\breve{A}$ is TUM, it can be shown that $B^{-1}N$ is also TUM (see [S86] page 272, Theorem 19.5), and so these are all $(0, 1, -1)$ vectors. ■

The special case of network flow problems with side constraints is particularly interesting.

**Corollary 72** *Let $P$ be the feasible space of a minimum cost network flow problem with integer data and side constraints. Let $x$ be an extreme point of $P$, and let $q$ be the number of linearly independent side constraints. Then $x$ can be decomposed into an integer flow satisfying all network flow (but not necessarily side) constraints and equal to $x$ wherever $x$ is integer, and a sum of no more than $q$ fractional cycle flows, where none of these cycles includes any edge for which $x$ was integer.*

**Proof:** Let $A$ be the node-arc incidence matrix, along with the bound constraints. The node-arc incidence constraints are always binding, and therefore the vectors $y \in N^x_{\breve{A}}$ all satisfy $Ay = 0$, i.e. they are circulations, which can be decomposed into cycle flows. The result now follows from Lemma 59. ■

It should be noted though that while there are no more than $q$ distinct fractional cycle flows in the decomposition of $x$, the number of distinct fractional values in $x$ can be exponential in $q$.

## B A Geometric Interpretation of the Algorithm Template

**Definition 73** *Consider an optimization problem*

$$(P): \ \max\{cx : x \in R^n, Ax \leq b, Dx = d\} \tag{42}$$

*where $D$ has $m$ rows, and for each $\mu \in R^m$, let*

$$L(\mu) = \max\{cx - \mu(Dx - d) : x \in R^n, Ax \leq b\} \tag{43}$$

*be the lagrangian relaxation of $(P)$ with respect to the $D$ constraints, and let*

$$\mathcal{L} = \{y \in R^{m+1} : y_{m+1} \geq L(y_1, \ldots, y_m)\} \tag{44}$$

*be the epigraph of $L$. Consider also a sequence of optimization problems*

$$(P^i): \ \max\{cx : x \in R^n, Ax \leq b, Dx = d, H^i x = h^i\} \tag{45}$$

*with lagrangians*

$$L^i(\mu) = \max\{cx - \mu(Dx - d) : x \in R^n, Ax \leq b, H^i x = h^i\} \tag{46}$$

*and epigraphs*

$$\mathcal{L}^i = \{y \in R^{m+1} : y_{m+1} \geq L^i(y_1, \ldots, y_m)\} \tag{47}$$

*where each $H^i$ is a matrix and each $h^i$ is a vector such that all $x$ satisfying $H^i x = h^i$ also satisfy $H^{i+1}x = h^{i+1}$, and such that the following property is satisfied: Let $\{\mu^i, i > 1\}$ be a sequence of dual vectors corresponding to the $D$ constraints such that $\mu^i$ is dual optimal for $(P^{i-1})$, and let $\{\bar{x}^i\}$ be a sequence such that $\bar{x}^i$ is an optimal solution to $L(\mu^i)$, and assume that for all $1 \leq j \leq i$, $\bar{x}^j$ satisfies $H^i \bar{x}^j = h^i$. In words, each subsequent $(P^i)$ is an increasingly relaxed restriction of $(P)$ chosen so that all prior lagrangian solutions $\bar{x}^j$ satisfy the current restriction $H^i x = h^i$.*

*We will assume throughout that $Ax \leq b$ contains constraints of the form $l_i \leq x_i \leq u_i$, $i = 1, \ldots, n$, where $l$ and $u$ are finite.*

**Lemma 74** *For all $\mu \in R^m, L^i(\mu) \leq L^{i+1}(\mu) \leq L(\mu)$, and for all $j \leq i$, $L^i(\mu^j) = L(\mu^j)$.*

**Proof:** The first statement follows from the fact that $L^i$ is a restriction of each subsequent $L^{i+1}$ and also of $L$, and the second follows from the fact that for each $j \leq i$, the solution $\bar{x}^j$ that maximizes $L(\mu^j)$ also satisfies $H^i \bar{x}^j = h^i$ and therefore maximizes $L^i(\mu^j)$ as well. ■

**Lemma 75** *The sets $\mathcal{L}$ and $\mathcal{L}^i$ are all convex. Every point on the surface of $\mathcal{L}$ or $\mathcal{L}^i$ therefore lies on at least one face of the set.*

**Proof:** Note first that the function $L(\mu)$ is convex, since

$$L(\lambda u + (1-\lambda)v) = \max_{x:Ax \leq b}\{cx - (\lambda u + (1-\lambda)v)(Dx - d)\} = \tag{48}$$

$$\max_{x:Ax \leq b}\{\lambda(cx - u(Dx - d)) + (1-\lambda)(cx - v(Dx - d)\} \leq \tag{49}$$

$$\lambda(\max_{x:Ax \leq b}\{cx - u(Dx - d)\}) + (1-\lambda)(\max_{x:Ax \leq b}\{cx - v(Dx - d)\}) = \tag{50}$$

$$\lambda L(u) + (1-\lambda)L(v). \tag{51}$$

It now follows that for any two vectors $u, v \in \mathcal{L}$, where we write $u = (\bar{u}, u_{m+1})$ and $v = (\bar{v}, v_{m+1})$,

$$\lambda u_{m+1} + (1-\lambda)v_{m+1} \geq \lambda L(\bar{u}) + (1-\lambda)L(\bar{v}) \geq L(\lambda\bar{u} + (1-\lambda)\bar{v}) \Rightarrow \tag{52}$$

$$(\lambda\bar{u} + (1-\lambda)\bar{v}, \lambda u_{m+1} + (1-\lambda)v_{m+1}) \in \mathcal{L} \tag{53}$$

which proves that $\mathcal{L}$ is convex, which implies that every point on its surface lies on a supporting hyperplane of the set. The argument for $\mathcal{L}^i$ is similar. ■

**Corollary 76** *Given $i > 1$, for every point $(\mu^j, L(\mu^j))$, $j \leq i$, there is a face of $\mathcal{L}^i$ containing that point.*

**Notation 77** *We will describe hyperplanes in $R^{m+1}$ as being of the form $z = a\mu + b$ where $z$ represents the $m+1$'st coordinate, $a$ and $\mu$ are in $R^m$ and $b$ is scalar.*

**Lemma 78** *Let $F$ be a face of $\mathcal{L}^i$ containing $(\mu^i, L(\mu^i))$, then $F$ is also a face of $\mathcal{L}$ and of every subsequent $\mathcal{L}^k,\ k > i$.*

**Proof:** Represent the halfspace associated with $F$ as $z \geq a\mu + b$. As $(\mu^i, L(\mu^i))$ is on the surface of $\mathcal{L}$, the only way $F$ can fail to be a face of $\mathcal{L}$ is if there exists $y = (\bar{y}, y_{m+1}) \in \mathcal{L}$ such that $y_{m+1} < a\bar{y} + b$. But were this to be the case then by definition of $\mathcal{L}$, $L^i(\bar{y}) \leq L(\bar{y}) \leq y_{m+1} < a\bar{y} + b$ which implies that $(\bar{y}, L^i(\bar{y}))$, which is in $\mathcal{L}^i$, also fails to belong to $F$, which is a contradiction. The argument for $\mathcal{L}^k,\ k > i$ is the same. ■

Thus for each subsequent $L^i$, the epigraph of the lagrangian is tangent to $\mathcal{L}$ at an additional point. We will see the consequences of this soon, but first we will note that the faces of the epigraph can also be cast as subgradients and “violation vectors”.

**Observation 79** *Let $F$ be a face of $\mathcal{L}$ defined by $z \geq a\mu + b$ containing a point $(\tilde{\mu}, \tilde{z}) \in \mathcal{L}$. Then $a$ is a subgradient of $L$ at $\tilde{\mu}$. Conversely if $a$ is a subgradient of $L$ at $\tilde{\mu}$ then $\mathcal{L}$ has a face $z \geq a\mu + L(\tilde{\mu}) - a\tilde{\mu}$ tangent at $(\tilde{\mu}, L(\tilde{\mu}))$. Similar statements hold for each $\mathcal{L}^i$ and $L^i$.*

**Proof:** If $(\tilde{\mu}, \tilde{z}) \in \mathcal{L}$ lies on $F$ then it is on the surface of $\mathcal{L}$, which implies that $a\tilde{\mu} + b = \tilde{z} = L(\tilde{\mu})$ (or else it would be in the interior). Thus since $F$ is a face, for each $\nu \in R^m$ we have $L(\nu) \geq a\nu + b = a\tilde{\mu} + b + a(\nu - \tilde{\mu}) = L(\tilde{\mu}) + a(\nu - \tilde{\mu})$, which shows that $a$ is a subgradient of $L$ at $\tilde{\mu}$. Conversely if $a$ is a subgradient of $L$ at $\tilde{\mu}$ then every point $(\nu, z') \in \mathcal{L}$ satisfies $z' \geq L(\nu) \geq L(\tilde{\mu}) + a(\nu - \tilde{\mu}) = a\nu + L(\tilde{\mu}) - a\tilde{\mu}$. ■

**Definition 80** *Given a vector $x$ satisfying $Ax \leq b$, the “violation vector” $a \in R^m$ of $x$ is defined by $a_j = d_j - D^j x,\ j = 1, \ldots, m$, where $D^j$ is the $j$'th row of $D$.*

**Lemma 81** *Let $\mu \in R^m$, and let $\{x^1, \ldots, x^t\}$ be the set of all extreme point optimal solutions to $L(\mu)$, and let $\{a^{v_i},\ i = 1, \ldots, t\}$ be the associated violation vectors. Then for some $\epsilon > 0$, there exists an $\epsilon$ ball $N$ around $\mu$ in which*

$$L(\nu) = L(\mu) + \max_{i=1,\ldots,t} a^{v_i}(\nu - \mu),\ \forall \nu \in N. \tag{54}$$

**Proof:** Suppose $y$ is an extreme point of $\{x \in R^n : Ax \leq b\}$ that is suboptimal for $L(\mu)$. Then small enough changes to $\mu$ will leave it suboptimal. Since there are finitely many extreme points, it follows that in some small enough ball $N$ around $\mu$, no additional extreme points become optimal, and thus no new points become optimal. The objective value of each $x^i$ for $\nu \in N$ is $L(\mu) + a^{v_i}(\nu - \mu)$, and thus the lagrangian value at $\nu$ is just the maximum of these quantities taken over $1, \ldots, t$. ■

**Lemma 82** *A vector $a \in R^m$ is a subgradient of $L$ at $\mu \in R^m$ if and only if there exists an optimal solution $x$ to $L(\mu)$ for which $a$ is the violation vector of $x$.*

**Proof:** Suppose $x$ is an optimal solution to $L(\mu)$ with violation vector $a$. For any $\nu \in R^m$,

$$L(\nu) \geq cx - \nu(Dx - d) = cx - \mu(Dx - d) + (\mu - \nu)(Dx - d) = L(\mu) + a(\nu - \mu), \tag{55}$$

which shows that $a$ is a subgradient at $\mu$. Suppose now that $a$ is a subgradient of $L$ at $\mu$ and let $\{x^1, \ldots, x^t\}$, $\{a^{v_1}, \ldots, a^{v_t}\}$ and $N$ all be as in Lemma 81. Then by (54), for all $\nu \in N$,

$$a(\nu - \mu) \leq \max_{i=1,\ldots,t} a^{v_i}(\nu - \mu). \tag{56}$$

Suppose that $a$ is not in the convex hull of $\{a^{v_1}, \ldots, a^{v_t}\}$, then there exists a separating hyperplane, i.e. there exists a vector $p$ and a scalar $q$ such that $ap > q$ and such that $a^{v_i}p < q,\ i = 1, \ldots, t$. But then we could choose $\nu \in N$ so that $\nu - \mu$ is a scalar multiple of $p$, and this would contradict (56). We conclude that there exist $\lambda_1, \ldots, \lambda_t \geq 0$ with $\sum_{j=1}^t \lambda_j = 1$ such that $a = \sum_{j=1}^t a^{v_j}$. Define now $x = \sum_{j=1}^t \lambda_j x^j$. Observe that $x$ is optimal for $L(\mu)$ and its violation vector is

$$d - Dx = d - \sum_{j=1}^t \lambda_j Dx^j = \sum_{j=1}^t \lambda_j (d - Dx^j) = \sum_{j=1}^t \lambda_j a^{v_j} = a \tag{57}$$

which proves the remainder of the lemma. ■

**Observation 83** *Given a face $z \geq a\mu + b$ of $\mathcal{L}$, $||a||$ can be thought of as the slope of the face insofar as maximal change in $z$ per unit movement along a vector $\nu - \mu \in R^m$ is $||a||$. At the point $\mu^*$ which minimizes $L(\mu)$, the flat hyperplane $z = L(\mu^*)$ defines a face for which the slope is zero, the subgradient is zero, and the vector of violations is zero. More generally, a small slope means that the violations are small and that the lagrangian cannot get much lower in the vicinity.*

**Definition 84** *Let $\{a^i, i \geq 1\}$ be a sequence of vectors in $R^m$ such that each $a^i$ is a subgradient of $L^i$ at $\mu^i$, and is therefore a subgradient of $L$ and of every $L^k, k > i$, at $\mu^i$ as well. We will describe the associated face by $z \geq a^i\mu + b_i$. Define the sequence $\{LB_i, i > 1\}$ so that $LB_i$ is the value of the restricted LP $(P^{i-1})$, and note that this is equal to $L^{i-1}(\mu^i)$. Note also that this sequence is nondecreasing and is a lower bound on $L$.*

The observations that motivate the following results are that each face $z \geq a^i\mu + b_i$ of $\mathcal{L}^i$ and $\mathcal{L}$ is also a face of every $\mathcal{L}^k$, $k \geq i$, and thus it does not cut off the minimum point $(\mu^k, L^{k-1}(\mu^k)) = (\mu^k, LB_k)$ of any future restricted lagrangian ($k > i$). Moreover any face $z \geq a^k\mu + b_k$ of $\mathcal{L}^k$ and $\mathcal{L}$ cannot cut off any $(\mu^i, L(\mu^i))$, as this is a point on the surface of $\mathcal{L}$. So where $F^i$ is the hyperplane of the face associated with $a^i$, and $F^k$ is the hyperplane of the face associated with $a^k$, $k > i$, then the gap between the lower bound $LB_k$ and the upper bound $L(\mu^k)$, *plus* the vertical distance (w.r.t. the last coordinate) covered by sliding up $F^k$ from $\mu^k$ (where it intersects $\mathcal{L}$ at value $L(\mu^k)$) to $\mu^i$ (where its value in the last coordinate is $\leq L(\mu^i)$,) does not exceed the gap between $L(\mu^i)$ and the same lower bound $LB_k$, nor does it exceed the vertical distance covered by sliding down $F^i$ from $\mu^i$ (where the value in the last coordinate is $L(\mu^i)$) to $\mu^k$ (where its value does not exceed that of the lower bound $LB_k$). This would seem to indicate that either the gap is shrinking or the slope is shrinking, though this is not always strictly true, as we will see.

**Lemma 85** *Where $k > i$,*

1. $a^i\mu^i + b_i = L(\mu^i)$
2. $a^i\mu^k + b_i \leq LB_k$
3. $a^k\mu^i + b_k \leq L(\mu^i)$

**Proof:** The first statement follows from the fact the the point $(\mu^i, L(\mu^i))$ lies on the face $z \geq a^i\mu + b_i$. The second follows from the fact that $z \geq a^i\mu + b_i$ is a face of $\mathcal{L}^{k-1}$, and so the minimizer $\mu^k$ of $L^{k-1}$ with value $LB_k$ must not be cut off by this inequality. The third statement follows from the fact that $z \geq a^k\mu + b_k$ is a face of $\mathcal{L}$ and therefore cannot cut off $(\mu^i, L(\mu^i))$. ■

**Corollary 86** *Where $k > i$,*

$$a^i(\mu^i - \mu^k) \geq L(\mu^i) - LB_k \geq 0 \tag{58}$$

$$a^k(\mu^i - \mu^k) + (L(\mu^k) - LB_k) \leq L(\mu^i) - LB_k \tag{59}$$

$$a^k(\mu^i - \mu^k) + (L(\mu^k) - LB_k) \leq a^i(\mu^i - \mu^k) \tag{60}$$

**Proof:** The final inequality in (58) follows from the fact that $LB_k$ is a lower bound on $L$. Statement (60) follows directly from (59) and (58). ■

**Observation 87** *We could have obtained Lemma 85 even without the assumption in Definition 73 that the constraints $H^ix = h^i$ are progressively more relaxed. If we tighten the definition of the sequence $\{a^i\}$ in Definition 84 so that $a^i$, rather than being any arbitrary subgradient, is the violation vector of $\bar{x}^i$, then it would be sufficient if $H^ix = h^i$ is defined so that for all $j \leq i$, $H^i\bar{x}^j = h^i$. To maintain the monotonicity of $\{LB_i\}$ we need also add that for each $i > 0$, the solution $x^{i-1}$ to $(P^{i-1})$ also satisfies $H^ix^{i-1} = h^i$.*

**Proof:** Let $k > i$. $LB_k$ is the minimum of $L^{k-1}$. But $L^{k-1}(\mu^i) = L(\mu^i)$ since the maximizer $\bar{x}^i$ of $L(\mu^i)$ satisfies $H^kx = h^k$ even for the revised definition of $H^k$, and so the violation vector $a^i$ of $\bar{x}^i$ is a subgradient of $L^{k-1}$ and defines the same face $z \geq a^i\mu^i + b_i$ (since $b^i$ is fixed by the fact that the face contains the point $(\mu^i, L(\mu^i)))$, and so this face cannot cut off $(\mu^k, LB_k)$. ■

**Definition 88** *Given $k > i > 1$, define $\theta_i^{i,k}$ to be the angle between $\mu^i - \mu^k$ and $a^i$, and define $\theta_k^{i,k}$ to be the angle between $\mu^i - \mu^k$ and $a^k$ (see the diagram later on). Define*

$$\pi = \frac{1}{1 + \frac{\cos\theta_k^{i,k}}{\cos\theta_i^{i,k}}} \tag{61}$$

*and define*

$$\alpha = \pi \frac{\cos\theta_k^{i,k}}{\cos\theta_i^{i,k}} \tag{62}$$

*and note that $\pi + \alpha = 1$, so that*

$$\pi = 1 - \alpha. \tag{63}$$

**Definition 89** *Define*

$$G = \max_{x,y:Ax\le b, Ay\le b} c(x-y) \tag{64}$$

$$V = \max_{x:Ax\le b} ||d - Dx|| \tag{65}$$

$$Q = \max\{G, V\} \tag{66}$$

*and note that by last assumption of Definition 73, $Q$ is bounded by a linear function of $A$, $b$ and $c$.*

**Observation 90** *Let us assume that $L(\mu^i) > LB_k$, or else $LB_k$ would already be the optimal solution to $(P)$. By (58), $\cos\theta_i^{i,k} > 0$, i.e. $\theta_i^{i,k}$ is acute. Thus if $\cos\theta_k^{i,k} > 0$ as well (i.e. it is also acute), then $0 < \pi < 1$. If $\theta_k^{i,k}$ is indeed acute, then if it equals $\theta_i^{i,k}$ then pi $= .5$. If it is sharper then $\pi < .5$ and if it is duller then $\pi > .5$.*

**Observation 91** *The left hand side of both (59) and (60) is the vertical distance (i.e. the distance in the last coordinate) when one moves on the face hyperplane $z = a^k\mu + b_k$ from $\mu^k$ to $\mu^i$, plus the gap between the upper bound $L(\mu^k)$ and the lower bound $LB_k$. The right hand side of (59) is the gap between the upper bound $L(\mu^i)$ and the lower bound $LB_k$, and the right hand side of (60) is the vertical distance (i.e. the distance in the last coordinate) when one moves on the face hyperplane $z = a^i\mu + b_i$ from $\mu^k$ to $\mu^i$. If we refer to this vertical distance as the "slope" from $\mu^k$ to $\mu^i$, then this says that the new slope plus the new gap $\le$ the old gap, and the new slope plus the new gap $\le$ the old slope.*

**Theorem 92** *Where $k > i$, if $\cos\theta_k^{i,k} > 0$ then*

$$L(\mu^k) - LB_k \le L(\mu^i) - LB_k, \tag{67}$$

*and either*

$$L(\mu^k) - LB_k \le \pi(L(\mu^i) - LB_k) \tag{68}$$

*or*

$$||a^k|| \le \pi||a^i||. \tag{69}$$

*If additionally $\cos\theta_k^{i,k} \ge \cos\theta_i^{i,k}$, then $\pi \le .5$ as noted above, and we also have*

$$||a^k|| \le ||a^i||. \tag{70}$$

**Proof:** If $\cos\theta_k^{i,k} > 0$ then $a^k(\mu^i - \mu^k) \ge 0$ and so (67) follows from (59). Since we always have $L(\mu^k) \ge LB_k$, it follows from (60) that

$$||a^k|| \cos\theta_k^{i,k} \le ||a^i|| \cos\theta_i^{i,k} \tag{71}$$

and so if $\cos\theta_k^{i,k} \ge \cos\theta_i^{i,k}$ then (70) holds as well. Assume now that (68) does not hold, i.e.

$$L(\mu^k) - LB_k > \pi(L(\mu^i) - LB_k). \tag{72}$$

Then subtracting $L(\mu^k) - LB_k$ from the left side of (59) and $\pi(L(\mu^i) - LB_k)$ from the right side yields

$$a^k(\mu^i - \mu^k) \leq \alpha(L(\mu^i) - LB_k) \leq \alpha a^i(\mu^i - \mu^k) \Rightarrow \tag{73}$$

$$||a^k|| \cos\theta_k^{i,k} \leq \pi||a^i|| \cos\theta_k^{i,k} \Rightarrow \tag{74}$$

$$||a^k|| \leq \pi||a^i||. \blacksquare \tag{75}$$

**Corollary 93** *If $m = 1$, i.e. there is only one dualized constraint, then for any $r > 0$, and $t \geq 4r$ either the minimum gap*

$$\min_{j=1}^{t}\{L(\mu^j)\} - LB_t \leq G/2^r \tag{76}$$

*or the minimum norm of the violation vector*

$$\min_{j=1}^{t}\{||d - D\bar{x}^j||\} \leq V/2^r. \tag{77}$$

*Thus in the worst case, either $\{LB_i,\ i > 1\}$ converges to the value of $(P)$ pseudologorithmically, or $\{\min_{j=1}^{i}\{||d - D\bar{x}^j||,\ i \geq 1\}$ converges to zero pseudologorithmically. This result moreover continues to hold even if we do not assume that the constraints $H^i x = h^i$ become increasingly more relaxed as $i$ increases. Rather we need only assume that for each $i$, the solution $x^{i-1}$ to $(P^{i-1})$ and each of $\bar{x}^i$, $\bar{x}^{i-1}$ and $\bar{x}^{i-2}$ satisfy $H^i x = h^i$.*

**Proof:** If $m = 1$ then all $a^i$ and $\mu^i$ are scalars. Consider that for any $i, k,\ i < k$, if $a^i = 0$ or if $\mu^i = \mu^k$ then by (58), $L(\mu^i) - LB_k = 0$, so let us assume for all indices $i, k$ that we consider that neither of these situations ever occurs. Consider now two indices $i$ and $k$, $i < k$. As all $a$ and $\mu$ are scalar, the "angle" between any two quantities is always either 0 (if they have the same signum) or 180 (if they have opposite signum), and by (58) the "angle" $\theta_i^{i,k}$ is always 0. Thus if $a^k$ has the same signum as $a^i$ then $\theta_k^{i,k} = 0$ as well. It now follows that for any $4r$ consecutive terms $a^1, \ldots, a^{4r}$, there is a sequence of length at least $2r$ that have the same signum. Let us write the first $2r$ members of this sequence as $\{a^{p_1}, \ldots, a^{p_{2r}}\}$. Thus by Theorem 92, for each $j = 2, \ldots, 2r$, either

$$|a^{p_j}| \leq .5|a^{p_{j-1}}| \tag{78}$$

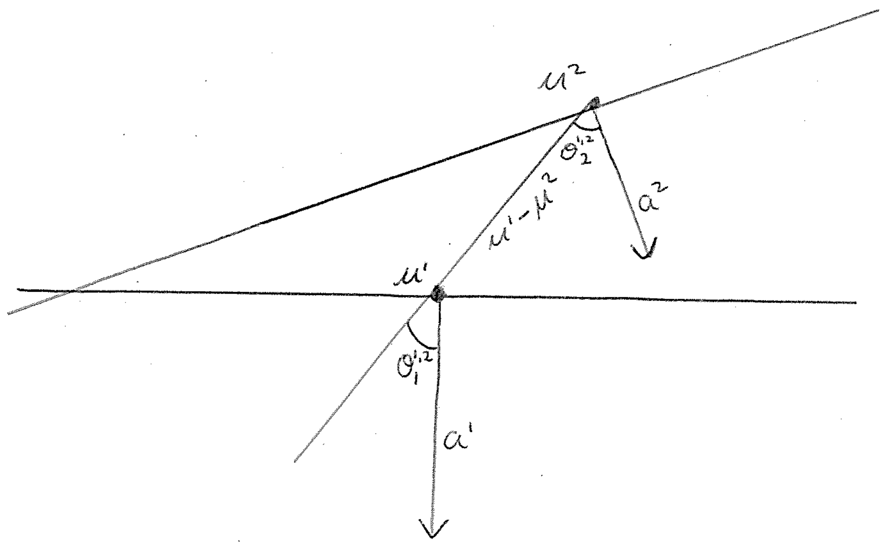
or

$$L(\mu^{p_j}) - LB_{p_j} \leq .5(L(\mu^{p_{j-1}}) - LB_{p_j}) \tag{79}$$

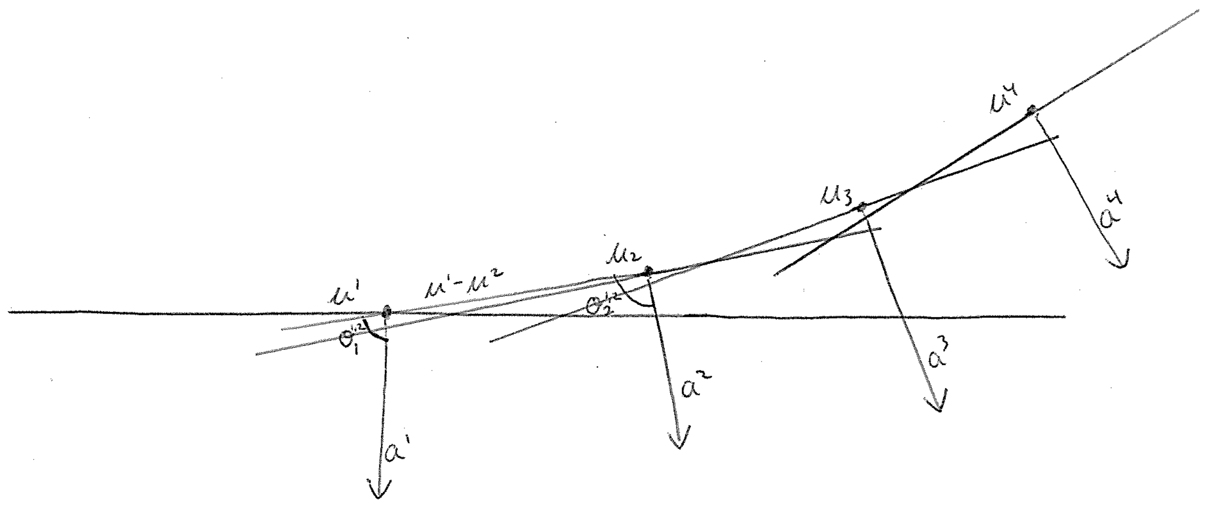
and thus by the time we reach the index $p_{2r} \leq 4r$ either the norm or the gap will have been halved at least $r$ times, proving the first statement of the corollary. The second statement follows easily in light of Observation 87. $\blacksquare$

If $m > 1$ then there is no obvious small worst-case bound for the convergence of $\{LB_i\}$. Expression (58) shows that if we draw a hyperplane in $R^k$ with normal $a^i$ passing through the point $\mu^i$, then each $\mu^k,\ k > i$, is on the other side of the hyperplane from the normal, as in the diagrams below. In this sense we can imagine the direction towards the future $\mu$ points as being opposite $a^i$. So we know that each future $\mu^k$ is "ahead" of $\mu^i$ with respect to $-a^i$. If we draw the hyperplane with normal $a^k$ passing through $\mu^k$, and $\mu^i$ is "behind" $\mu^k$ with respect to $-a^k$ as well (i.e. if $\mu^i$ is on the same side of the hyperplane as $a^k$) then $\theta_k^{i,k}$ is acute and Theorem 92 is applicable.

Good Case



Bad Case



There is no particular guarantee however that this will happen, even in two dimensions, as is indicated in the "bad case" in the diagram.

In practice it seems that the angles $\theta_i^{i,k}$ and $\theta_k^{i,k}$ are both often in the vicinity of 90 degrees, and that the latter are almost never as dull or duller than the former. Even in problems that converge quickly it is often the case that only a minority of the angles $\theta_k^{i,k}$ are acute, and in harder problems it may never happen. It is also interesting to note that in harder problems all of the angles tend to be extremely close to 90 degrees.

## References

[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.

[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).

[CEGMR12] R. Chicoisne, D. Espinoza, M. Goycoolea, E. Moreno and E. Rubio, A New Algorithm for the Open-Pit Mine Production Scheduling Problem, *Operations Research* **60** (2012), 517-528.

[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.

[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.

[EGMM12] D. Espinoza, M. Goycoolea, E. Moreno and G. Munoz, A study of the Bienstock-Zuckerberg algorithm for solving large-scale open-pit mining problems, *Integer Programming Workshop*, Valparaiso, March, 2012.

[EGMN12] D. Espinoza, M. Goycoolea, E. Moreno and A. Newman, MineLib: A Library of Open Pit Mining Problems, *Annals of Operations Research*, published online December 6th, 2012. Direct access to the MineLib repository http://mansci.uai.cl/minelib

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

[GT88] A. Goldberg and R. Tarjan, A new approach to the Maximum Flow Problem, *Journal of the Association for Computing Machinery* **35:4** (1988), 921–940.

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

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

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

[NRCWE10] A. Newman, E. Rubio, R. Caro, A. Weintraub, K. Eurek, A review of operations research in mine mlanning, *Interfaces* **40** *(2010), 222–245.*

*[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.*

*[PQ80] J.C. Picard, M. Queyranne, On the structure of all minimum cuts in a network and applications,* Mathematical Programming Study **13** *(1980), 8 – 16.*

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

*[S86] A. Schrijver, Theory of Linear and Integer Programming, Wiley (1986).*

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

| | **LargeProb** | **Mclaughlin** | **Zuck_small** | **Zuck_med** | **Zuck_large** | **KD** |
|---|---|---|---|---|---|---|
| **Blocks** | 87818 | 180749 | 9399 | 27387 | 96821 | 12154 |
| **Periods** | 100 | 20 | 20 | 15 | 30 | 12 |
| **Destinations** | 2 | 2 | 2 | 2 | 2 | 2 |
| **Variables** | 12236800 | 7229960 | 375960 | 821610 | 5809260 | 291696 |
| **Variables Cpx presol** | —** | —** | 3203979 | 17944902 | 36104201 | 2595370 |
| **Constraints** | 295558518 | 103788589 | 2921599 | 17367147 | 31690031 | 2313118 |
| **Constraints Cpx presol** | —** | —** | 282380 | 577755 | 4414170 | 282252 |
| **Problem arcs** | 307619482 | 110657031 | 3278721 | 18133953 | 37305589 | 2580494 |
| **Side constraints** | 200 | 20 | 40 | 30 | 60 | 12 |
| **Homogen side con** | 0 | 0 | 0 | 0 | 0 | 0 |
| **Pos dual side con at opt** | 134 | 12 | 14 | 11 | 23 | 9 |
| **Cplex sec** | —** | —** | — | — | —* | — |
| **Gurobi sec** | —** | —*** | — | —* | —*** | 43412 |
| **Algorithm Performance** | | | | | | |
| **Van Gap at term.** | -1.0E-12 | -1.2E-12 | 1.3E-13 | 2.4E-14 | -1.1E-12 | -1.3E-14 |
| **Van Lagran,Subprob sec** | 44415, 40 | 691, 1 | 18, 0 | 116, 0 | 649, 0 | 10, 0 |
| **Van Iters,Sec to 1e-5 opt** | 169, 42864 | 28, 773 | 28, 24 | 277, 133 | 35, 655 | 19, 11 |
| **Van Iters,Sec to opt** | 194, 47294 | 34, 942 | 31, 26 | 29, 142 | 42, 818 | 23, 14 |
| **Van Iters,Sec to term.** | 194, 47294 | 34, 942 | 31, 26 | 29, 142 | 42, 818 | 23, 14 |
| **VanPert Gap at term.** | -1.0E-12 | -1.2E-12 | -1.3E-13 | 2.4E-14 | -1.1E-12 | -1.3E-14 |
| **VanPert Lagran,Subprob sec** | 43366, 45 | 693, 1 | 19, 0 | 115, 0 | 654, 0 | 10, 0 |
| **VanPert Iters,Sec to 1e-5 opt** | 164, 41646 | 28, 774 | 28, 24 | 27, 132 | 35, 656 | 19, 11 |
| **VanPert Iters,Sec to opt** | 189, 46183 | 34, 944 | 31, 27 | 29, 142 | 42, 826 | 23, 14 |
| **VanPert Iters,Sec to term.** | 189, 46183 | 34, 944 | 31, 27 | 29, 142 | 42, 826 | 23, 14 |
| **Rich Gap at term.** | -5.4E-13 | -1.3E-12 | 1.3E-13 | 4.4E-14 | -1.1E-12 | -3.9E-14 |
| **Rich Lagran,Subprob sec** | 2764, 19 | 311, 1 | 6, 1 | 40, 0 | 272, 1 | 6, 0 |
| **Rich Iters,Sec to 1e-5 opt** | 9, 1341 | 10, 288 | 8,7 | 8, 34 | 9, 180 | 9, 6 |
| **Rich Iters,Sec to opt** | 18, 3098 | 15, 424 | 11, 10 | 11, 47 | 16, 333 | 12, 8 |
| **Rich Iters,Sec to term.** | 18, 3098 | 15, 424 | 11, 10 | 11, 47 | 16, 333 | 12, 8 |
| **RichPert Gap at term.** | -5.6E-13 | -1.3E-12 | 1.3E-13 | 4.4E-14 | -1.1E-12 | -3.9E-14 |
| **RichPert Lagran,Subprob sec** | 2670, 20 | 335, 1 | 6, 1 | 40, 0 | 276, 1 | 6, 0 |
| **RichPert Iters,Sec to 1e-5 opt** | 9, 1341 | 10, 282 | 8, 7 | 8, 34 | 9, 182 | 9, 6 |
| **RichPert Iters,Sec to opt** | 17, 2816 | 15, 428 | 10, 9 | 10, 43 | 15, 317 | 11, 7 |
| **RichPert Iters,Sec to term.** | 18, 3020 | 16, 456 | 11, 10 | 11, 47 | 16, 340 | 12, 8 |
| **Tuned Gap at term.** | -9.9E-13 | -1.3E-12 | 1.5E-13 | 4.4E-14 | -1.1E-12 | -3.9E-14 |
| **Tuned Lagran,Subprob sec** | 2968, 12 | 311, 0 | 7, 0 | 40, 0 | 289, 0 | 5, 0 |
| **Tuned Iters,Sec to 1e-5 opt** | 10, 1484 | 9, 257 | 8, 6 | 8, 34 | 9, 190 | 9, 5 |
| **Tuned Iters,Sec to opt** | 19, 3127 | 14, 400 | 11, 9 | 10, 42 | 15, 325 | 11, 7 |
| **Tuned Iters,Sec to term.** | 20, 3299 | 15, 427 | 12, 10 | 11, 47 | 16, 352 | 12, 8 |

Table 1: ***Sample problems 1***

| | **Newman1** | **SM2** | **Marvin** | **ManySC** | **Coal1** | **Coal2** | **W23** |
|---|---|---|---|---|---|---|---|
| **Blocks** | 1059 | 18388 | 8516 | 3165 | 34174 | 33773 | 74260 |
| **Periods** | 6 | 30 | 20 | 6 | 9 | 9 | 20 |
| **Destinations** | 2 | 2 | 2 | 4 | 15 | 17 | 4 |
| **Variables** | 12708 | 1103280 | 340640 | 24504 | 1683393 | 1705498 | 3564480 |
| **Variables Cpx presol** | 12552 | 894090 | 2064496 | 20710 | 1677451 | 1699498 | 3476640 |
| **Constraints** | 24603 | 545008 | 1726636 | 36830 | 289329 | 291391 | 9251776 |
| **Constraints Cpx presol** | 24603 | 545008 | 337860 | 36738 | 249664 | 250001 | 9251776 |
| **Problem arcs** | 35181 | 1611452 | 2050204 | 48416 | 1977327 | 2001301 | 12667652 |
| **Side constraints** | 12 | 60 | 40 | 6832 | 3092 | 3573 | 84 |
| **Homogen side con** | 0 | 0 | 0 | 12 | 936 | 1278 | 48 |
| **Pos dual side con at opt** | 3 | 36 | 13 | 124 | 463 | 609 | 15 |
| **Gurobi sec** | 4 | 589 | — | 12 | 3580 | 3061 | —* |
| **Cplex sec** | 4 | 681 | — | 21 | 1460 | 1214 | — |
| **Algorithm Performance** | | | | | | | |
| **Van Gap at term.** | -6.4E-15 | 4.6E-14 | 2.3E-15 | -6.7E-15 | .07 | .84 | 3.4E-13 |
| **Van Lagran,Subprob sec** | 0, 0 | 27, 0 | 13, 0 | 1, 4 | 235, 47816 | 7, 49859 | 165, 25 |
| **Van Iters,Sec to 1e-5 opt** | 9, 0 | 34, 47 | 27, 18 | 41, 20 | — | — | 43, 234 |
| **Van Iters,Sec to opt** | 11, 0 | 39, 52 | 30, 20 | 44, 22 | — | — | 46, 250 |
| **Van Iters,Sec to term.** | 11, 0 | 39, 52 | 30, 20 | 44, 22 | 1582, 50071 | 50, 50116 | 46, 250 |
| **VanPert Gap at term.** | -6.4E-15 | 4.6E-14 | 2.4E-15 | -5.4E-15 | 2.6E-6 | 3.7E-8 | 3.0E-13 |
| **VanPert Lagran,Subprob sec** | 0, 0 | 27, 1 | 13, 0 | 1, 5 | 25, 6408 | 27, 20654 | 160, 23 |
| **VanPert Iters,Sec to 1e-5 opt** | 9, 0 | 34, 47 | 27, 18 | 42, 21 | 124, 6379 | 156, 21136 | 43, 228 |
| **VanPert Iters,Sec to opt** | 10, 0 | 38, 51 | 29, 19 | 43, 22 | 129, 6394 | 160, 21141 | 45, 239 |
| **VanPert Iters,Sec to term.** | 11, 0 | 39, 53 | 30, 20 | 44, 23 | 131, 6408 | 161, 21142 | 46, 244 |
| **VanGT Gap at term.** | N/A | N/A | N/A | -5.4E-15 | 7.8E-7 | 1.3E-8 | N/A |
| **VanGT Lagran,Subprob sec** | N/A | N/A | N/A | 1, 5 | 22, 2674 | 26, 11644 | N/A |
| **VanGT Iters,Sec to 1e-5 opt** | N/A | N/A | N/A | 42, 21 | 109, 2875 | 144, 11997 | N/A |
| **VanGT Iters,Sec to opt** | N/A | N/A | N/A | 43, 22 | 117, 2896 | 144, 11997 | N/A |
| **VanGT Iters,Sec to term.** | N/A | N/A | N/A | 44, 23 | 120, 2912 | 151, 12057 | N/A |
| **Rich Gap at term.** | 1.4E-15 | 2.8E-14 | 4.3E-15 | -6.9E-15 | .997 | -5.3E-11 | 3.8E-13 |
| **Rich Lagran,Subprob sec** | 0, 0 | 9, 1 | 4, 0 | 0, 2 | 1, 87354 | 63, 34550 | 65, 9 |
| **Rich Iters,Sec to 1e-5 opt** | 6, 0 | 10, 14 | 8, 5 | 19, 19 | — | 193, 32447 | 14, 75 |
| **Rich Iters,Sec to opt** | 8, 0 | 17, 21 | 10, 6 | 22, 22 | — | 327, 38377 | 18, 96 |
| **Rich Iters,Sec to term.** | 8, 0 | 17, 21 | 10, 6 | 22, 22 | 11, 87391 | 327, 38377 | 18, 96 |
| **RichPert Gap at term.** | 1.4E-15 | 1.5E-14 | 3.7E-15 | -7.7E-15 | 3.7E-7 | 2.0E-9 | 3.9E-13 |
| **RichPert Lagran,Subprob sec** | 0, 0 | 9, 1 | 4, 0 | 0, 2 | 6, 23408 | 5, 12133 | 65, 13 |
| **RichPert Iters,Sec to 1e-5 opt** | 6, 0 | 10, 14 | 8, 45 | 18, 15 | 32, 23541 | 26, 12416 | 14, 79 |
| **RichPert Iters,Sec to opt** | 7, 0 | 16, 20 | 10, 6 | 18, 15 | 32, 23541 | 26, 12416 | 17, 95 |
| **RichPert Iters,Sec to term.** | 8, 0 | 17, 22 | 10, 6 | 19, 16 | 34, 23607 | 27, 12420 | 18, 101 |
| **RichGT Gap at term.** | N/A | N/A | N/A | -6.3E-15 | 9.6E-7 | 1.3E-8 | N/A |
| **RichGT Lagran,Subprob sec** | N/A | N/A | N/A | 0, 2 | 9, 2221 | 7, 1218 | N/A |
| **RichGT Iters,Sec to 1e-5 opt** | N/A | N/A | N/A | 18, 13 | 40, 2154 | 32, 1370 | N/A |
| **RichGT Iters,Sec to opt** | N/A | N/A | N/A | 20, 15 | 44, 2228 | 35, 1377 | N/A |
| **RichGT Iters,Sec to term.** | N/A | N/A | N/A | 21, 16 | 49, 2373 | 38, 1387 | N/A |
| **Tuned Gap at term.** | 1.4E-15 | 2.5E-14 | 1.3E-14 | -7.9E-15 | 3.6E-7 | 1.3E-8 | 4.04E-13 |
| **Tuned Lagran,Subprob sec** | 0, 0 | 8, 1 | 4, 0 | 0, 2 | 11, 1702 | 9, 485 | 71, 5 |
| **Tuned Iters,Sec to 1e-5 opt** | 6, 0 | 10, 12 | 8, 5 | 16, 12 | 50, 1367 | 42, 586 | 15, 79 |
| **Tuned Iters,Sec to opt** | 7, 0 | 13, 16 | 9, 5 | 19, 15 | 54, 1485 | 47, 597 | 18, 94 |
| **Tuned Iters,Sec to term.** | 8, 0 | 14, 17 | 10, 6 | 20, 16 | 59, 1835 | 50, 612 | 19, 99 |

Table 2: ***Sample problems 2***