A New LP Algorithm for Precedence Constrained Production Scheduling

Daniel Bienstock* Mark Zuckerberg†

August, 2009
Version Tues Aug 18 09:41:12 AEST 2009

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 we will make two further complicating assumptions:

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 mineralized earth from an open pit mine. The principal structural constraint associated with open

*Department of Industrial Engineering and Operations Research, Columbia University
†Resource and Business Optimization Group Function, BHP Billiton Ltd.

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 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.

To obtain a reasonably accurate model of the slope constraint however, it is necessary that the scheduling units 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)[1]

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, 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, the linear programming problem itself is well beyond the reach of commercial linear programming solvers.

1.1.3 Traditional Approaches in Mine Scheduling

The traditional approach to this problem 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 further assumes that any block of ore is defined by a single value v, and that the objective 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 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. This problem can be solved efficiently yielding 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 λ times the capacity consumption of the unit (for some chosen capacity), and then to increase λ 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.

This is obviously a very rough approach, 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

[1]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.

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] and [F06] for an overview), but there is little that comes with a proof of optimality. Caccetta and Hill ([CH03]) have published a paper describing a branch and cut based algorithm designed for large scale problems, but due to commercial confidentiality considerations they have not released details of their method.

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. 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 are of this form is known as a "maximum weight 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), whose solution is in turn used to update the lagrangian relaxation. The algorithm will be proved to converge, solving both the lagrangian and the original LP.

In practice convergence is fast – 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. We will then describe a subclass of this problem which we refer to as the "Parcel Assignment Problem". In this subclass 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.

In mine planning terms the interpretation of the parcel assignment problem is as follows. We aggregate the scheduling units into a small number of very large units so that a single extraction decision is made in each period for each very large aggregated unit of earth, but we still make separate processing decisions for each of the original units that are components of these large aggregated units. Thus there are a small number of extraction decisions (and a small number of precedence constraints) but there are still just as many processing decisions.

In Section 3 we show that the solution to the parcel assignment problem, despite having potentially millions of variables, can be derived from the solution to a much smaller LP *had we known* the values of a small set of optimal duals. These duals are unknown, but we use this observation to derive a "guessing algorithm", which when applied to the simpler problem described in [BDFG09] is broadly similar to their algorithm. The algorithm guesses these duals in order to generate this smaller LP, and then uses dual information from that smaller LP's optimal solution to update its

guess. We then prove the convergence of one version of this algorithm and prove a theorem that indicates that its convergence is likely to be fast.

In Section 4 we return to the original precedence constrained production scheduling problem. In Section 4.1 we show that precedence constrained production scheduling can be reformulated as a min cut problem with the same number of side constraints.

Then in Section 4.2 we prove a theorem that is crucial to the application of the algorithm defined with respect to the parcel assignment problem to the precedence constrained production scheduling algorithm. This theorem states 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. This acts as an analogue of the parcel assignment problem result stating that the knowledge of a small number of optimal dual values can be used to generate the full parcel assignment problem solution, and it allows us to apply a variation of the same algorithm to the full precedence constrained production scheduling problem.

We will 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, thus opening up the possibility that a similar algorithm can be developed for min cost network flow problems plus side constraints.

In Section 4.3 we motivate, define and prove convergence of the algorithm, and in Section 4.4 we prove several theorems that show a deep relationship between LP solutions and the lagrangian solutions in order to shed further light on the algorithm's convergence properties. Finally, in section 5 we present the results of computational experiments.

2 Definitions and Preliminaries

2.1 The Precedence Constrained Production Scheduling Problem

Definition 1 *The Precedence Constrained Production Scheduling Problem 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 ($(i,j) \in \mathcal{A}$ means that job j can be performed no later than job i).

Given R processing options for each job.

Given T scheduling periods.

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

Let $x_{j,t,d} \in [0,1]$ represent the proportion of job j performed in period t, and processed according to processing option, or "destination", d

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

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

maximize $c^T x$ subject to:

$$\sum_{\tau=1}^{t} y_{i,\tau} \leq \sum_{\tau=1}^{t} y_{j,\tau}, \ \forall (i,j) \in \mathcal{A}, \ t = 1, \ldots, T \tag{3}$$

$$Dx \leq d \tag{4}$$

$$y_{j,t} = \sum_{d=1}^{R} x_{j,t,d}, \ \forall j \in \mathcal{N}, \ t = 1, \ldots, T \tag{5}$$

$$\sum_{t=1}^{T} y_{j,t} \leq 1, \ \forall j \in \mathcal{N} \tag{6}$$

$$x \geq 0$$

2.2 The Parcel Assignment Problem (PAP)

As mentioned, 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, but the processing decisions are separate for each parcel. Allowing for parcels yields the following generalization of Definition 1.

Definition 2 *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 $Dx \leq d$ 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:

$$\sum_{\tau=1}^{t} y_{i,\tau} \leq \sum_{\tau=1}^{t} y_{j,\tau}, \ \forall (i,j) \in \mathcal{A}, \ t = 1, \ldots, T \tag{7}$$

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

$$y_{J,t} = \sum_{d=1}^{R} x_{j,t,d}, \ \forall j \in P(J), \ J \in \mathcal{N}, \ t = 1, \ldots, T \tag{9}$$

$$\sum_{t=1}^{T} y_{j,t} \leq 1, \ \forall j \in \mathcal{N} \tag{10}$$

$$x \geq 0$$

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 or 3.
- Typical number of side constraints : 20 – 200.
- Typical number of blocks : $50,000$ – 5 million.
- Typical number of parcels : 1 million – 10 million.
- Typical number of precedences : 1 – 100 million.

These numbers indicate that the number of constraints of the form (7), (9) and (10) can be expected to be very large. Let us consider however a variety of problem for which the number of parcels remains large but the number of *jobs* is small (so that only the number of constraints (9) 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 9).

Definition 3 *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.

3 An Algorithm for PAP

3.1 Cutoffs

Definition 4 *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. We will describe an algorithm that when applied to their problem works quite similarly, but which is defined in quite a bit more generality. We will then show that this more general algorithm can be adapted to the full precedence constrained production scheduling problem.

Definition 5 *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*

$$v_{j,t} = c_{j,t,0} - c_{j,t,1} \tag{11}$$

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 6 *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*

$$v_{j',t}/w_{j',t} > v_{j,t}/w_{j,t}, \tag{12}$$

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 7 *The optimal dual* μ_t *corresponding to the capacity constraint in period* t *is a cutoff* V_t*, where "cutoffs" are as defined in Lemma 6.*

Proof: Let μ_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

$$c_{j,t,0} = \mu_t w_{j,t,0} + \delta_{j,t} - \pi_{j,t,0} \tag{13}$$

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

$$c_{j,t,1} = \delta_{j,t} - \pi_{j,t,1}. \tag{14}$$

Subtracting the latter constraint from the former constraint yields:

$$v_{j,t} = \mu_t w_{j,t,0} - \pi_{j,t,0} + \pi_{j,t,1} \tag{15}$$

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

$$x_{j,t,0} > 0 \Rightarrow v_{j,t}/w_{j,t,0} \geq \mu_t \tag{16}$$

$$x_{j,t,1} > 0 \Rightarrow v_{j,t}/w_{j,t,0} \leq \mu_t \ \square \tag{17}$$

3.2 Generalized Cutoffs

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

Theorem 8 *Given a general parcel assignment problem, let μ be the optimal dual values corresponding to the side constraints, and given a parcel j and 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*

$$c_{j,t,d} - \mu^T D^{j,t,d} \tag{18}$$

is maximized. The 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 μ be the optimal dual vector associated with the side constraints and let δ and π be as in the proof of Theorem 7. Then for each variable $x_{j,t,d}$ the dual constraint is

$$c_{j,t,d} = \mu^T D^{j,t,d} + \delta_{j,t} - \pi_{j,t,d}. \tag{19}$$

By the nonnegativity of π observe that for all destinations d

$$c_{j,t,d} - \mu^T D^{j,t,d} \leq \delta_{j,t} \tag{20}$$

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

$$c_{j,t,d} - \mu^T D^{j,t,d} = \delta_{j,t}. \ \square \tag{21}$$

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 9 *Given the optimal side-constraints duals μ, and a variable $x_{j,t,d}$, with parcel j in job J, if d does not maximize (18) then $x_{j,t,d}$ can be replaced with 0, and if d uniquely maximizes (18) then it can be replaced with $y_{J,t}$. Thus where the number of "ties" 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.*

3.3 Guessing Algorithms

Idea 10 *What if we "guess" the optimal duals? This is tantamount to guessing cutoffs, and it yields a small LP. While the resulting problem may be inaccurate, maybe* its *optimal duals will be a better guess than we started with.*

Definition 11 *Given a vector μ with an entry for each side constraint, we shall say that μ induces the "cutoff rule" that for each parcel j and period t, any destination d that does not maximize (18) satisfies $x_{j,t,d} = 0$. We will also use the term "applying the cutoff μ" to mean applying the cutoff rule induced by μ.*

Proto-Algorithm

Given a PAP, denoted P,

1. Let $k = 1, \mu^0 = 0$.
2. Define the reduced LP $P2(\mu^{k-1})$ by replacing variables as per Observation 9 and μ^{k-1}.
3. Solve $P2(\mu^{k-1})$. Define x^k to be the optimal solution, and set μ^k to be the optimal dual vector corresponding to the side constraints.

4. If $k > 1$ and $\mu^k = \mu^{k-1}$ then STOP.

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

Note that by adding error terms with large penalties to the side constraints in the original problem, all reduced problems $P2(\mu^{k-1})$ will be feasible.

Lemma 12 *Let P be a parcel assignment problem, and let $L(P, \mu)$ be the lagrangian relaxation of P with the side constraints dualized with penalties μ. Every optimal solution to $L(P, \mu)$ satisfies the cutoff rule (18) induced by μ.*

Proof: Given a parcel j in job J and a period t, for each destination d the variable $x_{j,t,d}$ only appears in $L(P, \mu)$ in the constraint $\sum_{d=1}^{r} x_{j,t,d} = y_{J,t}$, and in the nonnegativity constraint $x_{j,t,d} \geq 0$, and its objective value is

$$c_{j,t,d} - \mu^T D^{j,t,d}. \tag{22}$$

It is therefore always optimal to choose $x_{j,t,d'} = y_{J,t}$ for some d' that maximizes (22), and it is always suboptimal to choose $x_{j,t,d'} > 0$ if d' does not maximize (22). □

The lemma says that when lagrangian relaxation penalties are chosen equal to μ, then in any optimal solution to the lagrangian relaxed problem the destination chosen for any parcel is one that satisfies the cutoff rule induced by μ.

Consider now the restriction $P2(\mu)$ of the original problem in which we require the cutoff rule induced by μ to be obeyed. Obviously this could only reduce the value of the lagrangian relaxation, but since even without the restriction the cutoff rule is obeyed by the lagrangian with penalties μ, it follows that any optimal solution to $L(P, \mu)$ is also optimal for $L(P2(\mu), \mu)$ and conversely.

Observe now that the optimal solution value of the restricted LP in the k^{th} algorithm iteration, $P2(\mu^{k-1})$, is equal to the minimum of its lagrangian $L(P2(\mu^{k-1}), \mu)$ taken over all penalties μ, which is $L(P2(\mu^{k-1}), \mu^k)$. So if the algorithm reached a stopping condition, i.e. if $\mu^k = \mu^{k-1}$, then where we write $z(\cdot)$ to refer to the optimal value of a problem, then

$$cx^k = z(P2(\mu^{k-1})) = z(L(P2(\mu^{k-1}), \mu^{k-1})) = z(L(P, \mu^{k-1})) \geq z(P) \tag{23}$$

which proves that x^k is optimal (the rightmost inequality follows from lagrangian duality). This proves the following lemma.

Lemma 13 *If the algorithm stops at some iteration k then x^{k-1} is an optimal solution to the original problem P.*

In practice we observed the following algorithm behavior:

- Extremely fast convergence – 4 to 5 iterations to a solution within .01% of optimality.
- Afterwards the algorithm often bounced rather than converged.
- Nevertheless we found that the algorithm *always provided an improving direction* $\mu^{k+1} - \mu^k$ for the lagrangian of the problem.

We will now turn to explain the causes of this behavior.

The next lemma states that the value of the lagrangian of the original problem P and of the lagrangian of the restricted problem $P2(\mu)$ do not only match at μ, *they match everywhere inside of a ball around* μ. This follows from the fact that given parcel j, period t and destination d, if d does not maximize $c_{j,t,d} - \mu^T D^{j,t,d}$, then neither will it maximize $c_{j,t,d} - \nu^T D^{j,t,d}$ for any ν within a small enough (but positive) radius of μ. Thus the restriction on $P2(\mu)$ enforcing the cutoff rule induced by μ will not cut off any optimal solutions of the lagrangian of the original problem P at any penalty vector ν close to μ. Formally,

Lemma 14 *Let $P2(\mu)$ be the restricted LP defined by applying cutoffs μ. There exists $\epsilon > 0$ such that for all $\nu : \|\nu - \mu\| < \epsilon$, every optimal solution to $L(P2(\mu), \nu)$ is also an optimal solution to $L(P, \nu)$ and conversely.*

It now follows from the convexity of the lagrangian that the direction from μ to the optimal lagrangian penalties μ^* for the restricted problem $L(P2(\mu))$, must also define an improving direction for the lagrangian of the unrestricted problem $L(P)$, since $L(P)$ and $L(P2(\mu))$ match inside of a positive radius ball around μ.

We state this in greater generality as the following theorem:

Theorem 15 *Consider an arbitrary linear program $P1$ defined by:*

$$(P1) \quad max\ v^T y$$

subject to

$$Ay \leq b \tag{24}$$

$$Dy \leq d \tag{25}$$

and let $L(P1, \mu)$ be the lagrangian relaxation in which the constraints (25) are dualized with penalties μ.

Assume that an optimal solution $y(P1, \mu)$ to $L(P1, \mu)$ also satisfies additional constraints $Hy = h$ and define a restricted LP

$$(P2) \quad max\ v^T y$$

subject to

$$Ay \leq b \tag{26}$$

$$Dy \leq d \tag{27}$$

$$Hy = h \tag{28}$$

and note that $L(P1, \mu) = L(P2, \mu)$. Assume now that for some $\epsilon > 0$ and all $\nu : \|\mu - \nu\| < \epsilon$ it remains the case than an optimal solution $y(P1, \nu)$ satisfies (28). Let μ^* be the optimal penalties for the lagrangian $L(P2)$, then either μ^* optimizes $L(P1)$ or $\mu^* - \mu$ defines a strictly improving direction for $L(P1)$.

Theorem 15 explains why the proto-algorithm always gave an improving direction.

It also gives insight into why it got close to optimality fast, but once close to optimality tended to bounce:

Observation 16

- *For the parcel assignment problem there are only a finite number of cutoff rules $Hx = h$, one of which holds for every lagrangian penalty vector μ.*
- *Each such cutoff rule has an "ϵ validity ball" of positive size.*
- *While ϵ may be small, the rule may remain "mostly valid" over a fairly large region.*

Thus when far from optimality, the algorithm gives excellent direction, as the lagrangian behavior of the restricted LP's that it solves is a reasonable approximation to the lagrangian behavior of the original problem. When close however it lacks the precision to return the actual optimum, though the iterates μ^k still define improving directions for the lagrangian.

3.4 Generalizing the Proto-Algorithm

3.4.1 A General Template

Theorem 15 suggests the following generalization of the proto-algorithm.

Given an arbitrary LP:

$$(P1) \quad \max\ c^T x \text{ subject to}$$

$$Ax \leq b \tag{29}$$

$$Dx \leq d \tag{30}$$

- Assume that $\{x : Ax \leq b\} \neq \emptyset$.
- Let $L(P1, \mu)$ be the lagrangian relaxation in which constraints (30) are dualized with penalties μ.

Algorithm Template:

1. Set $\mu^0 = 0$ and set $k = 1$.
2. Solve $L(P1, \mu^{k-1})$ to obtain optimal solution w^k. If $k > 1$ and w^k satisfies $H^{k-1}(x) = h^{k-1}$ then STOP.
3. Find some constraint $H^k(x) = h^k$ that is satisfied by w^k.
4. Define the restricted problem

$$(P2(k)) \quad \max c^T x \text{ subject to}$$

$$Ax \leq b \tag{31}$$

$$Dx \leq d \tag{32}$$

$$H^k(x) = h^k \tag{33}$$

5. Solve $P2(k)$ to get optimal primal x^k with value z^k and optimal dual μ^k (corresponding to constraints (32)). If $\mu^k = \mu^{k-1}$ STOP.
6. Set k=k+1 and GOTO step 2.

Note: We will assume that the system of constraints defined by (31) is feasible (so that the system defined by (31) and (33) is also). For convenience we will also assume that adding in constraints (32) maintains feasibility, though if this fails we could just add in slacks into the D constraints with large penalties in the original formulation.

3.4.2 Stopping Conditions

Theorem 17 *If the algorithm reaches a stopping condition then its current iterate x^k is optimal.*

Proof: If $H^{k-1}(w^k) = h^{k-1}$, then w^k is a feasible solution to the lagrangian problem $L(P2(k-1), \mu^{k-1})$ with solution value z. Now letting z^* be the optimal solution value for $P1$, and z^k be the optimal solution value for $P2(k-1)$, it follows from lagrangian duality that

$$z^* \leq z \leq z^k \leq z^*. \tag{34}$$

As for the other stopping condition, $\mu^k = \mu^{k-1}$ implies that w^k optimally solves $L(P1, \mu^k)$, so that we could choose $w^{k+1} = w^k$ and so $H^k(w^{k+1}) = h^k$, which is the first stopping condition. □

Observation 18 *At each iteration we are choosing constraints that hold for the solution to $L(P1, \mu^{k-1})$ and imposing them on the LP. If the algorithm reaches a stopping condition, then at the final iteration the constraints imposed were valid both for the optimal lagrangian solution and for the optimal LP solution.*

So the algorithm essentially is searching for a set of constraints to simplify the LP, *induced by a lagrangian solution*, that will hold both for the optimal lagrangian and the optimal LP.

Observation 19 (Heuristic Observation) *If the constraints $H^k(x) = h^k$ induced by the lagrangian solution $L(P1, \mu^{k-1})$ describe the optimal LP solution with increasing accuracy as μ^{k-1} gets closer to the optimal μ, then we can expect the algorithm to do very well, as we will see a "virtuous cycle" effect.*

Broadly speaking the heuristic observation generally holds for PAP and we do see a virtuous cycle effect when far from optimality, but when close to optimality the duals provided by the algorithm seem to overstep and the algorithm begins to bounce.

3.5 Making the Algorithm Converge

3.5.1 Building a Lagrangian Net

Observe that by construction, the restricted problem's lagrangian $L(P2(k))$ (seen as a function of the penalties μ) matches the original problem's lagrangian $L(P1)$ at μ^{k-1} and more generally at every μ for which there is an optimal solution w to $L(P1, \mu)$ that satisfies $H^k(x) = h^k$. If it would match at μ^k also (which is the global minimizer of $L(P2(k))$), i.e. if there would be an optimal solution to $L(P1, \mu^k)$ that satisfies $H^k(x) = h^k$, then we would reach a stopping condition and would conclude that x^k is optimal.

But even if it does not match we can still modify those constraints that defined $P2(k)$ so that it will, i.e. we can "*correct*" the current iterate's constraints $H^k(x) = h^k$ by *loosening them* to form constraints $H^{k+1}(x) = h^{k+1}$ for which a solution to $L(P1, \mu^k)$ indeed satisfies $H^{k+1}(x) = h^{k+1}$.

Naturally in modifying H, we have also modified $P2$, and μ^k can no longer be assumed to be its global optimizer, but we have still accomplished that for the new restricted problem $P2(k+1)$, the lagrangian matches the unrestricted lagrangian both at $\mu^k - 1$ and at μ^k. If we do this repeatedly then as k increases, $L(P2(k))$ will match $L(P1)$ at an increasingly large net of points $\{\mu^j\}$.

So at every iteration either we attain optimality or this net gets strictly tighter and the constraints $H^k(x) = h^k$ get strictly looser, and thus the algorithm cannot bounce.

3.5.2 Releasing the Net

But the constraints H^k may not only be too strong, they may also be unnecessarily weak, as we only want constraints that are valid at the optimal lagrangian and not necessarily at every lagrangian. We don't necessarily want constraints that are valid at every lagrangian solution, just at the "good ones", and thus we would want to throw away the "bad points" from the net periodically.

One way to interpret this is as follows. We can think of the idea of solving $L(P2(k))$ to be that even if $L(P2)$ does not match $L(P1)$ exactly, maybe the μ^k that optimizes $L(P2(k))$ will still constitute an improvement for $L(P1)$. Thus once we reach an iterate μ^k that improves $L(P1)$ we can ignore the old constraints $H^{k-1}(x) = h^{k-1}$, throw away the net, and start again.

Note moreover that even if we fail to ensure that the constraints $H^k(x) = h^k$ are weaker than the constraints $H^{k-1}(x) = h^{k-1}$, so long as we ensure that they are weak enough so that x^{k-1} will satisfy $H^k(x) = h^k$, we will be assured that the LP solution values z^k will be nondecreasing.

Observation 20 *If constraints $H^k(x) = h^k$ are always chosen so that x^{k-1} satisfies $H^k(x) = h^k$, then the solution values z^k are monotone nondecreasing. In particular this will hold if the constraints $H^k(x) = h^k$ are weaker than the constraints $H^{k-1}(x) = h^{k-1}$.*

3.6 The Parcel Assignment Problem: Ensuring convergence

3.6.1 Cutoffs and Reduced Costs

The following theorem gives insight into the performance of the proto-algorithm on the parcel assignment problem, and suggests a way to make it converge along the lines of Observation 20:

First define the expression

$$E(\mu, j, t, d) = c_{j,t,d} - \mu^T D^{j,t,d} \tag{35}$$

Observation 21 *The restricted General Parcel Assignment Problem with cutoff rule induced by μ can be written as follows:*

$$max\ c^T x \quad subject\ to$$

$$\sum_{\tau=1}^{t} y_{J,\tau} \leq \sum_{\tau=1}^{t} y_{J',\tau},\ \forall (J, J') \in \mathcal{A},\ t = 1, \ldots, T \tag{36}$$

$$Dx \leq d \tag{37}$$

$$y_{J,t} = \sum_{d=1}^{R} x_{j,t,d},\ \forall j \in J,\ \forall J \in \mathcal{N},\ t = 1, \ldots, T \tag{38}$$

$$\sum_{t=1}^{T} y_{J,t} \leq 1, \ \forall J \in \mathcal{N} \tag{39}$$

$$x \geq 0 \tag{40}$$

$$x_{j,t,d} = 0, \ \forall j,t,d \ s.t. \ d \ does \ not \ maximize \ E(\mu, j, t, d) \tag{41}$$

- Let x^* be an optimal primal solution to the restricted problem.
- Let (μ^*, δ^*, π^*) be an optimal dual vector with μ^* corresponding to constraints (37), δ^* corresponding to (38) and π^* corresponding to (40).
- Note that constraints (41) imply that those x variables that are restricted to zero can have negative duals π^*.
- Note that π^* are reduced costs for the unrestricted problem, and if $\pi^* \geq 0$ then x^* is optimal for the original problem as well.

Theorem 22 *For each j,t, the destination d that minimizes $\pi^*_{j,t,d}$ maximizes $E(\mu^*, j, t, d)$.*

Proof:

$$E(\mu^*, j, t, d) + \pi^*_{j,t,d} = \delta^*_{j,t}. \ \square \tag{42}$$

We conclude that changing the cutoff rule to μ^* from μ, which means that the new destination d chosen for each j and t will be that which maximizes $E(\mu^*, j, t, d)$, will lead to the best possible solution in which the new destination chosen for each j,t is that which has the most negative reduced cost in the current solution.

Thus the improvement can be very steep, but it can also overshoot its target. Evidently we need a strategy that *allows* us to increase the values of the variables that were forced to zero by cutoffs μ and which would be forced to their upper bound by cutoffs μ^* (in line with Theorem 22), but which does not *require* us to force all of them uniformly to their upper bound (and similarly for variables that we wish to decrease). We wish to learn from μ^* what changes may be necessary, but we should try to conserve any information we might have learned from μ, or else we are at risk for bouncing.

3.6.2 Partitions Rather than Cutoffs

Idea 23 *Instead of enforcing a cutoff at each iteration, and changing the cutoff from iteration to iteration, let us imagine that a cutoff μ suggests for each job J and each period t a* partition *of the variables $x_{j,t,d}, \ j \in J$ associated with each of the parcels into those that the cutoff suggests are at their upper bound (those corresponding to a uniquely selected destination), those that the cutoff suggests are at their lower bounds (those corresponding to unselected destinations) and those that are tied. Aggregate all variables in each of these three parts of the partition into a single variable.*[2]

3.6.3 Refining the Partition

Applying Observation 20 to Theorem 22 suggests that rather than change the cutoff, we *refine the current partition* by intersecting each part in the partition with each part in the partition suggested by using the optimal duals μ^* as cutoffs. In this way the cutoff rule μ^* is not applied strictly, but merely indicates where extra flexibility is needed to separate variables from a set whose values should be allowed to be different from that of the other variables in the set as per cutoff rule μ^*.

This step corresponds to the "building of the lagrangian net" in Section 3.5.1.

[2]We will not discuss ties extensively, but tied variables can be aggregated to a single variable as described. Alternatively if there are not many of them they can be left unaggregated, or they can be aggregated arbitrarily into one of the other two sets.

3.6.4 Aggregating the Partition

Observation 24 *Given a partition, the aggregated problem is itself a parcel assignment problem and its solution therefore itself satisfies some cutoff rule as per Theorem 8, i.e. the solution will lump together all parts of the partition into three categories: those at their upper bound, those at their lower bound, and those which are tied.*

This new cutoff yields a "coarsification" of the current partition - i.e. it suggests a way to aggregate together parts in the current partition while maintaining the feasibility of the current solution, as the extra flexibility afforded by the finer partition was not used by the current solution.

This step corresponds to the "releasing of the lagrangian net" in Section 3.5.2.

3.6.5 Conclusions

Theorem 25 *Applying the algorithm template to the parcel assignment problem we conclude that:*

Choosing "partitioning constraints" as our constraints $H(x) = h$, yields an LP which for each job and period has as many variables as there are parts in the associated partition.

- *After each iteration the current partition can be coarsified (if desired) as per the cutoffs that hold at the current solution, reducing the partition size back to 2 for each job and period (if ties are aggregated arbitrarily).*
- *The solution values are monotonically nondecreasing.*
- *If the coarsification step is not performed then the partitioning constraints at each iteration are weaker than those used in the previous iteration and therefore sub-optimal partitions will never be repeated.*
- *There are finitely many partitions and therefore if the coarsification step is only performed at an iteration in which the objective value strictly increased then the algorithm must converge.*

4 An Algorithm for General Precedence Constrained Production Scheduling

4.1 Precedence Constrained Production Scheduling and Maximum Closure

The effectiveness of the proto-algorithm applied in the previous sections to the parcel assignment problem relies on the assumption that after removing (most of) the x variables (from Definition 2) as per the cutoff rules that the resulting LP will be small, i.e. it relies on the assumption that while the number of parcels may be large, the number of jobs is small. In general however this assumption does not hold. Nevertheless we will see that an analysis of the solution structure of the general precedence constrained production scheduling problem will show that analogous results can still be obtained.

Definition 26 *Given a directed graph $G = (N, A)$ and a vector of node weights $w \in R^N$, the Maximum Weight Closure Problem is that of finding a set $C \subseteq N$ of maximum weight such that for all $(i, j) \in A,\ i \in C \Rightarrow j \in C$. It can be formulated as the following integer program: max $w^T x : x_i - x_j \leq 0,\ \forall (i, j) \in A,\ x \in \{0, 1\}^N$.*

Observation 27 *The feasible space of the linear relaxation of the max closure problem, max $w^T x : x_i - x_j \leq 0,\ \forall (i, j) \in A,\ 0 \leq x \leq 1$ is defined by a totally unimodular matrix with integer right hand side and integer bounds and is therefore an integer polytope.*

Theorem 28 *The Precedence Constrained Production Scheduling Problem (PCPSP) can be recast as the linear relaxation of a maximum weight closure problem with side constraints, with the same number of variables and constraints overall and the same number of side constraints.*

Proof (sketch): Define a linear system $L1$ in which for each $j \in \mathcal{N},\ t = 1, \ldots, T,\ d = 1, \ldots, R$, there is a variable $y_{j,t,d}$ with bound constraints as follows:

$$y_{j,1,1} \geq 0 \tag{43}$$

$$y_{j,t,1} \geq y_{j,t-1,R}, \quad \text{if } t > 1 \tag{44}$$

$$y_{j,t,d} \geq y_{j,t,d-1}, \quad \text{if } d > 1 \tag{45}$$

$$1 \geq y_{j,T,R} \tag{46}$$

Define now a second linear system $L2$ in which there is a variable $x_{j,t,d}$ for each $j \in \mathcal{N},\ t = 1, \ldots, T,\ d = 1, \ldots, R$ satisfying

$$x \geq 0, \quad \sum_{t=1}^{T} \sum_{d=1}^{R} x_{j,t,d} \leq 1,\ \forall j \in \mathcal{N} \tag{47}$$

Then the feasible spaces of $L1$ and $L2$ are in one-to-one correspondence, and for each feasible x there is a feasible y satisfying

$$y_{j,t,d} = \sum_{t'=1}^{t-1} \sum_{d'=1}^{R} x_{j,t',d'} + \sum_{d'=1}^{d-1} x_{j,t,d'} \tag{48}$$

Replacing the x variables in PCPSP by the y variables thus defined yields the desired result. □

From now on we will no longer focus on PCPSP and will instead focus on what we now know to be the equivalent problem: max closure with side constraints. We will refer to this problem as the General Precedence Constrained Problem (GPCP).

Definition 29 *The General Precedence Constrained Problem (GPCP) is defined as follows: Let $G = (\mathcal{N}, \mathcal{A})$ be a digraph,*

maximize $c^T x$ subject to:

$$x_i \leq x_j,\ \forall (i,j) \in \mathcal{A} \tag{49}$$

$$Dx \leq d \tag{50}$$

$$0 \leq x \leq 1$$

The following theorem together with Observation 27 says that the GPCP with no side constraints, or equivalently, the GPCP with lagrangian relaxation of its side constraints, can be solved with a min cut algorithm.

Theorem 30 *[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 it can be shown by reduction from max clique that adding a single cardinality constraint to a max closure problem is enough to make it NP-hard.

4.2 Distinct Fractional Values and Decomposition Theorems

4.2.1 Distinct Fractional Values

The key structural feature of *GPCP* that will enable the application of the algorithm template of Section 3.4.1 is given in the following theorem.

Theorem 31 *Let P be the feasible space of any GPCP, and let x be an extreme point of P at which q linearly independent side constraints are binding, then x contains no more than q distinct fractional values.*

Before we prove the theorem we need some lemmas.

Lemma 32 *Let $P = \{x \in R^n : Ax \leq b\}$. Let x be an extreme point of P. Let $\bar{A}x = \bar{b}$ be the binding constraints of P at x, and let N^x be the null space of $\bar{A}$. Then $N^x = \{0\}$.*

Lemma 33 *Let $P = \{x \in R^n : Ax \leq b,\ Dx \leq d\}$. Let x be an extreme point of P. Let $\bar{A}x = \bar{b}$, $\bar{D}x = \bar{d}$ be the binding constraints of P at x, and assume $\bar{D}$ has q linearly independent rows, and let N^x be the null space of $\bar{A}$. Then N^x can be spanned by q vectors.*

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

Lemma 34 *Let P be the feasible space of a GPCP with q side constraints. Refer to the portion of the constraint matrix containing the precedence constraints and the constraints $0 \leq x \leq 1$ as $Ax \leq b$, and to the side constraints as $Dx \leq d$. Let x be an extreme point of P with k distinct fractional values $\{\alpha_1, \ldots, \alpha_k\}$ and say that the indicator vector for $x = \alpha_j$ is denoted θ^j, so that where x^i is the integer part of x (i.e. the floor of x), x is decomposed as*

$$x = x^i + \sum_{r=1}^{k} \alpha_r \theta^r. \tag{51}$$

Let $\bar{A}$ be the submatrix of A containing the binding constraints at x. Then the vectors θ^r are linearly independent and belong to the null space of $\bar{A}$.

Proof: First we prove that $\bar{A}\theta^r = 0$. Given a precedence constraint $x_i - x_j \leq 0$, if the constraint is binding then $x_i = x_j$. Thus if $x_i = \alpha_r$, so that $\theta^r_i = 1$, then $x_j = \alpha_r$ also, and so $\theta^r_j = 1$ also, and so $\theta^r_i - \theta^r_j = 0$, and by the same token if $x_i \neq \alpha_r$ then $x_j \neq \alpha_r$ and again $\theta^r_i - \theta^r_j = 0$. If a constraint $x_i \geq 0$ or $x_i \leq 1$ is binding then naturally $\theta^r_i = 0$ for all r as it is not fractional. As to linear independence, this is clear from the fact that the supports of the θ^r vectors are disjoint. □

Lemmas 33 and 34 prove Theorem 31 as a corollary, since there cannot be more θ^r vectors than there are linearly independent rows in $\bar{D}$.

4.2.2 A Decomposition Theorem for TUM problems with Side Constraints

The result stating that the number of distinct fractional values in any extreme point solution to GPCP is no more than the number of binding side constraints can be interpreted as saying that the amount of "fractionality" in the extreme point solution is a (very slowly increasing) function of the number of binding side constraints. In this section we will prove a similar result for all problems for which the constraint matrix is totally unimodular, but for which extra side constraints exist.

Theorem 35 *Let P, A, $\bar{A}$, D, q, x and N^x be as in Lemma 33, and assume additionally that A is TUM and that b is integer. Define*

$$I^x = \{y \in R^n : y_i = 0,\ \forall i \text{ s.t. } x_i \text{ is integer}\} \tag{52}$$

and let $\{\theta^1, \ldots, \theta^q\}$ be any spanning set for $N^x \cap I^x$. Then there exists an integer vector x^i that satisfies:

1. $Ax^i \leq b$

2. $\bar{A}x^i = \bar{b}$

3. $x^i_j = x_j,\ \forall j : x_j$ *is integer*

and a vector $\alpha \in R^q$ *such that*

$$x = x^i + \sum_{r=1}^{q} \alpha_r \theta^r. \tag{53}$$

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

4. $x^i_j = 0,\ \forall j : x_j$ *is fractional.*

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 coordinate 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 $\bar{A}x^i = \bar{b}$. Clearly $x - x^i$ belongs to I^x, and moreover $\bar{A}(x - x^i) = 0$ so that it belongs to N^x as well, and so it can be decomposed as

$$x - x^i = \sum_{r=1}^{q} \alpha_r \theta^r \tag{54}$$

which proves (53).

For the special case of $GPCP$ we have already described a decomposition (51) for which x^i equals x everywhere that x is integer and is zero elsewhere. □

Observation 36 *Sets* $\{\theta^1, \ldots, \theta^q\}$*, as defined in Theorem 35, exist for which all values for each* θ^r *are integers between* $-m$ *and* m*, where* m *is the row rank of* $\bar{A}$*. Thus Theorem 35 implies that* x *can be decomposed as an integer vector (matching with* x *wherever* x *is integer) and a linear combination of no more than* q *integer* $\{-m, \ldots, m\}$ *vectors, where* q *is the number of binding side constraints at* x*.*

Proof: Let x and $\bar{A}$ be is as in Theorem 35, and let $\tilde{A}$ be the same as $\bar{A}$ but with all columns from $\bar{A}$ for which x_i is integer removed, and with a minimum collection of rows removed so as to make $\tilde{A}$ of full row rank. Note that $\tilde{A}$ is also TUM and where N^x and I^x are as in Theorem 35 then the null space of $\tilde{A}$ is $N^x \cap I^x$. Let us say that $\tilde{A}$ is a $m' \times p$ matrix ($m' \leq m$). Find a collection of m' linearly independent columns of $\tilde{A}$, and denote the nonsingular matrix thus defined as B and the remaining columns as N. The null space of A is spanned by the vectors $\{(B^{-1}N^j, e^j)^T\}$, where N^j is the j^{th} column of N and e^j is the j^{th} unit vector in R^{p-m}. Since B is TUM, B^{-1} is a $0, 1, -1$ matrix, and N^j is $0, 1, -1$ as well, and so these vectors are all integer with absolute value $\leq m'$. □

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

Corollary 37 *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 that are binding at* x*. 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 I^x and N^x be as defined in Theorem 35, and let A be the node-arc incidence matrix. Restricting ourselves to I^x is equivalent to just removing some arcs from the graph, and any $y \in I^x : Ay = 0$ is a circulation, which can be decomposed into cycle flows in the reduced graph.

$N^x \cap I^x$ can therefore be spanned by cycle flow vectors, and the result now follows from Theorem 35. □

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.

4.3 Applying the Algorithm Template

4.3.1 Partitioning Constraints for GPCP

The result stating that the number of distinct fractional values in an extreme point solution is no more than the number of binding side constraints (Theorem 31) suggests a way in which the algorithm template can be applied to GPCP:

Idea 38 *Let the constraints $H^k(x) = h^k$ defined in the algorithm template in Section 3.4.1 be partitioning constraints that partition the node set into parts such that every node in a part has a common solution value.*

Formally, let Θ be the matrix whose r^{th} column θ^r is the indicator vector for the r^{th} part in the partition; add variables y_r to the problem for each part in the partition, and add the constraints

$$x = \Theta y. \tag{55}$$

Obviously however, the point of these constraints is that adding them explicitly in this manner is equivalent to just collapsing the nodes in each part into a single node, which is what we would do in practice.

Theorem 31 says that we never need more than a small number of parts (no more than two more than the number of side constraints) in order to capture the optimal solution. We do not know what the optimal partition is, but if we had a way of "guessing" then we never need our guessed partitions to have a large number of parts, and thus each would collapse the problem to a small LP.

Recall now from Sections 3.5.1 and 3.6.3 that in order for the algorithm to converge we need to refine the partition yielded by each iteration's guess so that it is compatible with the solution yielded for the previous iteration's partition. One way to do this is to refine the new guessed partition by intersecting each part with each part of the previous iteration's partition. This could quickly lead however to a partition with a very large number of parts, and thus we need a way to "coarsify" an iteration's partition so that it will still be compatible with its optimal solution but will have a smaller number of parts (as in Sections 3.5.2 and 3.6.4).

Here is another place where the small number of distinct values in the solution is crucial. Observe that after collapsing the nodes in each part into a single node, the restricted problem obtained is *still a GPCP*, and thus the number of fractional elements in its optimal extreme point solution is always small. Since the optimal solution to the restricted problem did not need more parts than it has distinct solution values, this says that we can recoarsify the partition to have no more parts than two more than the number of side constraints.

We still need to discuss what our "guesses" should be, but first we formalize what we know so far.

Lemma 39 *Given a GPCP, and given a partition of the elements defined by a matrix Θ whose r^{th} column θ^r is the indicator vector for the r^{th} part in the partition, then if we collapse each part in the partition to a single element*

- *The resulting problem is itself a GPCP in which the arcs are obtained for the collapsed graph in the natural way, and the side constraints are*

$$D\Theta \leq d. \tag{56}$$

- *$D\Theta$ has the same number of rows as D and as many columns as there are parts in the partition.*

- *The optimal solution x to this restricted problem induces a feasible solution to the unrestricted problem which can itself be decomposed as*

$$x = \sum_{j=1}^{q} \alpha^j \theta^j \tag{57}$$

where α is the vector of the distinct values in x, θ^j is the indicator vector for $x = \alpha_j$, and q is no more than 2 more than the number of side constraints.

- *If we combine elements in the partition to form a new partition in which each part is the support of $\theta^j,\ j = 1, \ldots, q$ (from expression (57)), then x is still an optimal solution to the restricted problem defined by this new partition.*

4.3.2 Guessing Partitions

The algorithm template states that every iteration k is defined by the lagrangian with penalties μ^k, and that we ought to obtain our restricting simplifying constraints from the lagrangian solution. So considering that the simplifying constraints we are looking for are partitionings, we are looking for a partitioning of the elements that is induced by the lagrangian solution. The lagrangian problem however is just a conventional max weight closure problem (Observation 27), and its solution is a closure, i.e. a subset of the nodes. The most natural partition induced by the lagrangian solution is thus certainly the partition of the node set into that closure and its complement. Later we will see why this partition is particularly appropriate, though for now we just note that it is natural.

There are thus two natural ways in which the partitioning constraints may be updated at each iteration based on the lagrangian solution as per the algorithm template:

Idea 40 *Using the notation of the algorithm template of Section 3.4.1, given optimal primal x^k for $P2(k-1)$, decomposed as per expression (57), and optimal duals μ^{k-1}, and an optimal lagrangian solution w^k generated from μ^{k-1}, create a new partition as follows: Start with the partition $\mathcal{C}$ in which each part is the support of $\theta^j,\ j = 1, \ldots, q$, and refine it by splitting each part into its portion that is included in the max closure defined by w^k and into its portion that is excluded from the max closure defined by w^k.*

Alternatively,

Idea 41 *Refine the previous iteration's partition by splitting each part into its portion that is included in the max closure and into its portion that is excluded from the max closure.*

Refining the partition as per Idea 41 yields partition constraints $H^k(x) = h^k$ that are weaker than $H^{k-1}(x) = h^{k-1}$, and refining the partition as per Idea 40 yields partition constraints, which while not necessarily weaker than $H^{k-1}(x) = h^{k-1}$, are still such that x^{k-1} satisfies $H^k(x) = h^k$, and which are therefore still sufficient to maintain algorithm monotonicity.

Observation 42 *If there are p side constraints then applying Idea 40 yields a partition with no more than $2(p+2)$ parts, and applying Idea 41 yields a partition with no more than twice the number of parts in the previous iteration's partition.*

4.3.3 The Algorithm

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

- At every iteration, solve the lagrangian with penalties μ^{k-1} equal to the optimal duals for the restricted LP, $P2(k-1)$, 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 $P2(k-1)$ 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 and expression (57).

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 $P1$.

2. Get optimal solution y^k to $L(P1, \mu^{k-1})$, and define

$$I(y^k) = \{n \in \mathcal{N} : y^k_n = 1\} \tag{58}$$

and define

$$O(y^k) = \{n \in \mathcal{N} : y^k = 0\}. \tag{59}$$

If $k > 1$ and $\mathcal{C}^{k-1}$ is represented as

$$\mathcal{C}^{k-1} = \{C^{k-1}_1, \ldots, C^{k-1}_{r_{k-1}}\} \tag{60}$$

and no set C^{k-1}_j overlaps both $I(y^k)$ and $O(y^k)$ then STOP.

3. Let

$$\mathcal{L} = \{L_1, \ldots, L_{l_k}\} \tag{61}$$

be a partition of $\mathcal{N}$ refining $\{I(y^k), O(y^k)\}$ and define the partition

$$\mathcal{C}^k = \{L_i \cap C^{k-1}_j : i = 1, \ldots, l_k,\ j = 1, \ldots, r_{k-1}\} \tag{62}$$

of $\mathcal{N}$. Denote the nonempty sets in $\mathcal{C}^k$ as $\{C^k_1, \ldots, C^k_{r_k}\}$, and define corresponding incidence vectors $\theta^1, \ldots, \theta^{r_k}$.

4. Solve the new GPCP problem $P2(k)$ defined by collapsing all nodes in each C^k_j into a single supernode and replacing the side constraints $Dx \leq d$ by $D\Theta x \leq d$, where Θ is the matrix whose columns are the vectors θ^j.

   Let x^k be the optimal solution to $P2(k)$ represented in terms of the original graph, let μ^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) If $z^k > z^{k-1}$, let $\alpha_1, \ldots, \alpha_q$ be the distinct values in x^k (and note that q is no more than the number of side constraints plus 2), and for each r, let σ^r be the indicator vector for $x^k = \alpha_r$, and decompose x as $x = \sum \alpha_r \sigma^r$.

   Let $\mathcal{C} = \{C^r\}$ be the partition of $\mathcal{N}$ in which each C^r is the support of σ^r.

   Update $\mathcal{C}^k := \mathcal{C}$, and $r_k = q$.

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

Lemma 43 *If $P2(1)$ is feasible then the algorithm converges.*

Proof: The fact that x^k is always a feasible solution to $P2(k+1)$ implies that if $P2(1)$ is feasible then so is every $P2(k)$, and it also implies that the solution values $\{z^k\}$ increase monotonically. Since a coarsification step (step 6) can only be performed at an iteration k for which $z^k > z^{k-1}$, monotonicity implies that a suboptimal partition can never be repeated. Since there are finitely many partitions the algorithm must terminate, and by Theorem 17 it terminates with an optimal solution. □

Note that the algorithm can be formulated such that we always have $P2(1)$ feasible by putting slacks into the side constraints with large penalties in the original problem $P1$.

Note also that in principle the algorithm allows us to choose at each Step 3 in each iteration any partition $\mathcal{L}$ that refines the partition $\{I(y^k), O(y^k)\}$.

4.4 Analyzing the GPCP Algorithm

The algorithm performs extremely well in practice. Problems with millions of variables and tens of millions of constraints converge in 10 – 25 iterations. Max closure problems moreover can be solved very quickly with max flow algorithms, and so even with problems of this size each iteration solves in a matter of seconds.

The algorithm only guarantees finite convergence, but there are features of GPCP that enhance the algorithm's performance, which will be discussed in this section.

4.4.1 The Connection Between the LP Solution and the Lagrangian Solution

Observation 44 *We had three observations in our initial discussion of the algorithm template that were relevant to how one ought to choose the constraints $Hx = h$ to be applied at each iteration. These are summarized as follows:*

- *As per Observation 18, considering that the algorithm will be applying these constraints to the LP, and can only terminate if it finds constraints that are satisfied both by the lagrangian solution (with optimal penalties) and by the optimal LP solution, then the constraints we pick should be a feature that, at least for the lagrangian with optimal penalties, will characterize both the lagrangian solution as well as the optimal LP solution.*
- *In line with our heuristic observation (Observation 19), we should be looking for a feature of the lagrangian solution which even when drawn from a lagrangian solution with suboptimal penalties may still approximately characterize the optimal LP solution, and which can be expected to better approximate it as the penalties get closer to optimal.*
- *In line with Observation 16, we should choose a feature of the lagrangian solution that we can expect to remain fairly unchanged even as the penalties change, especially when far from the optimal penalties.*

We will show now that the choice of constraints $Hx = h$ that the GPCP algorithm makes is a good one from the perspective of Observation 44.

Note first that the partitioning paradigm in implementing the algorithm template is to derive a partition from the lagrangian solution and to use that partition to refine our current partition. In one sense then, the first "good constraint choice criterion" of Observation 44 is satisfied for free, as we only use the lagrangian solution to refine the partition, and we never use it to impose a new constraint.

Nevertheless, the more detailed the features that we derive from the lagrangian solution to refine the partition that we will apply to the LP solution, and the more characteristic they are of the optimal LP solution (at least where the lagrangian penalties were optimal), the stronger the LP approximation will be. The algorithm for PAP can be thought of as setting the gold standard in this regard. In that case the feature that we borrow from the lagrangian solution and apply to the LP is the cutoffs, i.e. the rules that govern where extracted parcels are to be processed. These rules apply both to the lagrangian solution (with optimal penalties) and to the optimal LP solution, and they *completely determine the LP solution* (except for the "tied" variables, of which there are typically few, and which can be left unaggregated) in the sense that when applied to form a restricted LP, the solution yielded is an optimal solution to the original LP .

We will prove a theorem that shows that in the case of GPCP we have a similar situation. The lagrangian of the GPCP is a max closure problem and its solution is a closure. The theorem shows that in a similar manner to the PAP situation, there is also a close relationship between these lagrangian max closures (when given optimal penalties) and the optimal LP solution for the GPCP.

First we prove a few lemmas.

Lemma 45 *Given a graph G with node weights and given two max closures C and C'. The union $C \cup C'$ as well as the intersection $C \cap C'$ are also max closures.*

Proof: Given an arbitrary set of nodes S, denote the sum of the weights of the nodes in S as $w(S)$. Since $C \cup C' = (C - C') \cup C'$ is also a closure, by maximality of C' we must have $w(C - C') \leq 0$, which implies that $w(C \cap C') \geq w(C)$. But by maximality of C, since $C \cap C'$ is also a closure, it must be that $w(C \cap C') = w(C) = w(C')$, and $w(C - C') = 0$. But this in turn implies that $w(C \cup C') = w(C') = w(C)$. □

Corollary 46 *Given a graph G with node weights, there is a unique inclusion minimal max closure C^m and a unique inclusion maximal max closure C^M, and for every max closure C we have $C^m \subseteq C \subseteq C^M$.*

Theorem 47 *Let x be an optimal solution to a GPCP with optimal duals μ corresponding to the side constraints. Let*

$$x = x^i + \sum_{r=1}^{q} \alpha_r \theta^r \tag{63}$$

be the decomposition of x into its integer and fractional parts as in Lemma 34. Assume further that $\alpha_1 > \alpha_2 > \cdots > \alpha_q$, and let C^i be the support of x^i and for each θ^j let C^j be its support. Then

1. *C^i is a max closure for the lagrangian problem with penalties μ.*
2. *For each $j = 1, \ldots, q$, C^j has penalized value of zero, and*
3. *for each $j = 1, \ldots, q$, $C^i \cup \bigcup_{r=1}^{j} C^r$ is also a max closure.*

Proof: It is clear that for all j, $C^i \cup \bigcup_{r=1}^{j} C^r$ is a closure, since any arc originating at a node in that set must point to a node with a solution (x) value $\geq \alpha^j$, which is also in that set.

Let $\bar{D}x \leq \bar{d}$ be the matrix of binding side constraints at x, and let $\bar{A}x \leq \bar{b}$ be the binding precedence constraints. Let ρ be the vector of optimal duals corresponding to the precedence constraints; let δ be the vector of optimal duals corresponding to the constraints $x \leq 1$, and let π be the vector of optimal duals corresponding to $x \geq 0$. Note first that for all r, $\bar{A}\theta^r = 0$ as proven above in Lemma 34, and we also have $\delta\theta^r = \pi\theta^r = 0$ since these duals can only be nonzero for integer valued variables and for all such coordinates θ^r is zero. Therefore,

$$cx = c(x^i + \sum_{r=1}^{q} \alpha_r \theta^r) = \tag{64}$$

$$cx^i + \sum_{r=1}^{q} \alpha_r (\rho A + \mu D + \delta - \pi)\theta^r = \tag{65}$$

$$cx^i + \mu D \sum_{r=1}^{q} \alpha_r \theta^r = \tag{66}$$

$$cx^i + \mu D(x - x^i) = cx^i + \mu Dx - \mu Dx^i = \tag{67}$$

$$cx^i + \mu d - \mu Dx^i. \tag{68}$$

Expression (68) is the objective value of the lagrangian with penalties μ evaluated at x^i, and since the optimal value of this lagrangian problem is cx we conclude that x^i is indeed optimal for that lagrangian. Finally, since $c\theta^r = \mu D\theta^r$ we conclude that the penalized value of θ^r is zero, which proves the remainder of the theorem. □

Theorem 47 and Corollary 46 imply the following corollary.

Corollary 48 *Given optimal lagrangian penalties μ, for all nodes n included in the minimal max closure we have $x_n = 1$ in every optimal LP solution, and for all n not included in the maximal max closure we have $x_n = 0$ in every optimal LP solution.*

The feature then which is clearly shared by the optimal lagrangian solution (with optimal penalties) and the optimal LP solution is that *they both distinguish between the elements contained in the minimal max closure and the elements not contained in the maximal max closure.* The algorithm captures this feature by refining the partition at each iteration of the algorithm by splitting each part in the partition into the part which lies in the current iterate's lagrangian max closure, and the part that does not.

4.4.2 Partitionings for PAP and Partitionings for GPCP: A Qualitative Analysis

The partitions used by GPCP are thus broadly similar to the cutoff induced partitions we had for PAP in that the elements that lie inside of the smallest max closure and those that lie outside of the largest max closure (which correspond loosely to the (parcel, period, destination) triples that lie "above" or "below" cutoff) are completely determined for the LP as well. But in this case there are two disadvantages:

1. We do not in general know what the smallest max closure C^m and the largest max closure C^M are, and so we cannot avoid aggregating the elements in $C^M - C^m$.

2. There are typically very many elements in $C^M - C^m$, and so we do not want to avoid aggregating them. In particular, all elements that are fractional in any optimal extreme point LP solution are in this category, and there are typically many such elements.[3]

Even if we make no effort to avoid aggregating the elements in $C^M - C^m$ however, the algorithm is still well suited to identify the collections of elements in that set with a common value. Consider that Theorem 47 and Corollary 46 also imply the following corollary.

Corollary 49 *Given optimal lagrangian penalties μ, and the collection of max closures for the associated lagrangian problem, define the algebra $\mathcal{A}$ generated by $\{C - C^m : C \text{ is a max closure}\}$ where complementation is defined with respect to $C^M - C^m$. Then*

1. *The atoms of $\mathcal{A}$ are all sets with zero (penalized) value.*

2. *Where arcs are inherited from the underlying graph, every closed nonempty collection of atoms defines a max closure for the underlying graph (after adding the smallest max closure back in) and conversely.*

The atoms of the algebra $\mathcal{A}$ are the candidate sets that may or may not be included in the max closure, and these are also the candidate sets that can be assigned common fractional values in the optimal LP solution. Minute changes in the optimal penalties will drive some of these atoms into negative value and some into positive value, which means that solving lagrangian problems with penalties in the vicinity of the optimal solution will tend to identify these sets as well.

Thus despite the fact that even optimal penalties μ do not give us sufficient information in the lagrangian solution to construct the full LP solution (unlike the case for the PAP), it is still the case that by solving several closure problems the algorithm can identify all of the sets of elements which have a common value in the optimal LP solution. This *will* in fact happen because any *suboptimal* LP solution based on a *suboptimal* partition will yield duals μ for which every optimal lagrangian solution *will split a part in that suboptimal partition* (or else we would have reached a stopping condition, which would mean that the LP solution was not suboptimal). The partition induced by improving μ will thus give increasingly good approximations of what the optimal LP solution needs to *split*. Each iteration of the algorithm thus uncovers relevant places in which the current partition suffers as a result of not separating between elements in different atoms, and it splits accordingly.

[3]Note that for PAP it is also true that all parcel, period, destination triples (j, t, d) for which $x_{j,t,d}$ is strictly between 0 and the proportion of j extracted in t are "tied" and not fixed to a destination, but there are usually very few such elements. This is easiest to see in the case of the "simple" PAP where the two destinations are waste and processing plant, and the side constraints are periodic plant capacity constraints. Only the parcels whose value per unit capacity are exactly equal to the cutoff value can be split. So unless the values are very lumpy this will not happen often.

Note that here once again we benefit from the fact that the optimal LP solution needs not have a large number of distinct fractional values, so that one need not necessarily perform many such steps in order to obtain an optimal LP solution.

As for the latter two "good constraint choice criteria" of Observation 44, it seems reasonable to expect that the separations between elements inside and outside of the max closure yielded by the lagrangian solutions taken even at suboptimal penalties will also approximate separations that the LP needs to make, and that this will improve as the penalties approach optimality.

Finally it also seems reasonable to expect that the partitions yielded by the lagrangian max closure, which ultimately reflect the structure of the underlying graph, will not change drastically with changes in the penalties. There may be large "balanced" sets corresponding to the elements with a common fractional value in the optimal LP solution, which can move entirely in or entirely out of the max closure with small changes in the penalties when the penalties are near optimal, but once these have been captured in the partition already then any other changes could be expected to be generally small.

4.4.3 Improving the Partition: A Sensitivity Analysis

Lemma 50 *Let $P2(k)$ be the restricted LP at the k^{th} iteration of the GPCP algorithm, and let μ^k be the optimal duals for $P2(k)$ corresponding to the side constraints, the optimal solution x^k, when written in the original space, can be decomposed as*

$$x^i + \sum_{r=1}^{q} \alpha_r \theta^r, \tag{69}$$

where x^k has q distinct fractional values $\alpha_1, \ldots, \alpha_q$, x^i is the integer vector which equals x^k at every coordinate for which x^k is integer, and θ^r is the indicator vector for $x^k = \alpha_r$. Denote the support of x^i as C^i, and the support of each θ^r as C^r and assume $\alpha_1 > \cdots > \alpha_q$. Then C^i and every

$$C^i \cup \bigcup_{r=1}^{j} C^r \tag{70}$$

are all closures and the penalized value of each C^r w.r.t. μ^k is 0. Moreover if C^i is a max closure w.r.t. μ^k then x^k is optimal for the unrestricted problem as well.

Proof: The last statement that if C^i is a max closure then x^k is optimal for the unrestricted problem, is a result of the fact that in this case we could consider C^i and $\{C^r\}$ to constitute the current iteration's partition, and C^i would be an optimal lagrangian solution to the lagrangian relaxation in the next algorithm iteration. Since none of the sets C^r or C^i overlap both C^i and its complement, we would hit a stopping condition. The rest of the lemma follows from Theorem 47. □

So if after solving the restricted problem we find that the closure C^i induced by the solution x^k is a max closure for the lagrangian at μ^k then we are done. Otherwise C^i is not a max closure, and so there must exist either closed subsets R of C^i in the reverse graph with negative value that may be excluded from the closure, or else, letting C^0 denote the collection of nodes with value 0 in x, there must exist closed subsets C of C^0 or some C^r of positive value that may be included into the closure.

We will show that slicing off the appropriate R and/or C sets to obtain a max closure can be seen as advantageous to the LP solution from a sensitivity analysis perspective.

Lemma 51 *Let C^i and $\{C^r; r = 1, \ldots, q\}$ be as in Lemma 50, and let C^0 represent the elements for which x^k in Lemma 50 was zero, and let the columns of the matrix $\Theta = (\theta^i, \theta^0, \theta^1, \theta^2, \ldots \theta^q)$ be the indicator vectors for the parts in the partition $\{C^i, C^0, \{C^r,\ r = 1, \ldots, q\}\}$. Then the restricted GPCP for which all elements in each C^r are required to have the same value can be written as follows:*

$$max\ cx : x - \Theta y = 0,\ A\Theta y \leq 0,\ Dx \leq d,\ 0 \leq y \leq 1. \tag{71}$$

The optimal duals corresponding to the D constraints are still μ^k, and if we let δ be the optimal duals corresponding to the $x - \Theta y = 0$ constraints, we obtain

$$c = \mu^k D + \delta. \tag{72}$$

Definition 52 *Given a set T of nodes in a graph, we shall say that a subset $S \subseteq T$ is "closed with respect to" T, if no arcs point from S to $T - S$.*

By a rough sensitivity analysis, for a given element e, δ_e is positive if an increase in x_j would raise the objective, and negative if an increase in x_j would decrease the objective.

In this sensitivity analysis sense we would say that it is "beneficial" to split C^0 and/or some C^r sets by increasing the value of all elements in some $C' \subseteq (C^i)^c$ closed w.r.t. $(C^i)^c$ (so that the new solution remains precedence feasible) if $(c - \mu^k D)\theta^C > 0$ (where θ^C is the indicator vector of C). Similarly it would be "beneficial" to split C^i and/or some C^r sets by decreasing the value of all elements in some $R' \subseteq (C^0)^c$ closed w.r.t. $(C^0)^c$ in the reverse graph if $(c - \mu^k D)\theta^{R'} < 0$.

At each iteration after performing the max closure step, the algorithm (in its most basic implementation) splits the parts in its current partition into the portion of each part that belongs to the max closure and into the portion of the part that does not belong to the max closure.

The first statement in the following lemma says that there is no way to alter this split which is "beneficial". Intuitively this is obvious, since "beneficence" is measured by $(c - \mu^k)D$ which is the objective function of the lagrangian.

The second statement says that given any set C' whose elements' solution values it would be "beneficial" to increase, if C' is such that it could only be less beneficial to have chosen one of its subsets instead, then C' is a subset of $\bar{C} - C^i$, where $\bar{C}$ is the lagrangian's max closure. A similar result holds for sets of elements R' whose elements it would be beneficial to decrease.

The lemma thus implies that in this sense *every* improvement in the partition is captured by the algorithm's choice, as every improving C and R (that only get worse if a portion is left out) is subset to the C and R suggested by the max closure.

The lemma relies ultimately on the property of max closure problems that every closed set of positive value contributes to every max closure, and therefore *every* improvement in the current partition is captured by the new max closure.

Lemma 53 *Under the conditions of Lemma 51, let $\bar{C}$ be a max closure for the lagrangian problem with penalties μ^k, and let*

$$C = \bar{C} - C^i \tag{73}$$

$$R = (C^0)^c - \bar{C}. \tag{74}$$

For any set S of nodes in the graph, denote its incidence vector as θ^S. There is no $\tilde{R} \subseteq \bar{C}$ that is closed w.r.t. $\bar{C}$ in the reverse graph for which $(c - \mu^k D)\theta^{\tilde{R}} < 0$, and there is no $\tilde{C} \subseteq \bar{R}$ that is closed w.r.t. $\bar{R}$ in the reverse graph for which $(c - \mu^k D)\theta^{\tilde{C}} > 0$.

Moreover, if some $C' \subseteq (C^i)^c$ which is closed with respect to $\mathcal{C}^i)^c$, satisfies that there is no $\tilde{R} \subseteq C'$ that is closed w.r.t. C' in the reverse graph for which $(c - \mu^k D)\theta^{\tilde{R}} \leq 0$, then $C' \subseteq C$. Similarly if some $R' \subseteq (C^0)^c$ is closed with respect to $(C^0)^c$, and satisfies that there is no $\tilde{C} \subseteq R'$ that is closed w.r.t. R' in the reverse graph for which $(c - \mu^k D)\theta^{\tilde{C}} \geq 0$, then $R' \subseteq R$.

Proof: Suppose that there is a $\tilde{R} \subseteq \bar{C}$ that is closed w.r.t. $\bar{C}$ in the reverse graph for which $(c - \mu^k D)\theta^{\tilde{R}} < 0$. Then $\bar{C} - \tilde{R}$ would be a closure, and if we denote its incidence vector as x and the incidence vector of the max closure $\bar{C}$ as $\bar{x}$, then

$$(c - \mu^k D)x = (c - \mu^k D)\bar{x} - (c - \mu^k D)\theta^{\tilde{R}} > (c - \mu^k D)\bar{x} \tag{75}$$

which contradicts the optimality of $\bar{x}$. The proof for the statement regarding $\tilde{C}$ is similar. For the second part of the lemma, suppose that $H = C' - C \neq \emptyset$, then by assumption, since H is closed w.r.t. C' in the reverse graph (i.e. no arc points from a node in $C' \cap C$ to a node in $C' - C$), we have $(c - \mu^k D)\theta^H > 0$. But H is disjoint from $\bar{C}$, which implies that $(c - \mu^k D)(\bar{x} + \theta^H) > (c - \mu^k D)\bar{x}$, and $\bar{C} \cup H = C^i \cup C \cup C'$ is closed, which contradicts the optimality of $\bar{C}$. The proof for the case $R' - R \neq \emptyset$ is similar. □

4.4.4 Lagrangian Improvement and LP improvement: A More General Analysis

Recall from Lemma 50 that the integer part x^i of the solution x^k of the restricted problem $P2(k)$ is typically a suboptimal solution to the lagrangian of the unrestricted problem with penalties μ^k. In iteration $k+1$ we evaluate an optimal solution y^{k+1} to this lagrangian. In this section we will show that the direction from x^i to y^{k+1} corresponds with an improving vector for x^k as well which under certain conditions is guaranteed to be feasible, and that the algorithm allows the LP solution to move in that direction.

Lemma 54 *Let P be a GPCP with an extreme point x^*. Let k be the number of distinct fractional values in x^*. Recall from Theorem 31 that there must therefore be at least k binding side constraints at x^*. Assume now that the number of binding side constraints is exactly k. Then where Θ is the matrix whose columns are the indicator vectors for the distinct fractional values in x^*, and $\bar{D}$ is the matrix of binding side constraints, $\bar{D}\Theta$ is nonsingular.*

Proof: Consider the restricted problem in which all elements in the support of each θ^r have a common value, and in which all elements e for which x^*_e is integer are restricted to that integer value. For each of the k distinct values α_r we therefore have one variable y_r. We can now perform the substitution $x = x^i + \Theta y$, where x^i is a constant integer vector with a 1 in positions (and only in positions) where x^* is 1. Notice that the bound constraints are just $0 \leq y \leq 1$, and for any arc in the original graph (i,j), if x^*_i and x^*_j were both fractionally valued, then i is in the support of some θ^p and j is in the support of some θ^q, and so the associated precedence constraint $x_i \leq x_j$ is just replaced by $y_p \leq y_q$. If however x^*_i and x^*_j are both integer then they both are fixed and there is no constraint, and if x^*_i alone is fractional then x^*_j is 1 and we can throw away the constraint, and similarly if x^*_j alone is fractional. Thus this restricted problem is also a GPCP, and the binding side constraints are

$$D\Theta y \leq d - Dx^i. \tag{76}$$

Observe now that y^* with $y^*_r = \alpha_r$ is an extreme point solution with the same binding constraints, and thus $\bar{D}\Theta$ must have k linearly independent rows by Theorem 31. The lemma now follows from the fact that $D\Theta$ has k columns. □

There is no guarantee that the number of binding side constraints will not exceed k but this is typically the case.

Lemma 55 *Let x^k be the optimal solution obtained to the algorithm's restricted LP in the k^{th} iteration $P2(k)$ (with x^k represented in the original space). Let μ^k be the optimal duals for the side constraints in $P2(k)$, and let*

$$x^k = x^i + \Theta\alpha \tag{77}$$

be its decomposition into integer and fractional parts as in Lemma 34. Then where $\bar{D}$ are the binding side constraints, c is the objective vector and Θ is as in Lemma 54,

$$c\Theta = \mu^k D\Theta \tag{78}$$

and

$$cx^k = cx^i + \mu^k(d - Dx^i) \tag{79}$$

Proof: Consider the restricted problem in which all nodes with a common value in x^k are collapsed to a single node, and in which the new "super" nodes corresponding to the x^k values 0 and 1 are eliminated (as in the proof of Lemma 54). Clearly if we represent x^k in this reduced space then it is still an optimal extreme point solution. Note moreover that no precedence constraints are binding (since all variables in the reduced space have distinct value), and no bound constraints are binding (since all values are fractional), and μ^k is still the optimal dual vector corresponding to the side constraints. The objective function for this problem is $c\Theta$, and the side constraints are given by (76). Duality therefore implies (78). Where $y = \Theta x^k$ is x^k represented in the reduced space, then

multiplying both sides of (78) by y gives 79 by (76) and complementary slackness. □

Expression (79) says that the LP solution x^k is equal to the lagrangian value of its associated closure C^i in the lagrangian problem $L(P1, \mu^k)$. If C^i is optimal for that problem then the algorithm reaches a stopping condition and x^k is optimal for the unrestricted LP.

The following theorem says that moving $\delta > 0$ distance in any lagrangian feasible direction θ from the closure x^i associated with x^k, gives the same improvement to the lagrangian solution as the improvement given to the LP solution by moving δ in the θ direction from x^k plus some corrective movement in the fractional values $\Theta\gamma$, and this movement from x^k is *feasible for the LP* if δ is small enough and $\bar{D}\Theta$ is nonsingular. Thus in particular if θ is the vector such that $x^i + \theta$ maximizes the lagrangian $L(P1, \mu^k)$, then moving in the direction θ from x^k maximizes the improvement in the LP as well (though naturally the distance δ we can move while maintaining feasibility, while positive, may be small).

Where θ is defined such that $x^i + \theta$ optimizes the lagrangian problem $L(P1, \mu^k)$ (i.e. $x^i + \theta$ is the vector we refer to as y^k in the algorithm), then where C^i is the support of x^i and C is the optimal closure (the support of $x^i + \theta$), θ is the vector with 1 in positions corresponding to elements in $C - C^i$ and -1 in positions corresponding to elements in $C^i - C$ and zeros elsewhere. To allow the solution to move from x^k to $x^k + \Theta\gamma + \delta\theta$ we need only update the partition by splitting each part in the partition into those elements that were included in C and into those elements that were excluded from C. *This is exactly what the algorithm does.*

Moreover, this argument holds for all of the closures defined by x^k (all of which have the same value, which is cx^k) and not just for x^i. In particular, where $\alpha_1 > \cdots > \alpha_q$ are the distinct fractional values in x^k and θ^r is the incidence vector for $x^k_j = \alpha_r$, then for any $r \in \{1, \ldots, q\}$, if θ is defined such that $x^i + \sum_{j=1}^{r} \theta^j + \theta = y^k$ then to move from x^k to the appropriate $x^k + \Theta\gamma + \delta\theta$ we need only update the partition by splitting each part in the partition into those elements that were included in C and into those elements that were excluded from C.

Theorem 56 *Under the conditions of Lemma 55, given any vector θ such that $x^i + \theta$ is lagrangian feasible (for the lagrangian with penalties μ^k), the lagrangian value of $x^i + \theta$ is*

$$cx^k + (c - \mu^k D)\theta. \tag{80}$$

If $D\Theta$ is nonsingular, then there exists a scalar $\delta > 0$ and a vector γ such that $x^k + \Theta\gamma + \delta\theta$ is feasible for $P1$, and

$$c(x^k + \Theta\gamma + \delta\theta) = cx^k + \delta(c - \mu^k D)\theta. \tag{81}$$

Proof: The lagrangian value $x^i + \theta$ is

$$c(x^i + \theta) + \mu^k(d - D(x^i + \theta)) = cx^i + \mu^k(d - Dx^i) + c\theta - \mu^k D\theta = cx^k + (c - \mu^k D)\theta \tag{82}$$

where the final equality follows from (79).

Define γ by

$$\gamma = -\delta(\bar{D}\Theta)^{-1}\bar{D}\theta. \tag{83}$$

Observe first that for any coordinate e for which $x^k_e = 0$ we must have $\theta_e \geq 0$ and for any coordinate e for which $x^k_e = 1$ we must have $\theta_e \leq 0$ or else $x^i + \theta$ would not have been lagrangian feasible. The columns of the matrix Θ moreover only have nonzero elements corresponding to fractional x^k_e, so for small enough $\delta > 0$, $x^k + \Theta\gamma + \delta\theta$ still satisfies all bound constraints. For any binding precedence constraint $ax \leq 0$, we have $a\Theta = 0$ by Lemma 34. Writing this binding precedence constraint as $x_h \leq x_j$, if x^k_h and x^k_j are fractional then $x^i_h = x^i_j = 0$ and so $\theta_h \leq \theta_j$ or else $x^i + \theta$ would not have been lagrangian feasible, and clearly for the same reason we would have $\theta_h \leq \theta_j$ if x^k_h and x^k_j are integer (as in that case they match x^i on those values). Thus $x^k + \Theta\gamma + \delta\theta$ satisfies all binding precedence constraints and for small enough δ it satisfies the other precedence constraints as well. As for the side constraints,

$$\bar{D}(x^k + \Theta\gamma + \delta\theta) = \bar{d} + \bar{D}\Theta\gamma + \delta\bar{D}\theta = \bar{d} \tag{84}$$

by (83), and so for small enough δ they are also satisfied, which completes the proof of feasibility. The objective value is

$$c(x^k + \Theta\gamma + \delta\theta) = cx^k + c\Theta\gamma + \delta c\theta = \text{ by (78)} \tag{85}$$

$$cx^k + \mu^k D\Theta\gamma + \delta c\theta = \text{ by (83)} \tag{86}$$

$$cx^k - \delta\mu^k D\theta + \delta c\theta = \tag{87}$$

$$cx^k + \delta(c - \mu^k D)\theta. \ \square \tag{88}$$

5 Computational Experiments

In this section we present results from some of our experiments. A more complete set of results will be presented in the full paper. All these tests were conducted using a single core of a dual quad-core 3.2 GHz Xeon machine with 64 GB of memory. The LP solver we used was Cplex, version 10.2 and the min cut solver we used was our implementation of Hochbaum's pseudoflow algorithm ([H08]).

The tests reported on in Tables 1, 2 and 3, are based on three real-world examples provided by BHP Billiton[4], to which we refer as 'Mine1', 'Mine2' and 'Mine3' and a synthetic but realistic model called 'Marvin' which is included with Gemcom's Whittle [W] mine planning software. 'Mine1B' is a modification of Mine1 with a denser precedence graph. Mine3 comes in two versions to which we refer as 'big' and 'small'. Using Mine1, we also obtained smaller and larger problems by modifying the data in a number of ways, as follows:

- **Mine1 very small, small, medium, large**. These are all derived by aggregating the blocks in Mine1 into larger units to reduce the number of scheduling units.
- **Mine1 x7 small**. The two "x7 small" cases consist of 7 copies of the block graph for Mine1, but with blocks aggregated into larger units. The "global capacity" case has a single mining capacity per period and a single processing capacity per period and the "separate capacities" case has a separate mining constraint for each of the seven block graphs in addition to the processing capacity constraint in each period.
- **Mine1 full**. No aggregation; the actual Mine1 case.
- **Mine1 double, triple**. Obtained by taking 2 and 3 copies of Mine1 (resp.) but binding them by common capacity constraints.
- **Mine1 triple 23, triple 34 and triple 100**. Identical to the Mine1 triple case except that the number of periods was increased from 12 to 23, 34 and 100, respectively.

Some of the row entries in these tables 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 1.
- **Block arcs**. The number of arcs in the job (block) precedence graph G, as in Definition 1.
- **Problem arcs**. The number of arcs in the graph that the algorithm creates to represent the scheduling problem (as per Theorem 28 and Definition 29).
- **Binding side constraints at optimum**. The number of side constraints at the optimum solution as computed by our algorithm.
- **Iterations, time to 10^{-5} optimality**. The number of iterations (resp., the CPU time) taken by algorithm until it obtained a solution it could certify as having $\leq 10^{-5}$ **relative** optimality error.

[4]Data was masked.

- **Iterations, time to combinatorial optimality**. The number of iterations (resp., the CPU time) taken by algorithm until it terminated, obtaining a solution it could certify as *optimal* as per the algorithm's termination criteria. Notice that this implies that the solution is optimal as per the numerical tolerances of Cplex.
- **Lagrangian time**. The total CPU time expended by the Lagrangian procedure until termination at optimality.
- **Subproblem LP time**. The total CPU time expended solving auxiliary linear programs until termination at optimality.

Finally, an entry of "—" entry means that Cplex was unable to terminate the task after 100000 seconds of CPU time.

We ran our algorithm in all cases with the same settings, with the exception of the 'Mine3 large' case, for which we reduced the frequency at which coarsification (Step 6 in the algorithm) is performed. This reduced the number of iterations required until algorithm termination from 61 to 39, though it only reduced the overall solution time from 1680 seconds to 1592 due to the extra time spent solving the larger auxiliary linear programs at each algorithm iteration.

6 Further Work

The algorithm applied to PCPSP converges very quickly in practice. Problems with millions of variables and tens of millions of constraints typically converge in under 20 iterations, with each iteration requiring only seconds to solve. But though the observed convergence is fast, and we have derived a variety of results that indicate why this should be expected, we do not yet have a proof of fast convergence.

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 a totally unimodular matrix with 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.

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.

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.

[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.

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

[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.

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

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

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

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

	Marvin	Mine1B	Mine2	Mine3 small	Mine3 big
Blocks	9400	29277	96821	675	108264
Parcels	9400	29277	96821	2975	177843
Block arcs	145640	1271207	1053105	1748	2762864
Periods	14	14	25	8	8
Destinations	2	2	2	8	8
Variables	199626	571144	3782250	18970	3503095
Variables Cplex presolved	197666	568890	—	17056	—
Constraints	2048388	17826203	26424496	9593	19935500
Constraints Cplex presolved	2047939	17822237	—	9353	—
Problem arcs	2229186	18338765	3001354	24789	23152350
Side constraints	28	28	50	120	132
Non-knapsack side constraints	0	0	0	10	13
Binding side const. at optimum	14	11	23	33	44
Cplex time (sec)	55141	—	—	52	—
Algorithm Performance					
Iters. to 10^{-5} optimality (sec)	8	8	9	14	30
Time to 10^{-5} optimality (sec)	10	60	344	1	1117
Iters. to comb. optimality	11	12	16	15	39
Time to comb. optimality (sec)	15	95	649	1	1592
Lagrangian time (sec)	13	83	621	0	725
Subproblem LP time (sec)	1	0	6	1	709

Table 1: ***Sample problems***

	Mine1 very small	Mine1 small	Mine1 x7 small global cap	Mine1 x7 small separate cap	Mine1 medium	Mine1 large
Blocks	105	469	3025	3025	3280	7892
Parcels	755	2650	17072	17072	7636	15003
Block arcs	222	1311	8229	8229	22671	113703
Periods	12	12	12	12	12	12
Destinations	2	2	2	2	2	2
Variables	14282	52916	378236	344852	160944	292800
Variables Cplex presolved	14274	52909	378224	344845	160944	292560
Constraints	8834	37845	268963	244425	327628	1457684
Constraints Cplex presolved	8834	37845	268963	244425	327628	1457539
Problem arcs	22232	87618	627178	569184	477632	1727565
Side constraints	24	24	24	96	24	24
Non-knapsack side constraints	0	0	0	0	0	0
Binding side const. at optimum	12	11	15	50	11	11
Cplex time (sec)	2	43	3643	4379	12904	98440
Algorithm Performance						
Iters. to 10^{-5} optimality (sec)	6	7	9	18	6	8
Time to 10^{-5} optimality (sec)	0	0	2	23	1	7
Iters. to comb. optimality	7	8	11	18	7	11
Time to comb. optimality (sec)	0	1	3	23	2	11
Lagrangian time (sec)	0	0	1	1	1	8
Subproblem LP time (sec)	0	0	0	16	0	0

Table 2: ***Mine1 problems, I***

	Mine1 full	Mine1 double	Mine1 triple	Mine1 triple 23	Mine1 triple 24	Mine1 triple 100
Blocks	29277	58554	87831	87831	87831	87831
Parcels	29277	58554	87831	87831	87831	87831
Block arcs	985011	1970022	2955033	2955033	2955033	2955033
Periods	12	12	12	23	34	100
Destinations	2	2	2	2	2	2
Variables	489552	979104	1468656	2814924	4161192	12238800
Variables Cplex presolved	487584	—	—	—	—	—
Constraints	11849433	23698842	35548251	68053636	1000559021	295591331
Constraints Cplex presolved	11847000	—	—	—	—	—
Problem arcs	12280407	24560814	36841221	70692852	104544483	307654269
Side constraints	24	24	24	46	68	200
Non-knapsack side constraints	0	0	0	0	0	0
Binding side const. at optimum	11	15	16	33	50	151
Cplex time (sec)	—	—	—	—	—	—
Algorithm Performance						
Iters. to 10^{-5} optimality (sec)	7	10	12	11	10	9
Time to 10^{-5} optimality (sec)	45	108	192	391	608	2737
Iters. to comb. optimality	9	14	17	14	16	20
Time to comb. optimality (sec)	61	155	280	507	1018	6500
Lagrangian time (sec)	54	132	244	439	863	5496
Subproblem LP time (sec)	0	4	5	12	52	570

Table 3: ***Mine1 problems, II***