Les Cahiers du GERAD ISSN: 0711–2440

Period Decompositions for the Capacity Constrained Lot Size Problem with Setup Times

S.A. de Araujo, B. De Reyck,
Z. Degraeve, I. Fragkos, R. Jans

G–2013–76

October 2013

Period Decompositions for the Capacity Constrained Lot Size Problem with Setup Times

Silvio Alexandre de Araujo
Departamento de Matemática Aplicada
Universidade Estadual Paulista, Brazil
saraujo@ibilce.unesp.br

Bert De Reyck
Department of Management Science and Innovation
University College London, UK
& Department of Management Science and Operations
London Business School, UK
bdereyck@london.edu

Zeger Degraeve
Melbourne Business School
University of Melbourne, Australia
& Department of Management Science and Operations
London Business School, UK
zdegraeve@london.edu

Ioannis Fragkos
Department of Management Science and Innovation
University College London, UK
i.fragkos@ucl.ac.uk

Raf Jans
GERAD
& HEC Montréal, Canada
raf.jans@hec.ca

October 2013

Les Cahiers du GERAD
G–2013–76

Abstract: We study the Capacity Constrained Lot Size Problem with Setup Times (CLST). Based on two strong reformulations of the problem, we present a transformed reformulation and valid inequalities that speed up column generation and Lagrange relaxation. We demonstrate computationally how both ideas enhance the performance of our algorithm and show theoretically how they are related to dual space reduction techniques. We compare several solution methods, and propose a new efficient hybrid scheme that combines column generation and Lagrange relaxation in a novel way. Computational experiments show that the proposed solution method for finding lower bounds is competitive with other state-of-the-art approaches found in the literature. Finally, a branch-and-price based heuristic is designed and computational results are reported. The heuristic scheme compares favorably or outperforms similar approaches.

Résumé : Nous étudions un problème de planification de production avec des contraintes de capacité et des temps de mise en route. En nous basant sur deux reformulations fortes du problème, nous présentons une reformulation transformée et des inégalités valides qui accélèrent la génération des colonnes et le processus de la relaxation Lagrangienne. Nous présentons des résultats qui indiquent que ces idées améliorent la performance de nos algorithmes et nous démontrons d'une façon théorique comment ces idées sont liées aux techniques de réduction de l'espace dual. Nous comparons plusieurs méthodes de résolution, et proposons une nouvelle approche hybride efficace. Les résultats indiquent que la méthode pour trouver des bornes inférieures est compétitive avec d'autres approches dans la littérature. Finalement, une nouvelle approche heuristique est proposée et nous rapportons des résultats favorables.

1 Introduction

The Multi-Item Capacity Constrained Lot Size Problem with Setup Times (CLST) is a well-known extension of the classical single-item uncapacitated Wagner-Whitin problem. The seminal paper of Trigeiro et al. (1989) was the first to introduce CLST and demonstrate that it is considerably harder than similar problems without setup times. In their study, they proposed a per item Lagrange relaxation and designed a smoothing heuristic (TTM) that was able to find good feasible solutions quickly. It is worth noticing that many heuristics approaches have been developed (Jans and Degraeve 2007), but most of them do not provide a lower bound and therefore the solution quality cannot be assessed directly. The aim of this paper is to design a fast heuristic for CLST that provides good solutions and a strong lower bound used to assess the solution quality. Per period Danzig-Wolfe decompositions of two network reformulations of CLST are proposed. The main advantage of the proposed decompositions is that they provide a lower bound which is stronger than these of the standard and network formulations. The potential downside is their computational tractability: when computed with column generation, the already large number of variables of the network formulations and their inherent degeneracy could lead to long solution times for the restricted master programs, and only minor bound improvements per iteration. Although there exists limited computational experience with decompositions of extended formulations on this problem, evidence suggests that simplex-based solvers do not exhibit good convergence behavior (Jans and Degraeve 2004). We propose a novel, considerably faster subgradient-based hybrid scheme that combines Lagrange relaxation and column generation. This scheme gives valid lower bounds of excellent quality and outperforms pure simplex-based column generation, Lagrange relaxation and subgradient-based column generation (in which the restricted master programs are solved with subgradient optimization). Further, we enhance the performance of our algorithm by utilizing two new dual space reduction techniques. First, we show how a primal space reformulation can lead to improved performance of dual based algorithms during column generation. Second, we employ a class of valid inequalities and show how this corresponds to adding dual optimal inequalities in the dual space of the restricted master program (Ben Amor et al. 2007). The new subproblems remain tractable and the performance of the hybrid column generation scheme is improved further. The new hybrid scheme is embedded in a heuristic branch-and-price framework, designed specifically to obtain good feasible solutions fast. To achieve this, we recover a primal solution of the restricted master using the volume algorithm of Barahona and Anbil (2000) and branch on the resulting fractional setup variables. Moreover, we integrate in a customized fashion recent MIP-based heuristic approaches, such as relaxation induced neighborhoods and selective dives (Danna et al. 2005), with established ones such as the forward/backward smoothing heuristic of Trigeiro et al. (1989). Extensive computational experiments show that the branch-and-price heuristic performs very well against other competitive approaches. The remainer of this paper is organised as follows. Section 2 provides a brief literature review. Section 3 describes CLST formulations, and Section 4 their Dantzig-Wolfe decompositions. Section 5 describes customized procedures for solving the subproblems. Section 6 describes the hybrid scheme and Section 7 the branch-and-price heuristic. Finally, Section 8 presents computational results and Section 9 concludes with comments and directions for future research.

2 Literature Review

The literature on capacity constrained lot size problems is vast. A broad categorization would distinguish two research streams: theoretical and computational. Theoretical results include mainly complexity proofs, polyhedral studies and approximation algorithms and are not related directly to the scope of this work. Computational studies address the development and computational assessment of exact and heuristic approaches. With respect to the more recent research on exact approaches, Van Vyve and Wolsey (2006) suggest an approximate extended formulation based on the network reformulation of Eppen and Martin (1987) that uses a single parameter to control the trade off between the number of variables and the lower bound strength. They show that selecting small values of that parameter is sufficient to solve hard problems, especially when

a redundant row that facilitates the solver to generate cuts is added in the formulation. Degraeve and Jans (2007) discuss the structural deficiency of the decomposition proposed by Manne (1958) which only allows the computation of a lower bound for the problem, show the correct implementation of the Dantzig-Wolfe decomposition principle for CLST and develop a branch-and-price algorithm. Pimentel et al. (2010) propose three Dantzig-Wolfe decompositions of the standard formulation of CLST, and compare the performance of the corresponding branch-and-price algorithms. Specifically, they describe and compare the per-item, per-period and simultaneous per-item and per-period decompositions. Belvaux and Wolsey (2000) developed BC-PROD, a specialized branch-and-cut system for generic lot size problems. Some of the cornerstone work that is less recent are the variable redefinition approach of Eppen and Martin (1987), the use of valid inequalities (Barany et al. 1984) and the simple plant location reformulation of Krarup and Bilde (1977). Recently, many authors (Alfieri et al. 2002, Pochet and Van Vyve 2004, Denizel et al. 2008 and Süral et al. 2009) have used such alternative formulations for the CLSP with stronger linear relaxations. The seminal paper of Trigeiro et al. (1989) proposed a per-item Lagrange relaxation and designed a smoothing heuristic (TTM) that was able to find good feasible solutions quickly. Süral et al. (2009) designed a Lagrange relaxation based heuristic that outperforms TTM for problems without setup costs. Other recent heuristic approaches include among others, the cross entropy-Lagrange hybrid algorithm of Caserta and Rico (2009), the adaptive large neighbourhood search algorithm of Müller et al. (2012), the LP-based heuristic and curtailed branch-and-bound of Denizel and Süral (2006) and the iterative production estimate (IPE) heuristic of Pochet and Van Vyve (2004). Regarding metaheuristic approaches, a good review can be found in Jans and Degraeve (2007). Relevant to the current work is also the decomposition approach proposed in Jans and Degraeve (2004). The authors propose a per-period decomposition of a strong reformulation proposed in Eppen and Martin (1987). They obtain improved lower bounds, but their computational experiments show that standard computation schemes may be very time consuming for hard problems. The main contributions of the present work are (a) the development and comparison of two period Dantzig-Wolfe network reformulations for CLST; (b) the development of a methodology that circumvents the computational difficulties of extended formulations through the design of a stabilization algorithm, and the use of problem-specific inequalities in the customized algorithm that solves the subproblems; (c) the development of a state-of-the-art branch-and-price heuristic that integrates and customizes several recent advances such as the volume algorithm (Barahona and Anbil 2000, 2002), relaxation induced neighbourhoods and selected dives (Danna et al. 2005) and finally (d) the presentation of computational results that suggest the competitiveness of the proposed scheme against other approaches. The next section gives an overview of several formulations that are used throughout the paper.

3 Formulations for the Capacity Constrained Lot Size Problem

In this section we present different formulations for the capacity constrained lot size problem: the regular formulation (CL); the shortest path formulation (SP) proposed in Eppen and Martin (1987); a transformed shortest path formulation (SPt); the facility location formulation (FL) studied in Krarup and Bilde (1977); and the facility location formulation with precedence constraints (FLp).

3.1 The Regular Formulation (CL)

The regular formulation for the Capacity Constrained Lot Size Problem with Setup Times is described by the following sets, parameters and decisions variables (Trigeiro et al. 1987, Degraeve and Jans, 2007).

Sets

$I = \{1, \ldots, n\}$: Set of items, indexed by i.
$T = \{1, \ldots, m\}$: Set of periods, indexed by t.

Parameters

d_{it}: demand of item i in period t, $\forall i \in I, \forall t \in T$.
sd_{itk}: sum of demand of item i from period t till period k, $\forall i \in I, \forall t, k \in T : t \leq k$.
hc_{it}: cost of holding inventory for item i from period $t-1$ to period t, $\forall i \in I, \forall t \in T$.

sc_{it}: setup cost of item i in period t, $\forall i \in I, \forall t \in T$.
vc_{it}: production cost of item i in period t, $\forall i \in I, \forall t \in T$.
st_{it}: setup time of item i in period t, $\forall i \in I, \forall t \in T$.
vt_{it}: variable production time of item i in period t, $\forall i \in I, \forall t \in T$.
M_{it}: big-M quantity, defined as $M_{it} = \min\{sd_{itm}, \frac{cap_t - st_{it}}{vt_{it}}\}, \forall i \in I, \forall t \in T$.
cap_t: time capacity in period t, $\forall t \in T$.

Decision Variables

x_{it}: production quantity of item i in period t, $\forall i \in I, \forall t \in T$.
s_{it}: inventory quantity of item i at the beginning of period t, $\forall i \in I, \forall t \in T \cup \{m+1\}$.
y_{it}: =1 if setup for item i in period t, =0 otherwise, $\forall i \in I, \forall t \in T$.

The mathematical formulation of CLST is then as follows:

$$\min \quad \sum_{i\in I}\sum_{t\in T} sc_{it}y_{it} + \sum_{i\in I}\sum_{t\in T} vc_{it}x_{it} + \sum_{i\in I}\sum_{t\in T} hc_{it}s_{it} \tag{1}$$

$$\text{s.t.} \quad s_{it} + x_{it} = d_{it} + s_{i,t+1} \qquad \forall i \in I, \forall t \in T \tag{2}$$

$$x_{it} \le M_{it}y_{it} \qquad \forall i \in I, \forall t \in T \tag{3}$$

$$\sum_{i\in I} st_{it}y_{it} + \sum_{i\in I} vt_{it}x_{it} \le cap_t \qquad \forall t \in T \tag{4}$$

$$x_{it}, s_{it} \ge 0, s_{i,m+1} = 0, y_{it} \in \{0,1\} \qquad \forall i \in I, \forall t \in T \tag{5}$$

The objective function (1) minimizes the setup cost, the variable production cost, the inventory holding cost and initial inventory cost. Constraints (2) are the demand constraints: inventory carried over from the previous period and production in the current period can be used to satisfy current demand and build up inventory. As in Vanderbeck (1998), we deal with possible infeasible problems by allowing for initial inventory (s_{i0}) which is available in the first period at a large cost, fc_i. There is no setup required for initial inventory. Constraint (3) forces the setup variable to one if any production takes place in that period. Next, there is a constraint on the available capacity in each period (4). Finally, we have the non-negativity and integrality constraints (5) and the ending inventory is set to zero.

3.2 The Shortest Path Formulation (SP)

Next, model (1)–(5) is reformulated using the variable redefinition approach of Eppen and Martin (1987).

Define the following parameters:

cv_{itk}: total production and holding cost for producing item i in period t to satisfy demand for the periods t until k, $cv_{itk} = vc_{it}sd_{itk} + \sum_{s=t+1}^{k}\sum_{u=t}^{s+1} hc_{iu}d_{is}$,
ci_{it}: total production and holding cost for initial inventory for item i to satisfy demand from period 1 up to period t, $ci_{it} = fc_i sd_{i1t} + \sum_{s=2}^{t}\sum_{u=1}^{s-1} hc_{iu}d_{is}$.

We also define the following new variables:

z_{itk}: fraction of the production plan for item i where production in period t satisfies demand from period t to period k, $x_{it} = \sum_{k=t}^{m} sd_{itk}z_{itk}, \forall i \in I, \forall t \in T$.
p_{it}: fraction of the initial inventory plan for item i where demand is satisfied for the first t periods, $s_{i0} = \sum_{t=1}^{m} sd_{i1t}p_{it}, \forall i \in I$.

The shortest path formulation (SP) is then as follows:

$$\min \quad \sum_{i\in I}\sum_{t\in T}(sc_{it}y_{it}+ci_{it}p_{it})+\sum_{i\in I}\sum_{t\in T}\sum_{k=t}^{m}cv_{itk}z_{itk} \tag{6}$$

$$\text{s.t.} \quad \sum_{k=1}^{m}(z_{i1k}+p_{ik})=1 \qquad \forall i\in I \tag{7}$$

$$\sum_{j=1}^{t-1}z_{ijt-1}+p_{it-1}=\sum_{k=t}^{m}z_{itk} \qquad \forall i\in I,\forall t\in T\backslash\{1\} \tag{8}$$

$$\sum_{i\in I}st_{it}y_{it}+\sum_{i\in I}\sum_{k=t}^{m}vt_{it}sd_{itk}z_{itk}\leq cap_t \qquad \forall t\in T \tag{9}$$

$$\sum_{k=t}^{m}z_{itk}\leq y_{it} \qquad \forall i\in I,\forall t\in T \tag{10}$$

$$y_{it}\in\{0,1\},p_{it}\geq 0 \qquad \forall i\in I,\forall t\in T \tag{11}$$

$$z_{itk}\geq 0 \qquad \forall i\in I,\forall t,k\in T:k\geq t \tag{12}$$

The objective function (6) minimizes the total costs. Constraints (7) and (8) define the flow balance constraints of each node (i,t), which ensure that demand is satisfied. For each item, a unit flow is sent through the network, imposing that its demand has to be satisfied without backlogging. The capacity constraints (9) limit the sum of the total setup times and production times to the available capacity in each period. Constraint (10) defines the setup forcing for each item. Finally, setup decisions are binary (11).

3.3 Transformed Shortest Path Formulation (SPt)

We transform formulation SP to an equivalent formulation, which leads to a better convergence of the Lagrange multipliers due to dual space reduction, as explained below. The idea is to substitute, for each item, the demand balance constraint of period t with the sum of the demand balance constraints of the first t periods. Doing so, constraints (7) and (8) are replaced with:

$$\sum_{k=t}^{m}p_{ik}+\sum_{j=1}^{t}\sum_{k=t}^{m}z_{ijk}=1 \qquad \forall i\in I,\forall t\in T \tag{13}$$

We denote the resulting transformed formulation SPt. Eppen and Martin (1987) showed that their network reformulation corresponds to a shortest path problem defined on an acyclic graph. The demand constraints (7) and (8) express the flow balance in each node of this graph. Equation (13) imposes that the flow of the cut-set defined by the $s-t$ cut $C_t=(\{0,\ldots,t-1\}\cup\{t,\ldots,m\})$ is equal to one. Figure 1 illustrates a single-item four period example with no initial inventory. Equation (8) for period 3 reads $z_{12}+z_{22}=z_{33}+z_{34}$, which corresponds to the flow balance of node 2. For the same period, (13) sets the total flow through the edges connecting the sets of nodes {0, 1, 2} and {3, 4} equal to one, i.e., $1=z_{13}+z_{14}+z_{23}+z_{24}+z_{33}+z_{34}$.

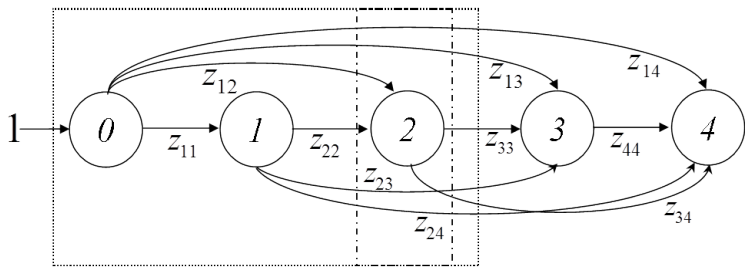

Figure 1: A single-item four period example with no initial inventory

The idea behind this transformation is that of *dual space reduction*. Let D be the feasible dual space associated with constraints (7) and (8), and D' the one associated with constraints (13). If v_{i1} and v_{it} are the dual values of (7) and (8) and the dual values of (13), then

$$D = \left\{ v_{it} \in \Re \;\middle|\; \begin{array}{c} v_{it} - v_{i,k+1} \leq cv_{itk}\ \forall i \in I, t, k \in T : t \leq k < m \\ v_{i1} - v_{it+1} \leq ci_{it}\ \forall i \in I, t \in T \backslash \{m\} \\ v_{it} \leq cv_{itm}\ \forall i \in I, t \in T \\ v_{i1} \leq ci_{im}\ \forall i \in I \end{array} \right\};$$

$$D' = \left\{ \bar{v}_{it} \in \Re \;\middle|\; \begin{array}{l} \sum_{l=1}^{t} \bar{v}_{il} \leq \min\{ci_{it}, cv_{i1t}\}\ \forall i \in I, t \in T \\ \sum_{l=t}^{k} \bar{v}_{il} \leq cv_{itk}\ \forall i \in I, t, k \in T : k \geq t \end{array} \right\}$$

Theorem 1 $D' \subseteq D$.

Proof. See Appendix. □

Transformed formulations like the SPt imply a denser form of the constraint matrix, which, in the context of the simplex method, might result in more pivots and therefore be less efficient. Since our algorithm works in the dual space, it is interesting to investigate computationally if the dual space reduction of subsystem (7) and (8) is beneficial.

3.4 Facility Location Formulation (FL)

CL can be reformulated using variables originally employed in Facility Location problems. To the best of our knowledge, the first paper that used facility location variables to formulate lot size models was the work of Krarup and Bilde (1977). The resulting model (FL) is described below.

Parameters:

cs_{itk}: total production and holding cost for producing item i in period t to satisfy demand of period k, $cs_{itk} = (vc_{it} + \sum_{u=t}^{k-1} hc_{iu})d_{ik}$

cu_{it}: initial inventory and holding cost for item i, to satisfy demand in period t, $cu_{it} = (fc_i + \sum_{u=1}^{t-1} hc_{iu})d_{it}$.

We also define the following variables:

w_{itk}: fraction of demand for item i in period k that is satisfied by production in period t, $x_{it} = \sum_{k=t}^{m} d_{ik} w_{itk}$, $\forall i \in I, \forall t \in T$,

sp_{it}: fraction of demand for item i in period k that is satisfied by initial inventory, $s_{i0} = \sum_{t=1}^{m} sp_{it} d_{it}, \forall i \in I$.

The facility location (FL) reformulation is then as follows:

$$\min \quad \sum_{i \in I} \sum_{t \in T} (sc_{it} y_{it} + cu_{it} sp_{it}) + \sum_{i \in I} \sum_{t \in T} \sum_{k=t}^{m} cs_{itk} w_{itk} \tag{14}$$

$$\text{s.t.} \quad sp_{it} + \sum_{k=1}^{t} w_{ikt} = 1 \qquad \forall i \in I, \forall t \in T \tag{15}$$

$$\sum_{i \in I} st_{it} y_{it} + \sum_{i \in I} \sum_{k=t}^{m} vt_{it} d_{ik} w_{itk} \leq cap_t \qquad \forall t \in T \tag{16}$$

$$w_{itl} \leq y_{it} \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{17}$$

$$y_{it} \in \{0, 1\}, sp_{it} \geq 0 \qquad \forall i \in I, \forall t \in T \tag{18}$$

$$w_{itk} \geq 0 \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{19}$$

The objective function (14) minimizes the total cost, which consists of the setup cost, the aggregated production and holding costs, and the initial inventory and holding costs. Equations (15) correspond to the flow balance constraints (2) and state that period t demand must be covered by a combination of initial inventory and production in periods $\{1,\ldots,t\}$. The capacity constraints (16) are in exact correspondence with (4). The setup constraints (17) do not allow any production in period t unless a setup is done and the non-negativity conditions (18) and (19) complete the FL formulation.

3.5 Facility Location Formulation with Precedence Constraints (FLp)

Many extended formulations are often degenerate or have multiple optimal solutions. This is also the case with the facility location formulation (14)–(19). The existence of multiple solutions in the extended formulation may degrade the efficiency of column generation. Hence, the addition of valid inequalities in the subproblem that cut off some of the primal space containing alternative optimal solutions, may lead to improved convergence, as it prevents the subproblem from generating columns that describe alternative solutions. A class of such valid inequalities is described below.

Observation 2 *There exists an optimal solution of FL with* $w_{it,k-1} \geq w_{itk}$ *for all* $i \in I, t, k \in T : t{+}1 \leq k \leq m$ *and* $sp_{it-1} \geq sp_{it}$ *for all* $i \in I$ *and* $t \in T\backslash\{1\}$.

We will refer to the above valid inequalities as *Precedence Constraints*. Observation 2 is used in Wolsey (1989) in his study of the facility location formulation in the context of lot size problems with start-up costs and no capacity constraints. A short proof can be found in the appendix. FL is not a minimal image of the $conv(CL)$ in the sense that there exists a subset of $conv(FL)$ that is the image of all extreme points of $conv(CL)$. This is important because $conv(CL)$ might have more extreme points when the precedence constraints are not considered. The precedence constraints $w_{it,k-1} \geq w_{itk}$ can be used alongside the setup forcing $w_{itt} \leq y_{it}$ in place of $w_{itk} \leq y_{it}$ for all $i \in I, t, k \in T : t+1 \leq k$. The FL formulation with the primal valid inequalities is written as:

$$\min \quad \sum_{i\in I}\sum_{t\in T}(sc_{it}y_{it} + cu_{it}sp_{it}) + \sum_{i\in I}\sum_{t\in T}\sum_{k=t}^{m} cs_{itk}w_{itk} \tag{20}$$

$$\text{s.t.} \quad w_{itt} \leq y_{it} \qquad \forall i \in I, \forall t \in T \tag{21}$$

$$w_{it,k-1} \geq w_{itk} \qquad \forall i \in I, \forall t, k \in T : k > t \tag{22}$$

$$(15) - (16), (18) - (19) \tag{23}$$

The idea behind the inclusion of constraints (22) is that of primal space reduction. Although it is reasonable to assume that a decomposition scheme that uses (22) in the subproblem would deliver an improved lower bound compared to not including them, we show that this is not the case. Rather, (22) accelerates the column generation convergence because the feasible space of eligible columns is reduced.

4 Period Dantzig-Wolfe Decompositions

4.1 Formulations

We develop period decompositions for formulations SP, SPt, FL and FLp. We denote each decomposition formulation by appending /P to the original notation. For example, SP/P denotes the period decomposition of the shortest path formulation. The demand balance constraints are the complicating constraints and the capacity and setup forcing constraints of each period form the period subproblems. We focus on period decompositions of network formulations because the resulting relaxations provide improved lower bounds compared to the linear programming (LP) relaxation of the extended formulations, since the period capacity polytope does not have the integrality property (Geoffrion, 1974).

The formulation of the per period decomposition of SP, SP/P, is described in detail in Jans and Degraeve (2004). The formulation of the per period decomposition of SPt, SPt/P is very similar to SP/P and we skip

it for brevity. Instead, we present the per period decompositions of FL and FLp. To this end, let us define by S_t the index set of extreme point production plans of the subproblem for period t, i.e.,

$$S_t := \left\{ q \in extr\left(conv\left\{ (w_{itk}, y_{it})_{i\in I, k\geq t} \mid (16) - -(19) \right\} \right) \right\}$$

We associate a decision variable with the fraction of the extreme point q of subproblem t that is used in a feasible solution. If we denote by $(\bar{w}_{itkq}, \bar{y}_{itq})$ the components of point q, its cost can be written as $ct_{tq} = \sum_{i\in I} (sc_{it}\bar{y}_{itq} + \sum_{k=t}^{m} cs_{itk}\bar{w}_{itkq})$. Then the master program FL/P can be formulated as follows:

$$\min \quad \sum_{t\in T}\sum_{q\in S_t} ct_{tq} wt_{tq} + \sum_{i\in I}\sum_{t\in T} cu_{it} sp_{it} \tag{24}$$

$$\text{s.t.} \quad sp_{it} + \sum_{l=1}^{t}\sum_{q\in S_l} \bar{w}_{iltq} wt_{lq} = 1 \qquad \forall i \in I, \forall t \in T \qquad [\pi_{it}] \tag{25}$$

$$\sum_{q\in S_t} w_{tq} = 1 \qquad \forall t \in T \qquad [\mu_t] \tag{26}$$

$$y_{it} = \sum_{q\in S_t} \bar{y}_{itq} wt_{tq} \qquad \forall i \in I, \forall t \in T \tag{27}$$

$$wt_{tq} \geq 0, \;\; y_{it} \in \{0,1\} \qquad \forall i \in I, \forall q \in S_t, \forall t \in T \tag{28}$$

The objective function (24) minimizes the total cost of the initial inventory and the cost of the production plans chosen in each period. Constraints (25) model demand and correspond to constraints (2) in the standard formulation. The convexity constraints (26) and the nonnegativity constraints (28) enforce a convex combination. The setup variables definition is given in (27). The integrality must be imposed on the original setup variables (28). The constraint coefficient parameters $(\bar{w}_{itkq}, \bar{y}_{itq})$ are defined by the subproblem extreme points. The subproblem objective function minimizes the reduced cost over the extreme points. Specifically, the period t subproblem (SUB) reads:

$$\min \quad \sum_{i\in I} sc_{it} y_{it} + \sum_{i\in I}\sum_{k=t}^{m} (cs_{itk} - \pi_{ik}) w_{itk} - \mu_t \tag{29}$$

$$\text{s.t.} \quad \sum_{i\in I} st_{it} y_{it} + \sum_{i\in I}\sum_{k=t}^{m} vt_{it} d_{ik} w_{itk} \leq cap_t \qquad \forall t \in T \tag{30}$$

$$w_{itk} \leq y_{it} \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{31}$$

$$y_{it} \in \{0,1\} w_{itk} \geq 0 \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{32}$$

The period decomposition of the formulation FLp has the same master problem. However, the subproblem, denoted SUBp, is different because it includes the precedence constraints (21) and (22) in place of (31).

4.2 Lower Bounds

It is interesting to investigate the lower bound quality of the proposed decompositions. To this end, let $\bar{v}_{FL/P}$ be the optimal objective value of the linear relaxation of the decomposed model FL/P (24)–(28). Also, let $\bar{v}_{FLp/P}, \bar{v}_{SP/P}$ and $\bar{v}_{SPt/P}$ be the corresponding lower bounds obtained by the linear relaxation of FLp/P, SP/P, and SPt/P respectively. To obtain a first result, we will need the following definition and lemma.

Definition 3 *Let D be the dual space polyhedron of a linear program and D^* be the set of its optimal solutions. Also, assume a new set of constraints $E^T\pi \leq \mathbf{e}$ that cuts off part of the dual space. If $D^* \subseteq \{\pi : E^T\pi \leq \mathbf{e}\}$, then $E^T\pi \leq \mathbf{e}$ is called a set of* dual-optimal inequalities. *If $E^T\pi \leq \mathbf{e}$ cuts off a nonempty set of dual-optimal solutions but is still satisfied by at least one dual-optimal solution, it is called a set of deep dual-optimal inequalities. (Ben Amor et al. 2007).*

Lemma 4 *Using subproblem SUBp is equivalent to using subproblem SUB and imposing constraints Using subproblem SUBp is equivalent to using subproblem SUB and imposing constraints*

$$(cs_{itk} - \pi_{ik})\, d_{i,k+1} \geq (cs_{it,k+1} - \pi_{i,k+1})\, d_{ik} \;\; \forall i \in P, \;\; t, k \in T : t \leq k \leq m-1 (*)$$

in the dual space of the LP master of FL/P. Moreover, these constraints are deep dual optimal inequalities.

Proof. It suffices to show that SUB has the same optimal solution as SUBp whenever holds. Note that can be added to both sides of the inequality, but is omitted since they cancel each other out. Therefore, states that the profit to weight ratio of should be greater than that of. It follows that there exists an optimal solution of the LP relaxation of SUB when holds which is also optimal for the LP relaxation of SUBp. Also, SUB has the same structure after branching, so if holds, the precedence constraints hold at any node of the branch-and-bound tree, and therefore at an integer optimal solution. Hence, from construction, imply the precedence constraints, that do not cut-off all optimal solutions. Their equivalence with the precedence constraints and definition 1 imply that they are deep dual optimal inequalities. □

Proposition 5

$$\bar{v}_{SP/P} = \bar{v}_{SPt/P} = \bar{v}_{FLp/P} = \bar{v}_{FL/P}$$

Proof. First, note that because the corresponding subproblems have the same set of extreme points, and the linking constraints of SPt/P are linear combinations of those of SP/P. Second, consider the subproblem SUBp. The linear transformation $z_{itt} = w_{tt}$; $z_{itk} = w_{itk} - w_{it,k-1}$, $k > t$ maps every extreme point (y_{it}, w_{itk}) of SUBp to a unique extreme point (y_{it}, z_{itk}) of the SP/P subproblem. Using this transformation in FLp/P, the resulting model is exactly SPt/P. On the other hand, every feasible solution of the SPt/P subproblem can be mapped to FLp/P with the inverse transformation, i.e., $w_{itt} = z_{tt}$; $w_{itk} = \sum_{l=t}^{k} z_{itl}$. Hence, $\bar{v}_{SPt/P} = \bar{v}_{FLp/P}$. Finally, we show that $\bar{v}_{FLp/P} = \bar{v}_{FL/P}$. To this end, notice that $\bar{v}_{FLp/P} \geq \bar{v}_{FL/P}$, since adding constraints to the subproblem can never lead to a worse bound. Denote by $\bar{v}'_{FL/P}$ the optimal value obtained when solving the dual of FL/P amended with the dual restrictions $*$. Then $\bar{v}'_{FL/P} = \bar{v}_{FLp/P}$ from lemma 1 and $\bar{v}'_{FL/P} \leq \bar{v}_{FL/P}$, since adding cuts to the dual of a primal minimization problem can never increase its optimal value. The result follows. □

Interestingly, the above lower bound is stronger than the one obtained by the simultaneous per item and per period decomposition of formulation CL studied by Pimentel et al. (2010). Let us denote the latter lower bound by $\bar{v}_{CL/P/I}$.

Proposition 6

$$\bar{v}_{CL/P/I} \leq \bar{v}_{SP/P}$$

Proof. See appendix. □

5 Solving the Subproblems

This section describes two fast customized algorithms used to solve subproblems SUB and SUBp. Note that since the feasible solutions of FLp are in exact correspondance with those of SPt, and since the subproblem of SP and SPt is common, solving SUB and SUBp implies a solution for all four subproblems.

5.1 A Customized Algorithm for SUB

For SUB, the following simple observation plays a key role in the subsequent solution approach.

Observation 7 *In an optimal solution of the LP relaxation of SUB there exist some $k_i \geq t$ such that $w_{itk_i} = y_{it}$, $\forall i \in I$.*

This implies that the subset of tight inequalities (31) can be used to substitute out the y_{it} variables in (30). As a result, the LP relaxation of SUB reduces to a linear knapsack problem which admits a greedy solution, similar to the one proposed in Holmberg and Yuan (2000). It follows that an optimal solution of SUB has fractional continuous variables for at most one item. The following algorithm identifies the subset of tight inequalities in Phase I and solves a regular linear knapsack problem in Phase II.

Algorithm 1 Algorithm for Subproblem SUB

Phase I

Initialize $r_{itk} \leftarrow \frac{cs_{itk}-\pi_{ik}}{vt_{it}d_{ik}}, \forall i \in I, \forall t, k \in T : t \geq k$

Initialize $K_i = \emptyset, P = \{t, \ldots, m\}, Best_i = False, \forall i \in I$

for $i \in I$ **do**
  while $Best_i = False$ **do**
    $r_{it} \leftarrow \min_{k \in P \setminus K_i} \{r_{itk}\}$; $k_i \leftarrow \arg r_{it}$ \\ Calculate min ratio
    $K_i \leftarrow K_i \cup \{k_i\}$
    $r_{it} \leftarrow \frac{sc_{it}+\sum_{k\in K_i}(cs_{itk}-\pi_{ik})}{st_{it}+\sum_{k\in K_i} vt_{it}d_{ik}}$ \\ Update set/ratio of merged periods
    if $r_{it} > \min_{k \in P \setminus K_i} \{r_{itk}\}$ or $K_i = P$ **then** \\ If ratio is min, exit
      $Best_i = True$
    end if
  end while
end for

Phase II

$csm_{it} := sc_{it} + \sum_{k \in K_i} (cs_{itk} - \pi_{ik})$

$vtm_{it} := st_{it} + \sum_{k \in K_i} vt_{it} d_{ik}$

Solve a linear Knapsack with weights $\{ vtm_{it}; vt_{it}d_{ik}\ \forall k \notin K_i, i \in I \}$ and costs $\{ csm_{it};\ cs_{itk} \forall k \notin K_i, i \in I \}$. Let $\{ wm_{it};\ w_{itk}\ \forall k \notin K_i, i \in I \}$ denote the optimal solution.

$y_{it} \leftarrow wm_{it}, \forall k \in K_i, i \in I$

Return $(y_{it}, w_{itk})\ \forall i \in I, \forall k \geq t$

Phase I identifies which of the variable upper bound constraints (31) are satisfied as equalities. Then, the value of the setup variable is set equal to the value of the production variable with the most attractive cost-to-weight ratio. If the new ratio is still the most attractive after the substitution, *Phase I* terminates. If it is not, another constraint (31) is tight for the same item, and the corresponding substitution is made. *Phase II* solves a linear knapsack problem where all w_{itk} with $k \in K_i$ are equated with y_{it} and substituted out. Note that adjusting the algorithm to tackle items with zero setup time or cost is straightforward. Due to observation (7), algorithm SUB produces an optimal solution of the SUB linear relaxation.

After solving the relaxation, a depth-first branch-and-bound algorithm is implemented. Branching is needed only when some wm_{it} was the last variable to enter the knapsack at fractional level, and therefore $y_{it} < 1$. Thus, at most one setup variable is fractional. In addition, branching does not change the problem structure because it either fixes w_{itk} to zero when $y_{it} = 0$ or reduces the problem capacity by st_{it} and fixes sc_{it} in the objective, when $y_{it} = 1$. For this reason, algorithm SUB can be called at each node of the branch-and-bound tree, with different input data. Computational experiments confirm the superiority of this approach compared to simplex-based branch-and-bound.

5.2 A Customized Algorithm for SUBp

The addition of the precedence constraints changes the structure of subproblem SUB and the previous algorithm cannot be employed. However, we can take advantage of the fact that each solution of SUBp corresponds to a solution of the SP/P subproblem, whose relaxation is a linear multiple choice knapsack problem (Jans and Degraeve 2004). Therefore, we solve the latter subproblem and then map back the solution to SUBp. A customized depth-first branch-and-bound algorithm is also developed for this subproblem. The difference with the usual multiple choice knapsack algorithms is that when some item is branched to 1, the dominated periods change (Pisinger 1995). We use an efficient variant of the algorithm suggested by Graham

(1972) to construct and update the convex hulls, and then solve the linear relaxations using the algorithm of Sinha and Zoltners (1979).

6 Solving the Master Problem: A New Hybrid Algorithm

In many practical cases the convergence of column generation can be slow as a tailing-off effect is observed (Vanderbeck and Savelsbergh 2006). This behavior is mainly due to high degeneracy of the restricted master. Specifically, when the restricted master linear programs are solved with a simplex algorithm, two problems may arise. First, the solution time can be large, and second, the optimal dual values that are passed to the subproblems may be of bad quality. The latter issue stems from the fact that degeneracy implies multiple dual optimal solutions, and simplex-based solvers typically return an optimal extreme point of the dual polyhedron. Such corner points may not provide an accurate representation of the optimal face, whereas a point that lies within the optimal face may lead to faster convergence (Lübbecke and Desrosiers 2005 and Mitchell 2005). To tackle this problem, some authors have proposed to solve the linear programs with an interior point method, or to employ stabilization techniques, such as those described in du Merle et al. (1999), Ben Amor et al. (2005), Elhallaui et al. (2005) and Gondzio et al. (2013). Recently, Subramanian and Sherali (2008) gave insights in the way that poor dual values result in the generation of poor quality columns, and designed a subgradient-based scheme to solve large set-partitioning problems arising in aircraft crew scheduling. Columns coming from SUB or SUBp can have some fractional components. Therefore, the row aggregation technique of Elhallaui et al. (2005) cannot be used, since it is developed for pure set-partitioning problems.

Huisman et al. (2005) discuss how the relationship between Dantzig-Wolfe decomposition and Lagrange relaxation can lead to the development of improved algorithms that combine the strengths of both methods. They explore two different ways to combine column generation and Lagrange relaxation. First, Lagrange relaxation can be used to solve the master, by dualizing the linking constraints, and the resulting near-optimal dual values can be used to price out the subproblems. Second, Lagrange relaxation can be employed within column generation, exploiting the fact that both methods solve the same subproblem. Specifically, the restricted master programs are solved with simplex, and next the master dual prices are used as a starting point for a number of Lagrange relaxation iterations, during which several negative reduced cost columns can be found and added to the master at once. We propose a new method, which is a combination of the two previously discussed methods. Specifically, it uses Lagrange relaxation to solve the master programs, but also to generate new columns. Figure 2 gives a schematic representation of the algorithm.

We initialize the column generation process using the heuristic of Trigeiro et al. (1989). This heuristic returns a feasible solution, a lower bound and a set of lagrangian dual prices of the capacity constraints. Using the dual of the relaxation of SP or FL, we can then calculate the starting dual prices π_{it}. Also, the feasible solution $(y_{it}, x_{it})^F$ is split into per period columns of the SPt or FLp master programs. To perform this operation, it is necessary to transform the $(y_{it}, x_{it})^F$ solution to the space of the extended formulation variables. This can be done by fixing the setup variables to their given values and solving the corresponding formulation, which is a shortest path problem. Next, the Lagrange relaxation subroutine uses the modified subgradient method of Crowder (1976) to update the dual prices π_{it}. Moreover, for each updated set of dual prices we check if the subproblem solutions price out, using the last obtained dual prices of the convexity constraints, μ_t. The columns that price out are added to the master. Existing columns with a high reduced cost are removed from the master and kept into a separate column pool. Then, the restricted master problem is solved using subgradient optimization by relaxing the linking constraints (25) in the master (Huisman et al. 2005). The scheme iterates until no column prices out, where we invoke the volume algorithm to obtain an approximate primal solution. In this way, we are able to make better informed branching decisions in the subsequent branch-and-price phase.

Contrary to existing hybrid algorithms (Barahona and Jensen, 1998, Degraeve and Peeters 2003, Degraeve and Jans 2007), this scheme relies exclusively on Lagrange relaxation. It has the benefits that it (a) avoids the degeneracy issues of the simplex method and therefore exibits faster convergence and (b) returns good quality dual prices that help the column generation convergence. Note that the intermediate value of the

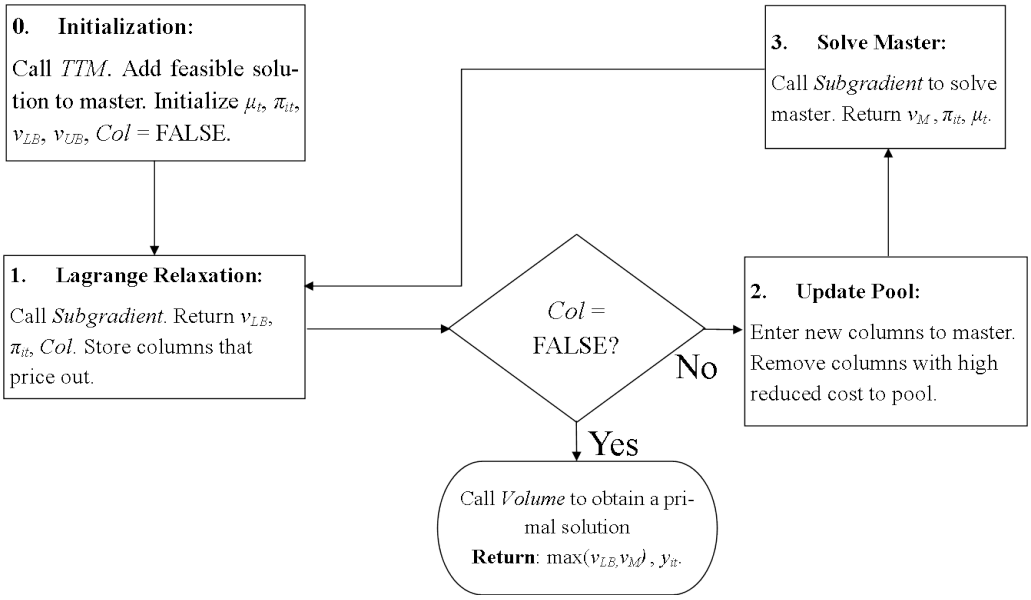

Figure 2: The main building blocks of the new hybrid algorithm

master programs, v_M, is not necessarily a valid upper bound of the column generation lower bound, as it is calculated using lagrange relaxation. It is a valid lower bound (Huisman et al. 2005) when no column prices out, and therefore the scheme returns the best bound that is obtained by the master and by Lagrange relaxation.

7 A Branch-and-Price Heuristic

We have combined the hybrid column generation algorithm with heuristic techniques, and integrated them in an enumeration scheme. Our experiments use formulation FLp. The goal is to find good feasible solutions fast by exploiting the structure of the network formulation and its subproblem. It is worth noting that the development of an exact approach is possible, but the nature of the lower bounding process would make it inefficient. This is because branching is needed even when the primal solution returned by the volume algorithm is integral. Therefore, nodes can be implicitly enumerated only by dominance and infeasibility pruning, and the resulting trees are large. Since a heuristic search is performed in each node and the branching decisions dictate the search space in the tree, the overall scheme can be seen as integration of exploration and exploitation heuristics. The next section describes the heuristics employed in each node of the tree.

7.1 Node Heuristics

We employ a successive rounding heuristic that uses a smoothing subroutine, which in turn employs some operations of the heuristic of Trigeiro et al. (1989). The successive rounding heuristic gets as input the primal setup variables produced by the volume algorithm, determines a threshold level and fixes the setup variables below and above that threshold to 0 and 1, respectively. The process iterates for increasing threshold values. Typically, feasible solutions obtained in this way follow a U-shaped fashion (Degraeve and Jans 2007). Therefore, when the solution quality starts to degrade the heuristic terminates. Starting from this solution, the smoothing heuristic that searches for an improved feasible solution is applied. The smoothing part starts from the last period and tries to push production backwards, such that the Lagrange costs are minimized. This is done iteratively until the first period, and if any capacity constraints are violated it performs a similar forward operation to recover feasibility. Thus, the rounding/smoothing heuristic scheme searches the feasible space by modifying incrementally the proposed setup schedules and production plans and can be considered an exploitation heuristic.

7.2 Node Lower Bounds

At each node we solve the restricted master problem using subgradient optimization, without generating any columns. If its objective value is lower than the incumbent value, we recover an approximate primal solution using the volume algorithm and use that solution to implement a branching strategy. Else, if the restricted master objective value is higher than the incumbent value, the hybrid algorithm is invoked to generate columns. During the hybrid algorithm iterations, a valid non-decreasing node lower bound is recorded. If that lower bound comes to exceed the incumbent value, the node is pruned and the columns generated at this node are deleted. When the hybrid algorithm terminates and the lower bound is below the incumbent value, branching is needed. In this case, the volume algorithm is invoked to recover an approximate primal solution of the restricted master, which is then used for branching. When the volume algorithm returns all-integer setup variables, we fix these variables and solve the resulting LP problem using formulation CL, which is a generalized network flow problem for fixed setups (Degraeve and Jans 2007). To ensure that the approximate primal solution is of good quality, additional columns are generated when the maximum violation of the linking constraints is higher than some predefined tolerance (we used 0.05). More details about this procedure can be found on the appendix of this chapter.

7.3 Tree Exploration Strategies

The below procedure summarizes the main building blocks the heuristic Branch and Price algorithm.

Algorithm 2 Summary of the heuristic branch-and-price procedure HBP

Inputs: *LowerBound, UpperBound, CGSolution, Incumbent*
Outputs: *LowerBound, UpperBound, Incumbent, NodesExplored*

1: $NodesExplored \leftarrow 0; Pass \leftarrow 0; NodeLimit \leftarrow 100;$ \\ Initializations
2: **Call** $RINS\_Variable\_Fixing(RINSDepth)$ \\ Fix Variables based on RINS. Store how many variables are fixed in RINSDepth
3: **Call** $Branch\_And\_Price$ \\ Terminates when $NodesExplored = NodeLimit$ or when $LowerBound = UpperBound$
4: **while** $Time < TimeLimit$ **or** $LowerBound = UpperBound$ **do**
5: $Pass = Pass + 1$
6: $NodeLimit = NodeLimit + (Pass1) * 50$ \\ Expand the explored space during each pass
7: **Call** $Unfix\_Variables(RINSDepth)$ \\ Free the variables that are fixed at the first RINSDepth levels of the tree
8: **Call** $Update\_Fixed\_Variables$ \\ Fix the remaining fixed variables to the values of the incumbent
9: **Call** $Branch\_And\_Price$ \\ Search again, in a larger neighbourhood
10: **end while**

The exploration of the search space is done in two ways. First, we employ the concept of *relaxation induced neighborhoods search* (RINS) (Danna et al. 2005). Although other MIP-based techniques could be employed, such as local branching (Fischetti and Lodi 2003), most of them destroy the subproblem structure. After solving the root node, a variable fixing phase follows. We fix to 1 (0) the setup variables that are 1 (0) in the incumbent and more (less) than 0.8 (0.2) in the approximate primal solution of the master program, given by the volume algorithm. Then branching is applied. Following Van Vyve and Wolsey (2006), we branch on the earliest period and on the most fractional item. The intuition behind this choice is that branching decisions in earlier periods are more important than in later periods because they are more likely to influence the subsequent production flow. During the first pass, we branch to 0 if both the incumbent has the branching variable at 0 and its fractional value at the root note is less than 0.2, else we branch to 1. The bias against 0-branches comes from the fact that when the capacity constraints are tight, it is more likely to produce more than needed in earlier periods, and therefore a setup is necessary.

Notice that the subtree invoked from RINS is guaranteed to have a leaf node that is at least as good as the incumbent, because the incumbent itself is under that subtree. If a better feasible solution exists, it is likely to be found at the early levels of the subtree from the rounding/smoothing heuristic. For this reason, we explore the subtree induced by the RINS heuristic only partially, by imposing a node limit. A

potential downside of exhaustively exploring this subtree, is that if some of the fixing decisions are not part of the optimal solution, the algorithm could stall or terminate without a significant improvement. Hence, it is neccesary to modify the search space. To do this, we note that at each node the fixed variables are either fixed by the RINS fixing decisions, or by branching. To explore a different but promising part of the setup variable space, we free the variables fixed by RINS and fix the variables that were fixed by branching to the values of the incumbent solution. The previously free variables remain free. In a sense, this scheme tries to replace dive decisions with branching decisions, while remaining close to the neighborhood of the incumbent. The same idea is applied iteratively: a number of nodes in each subtree is explored, the variables fixed early up in the free become free and the variables fixed from Branch and Price are fixed again to the value of the incumbent.

8 Computational Experiments

Computational experiments were performed on a 2.0 GHz / 2 GB machine. The CPU times listed in all tables refer to the time that the incumbent was found. Detailed results of all experiments can be found in the appendix of this chapter.

8.1 Solving the Root Node

To demonstrate the strength of the lower bound of the proposed hybrid scheme, we employ data instances from Trigeiro et al. (1989). These instances are among the hardest ones and have been tested extensively (Belvaux and Wolsey 2000, Van Vyve and Wolsey 2006, Degraeve and Jans 2007, Jans and Degraeve 2004 and Pochet and Van Vyve 2004). We use 4 different formulations and 3 solution methods. Table 1 shows the nomenclature of different combinations of formulations (period decomposition of the Facility Location formulation FL/P, period decomposition of the Facility Location formulation with Precedence constraints FLp/P, period decomposition of the Shortest Path formulation SP/P and period decomposition of the Transformed Shortest Path formulation SPt/P) and solution methods (Lagrange Relaxation LR, Approximate Column Generation ACG and the Hybrid Algorithm HB). In ACG, the master is solved with subgradient optimization. Each subproblem returns one column per iteration, as in standard column generation. ACG is listed to demonstrate how the hybrid scheme performs against a Lagrange-based implementation of column generation.

Table 1: Nomenclature of combinations of formulations and solution methods

	Method Used to Calculate the Lower Bound		
Formulation	**LR**	**ACG**	**HB**
FL/ P	LR	ACG	HB
FLp/P	LRp	ACGp	HBp
SP/P	SPP-LR	SPP-ACG	SPP-HB
SPt/P	SPtP-LR	SPtP-ACG	SPtP-HB

Table 2 compares the CPU time in seconds of combinations of formulations and solution methods. For the shake of brevity, we do not present full results for formulations SP/P and SPt/P, since their behavior is very similary to that of FL/P and FLp/P respectively. However, we do show that the hybrid solution method HB is superior to Lagrange Relaxation LR and that LR applied to the transformed formulation SPt/P is much faster than when applied to the standard formulation SP/P. We list the maximum absolute violation of the linking constraints obtained by the volume algorithm in Column 2. The small violations suggest that the approximate primal solution recovered by the volume algorithm is very close to the exact primal solution. Since the difference in lower bound values that each method returned are small, we compare the time needed to calculate that lower bound only, listing the time each method takes relative to the best implementation, which is the hybrid method applied to the FLp/P formulation, called HBp. In Column 3, we report the time in seconds that HBp needs to calculate the lower bound. Columns 4-8 and 11 show the time ratio between each algorithm and HBp. Columns 9 and 10 refer to the Shortest Path formulation and are therefore compared with the best implementation of the shortest path formulation, SPtP-HB. Both SPtP-HB and HBp use the hybrid scheme and solve the same subproblem. Consequently, the difference in

their time performance is small. Finally, the last column, JD, refers to the CPU times obtained by Jans and Degraeve (2004), who utilized decomposition SP/P and implemented a standard simplex-based column generation algorithm. Note that for G57 and G72, we compare to their Lagrange relaxation times (2000 iterations), as they were not able to solve these instances with simplex-based column generation.

Table 2: Computational performance of different algorithms for obtaining the lower bound

Dataset	HBp Max Violation	Time HBp (s)	HB/ HBp	LR/ HBp	LRp/ HBp	ACG/ HBp	ACGp/ HBp	SPP-LR/ SPtP-HB	SPtP-LR/ SPtP-HB	JD (SP/P)/ HBp
G30 (6-15)	0.032	0.18	2.0	8.9	3.1	9.6	9.3	4.89	1.52	8.0
G30b (6-15)	0.049	0.2	1.3	11.9	3.8	7.1	6.7	11.67	3.08	8.2
G53 (12-15)	0.016	1.6	0.9	4.6	1.2	3.5	3.5	6.45	2.90	5.7
G57 (24-15)	0.023	7.60	2.2	5.8	2.3	3.8	3.9	4.70	1.39	3.4
G62 (6-30)	0.029	0.55	1.1	7.2	2.6	15.3	12.6	4.76	1.53	681.0
G69 (12-30)	0.025	2.43	2.2	9.6	2.4	12.4	14.5	7.04	1.77	79.9
G72 (24-30)	0.031	15.77	2.3	6.9	1.6	12.0	11.1	2.73	1.40	18.4
Average	0.029	4.0	1.7	7.8	2.4	9.1	8.8	6.0	1.9	114.9

The comparison of the different approaches suggests some conclusions. The addition of the precedence constraints to the facility location formulation (column 4), and the transformation of the shortest path formulation (column 9 versus column 10) enhance computational performance. We see that on average HB needs 1.7 times the time of HBp to converge (column 4). Further, in columns 5-8 it is evident that the effect of the precedence constraints is much stronger when using LR instead of ACG. When comparing columns 5 and 6, we see that LRp is approximately 3 times faster than LR, whereas columns 7 and 8 reveal that the performance of ACG and ACGp is approximately the same. In addition, columns 9 and 10 reveal that the Lagrange relaxation of the shortest path formulation is approximately 3 times slower, on average, compared to its transformed version. Algorithms SPP-ACG, SPtP-ACG and SPP-HB showed similar behavior as ACG, ACGp and HB respectively and are not reported. The hybrid scheme outperforms all approaches. On average, HBp is 20 times faster than the other algorithms (5.2 times faster if JD is excluded). Finally, it is interesting to note JD, an implementation of simplex-based column generation. Although the runs are made on a slower computer (Pentium 750 MHz), the CPU times are disproportionally larger, due to the poor performance of simplex-based column generation.

On assessing the lower bound quality obtained by SP/P, Jans and Degraeve (2004) give evidence that for the 7 instances of Table 2, the lower bound is stronger than the one obtained by Trigeiro et al. (1989), Degraeve and Jans (2007), Belvaux and Wolsey (2000) and Miller et al. (2000), while the bound from Van Vyve and Wolsey (2006) seems to be stronger for most instances. Note however that there is no ground to support theoretical arguments for which bounds are stronger, with the exception of the bound given by Trigeiro et al. (1989) and Jans and Degraeve (2004). Therefore, the performance of each methodology is data dependent. Intuitively, the period decomposition should perform well when the capacity constraints are tight, because it takes advantage of the extreme points of the single-period polytopes, and when the number of items is small, which is likely to make the subproblems easier.

We compared the performance of our algorithm to the results obtained by Süral et al. (2009). They design a Lagrange-based heuristic for a reformulation of the capacity constrained problem with setup times but without setup costs. The demand constraint is relaxed. They modify the data from Trigeiro et al. (1989) as follows. First, they set all setup costs to zero. Second, they increase zero demands to 2 units. Third, they construct new instances by reducing the number of periods of some existing ones. Finally, they construct instances with unit inventory costs for all items, called *homogeneous* (denoted "hom"). Instances with the original inventory cost are called *heterogeneous* (denoted "het"). In total, 100 instances are generated. Since their approach usually terminates in a few seconds, we have chosen to compare it with our hybrid procedure only. Specifically, we use the hybrid process to obtain a lower bound, recover an approximate primal solution with the volume algorithm, fix the setup variables to 0 or 1 (as described in the diving heuristic) and call the successive rounding/smoothing heuristic once. In particular, no branching is performed, and the algorithm is actually the procedure that is performed at the root node of the branch-and-price tree. Table 3 displays the average duality gaps and the average CPU time for the best heuristic approach of Süral et al. (SDW) and

our extented hybrid heuristic (EHB). SDW was run on an Intel Pentium 4 machine, and the subproblems were solved with CPLEX 7.0.

Table 3: Comparison of EHB with the Lagrange-based heuristic of Süral et al. (2009)

Category	Gap EHB (%)	Gap SDW (%)	CPU EHB (s)	CPU SDW (s)
12 x 10 het	18.43	31.73	0.34	3.06
24 x 10 het	11.14	18.07	1.62	4.79
12 x 15 het	19.25	25.95	1.40	6.07
24 x 15 het	13.44	21.26	6.37	15.33
12 x 30 het	21.92	28.68	3.27	24.01
24 x 30 het	23.44	32.35	19.72	38.93
12 x 10 hom	22.97	42.08	0.53	2.69
24 x 10 hom	14.06	20.88	1.77	4.45
12 x 15 hom	18.44	28.00	1.19	5.64
24 x 15 hom	14.96	20.56	3.46	11.14
12 x 30 hom	21.68	24.33	2.99	19.10
24 x 30 hom	21.35	30.26	7.95	22.91
Average het	17.94	26.34	5.45	15.37
Average hom	18.91	27.69	2.98	10.99

It is interesting to notice the large gaps that result from problems without setup cost. Clearly, EHB outperforms SDW, both in terms of gap quality and CPU time, for both homogeneous and heterogeneous problems.

We also run EHB using the F and G instances from Trigeiro. The average gaps were 2.43% and 2.51% and the average CPU times 0.25 and 2.14 seconds, respectively. Note that Degraeve and Jans (2007) cite an average gap of 2.87% for the F instances, after exploring 2000 nodes in their branch-and-price tree.

We also tested EHB against the best implementation of the iterative production estimate (IPE) heuristic of Pochet and Van Vyve (2004). They run their experiments on a 350 MHz machine and list results on the six instances from the G dataset of Trigeiro et al. (1989) described in Table 2 (IPE was not tested on G30). The optimal value is listed to facilitate the comparisons. Table 4 presents the results.

Table 4: Comparison of EHB against the IPE heuristic

		Best Incumbent		CPU Time (s)	
Dataset	**Optimal**	**IPE**	**EHB**	**IPE**	**EHB**
G30b (6-15)	37721	38976	38162	1.31	0.27
G53 (12-15)	74634	78098	75035	3.53	1.4
G57 (24-15)	136509	137153	136884	6.01	7.21
G62 (6-30)	61746	63073	63018	1.7	0.67
G69 (12-30)	130596	131988	131668	6.03	3.38
G72 (24-30)	287929	290006	288313	21.15	15.5

Although comparison of CPU Times is hard since different machines were used, it is clear that the quality of feasible solutions is much better for EHB. Note that the above IPE results are the best that Pochet and Van Vyve cite, based on the BC-PROD cut generator. Also, despite that EHB seems to find a better feasible solution than IPE, more space for improvement exists, as shown by the optimal solutions. The branch-and-price heuristic performs EHB at the root node and tries to approach the optimal solution. Its computational peformance is described in Section 8.2.

8.2 Comparison of Heuristic Branch-and-Price with Other Approaches

We compared our approach with the most recent and successful heuristic and exact approaches found in the literature. In order to do this, we employed the 7 instances taken from Trigeiro et al. (1989) described in Section 8.1. A comparison based on 7 instances is limited. However, these 7 instances are specifically tested in many other papers and therefore it allows us to compare our results to various other results reported in the literature. Table 5 presents results of the following studies: Müller et al. (2012) (MS), Degraeve and Jans (2007) (DJ), Belvaux and Wolsey (2000) (BW) and our approach (HB&P). The optimal solution of each

instance is also listed, to facilitate the comparisons. MS is a randomized heuristic, so it does not provide any lower bounds and gaps.

Table 5: Comparison of branch-and-price with other approaches

		Best Incumbent				CPU Time (s)			Gaps(%)		
Dataset	**Optimal**	MS	DJ	BW	HB&P	DJ	BW	HB&P	DJ	BW	HB&P
G30 (6-15)	37809	-	37809	-	37809	33	-	5	1.90	-	1.00
G30b (6-15)	37721	37776.4	38162	37221	37721	29	-	2	2.51	1.35	0.90
G53 (12-15)	74634	74720.8	75035	74752	74634	66	189	9	1.61	1.33	0.93
G57 (24-15)	136509	130675	136860	136509	136509	44	55	101	0.36	0.10	0.07
G62 (6-30)	61746	61792.2	62644	61746	61746	359	55	140	2.79	1.24	0.88
G69 (12-30)	130596	130675	131234	130599	130599	215	102	131	0.81	0.32	0.20
G72 (24-30)	287929	287966	288383	287950	288016	306	298	46	0.22	0.07	0.07

Before analyzing the relative performance of each approach, some implementational details are presented. Müller et al. (2012) use a hybrid adaptive large scale neighborhood search strategy. They run their experiments on a 2.66 GHz / 8GB RAM machine and use CPLEX 10.2 to create repair neighborhoods. Since their approach is randomized, the listed values are based on an average of 100 runs, where each run lasts for 60 seconds. Degraeve and Jans develop an exact branch-and-price algorithm. They pose a limit of 2000 nodes and run their experiments on a 750 MHz processor. Finally, Belvaux and Wolsey use a relax-and-fix heuristic which they incorporate within BC-PROD, a customized branch-and-cut system for production planning problems. They use a 200 MHz machine to perform their computations.

In terms of quality of feasible solutions, it seems that the two most competitive approaches are BW and HB&P. It is interesting to notice that, although BW is the oldest approach and runs are made on a slow machine, it seems to provide much better feasible solutions compared to most other approaches. When compared to HB&P, it finds solutions of similar quality. CPU times are not directly comparable. HB&P produces a better gap however, as the lower bound is stronger and the solution quality similar. HB&P gives a stronger bound because most separation routines of BC-PROD use flow-based inequalities that incorporate information mainly from the convex hulls of the single - item uncapacitated polytopes. HB&P however, gives a lower bound that describes the intersection of the capacity and single-item uncapacitated polytopes and therefore it tends to be stronger for tightly constrained problems. Finally, DJ is outperformed both in terms of time and solution quality.

To the best of our knowledge, the best exact approach found in the literature for the above instances is the approximate extended formulation of Van Vyve and Wolsey (2006). Unfortunately, the authors do not cite the CPU time at which they obtain the lower bound at the root node and a comparison with our approach is not possible. However it is expected that a heuristic implementation of their approach would work better for problems with short time horizon, whereas our approach is better for long horizon problems. For example, they solve G62 (6 items, 30 periods) to optimality after 220400 nodes and 1078 seconds on a 1.6GHz machine whereas we need 4212 nodes and 140 seconds to find the optimal value (on a 2GHz machine).

Next, we compared our approach with the best branch-and-price algorithm of those suggested by Pimentel et al. (2010) (PAV). They used the Trigeiro et al. (1989) sets X11117 X12429. Each X set comprises of 5 instances and there are 30 X sets, giving a total of 150 instances. They applied a per period, per item and simultaneous per period and per item decomposition, solved the subproblems with CPLEX 8.1, and performed their computations on a Pentium 4 machine with 1 GB RAM. Table 6 presents results for the 10 hardest sets, i.e. those for which their algorithm gives the largest gaps. The bottom line lists average results for all 150 instances. Detailed results can be found in the appendix.

Note that HB&P outperforms PAV in all of the above sets except one. Moreover, HB&P terminated within the time limit of 150 seconds in 149 out of 150 instances. Also, the average gap and CPU times are much better for HB&P. An interesting observation is that Pimentel et al. (2010) do not get their best gaps from their simultaneous item/period decomposition, which theoretically gives a stronger bound compared to both their item and period decompositions, but from the item decomposition. This is because the master problems of the simultaneous item/period decomposition are very degenerate and time consuming. Therefore, they may not be able to obtain good feasible solutions and improve their gap within their time limit.

Table 6: Comparison with Pimentel et al. (2010)

	CPU Time PAV (s)	CPU Time HB&P (s)	Gap PAV (%)	Gap B&P (%)
x11419	3600	96.06	10.367	7.069
x11429	3600	106.30	9.599	4.993
x12429	3600	110.15	7.999	3.650
x12419	3600	84.07	5.967	4.250
x11428	3600	68.42	4.777	0.951
x12428	3600	61.83	3.908	1.563
x12229	3600	29.93	3.236	1.924
x11229	3600	66.01	3.045	2.071
x12219	3600	62.96	2.745	2.507
x11129	3600	63.37	2.525	2.874
Average (150 instances)	3600	74.91	5.417	3.185

In a final round of trials, we tested our algorithm on some new hard datasets against the CPLEX v12.2 solver (CPX), using the regular formulation CL. The purpose of this comparison is to give evidence on the relative strengths and weaknesses of a decomposition approach against a modern off-the-self branch-and-cut software. To this end, we constructed new harder instances by modifying the Trigeiro dataset. Specifically, we replicated the demand patterns to 60 periods, and reduced the capacity to 95% of its original value. We focused on instances with 6 and 12 items because the integrality gap of the extended formulations improves as the number of items increases, therefore problems with fewer items are more challenging to solve. Finally, we excluded instances that were infeasible without initial inventory because their gap was sensitive to the initial inventory cost.

The process described above led to the creation of 30 new 60-period instances, 21 of which have 6 items and 9 with 12 items. Table 7 shows the computational results. Both algorithms used 100 seconds of CPU time.

Table 7: Integrality gaps of heuristic branch-and-price and CPLEX v12.2

	CPX Gap (%)	HB&P Gap (%)	Gap Closed (%)
6 - 60 Average	2.29	2.44	17.90
6 - 60 Median	1.86	1.97	9.51
12 - 60 Average	1.79	1.57	12.58
12 - 60 Median	1.71	1.57	9.73
Total Average	2.14	2.18	15.29
Total Median	1.85	1.78	9.57

The computational experiments show that both algorithms achieve good performance in terms of integrality gaps. However, no instance was solved to optimality after 100 seconds. The total median gaps show that heuristic branch-and-price performs better overall, but the average gaps are slightly higher than those of CPLEX. We observed that the lower bound obtained by the period decomposition was always better. To demonstrate the efficiency of the lower bound, we calculated all gaps using the best feasible solution and report the amount of gap that is closed with our branch-and-price algorithm. The amount of gap closed is the difference between the CPX Gap and HB&P Gap, divided by the CPX Gap, where all gaps are calculated using the best feasible solution. The last column indicates the superiority of the lower bound obtained by the period decomposition, as the average gap improvement is about 15%. The lower bound obtained by CPLEX is weak, even after exploring a large part of the branch-and-bound tree. Specifically, CPLEX explores more than 28,000 nodes on average, whereas we explore an average of 644 nodes with our approach. The fact that CPLEX explores such a large part of the tree allows it to find better feasible solutions in most instances, so its integrality gaps are competitive to branch-and-price. In conclusion, the two approaches give similar results in terms of gap quality, but our approach dominates CPLEX in lower bounds, since it takes advantage of the special structure of the single period polytopes.

9 Conclusions and Future Research

We have presented period decompositions of the facility location and shortest path formulations of the capacity constrained lot size problem with setup times. The subproblems polytopes do not have the integrality property, and therefore an im proved lower bound is obtained. Customized branch-and-bound algorithms are developed to solve the single-period subproblems. It is shown that a standard column generation scheme is computationally inefficient, and a novel, subgradient-based algorithm is developed, that combines column generation and Lagrange relaxation. This scheme gives a valid lower bound which is a good approximation of the exact lower bound. Moreover, an approximate primal solution of the restricted master is recovered with the volume algorithm. The performance of the hybrid scheme is enhanced further with the proposition of a transformed shortest path formulation and with the addition of a class of valid inequalities in the subproblem. This class improves the quality of the columns that price out, and is equivalent to adding dual optimal inequalities in the dual of the restricted master program. In addition, the subproblem is still tractable with a fast customized branch-and-bound algorithm. Finally, a branch-and-price based heuristic is designed, that integrates relaxation induced neighborhoods, selective diving and successive rounding/smoothing within a novel strategy of node exploration. The latter explores promising parts of the branch-and-bound tree based on information obtained by new feasible solutions. Computational results show that the proposed approach outperforms or compares favorably with the most recent and successful approaches found in the literature.

There are several directions that need further research. The implementation of standard stabilization techniques (eg. du Merle et al. 1999) to extended formulations may make them more tractable computationally. Also, it could lead to the developement of an exact approach, for which an exact lower bound is needed. On a different line, the integration of approximate schemes such as the volume algorithm could lead to enhanced MIP-based heuristics that can tackle very large instances efficiently and give a good dual bound, used to assess their performance. Finally, the per-period decomposition could lead to the development of successful approximation algorithms. Recently, Levi et al. (2008) used the simple plant location formulation with added flow cover inequalities to derive the first 2-approximation algorithm for a variant of CLST without setup times. It would be interesting to explore whether a column generation-based relaxation would lead to similar approximation schemes for CLST.

10 Appendix

10.1 Proof of Theorem 1

Theorem 1 states that $D' \subseteq D$ where

$$D = \left\{ v_{it} \in \Re \;\middle|\; \begin{array}{ll} v_{it} - v_{i,k+1} \leq cv_{itk} \, \forall i \in I, t, k \in T : t \leq k < m & (A1) \\ v_{i1} - v_{it+1} \leq ci_{it} \;\forall i \in I, t \in T \backslash \{m\} & (A2) \\ v_{it} \leq cv_{itm} \;\forall i \in I, t \in T & (A3) \\ v_{i1} \leq ci_{im} \;\forall i \in I & (A4) \end{array} \right\};$$

$$D' = \left\{ \bar{v}_{it} \in \Re \;\middle|\; \begin{array}{ll} \sum_{l=1}^{t} \bar{v}_{il} \leq \min\{ci_{it}, cv_{i1t}\} \;\forall i \in I, t \in T & (A5) \\ \sum_{l=t}^{k} \bar{v}_{il} \leq cv_{itk} \;\forall i \in I, t, k \in T : k \geq t & (A6) \end{array} \right\}$$

Note that $D = \bigcup_{i=1}^{n} D_i$ and $D' = \bigcup_{i=1}^{n} D_i'$ and $\bigcap_{i=1}^{n} D_i = \bigcap_{i=1}^{n} D_i' = \emptyset$, therefore we can omit the item index in the context of this proof. Also, without loss of generality we can restrict our attention to the positive orthant of both dual spaces because (7) can be written as inequality ($\geq$) and also (8) can be written as inequality ($\leq$). We will first show that $D' \subseteq D$. For this, consider a point $v_{it} \in D'$. note that (A5) and (A6) imply (A3) and (A4). Supposing that (A6) holds, we show that (A1) holds as well. We write (A6) as $v_{it} + \underbrace{\sum_{l=t+1}^{k} v_{il}}_{U} \leq cv_{itk} \Leftrightarrow v_{it} + U \leq cv_{itk}$. Observe that if $v_{it} - v_{i,k+1} > cv_{itk}$, then

$v_{it} - v_{i,k+1} > cv_{itk} \geq v_{it} + U \Leftrightarrow U + v_{ik+1} < 0$, which cannot hold because $v_{it} \in \Re_+$ in both spaces. Therefore, (A6) implies (A1). Using the same argument, (A5) implies (A2). This means that $v_{it} \in D' \Rightarrow v_{it} \in D$ for an arbitrary point $v_{it} \in D'$ and thus $D' \subseteq D$. We show that the inclusion may be strict via an example. Consider the following single-item 2-period uncapacitated system with no initial inventory and its transformed version:

$$\begin{aligned}
\min \quad & c_{11}z_{11} + c_{12}z_{12} + c_{22}z_{22} && && [\mathbf{P}] && (33)\\
\text{s.t.} \quad & z_{11} + z_{12} && \geq 1 && [v_1] && (34)\\
& -z_{11} + z_{22} && \geq 0 && [v_2] && (35)\\
& z_{11}, z_{12}, z_{22} && \geq 0 && && (36)
\end{aligned}$$

$$\begin{aligned}
\min \quad & c_{11}z_{11} + c_{12}z_{12} + c_{22}z_{22} && && [\mathbf{P'}] && (37)\\
\text{s.t.} \quad & z_{11} + z_{12} && \geq 1 && [v_1] && (38)\\
& z_{12} + z_{22} && \geq 1 && [v_2] && (39)\\
& z_{11}, z_{12}, z_{22} && \geq 0 && && (40)
\end{aligned}$$

The corresponding dual formulations are:

$$\begin{aligned}
\max \quad & v_1 && && [\mathbf{D}] && (41)\\
\text{s.t.} \quad & v_1 - v_2 && \leq c_{11} && && (42)\\
& v_1 && \leq c_{12} && && (43)\\
& v_2 && \leq c_{22} && && (44)\\
& v_1, v_2 && \geq 0 && && \\
\max \quad & v_1 + v_2 && && [\mathbf{D'}] && (45)\\
& v_1 && \leq c_{11} && && (46)\\
& v_1 + v_2 && \leq c_{12} && && (47)\\
& v_2 && \leq c_{22} && && (48)\\
& v_1, v_2 && \geq 0 && && (49)
\end{aligned}$$

Figure 3 shows the feasible regions of (D) and (D'). Clearly $(D') \subset (D)$, and the proof is complete.

10.2 Complete Computational Results

Table 8 presents the complete results for the 36 Trigeiro instances X11117 X12429. For each set, the average CPU time and the average gap is reported. Runs were made on a 2.0GHz / 2GB RAM machine.

Table 8: Computational results for the Trigeiro X11117 - X12429 datasets

Set	CPU Time (s)	Gap	Set	CPU Time (s)	Gap	Set	CPU Time (s)	Gap
x11419	96.06	7.07%	**x12129**	33.98	0.92%	**x11128**	1.15	0.09%
x11429	106.30	4.99%	**x12418**	31.37	0.86%	**x11217**	12.35	0.07%
x12419	84.07	4.25%	**x12218**	38.67	0.59%	**x11118**	0.40	0.04%
x12429	110.15	3.65%	**x12417**	25.62	0.56%	**x12117**	0.21	0.04%
x12119	80.22	3.46%	**x12128**	1.68	0.38%	**x11227**	1.60	0.03%
x11119	55.94	3.13%	**x12427**	23.05	0.27%	**x12127**	0.10	0.03%
x11129	63.37	2.87%	**x11228**	36.55	0.26%	**x11117**	0.01	0.00%
x11219	72.62	2.51%	**x11427**	60.03	0.25%	**x11127**	0.01	0.00%
x12219	62.96	2.51%	**x11218**	37.93	0.24%	**x11418**	105.00	0.98%
x11229	66.01	2.07%	**x12228**	14.21	0.23%	**x11428**	68.42	0.95%
x12229	29.93	1.92%	**x11417**	57.89	0.22%	**x12217**	4.95	0.17%
x12428	61.83	1.56%	**x12118**	0.53	0.18%	**x12227**	3.64	0.11%

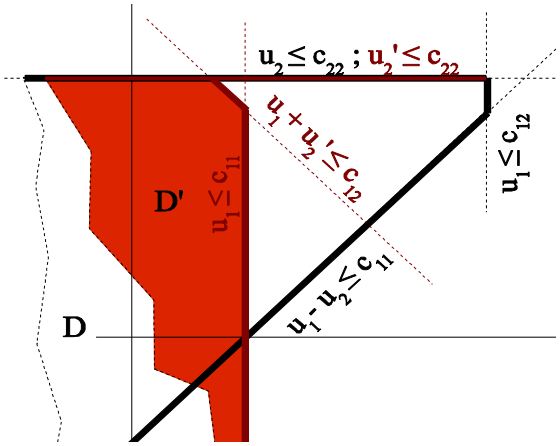

Figure 3: Dual space reduction in a small example

Tables 9 and 10 present results for the EHB heuristic tested on the homogeneous and heterogeneous instances constructed in Süral et al (2009) respectively. MGxx are the Gxx instances with 10 periods and MMG the Gxx instances with 15 periods.

Table 9: The performance of EHB on homogeneous instances

Instance	LB	UB	Gap (%)	CPU Time (s)	Instance	LB	UB	Gap (%)	CPU Time (s)
G51	185.8	218.8	17.74	1.22	**MG56**	66.6	80	20.14	1.66
G52	129.2	159.9	23.7	0.91	**MG57**	169.5	182.2	7.49	2.19
G53	123.3	148	19.98	0.83	**MG58**	194.6	210.5	8.18	1.77
G54	83.7	105	25.39	1.05	**MG59**	190.3	201.5	5.87	2.53
G55	289.9	312.6	7.84	1.51	**MG60**	142.9	151.5	6.01	1.89
G56	247.3	266.2	7.66	3.02	**MG66**	75.5	89.5	18.51	0.38
G57	327.3	356.9	9.03	4.20	**MG67**	41.7	49	17.42	0.43
G58	416.1	441.4	6.09	3.78	**MG68**	78.7	93.2	18.42	0.58
G59	463.9	495.5	6.82	4.03	**MG69**	11.4	18.6	63.16	0.48
G60	369.7	409.0	10.63	3.37	**MG70**	69.4	79.8	14.92	0.51
G66	353.9	445.4	25.87	3.11	**MG71**	62.1	71.3	14.85	1.76
G67	377.9	448.0	18.55	2.73	**MG72**	25.5	36.7	44.20	1.82
G68	738.8	841.9	13.96	3.41	**MG73**	163.2	176.4	8.07	2.17
G69	234.6	298.4	27.20	2.83	**MG74**	76.9	86.6	12.56	1.99
G70	398.9	490	22.85	2.88	**MG75**	145.3	163.8	12.73	1.91
G71	208.5	262	25.65	7.21	**MMG66**	113.8	134.8	18.46	1.04
G72	112.3	162.3	44.48	7.01	**MMG67**	157.3	183.3	16.57	0.82
G73	946.2	1017.2	7.51	8.00	**MMG68**	78.7	93.2	18.42	0.58
G74	398.9	485.6	21.74	6.79	**MMG69**	62.1	82.7	33.15	1.09
G75	1000.5	1074.1	7.35	10.75	**MMG70**	157.9	179.7	13.79	0.98
MG51	96.0	105.3	9.71	0.67	**MMG71**	76.0	89.1	17.27	2.30
MG52	58.8	67.6	14.91	0.37	**MMG72**	45.6	71.5	56.83	2.94
MG53	58.5	65.9	12.59	0.68	**MMG73**	363.3	390.6	7.51	3.61
MG54	24.2	36.3	49.94	0.66	**MMG74**	186.3	221.4	18.83	3.09
MG55	127.1	139.9	10.09	0.53	**MMG75**	345.3	376.2	8.96	4.27

10.3 Implementation Details

The hybrid optimization scheme is implemented as follows. First, it is initialized with a vector of dual prices and a lower bound, both coming either from TTM or from the LP relaxation of the SPL formulation. All

Table 10: The performance of EHB on heterogeneous instances

Instance	LB	UB	Gap (%)	CPU Time (s)	Instance	LB	UB	Gap (%)	CPU Time (s)
G51	354.5	392.8	10.82	1.22	**MG56**	112.4	118.3	5.23	1.27
G52	154.6	177.7	14.95	0.91	**MG57**	204.1	216.4	6.05	2.32
G53	194	233.1	20.15	0.83	**MG58**	281.6	305.7	8.54	1.04
G54	197.3	239	21.12	1.05	**MG59**	313.9	348.2	10.94	1.38
G55	500.9	583.4	16.48	1.51	**MG60**	261.7	288.5	10.24	1.96
G56	536.5	591.1	10.18	3.02	**MG66**	99	109.3	10.4	0.43
G57	427.8	463.6	8.36	4.2	**MG67**	35.8	48.6	35.86	0.17
G58	707.6	743.7	5.1	3.78	**MG68**	136	165.8	21.94	0.2
G59	970.2	1100.6	13.44	4.03	**MG69**	12.4	18.9	52.54	0.25
G60	830	917.6	10.55	3.37	**MG70**	189.3	202.1	6.77	0.41
G66	562.6	660.6	17.41	3.11	**MG71**	77.1	81.5	5.71	3.67
G67	753.2	948.2	25.88	2.73	**MG72**	27.5	37.6	36.73	0.92
G68	1636.9	1889.7	15.45	3.41	**MG73**	262.3	283.4	8.04	3.23
G69	335	427.4	27.57	2.83	**MG74**	89.1	99.2	11.34	1.12
G70	1261.4	1554.8	23.26	2.88	**MG75**	212.9	225.1	5.72	1.59
G71	313.5	403	28.55	7.21	**MMG66**	153.2	175.9	14.82	0.55
G72	114.3	172.4	50.78	7.01	**MMG67**	259	318	22.78	0.79
G73	1947.6	2171.7	11.51	8	**MMG68**	136	165.8	21.94	0.25
G74	507.3	606.3	19.51	6.79	**MMG69**	71	97.3	37.12	0.57
G75	2072.7	2214.2	6.82	10.75	**MMG70**	495.1	601.3	21.44	0.9
MG51	141.1	153	8.4	0.67	**MMG71**	94.5	110.6	16.99	3.44
MG52	69.3	76.3	10.09	0.37	**MMG72**	53.1	74.8	40.92	2.65
MG53	73.8	87.2	18.14	0.68	**MMG73**	673.6	752.3	11.69	4.7
MG54	38.6	42.4	9.82	0.66	**MMG74**	234.9	261.6	11.38	5.22
MG55	185.6	204.8	10.34	0.39	**MMG75**	601.5	636.1	5.75	4.26

subgradient algorithms use the standard step length formula with smoothing. In Lagrange relaxation, we perform 20 iterations if the number of hybrid iterations is less than 3 or greater than 20, and 4 iterations otherwise. The step length is 1.05 and if it is not improved at an iteration, it is multiplied by 0.6. We use a 90% smoothing weight in each violation. The upper bound in the subgradient formula is updated as 1.01 times the current restricted master value. The restricted master is solved similarly. The step length is 0.6 and is multiplied by 0.9 if there is no bound improvement for 10 iterations. The smoothing parameter is 50% initially and then it is halved when the bound improvement is less than 1% in the last 100 iterations. The volume algorithm uses a step length of 0.02 and it is multiplied by 0.9 if there is no bound improvement for 10 iterations. The *target value* T is initialized at 95% of the lower bound and it is updated as follows. Denote the lower bound by lb. Then whenever $lb > 0.95T$ we set $T = 1.005lb$. Finally, the volume algorithm is short-cutted whenever the step length falls bellow a specified tolerance (0.001) or if the maximum absolute violation is less than 3% (2% for the root node) and the gap between the lower bound and the primal solution is less than 3%.

The branch-and-price scheme is implemented in the following sequence. After obtaining a lower bound at the root node, a diving phase follows. The order of variable fixing is the following. We start from the last period. For each item, we calculate a cost to weight ratio, which is the setup cost over the setup time and the demand of some periods ahead. We sort the items and start fixing from the one that has the smallest ratio*and* complies with the fixing criteria described in the paper. All setup variables that comply with the fixing criteria will be fixed, but the fixing order is important. If the resulting subtree is solved completely, we need to backtrack to one of the nodes that are fixed and it is important to backtrack to an influential fixing decision. Therefore, we fix the variables in reverse period order and within each period we choose to fix the most "insignificant" items first.

When we examine the second node on the same level, we delete the columns that were generated on the fist node. We also delete columns when backtracking. At each node we first try to find a feasible solution and then we perform 60 subgradient iterations on the restricted master. If the its objective is higher than the incumbent, we generate new columns. Then, if the lower bound is higher than the incumbent we prune the node, and if it not we call the volume algorithm, with 200 iterations. If the maximum violation is more than 5% and we have not generated columns, we try to generate columns, check the new lower bound against

the incumbent and call the volume algorithm again, if needed. If the volume algorithm identifies a feasible solution, we fix the setup variables and solve the resulting extended network problem in the standard (x, y) space. The algorithm terminates in the following situations: a) The time limit is reached (we set 100 seconds for all runs) b) The algorithm backtracks to the root node For the small instances (ie, 6-15), the algorithm backtracks to the root node, while for the larger ones the time limit is reached.

References

Alfieri, A., P. Brandimarte, S. D'Orazio. 2002. LP-based heuristics for the capacitated lot-sizing problem: the interaction of model formulation and solution algorithm. *International Journal of Production Research* 40 441–458.

Barahona, F., D. Jensen. 1998. Plant location with minimum inventory. *Mathematical Programming* 83 101–111.

Barahona, F., R. Anbil. 2000. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming* 87 385–399.

Barahona, F., R. Anbil. 2002. On some difficult linear programs coming from set partitioning. *Discrete Applied Mathematics* 118 3–11.

Barany, I., T. Van Roy, L. Wolsey. 1984. Strong formulations for multi-item capacitated lot-sizing. *Management Science* 30 1255–1261.

Ben Amor, H., J. Desrosiers, J.M. Valério de Carvalho. 2007. Dual-optimal inequalities for stabilized column generation. *Operations Reseach* 54 454–463.

Belvaux, G., L. Wolsey. 2000. Bc-prod: A specialized branch-and-cut system for lot-sizing problems. *Management Science* 46 724–738.

Caserta, M., E.Q. Rico. 2009. A cross-entropy lagrangian hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Computers & Operations Research* 36 530–548.

Crowder, H. 1976. Computational improvements for subgradient optimization. *Symp. Mathematica* XIX 357–372

Danna, E., E. Rothberg, C. Le Pape. 2005. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming, Series A* 102 71–90.

Degraeve, Z., M. Peeters. 2003. Optimal integer solutions to industrial cutting-stock problems: Part 2, Benchmark results. *INFORMS Journal on Computing* 15 58–81.

Degraeve, Z., R. Jans. 2007. A new Danzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research* 55 909–920.

Denizel, M., F.T. Altekin, H. Süral, H. Stadtler. 2008. Equivalence of the LP relaxation of two strong formulations for the capacitated lot-sizing problem with setup times. *OR Spectrum* 30 773–785.

Denizel, M., H. Süral. 2006. On alternative mixed integer programming formulation and LB-based heuristics for lot-sizing with setup times. *Journal of the Operational Research Society* 57 389–399.

Du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen. 1999. Stabilized column generation. *Discrete Mathematics* 194 229–237.

Elhallaoui, L., D. Villeneuve, F. Soumis, G. Desaulniers. 2005. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research* 53 632–645.

Eppen, G.D., R.K. Martin. 1987. Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research* 35 832–848.

Fischetti, M., A. Lodi. 2003. Local branching. *Mathematical Programming, Series B* 98 23–47.

Geoffrion, A.M. 1974. Lagrangean relaxation for integer programming. *Mathematical Programming Study* 2 82–114.

Gondzio, J., P.G. Brevis, P. Munari. 2013. New developments in the primal-dual column generation technique. *European Journal of Operational Research* 224 41–51.

Graham, R.L. 1972. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters* 1 132–133.

Holmberg, K., D. Yuan. 2000. A Lagrangean heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research* 48 461–481.

Huisman, D., R. Jans, M. Peeters, A.P.M. Wagelmans. 2005. Combining column generation and lagrangian relaxation. G. Desaulniers, J. Desrosiers, M. Solomon, eds. *Column Generation*. Springer, New York, 247–270.

Jans, R., Z. Degraeve. 2004. Improved lower bounds for the capacitated lot-sizing problem with setup times. *Operations Research Letters* 32 185–195.

Jans, R., Z. Degraeve. 2007. Meta-heuristics for dynamic lot-sizing: a review and comparison of solution approaches. *European Journal of Operational Research* 177 1855–1875.

Krarup, J., I. Bilde. 1977. *Plant location, set covering and economic lot size: an O(mn) algorithm for structural problems.* Numerische methoden bei optimierungsaufgaben, Band 3: optimierung bei graphentheoretischen und ganzzahligen problemen, Vol. 36, Birkhauser Verlag, Basel and Stuttgart, 1977.

Levi, R. A. Lodi, M. Sviridenko. 2008. Approximation algorithms for the capacitated multi-item lot-sizing problems via flow-cover inequalities. Math. *Operations Research* 33 461–474.

Lübbecke, M., J. Desrosiers. 2005. Selected topics in column generation. *Operations Research* 53 1007–1023.

Manne, A.S. 1958. Programming of economic lot sizes. *Management Sciences* 4(2) 115–135.

Miller, A.J., G.L. Nemhauser, M.W. Savelsberg. 2000. Solving multi-item capacitated lot-sizing problems with setup times by branch-and-cut. CORE Discussion paper 2000/39, UCL, Louvain-la-Neuve, Belgium.

Mitchell, J. 2005. Cutting plane methods and subgradient methods. *INFORMS TutORials in Operations Research*, INFORMS New Orleans.

Müller, L.F., S. Spoorendonk, D. Pisinger. 2012. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research* 218 614–623.

Pimentel, C.M.O., F.P. Alvelos, J.M. Valério de Carvalho. 2010. Comparing danzig-wolfe decompositions and branch-and-price algorithms for the multi-item capacitated lot sizing problem. *Optimization Methods and Software* 25 299–319.

Pisinger, D. 1995. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research* 83 394–410.

Pochet, Y., M. Van Vyve. 2004. A generic heuristic for production planning problems. *INFORMS Journal on Computing* 16 316–327.

Sinha, P., A. Zoltners. 1979. The multiple-choice knapsack problem. *Operations Research* 27 (3) 503–515.

Subramanian, S., H. Sherali. 2008. An effective deflective subgradient optimization scheme for implementing column generation for large scale airline crew scheduling problems. *INFORMS Journal on Computing* 20 565–578.

Süral, H., M. Denizel, L.N. Van Wassenhove. 2009. Lagrangean relaxation based heuristics for lot sizing with setup times. *European Journal of Operational Research* 194 51–63.

Trigeiro, W., L.J. Thomas, J.O. McClain. 1989. Capacitated lot sizing with set-up times. *Management Sciences* 35 353–366.

Van Vyve, M., L. Wolsey. 2006. Approximate extended formulations. *Mathematical Programming, Series B* 105 501–522.

Vanderbeck, F. 1998. Lot-sizing with start-up times. *Management Sciences* 44 1409–1425.

Vanderbeck, F., M.W.P. Savelsbergh. 2006. A generic view of Danzig-Wolfe decomposition in mixed integer programming. *Operations Research Letters* 48 111–128.

Wolsey, L. 1989. Uncapacitated lot-sizing problems with start-up costs. *Operations Research* 37 741–747.