Product Decomposition in Supply Chain Planning

Braulio Brunaud[1], Maria Paz Ochoa[1], and Ignacio E. Grossmann[1]

[1]Carnegie Mellon University, Department of Chemical Engineering

July 11, 2018

Abstract

Lagrangean decomposition has been used to overcome the difficulties in optimizing large-scale supply chain planning models. Decomposing the problem by time periods has been established as a useful technique. In this paper a novel decomposition scheme is presented, in which the problem is decomposed by products. The decomposition is based on a reformulation of knapsack constraints in the problem. The new approach also allows for simultaneous decomposition by products and time periods, enabling the generation of a large number of subproblems, with the potential benefit of using parallel computing. In the case study performed temporal decomposition was more efficient than the other decomposition schemes. However, a sensitivity analysis indicates that the performance of product decomposition depends on the parameters of the model, and it can outperform temporal decomposition for some sets of parameters. Selecting different orders of products does not affect the result, while aggregating linking constraints can lead to an increase of the Lagrange dual gap. The novel decomposition schemes proposed provide new alternatives for modelers to design the most efficient optimization algorithms.

1 Introduction

The supply chains of the 21st century are highly complex and spread all across the world (Kearney, 2004). Globalization has brought both access to new markets and increased competition. In this scenario, making the best possible decisions is critical to remain competitive. Making optimal tactical plans involves balancing several trade-offs including inventory, transportation, and production costs. The main decisions are the assignment of production volume to manufacturing facilities, including third parties, inventory levels, and shipments between facilities. The planning horizons addressed range from 6 to 24 months, divided in monthly or even weekly time periods. At the same time, a typical supply chain handles hundreds or thousands of products. The resulting optimization models can be very large, and often require the use

of decomposition techniques to obtain good solutions in reasonable time.

We propose a novel Lagrangean decomposition scheme based on decomposing the problem by products, and also a decomposition scheme combining temporal and product decomposition. Lagrangean decomposition (Guignard and Kim, 1987) has been effectively applied to many large-scale optimization problems (Wang, 2003). Graves (1982) used these techniques for hierarchical planning, decomposing a large problem into two subproblems. Gupta and Maranas (1999) analyzed the importance of choosing the appropriate relaxation. Along the same lines. Jackson and Grossmann (2003) used temporal decomposition for solving a multi-site, multi-period planning problem. Terrazas-Moreno et al. (2011) compared temporal and spatial decomposition for supply chain planning. They concluded that temporal decomposition was superior because the relaxation of inventory continuity, generated in temporal decomposition, is closer to the optimal solution compared to the relaxation of material balances obtained when using spatial decomposition. Yongheng et al. (2014) used temporal decomposition to solve a planning problem under uncertainty. Calfa et al. (2013) combined temporal Lagrangean decomposition with bilevel decomposition for the integration of planning and scheduling. The opportunity of decomposing the problem by problems is also identified by Van Elzakker et al. (2014). However, they propose a heuristic procedure, while we generalize these ideas using Lagrangean decomposition.

The update of the Lagrange multipliers is the most critical step of the algorithm, hence, it has been a matter of intensive research. The update methods are based on three basic methods: subgradient (Fisher, 1985), cutting planes (Kelley, 1960), and bundle methods that combine the first two (Lemarechal et al., 1981). The cutting plane methods return many unbounded solutions, especially at the beginning at the first iterations. The bundle methods handle these issues adding stabilization terms. Within the cutting plane methods, Mouret et al. (2011) allows the multipliers to take values within a band of the value given by the subgradient, while Oliveira et al. (2013) allow the multipliers to take values in a band defined by the direction and the opposite direction given by the subgradient method. Both report convergence improvements compare to plain cutting plane methods. The improvement of the subgradient method has also been a matter of research. Erlenkotter (1978) proposed a multiplier adjustment method for the Lagrangian relaxation of the uncapacitated location problem, while Wu and Ierapetritou (2006) use the Nelder-Mead method to identify search directions for adjusting the Lagrangean multipliers. We propose a probing method to improve the step size when updating the mutlipliers using the subgradient method.

The paper is organized as follows: Section 2 presents an overview of Lagrangean decomposition. In

section 3 a representative instance of the problem addressed is described. In section 4 temporal decomposition, product decomposition, and simultaneous product and temporal decomposition are derived. The probing subgradient method is described in section 5. A case study is presented in Section 6, and the main results of the paper are summarized in section 7.

2 Lagrangean decomposition overview

Held and Karp (1970) originally proposed the Lagrangean Relaxation for the travelling salesman problem in which the subtour elimination constraints are dualized. The basic idea is to exploit the structure of many optimization problems that involve complicating constraints as shown in Fig. 1. These constraints are added to the objective function with a penalty term proportional to the amount of constraint violation. The resulting relaxation reduces the computational complexity of the solution procedure, because the Lagrangean problem is easier to solve than the original problem, and each subproblem can be solved independently.

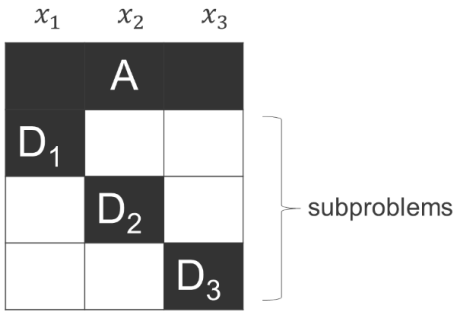

Figure 1. Constraint matrix structure exploited in Lagrangean relaxation

Consider the following optimization problem (Guignard, 2003):

$$P = \max_x = \{z = c^T x | Ax \leq b, Dx \leq e, x \in X\} \tag{1}$$

where $X = \mathbb{R}_+^{n-p} x \{0,1\}^p$, and where $Dx \leq e$ are considered to be the complicating constraints. Let λ be a vector of nonnegative Lagrangean multipliers. The Lagrangean relaxation of (P) is the following:

$$LR(\lambda) = \max_x = \{z = c^T x + \lambda^T (e - Dx) | Ax \leq b, x \in X\} \tag{2}$$

The feasible region of the relaxation is larger that the original the objective is overestimated, and hence

its optimal value provides an upper bound of the original problem (P). The best bound that can be obtained solving the Lagrangean Dual is:

$$LD = \min_{\lambda} LR(\lambda) = \min_{\lambda}\{\max_{x} = \{z = c^T x + \lambda^T (e - Dx) | Ax \leq b, x \in X\}\} \tag{3}$$

Furthermore, Geoffrion (1974) demonstrated the equivalence between the Lagreangean dual (LD) and primal relaxation of (P) denoted by (P^*).

$$P^* = \max_{x} = \{z = c^T x | Dx \leq e, x \in Conv(X) | Ax \leq b, x \in X\} \tag{4}$$

where $Conv(X)$ denotes the convex hull of the set X, and $v\{\cdot\}$ is the optimal solution of $\{\cdot\}$. Then.

$$v\{LP\} \geq v\{LR\} \geq LD = v\{P^*\} \geq v\{P\} \tag{5}$$

where LP is the LP relaxation of (P). This relationship is explained by the modification of the original structure of the problem in Lagrangean Relaxation, since the complicating constraints are removed from the constraint set and added to the objective function as a penalty term. To overcome this drawback, Guignard and Kim (1987) proposed Lagrangean Decomposition, which is a special case of the Lagrangean Relaxation. In Lagrangean Decomposition, instead of relaxing the complicating constraints, the set of variables that connects the set of complicating constraint to the other set of constraints is duplicated, generating new equality constraints. Then, the Lagrangean Relaxation is applied to the new problem by dualizing the new set of complicating variables, namely the new equality constraints. Now every constraint in the original problem appears in one of the sub-problems. Fig. 2 shows the process to obtain the Lagrangean Decomposition for an instance of problem (P), where the set X is the positive integers. It can also be proved that the upper bound generated from the Lagrangean Decomposition is always at least as tight as the one from the Lagrangean relaxation (Guignard and Kim, 1987). The bound of Lagrangean Decomposition can be further tightened by adding surrogate constraints, resulting in subproblems with overlapping constraints.

Another way of looking at Lagrangean decomposition is as a way to reach consensus in a disagreement between one or more parts. The subproblems share a connection expressed in terms of the linking constraints. When dualizing these constraints, the subproblems do not necessarily agree on the value

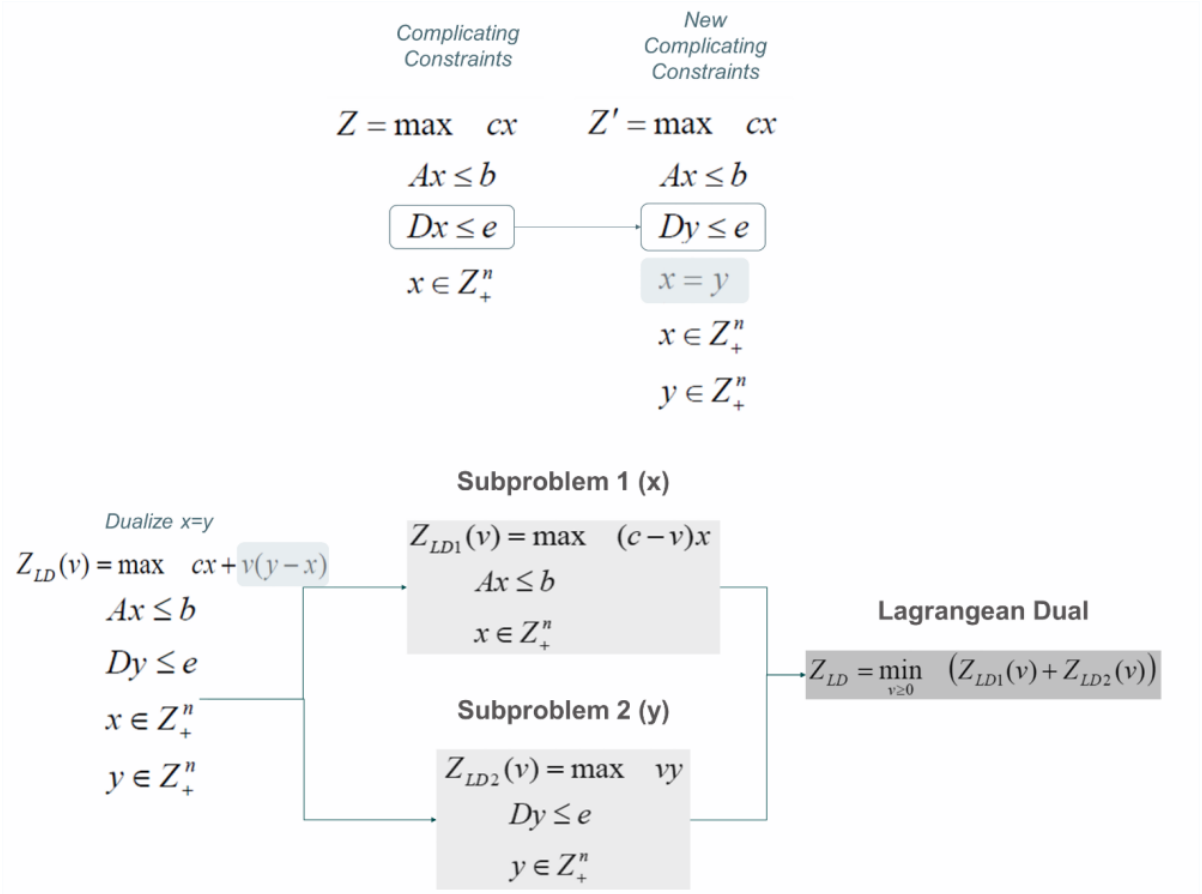

Figure 2. Process to obtain the Lagrangean Decomposition

of the shared variables. To resolve this disagreement they ask a third party to put a price on their disagreement. This price is the Lagrangean multiplier. The iterative procedure proposes new prices at each iteration until the parts agree as much as possible. This way of solving the disagreements makes Lagrangean decomposition the algorithm of choice for decentralized decision-making (Kelly and Zyngier, 2008; Granada et al., 2012; Boyd et al., 2011).

3 Problem description

In order to clarify the presentation of the algorithms described in the following sections it is convenient to define a motivating problem. The problem is later used as case study in Section 6. Consider a set of production facilities, $i \in I$, serving a number of customers, $j \in J$ as shown in Fig. 3. A maximum demand forecast, D_{jpt}, is given at each customer for every product, $p \in P$, at every time period, $t \in T$. The length of each time period, not necessarily uniform, is L_t. The objective is to determine the production and shipping plan to maximize the profit. The production plan is described by the production time θ_{ipt} and

the production amount x_{ipt}. Both variables are linked by a fixed production rate R_{ip}. The unit production cost per amount of product is γ_{ip}. When a product is produced in a given time period t, a setup time σ_{ip} with corresponding setup cost α_p is incurred. The binary variable indicating the occurrence of a production run is y_{ipt}. After production, the amount shipped to each customer, f_{ijpt}, is decided, with the corresponding transportation cost C_{ij}. The product that is not sent to customers is kept as inventory, s_{ipt}, with unit holding cost H_p. All the product arriving at a customer z_{jpt} is sold at price β_p.

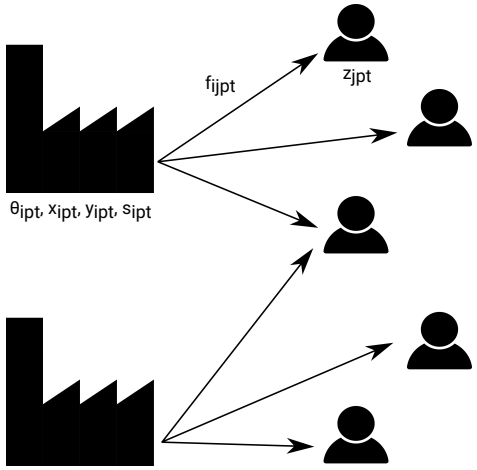

Figure 3. Supply chain network for the first problem

The problem is formulated as a mixed-integer linear programming model (MILP), defined by Eqs.

(6)–(12).

$$M: Max \quad \sum_j \sum_p \sum_t \beta_p z_{jpt} - \sum_i \sum_p \sum_t (\alpha_p y_{ipt} + \gamma_{ipt} x_{ipt} + H_p s_{ipt}) - \sum_i \sum_j \sum_p \sum_t C_{ij} f_{ijpt} \tag{6}$$

$$s.t. \quad R_{ip}\theta_{ipt} = x_{ipt} \qquad \forall i,p,t \tag{7}$$

$$\sum_p (\theta_{ipt} + \sigma_{ip} y_{ipt}) \leq L_t \qquad \forall i,t \tag{8}$$

$$\theta_{ipt} \leq y_{ipt} L_t \qquad \forall i,p,t \tag{9}$$

$$s_{ipt} = s_{ipt-1} + x_{ipt} - \sum_j f_{ijpt} \qquad \forall i,p,t \tag{10}$$

$$\sum_i f_{ijpt} = z_{jpt} \qquad \forall j,p,t \tag{11}$$

$$z_{jpt} \leq D_{jpt} \qquad \forall j,p,t \tag{12}$$

$$x_{ipt}, \theta_{ipt}, s_{ipt}, f_{ijpt}, z_{jpt} \geq 0, y_{ipt} \in \{0,1\} \tag{13}$$

The profit to be maximized is given by Eq. (6), and it is equal to the difference between the income and the costs, which include setup costs, production costs, inventory holding costs, and transportation costs. The production time and production amount are related by Eq. (7). Eq. (8) ensures the production time limit is not exceeded, while Eq. (9) ensures that the binary variable indicating when an item is produced is set to one if there is production time consumed. Eq. (10) represents the system inventory balance. Finally, Eqs. (11) and (12) define the amount sold and an upper bound for sales, respectively.

4 Lagrangean decomposition schemes

4.1 Temporal decomposition

Temporal decomposition is a well established algorithm to solve large instances of problem (M). The goal of temporal decomposition is to break the problem into individual subproblems of one or more time periods. For example, a model with a planning horizon of 6 months can be decomposed into 6 one-month subproblems, or three subproblems of two months each. In the problem, time periods are linked by inventory balance constraints (Eq. 10), which relate the inventory at the end of period t with the inventory at the end of period $t-1$. These constraints can be dualized to obtain the Lagrangean

relaxation (Geoffrion, 1974) of the problem, i.e. transferring the constraint to the objective function with a penalty term referred as Lagrange multiplier. Fig. 4 is a graphical representation of the inventory balance constraint.

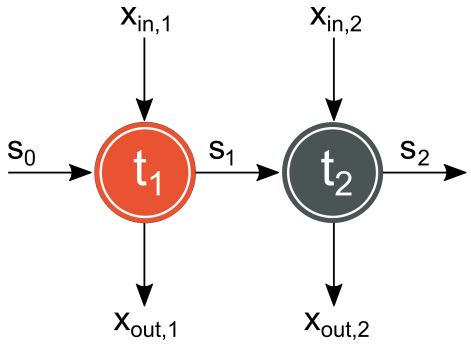

Figure 4. Inventory balance constraints

To obtain the Lagrangean decomposition (Guignard and Kim, 1987) of the problem, the model is reformulated to include auxiliary variables si_{ipt} for the initial inventory at period t, and sf_{ipt} for the inventory at the end of the period. In this case the linking constraint is the inventory continuity equation (Eq. 15) enforcing that the initial inventory at period t must be equal to the inventory at the end of period $t-1$. The problem is decomposed by dualizing the continuity constraints. Fig. 5 illustrates the formulation for this decomposition.

$$sf_{ipt} = si_{ipt} + x_{ipt} - \sum_{j} f_{ijpt} \qquad \forall i, p, t \tag{14}$$

$$sf_{ipt-1} = si_{ipt} \qquad \forall t > 1 \tag{15}$$

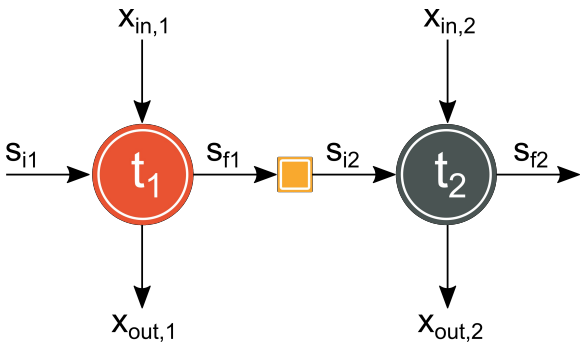

Figure 5. Reformulation of inventory balance constraints for Lagrangean decomposition

Applying this reformulation it is possible to obtain the temporal Lagrange dual ($TLD(t)$) of problem

(M).

$$TLD(t): Max \quad \sum_j \sum_p \beta_p z_{jpt} - \sum_i \sum_p (\alpha_p y_{ipt} + \gamma_{ipt} x_{ipt} + H_p s_{ipt})$$
$$- \sum_i \sum_j \sum_p C_{ij} f_{ijpt} + \sum_i \sum_p (\lambda_t s f_{ipt} - \lambda_{t-1} s i_{ipt}) \tag{16}$$

$$s.t. \quad R_{ip}\theta_{ipt} = x_{ipt} \qquad \forall i,p \tag{17}$$

$$\sum_p (\theta_{ipt} + \sigma_{ip} y_{ipt}) \leq L_t \qquad \forall i \tag{18}$$

$$\theta_{ipt} \leq y_{ipt} L_t \qquad \forall i,p \tag{19}$$

$$s f_{ipt} = s i_{ipt} + x_{ipt} - \sum_j f_{ijpt} \qquad \forall i,p \tag{20}$$

$$\sum_i f_{ijpt} = z_{jpt} \qquad \forall j,p \tag{21}$$

$$z_{jpt} \leq D_{jpt} \qquad \forall j,p \tag{22}$$

$$x_{ipt}, \theta_{ipt}, s_{ipt}, f_{ijpt}, z_{jpt} \geq 0, y_{ipt} \in \{0,1\} \tag{23}$$

The inventory available at the beginning of the planning horizon is the initial inventory. In other words, for $t = 1$, $si_{ip1} = InitInv_{ip}$.

4.2 Product decomposition

Temporal decomposition exploits the natural dynamic structure of time periods, one time period is linked to the next by the inventory balance. On the other hand, products do not exhibit the same structure, products usually share capacity of a limited resource, e.g.: production, transportation, and inventory. The underlying dynamic structure is not clear at first sight, but it can be exposed through reformulation. In general, products are linked by a knapsack constraint of the following form:

$$\sum_p \eta_p x_p \leq C \tag{24}$$

where x_p is the amount of product p, and η_p is the unit consumption factor of x_p with respect to resource C. In model (M), Eq. 8 has this structure, in which the production time is limited by the period length. To reformulate this constraint it is convenient to think of the planning process as making optimal decisions about one product at a time. For the first product, the full capacity is available. After the first product

has been optimized, it is required to know the remaining capacity for the next product. If we denote as c_p the available capacity for product p, then c_p is given by Eq. (25).

$$c_p = c_{p-1} - u_{p-1} \qquad \forall p > 1 \tag{25}$$

$$u_p = \eta_p x_p \qquad \forall p \tag{26}$$

To make this reformulation possible, we have also assigned an arbitrary order to products. Later in the paper we explore the impact of the order of the products in the computational efficiency.

As with temporal decomposition, it is convenient to define auxiliary variables for the initial and ending capacity for a given product, ci_p and cf_p, the reformulation obtained is shown in Fig. 6, and the constraints are shown in Eqs. (27) and (28).

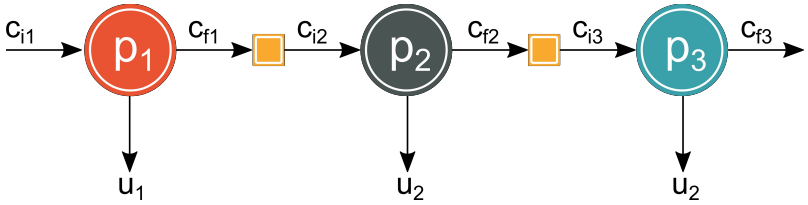

Figure 6. Representation of capacity balance constraints for product decomposition

$$cf_p = ci_p - u_p \qquad \forall p \tag{27}$$

$$cf_{p-1} = ci_p \qquad \forall p > 1 \tag{28}$$

$$ci_1 = C \tag{29}$$

In problem (P), Eq. (8) ensures that the sum of the production time of all products do not exceed the available production time. Even though it is expressed in time units, the available production time limits the production capacity. It is possible to decompose the problem using the same procedure described in the previous section for temporal decomposition. To accomplish this auxiliary variables hi_{ip} and hf_{ip} are defined to represent the number of production hours available at plant i before and after optimizing a

given product. The reformulated constraints are Eqs. (30)–(33).

$$\theta_{ipt} + \sigma_{ip} y_{ipt} \leq hi_{ipt} \qquad \forall pt \tag{30}$$

$$hf_{ipt} = hi_{ipt} - (\theta_{ipt} + \sigma_{ip} y_{ipt}) \qquad \forall pt \tag{31}$$

$$hf_{i(p-1)t} = hi_{ipt} \qquad \forall i, p > 1 \tag{32}$$

$$hi_{i1} = L_t \tag{33}$$

The resulting product Lagrange dual ($PLD(p)$) is given by Eqs. (34)–(43).

$$\begin{aligned} PLD(p): Max \quad & \sum_j \sum_t \beta_p z_{jpt} - \sum_i \sum_t (\alpha_p y_{ipt} + \gamma_{ipt} x_{ipt} + H_p s_{ipt}) \\ & - \sum_i \sum_j \sum_t C_{ij} f_{ijpt} + \sum_i \sum_t (\mu_{ipt} hf_{ipt} - \mu_{i(p-1)t} hi_{ipt}) \end{aligned} \tag{34}$$

$$s.t. \quad R_{ip}\theta_{ipt} = x_{ipt} \qquad \forall i, t \tag{35}$$

$$\sum_p (\theta_{ipt} + \sigma_{ip} y_{ipt}) \leq hi_{ipt} \qquad \forall i, t \tag{36}$$

$$\theta_{ipt} \leq y_{ipt} L_t \qquad \forall i, t \tag{37}$$

$$\theta_{ipt} + \sigma_{ip} y_{ipt}) \leq hi_{ipt} \qquad \forall t \tag{38}$$

$$hf_{ipt} = hi_{ipt} - (\theta_{ipt} + \sigma_{ip} y_{ipt}) \qquad \forall i, t \tag{39}$$

$$s_{ipt} = s_{ipt-1} + x_{ipt} - \sum_j f_{ijpt} \qquad \forall i, t \tag{40}$$

$$\sum_i f_{ijpt} = z_{jpt} \qquad \forall j, t \tag{41}$$

$$z_{jpt} \leq D_{jpt} \qquad \forall j, t \tag{42}$$

$$x_{ipt}, \theta_{ipt}, s_{ipt}, f_{ijpt}, z_{jpt} \geq 0, y_{ipt} \in \{0, 1\} \tag{43}$$

The derivation from this section has been motivated by the large number of products considered in supply chain planning. However, the reformulation can be applied to any set sharing a resource in a knapsack constraint. For example, in the optimization of HVAC systems (Heating, Ventilation and Air Conditioning), the capacity of the cooling unit is shared among all rooms (Álvarez et al., 2013).

4.3 Simultaneous product and temporal decomposition

After decomposing the problem (P) by time periods the resulting subproblems have multiple products. Similarly, when decomposing problem (M) by products, the resulting subproblems are multiperiod. Applying the same concepts reviewed in the previous sections, it is possible to decompose problem (M) by products and time periods simultaneously. The network structure of the resulting subproblems is shown in Fig. 7.

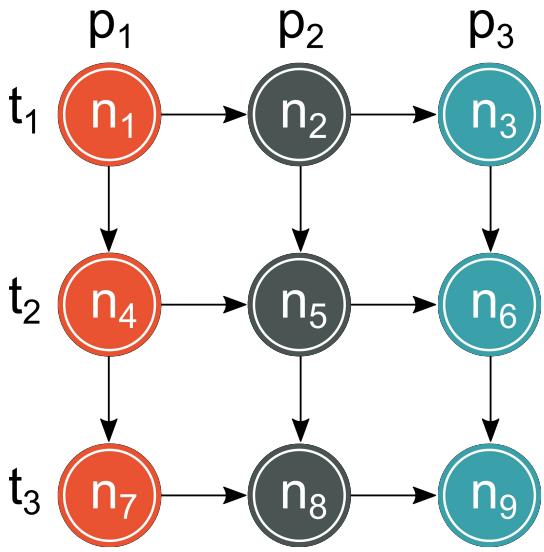

Figure 7. Subproblem network structure of the simultaneous product and temporal decomposition

In Fig. 7 node n_1 corresponds to the subproblem for the first product in the first period. In the structure, some nodes such as node n_5 have multiple inputs and multiple outputs. The problem is decomposed dualizing all the linking constraints, which are represented by the arcs in the diagram.

The resulting Lagrange dual is given by problem $(PTLD(p,t))$.

$$
\begin{aligned}
PTLD(p,t): Max \quad & \sum_j \beta_p z_{jpt} - \sum_i (\alpha_p y_{ipt} + \gamma_{ipt} x_{ipt} + H_p s_{ipt}) \\
& - \sum_i \sum_j C_{ij} f_{ijpt} + \sum_i (\mu_{ipt} hf_{ipt} - \mu_{i(p-1)t} hi_{ipt}) \\
& + \sum_i (\lambda_{ipt} sf_{ipt} - \lambda_{ipt-1} si_{ipt}) && (44) \\
s.t. \quad & R_{ip}\theta_{ipt} = x_{ipt} && \forall i \quad (45) \\
& \sum_p (\theta_{ipt} + \sigma_{ip} y_{ipt}) \leq hi_{ipt} && \forall i \quad (46) \\
& \theta_{ipt} \leq y_{ipt} L_t && \forall i \quad (47) \\
& \theta_{ipt} + \sigma_{ip} y_{ipt}) \leq hi_{ipt} && \forall t \quad (48) \\
& hf_{ipt} = hi_{ipt} - (\theta_{ipt} + \sigma_{ip} y_{ipt}) && \forall i \quad (49) \\
& sf_{ipt} = si_{ipt} + x_{ipt} - \sum_j f_{ijpt} && \forall i \quad (50) \\
& \sum_i f_{ijpt} = z_{jpt} && \forall j \quad (51) \\
& z_{jpt} \leq D_{jpt} && \forall j \quad (52) \\
& vars && (53)
\end{aligned}
$$

The ability to simultaneously decompose a problem into both time periods and products gives the flexibility to decide on the number of subproblems. For example, if a planning problem has only 12 time periods, the maximum number of subproblems obtained when applying temporal decomposition is 12. However, if the problem has also 50 products, the number of subproblems can be up to 600. This can be helpful for very large scale problems.

5 An improved subgradient method

The subgradient method has been the most popular choice to update the multipliers in Lagrangean decomposition. The update formula proposed by Fisher (1985) has been used in a large number of applications because of its simplicity and ease of computation.

The main drawback of the formula is that its application does not guarantee a descent in the Lagrange

dual (in case of maximization). Even though the subgradient gives a descent direction, the step size used might be too large. This could lead a decomposition algorithm diverging from the optimal Lagrange dual for some iterations. Several more iterations are required to bring the bound value back to the initial values. In the traditional subgradient the step is taken regardless of the reduction in the Lagrange dual, and if the function fails to decrease in a specified number of iterations, the step size is decreased. Usually, the initial step is chosen to be between 0 and 2, and it is halved when the Lagrange dual fails to decrease in 3 consecutive iterations.

We propose a simple algorithm to avoid taking ascending steps in the search process. The idea is to probe the value of the candidate step and then decide the length of the step based on the difference between the current value of the Lagrange dual (z_k) and its value at the candidate step (z_c). We can distinguish the following cases:

1. The candidate step improves the bound ($z_k - z_c > \delta$). Take full step α.
2. The candidate step is similar to the current solution ($|z_k - z_c| \leq \delta$). Take half step, $\frac{\alpha}{2}$.
3. The candidate step worsens the bound ($z_c - z_k > \delta$). Take a quarter step, $\frac{\alpha}{4}$.

The decisions on the step size rely on the piecewise-convex nature of the Lagrange dual function. Clearly, if the value of the function at the candidate step is similar to the current value, there is a point between the current and the candidate solution that minimizes the Lagrange dual function in the current search direction. Following the same idea, if the bound worsens at the candidate step the point that minimizes the Lagrange dual function in the current search direction is located closer to the current point. Fig 8 summarizes the cases. The method is computationally more intensive than the traditional subgradient

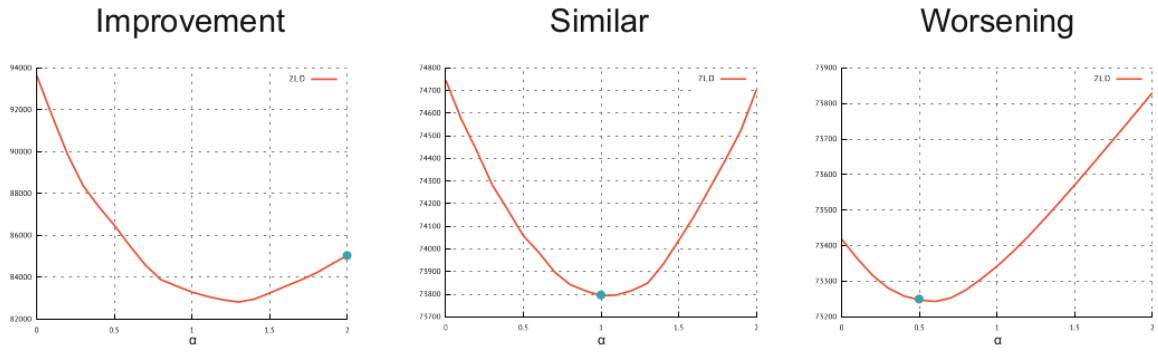

Figure 8. Probing subgradient method cases

because more Lagrange dual evaluations are required. Each evaluation is conducted by solving all the

subproblems for a given value of the Lagrange multipliers. To partially alleviate this issue, when the case 1 is found, i.e. the candidate step improves the bound, the value of z_k is immediately updated to save one evaluation. The extra effort in the computation is compensated by having optimization trajectories with less divergence, which can be especially useful when the time budget for the iterations is low, for example, in problems whose subproblems are hard to solve.

6 Case study

The computational performance of the Lagrangean decomposition algorithms proposed is assessed by applying them to the case study presented in Section 3. The problem considered involves 6 manufacturing sites and 10 markets, while the number of customers and time periods was varied from 6 to 20. The number of time periods considered is equal to the number of products to ensure a fair comparison between temporal and product decomposition. In the entire section, this number will be referred as the size of the problem. Each time period represents 720 hours. The MILP formulations are modeled in JuMP(Dunning et al., 2017) and solved with Gurobi 7.5(Gurobi Optimization, 2015) in an Intel i7 machine with 16 Gb of RAM. Model sizes are shown in Table 1. Since the intention is to evaluate the decomposition schemes, the optimal solution was provided as a lower bound instead of using a Lagrangean heuristic. The method used to update the Lagrangean multipliers is the standard subgradient with initial step size of 2, decreasing the step to half of the current step when no improvement is obtained in 3 consecutive iterations.

Table 1. Model sizes for the case study

Problem size	Constraints	Variables	Binaries
6	2,232	4,032	216
20	24,800	44,800	2,400

6.1 Decomposition schemes comparison

The different ways of decomposing the problem, by product (P), time periods (T), and both, products and time periods (PT), were compared applying the algorithms to the case study. The effect of the initial multipliers in the formulation was also studied. Two possible values for the initial multipliers were considered: 1) zero ($\lambda_0 = 0$), and 2) the multipliers obtained from solving the LP relaxation (λ_0 from LP). The results for solution time and optimality gap after 90 seconds are shown in Table 2.

In the case study, the temporal decomposition converged to better bounds than the ones obtained

Table 2. Comparison of the performance of decomposition schemes for different multiplier initialization strategies

	$\lambda_0 = 0$			λ_0 from LP		
	P	T	PT	P	T	PT
Size = 6						
Iterations	27	78	907	14	28	855
Avg .Iteration Time (s)	3.48	1.18	0.10	6.43	3.33	0.11
Gap (%)	0.29	0.09	0.46	0.39	0.03	0.37
Size = 20						
Iterations	80	113	97	75	104	135
Avg .Iteration Time	1.13	0.78	0.93	1.20	0.87	0.67
Gap (%)	26.36	20.55	66.79	1.24	0.77	2.43

with product decomposition. Even though the number of products and time periods was set to be the same, the product subproblems took longer to solve than the time period subproblems, which can be seen in the larger average iteration time. Allowing more iterations, showed that the minimum value for product decomposition in a problem of size 6 was 0.23%, which is still larger than the gap obtained with temporal decomposition. In the case study, the termination criterion was chosen to be a time limit, because when optimizing supply chain planning there is usually a limited time to obtain a solution, in order to allow the analysis of different scenarios. The temporal decomposition being more efficient than the product decomposition for the case study can not be generalized. The value of the Lagrangean relaxation for products decomposition strongly depends on the capacity. In the limiting case, when there is unlimited production capacity the problem for each product can be solved independently. Fig. 9 shows the sensitivity analysis of the gap with respect to capacity for product decomposition of size 20. For large values of capacities the gap decreases to reach very low gaps. The values do not reach zero gap because of the relative gap setting for solving the subproblems was set to 0.5%.

The sensitivity analysis explicitly shows that the gap obtained using product decomposition depends on the parameters of the model. For large capacity values, the gaps obtained are even lower than the gap obtained with temporal decomposition (0.78 %). For small values of capacity, the gap is also small, because the problem becomes too constrained and finding the optimal solution becomes easier.

Another noteworthy aspect is that the number of products was set to be the same as the number of time periods for comparison purposes. However, the number of products in supply chain planning models usually exceeds the number of time periods. In this situation, product decomposition leads to a larger number of smaller subproblems than temporal decomposition, making it better suited to take

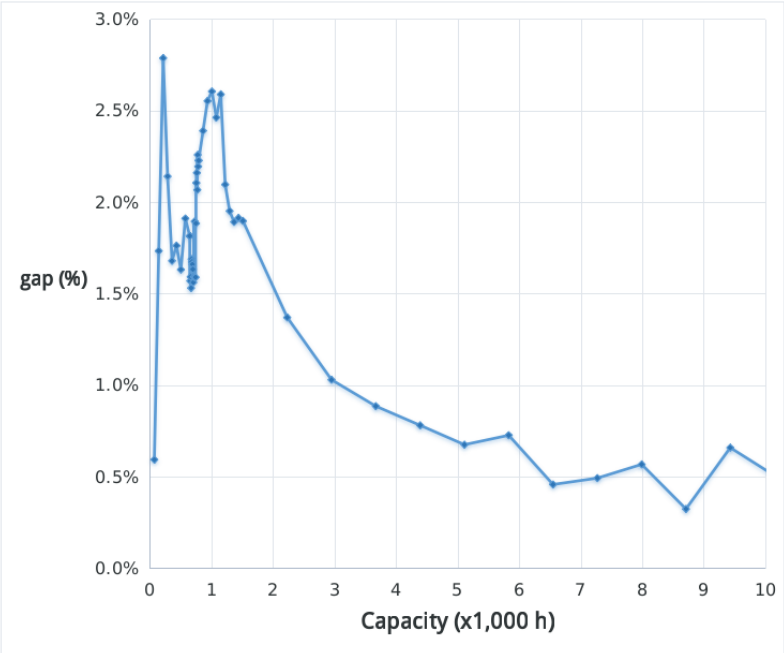

Figure 9. Sensitivity analysis of gap with respect to capacity for product decomposition of size 20

advantage of parallel computing capabilities. The simultaneous product and temporal decomposition for 6 products and time periods is an example of such situation; the number of subproblems obtained is much larger ($O(n^2)$ where n is the problem size), but with a much lower average iteration time. The tradeoff is that a larger number of subproblems comes with a larger number of Lagrangean multipliers that need to be optimized, which could slow down the overall solution time. For the problem with 20 products and time periods, the average iteration time for simultaneous product and temporal decomposition becomes similar to the average iteration time for temporal decomposition. A balance is reached in which solving 400 small subproblems takes almost the same time as solving 20 larger subproblems. This indicates that the number of subproblems must be carefully chosen to balance the average iteration time and the number of iterations required to converge to the optimal solution.

The success of Lagrangean decomposition is based on the exponential increase of the solution time with problem size. For example, if a problem is decomposed into two subproblems, the solution time required to solve both subproblems is much lower than the time required to solve the full problem. However, several iterations are required and subproblems are solved one time per iteration. If the problem is decomposed into more subproblems, more iterations are required to converge. The number of subproblems that minimizes the solution time is problem-dependent. The benefit of the decomposition schemes proposed is that it provides more options to the modeler in the goal of finding the most efficient algorithm to solve

a given problem.

The choice of the initial value of the Lagrangean multipliers is also an important factor impacting the efficiency of the algorithm. In all the cases but the product decomposition of size 6, choosing the initial multipliers as the multipliers obtained from the LP relaxation allowed to reach a smaller gap. Because the value of the initial Lagrange dual is smaller for the initial multipliers from the LP relaxation, the gap that needs to be closed is also smaller. Fig. 10 compares the Lagrange dual (ZLD) with respect to time for the temporal decomposition of size 20.

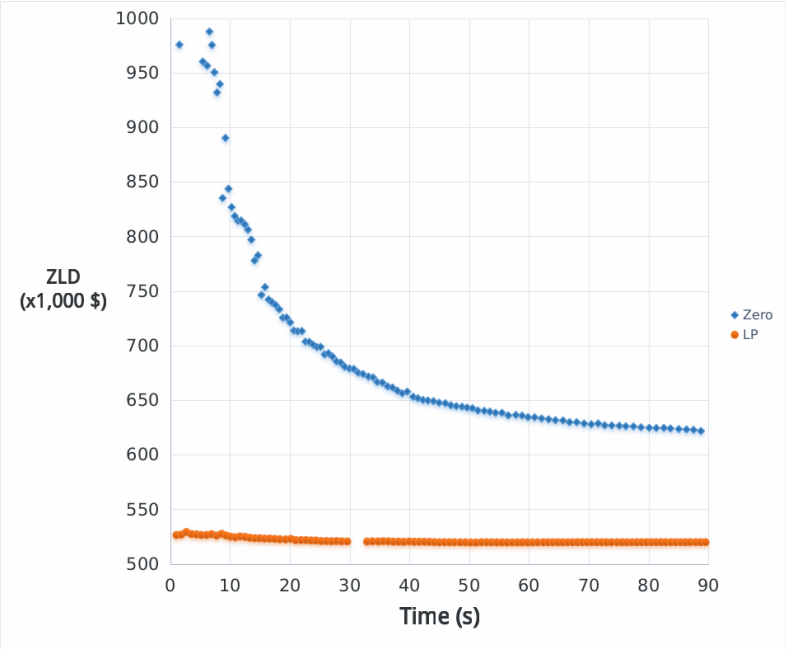

Figure 10. Comparison of Lagrange dual value (ZLD) with respect to time for different choices of initial multipliers

6.2 Aggregation of linking constraints

When solving a Lagrangean decomposition problem it is necessary to find a bound, upper bound in the case of maximization and lower bound in the case of minimization, that is as tight as the LP relaxation of the problem. Thus, if the decomposition is applied to a relaxation of the original problem, the result of the decomposition algorithm also leads to a valid bound for the original problem. Clearly, the quality of the bound obtained depends on how close is the solution of the relaxation chosen to the optimal solution. As mentioned before, a larger number of multipliers that need to be optimized translates into a larger number of iterations to converge to the optimal Lagrange dual. A relaxation of the problem that can help reducing the number of multipliers can be obtained creating surrogate constraints from the linking

constraints, i.e. aggregating the linking constraints.

Take for example, the linking constraint for temporal decomposition (Eq. (15)). The constraint is indexed by the manufacturing site i and the product p. The equation actually represents ip linking constraints between node t and node $t+1$. Surrogates can be constructed by adding the constraints by either site, product, or both. Fig. 11 provides an example of different aggregation possibilities for temporal decomposition.

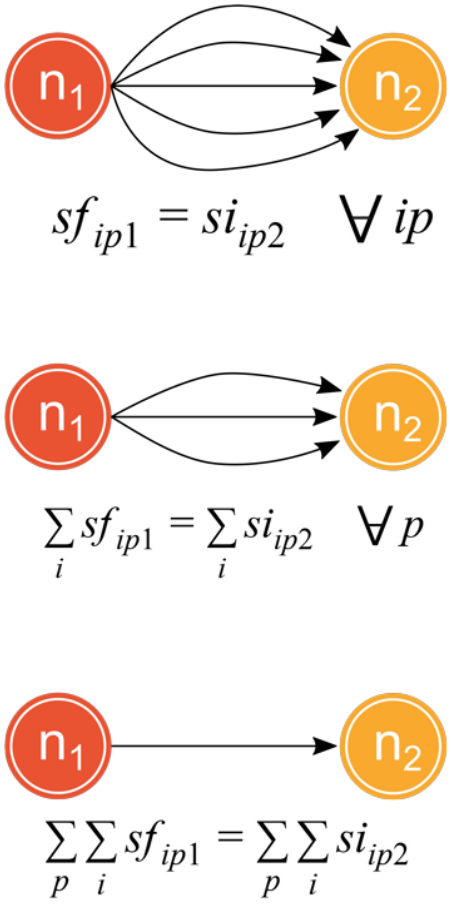

Figure 11. Reduction of number of multipliers by creating surrogates of linking constraints

The different aggregation strategies were applied to the case study, obtaining the results shown in Table 3. In the table *none* represents the original formulation without aggregation, i represents the constraints that were added by manufacturing site, t by time period, p by product, it by site and time period, and ip by site and product.

The result indicate that the aggregation of linking constraints does not improve the Lagrange dual gap. In the best case, for temporal decomposition of size 6, the results remain similar while decreasing

Table 3. Comparison of the results for different aggregation strategies

	P				T				PT	
	none	i	t	it	none	i	p	ip	none	i
Size = 6										
Num. Multipliers	180	30	30	5	180	30	30	5	360	60
Avg. Iteration Time (s)	6.43	0.52	11.12	0.56	3.33	1.05	3.24	1.00	0.11	0.09
Gap (%)	0.39	1.09	0.86	1.09	0.03	0.02	0.06	0.09	0.37	1.24
Size = 20										
Num. Multipliers	2280	380	114	19	2280	380	114	19	4560	760
Avg. Iteration Time (s)	1.20	5.29	0.75	3.00	0.87	0.82	0.69	0.54	0.67	0.57
Gap (%)	1.24	4.89	1.47	4.85	0.77	1.02	13.61	16.42	2.43	5.67

the average iteration time, as for the case of aggregation by sites. For temporal decomposition of size 20 aggregation of products significantly worses the Lagrange dual. For product decomposition, aggregations by site produce the largest increases in the Lagrange dual. For simultaneous product and temporal decomposition the aggregation reduces the number of multipliers, but the Lagrange dual also worsens. The observation that the aggregation of linking constraints can lead to the same bound reducing the number of multipliers is reported by Chen and Guignard (1998). Consistently, in the case study we obtained some scenarios where the bound is maintained and some where it is worsened.

6.3 Effect of product order

In order to obtain the product decomposition formulation the products are assigned an arbitrary order. This gives rise to the question about the impact of the order of the products in the results. To answer this question, product Lagrangean decomposition was applied to 100 random orders of products, for the problem of size 6 after 50 iterations. The histogram of the results is shown in Fig. 12 The results obtained

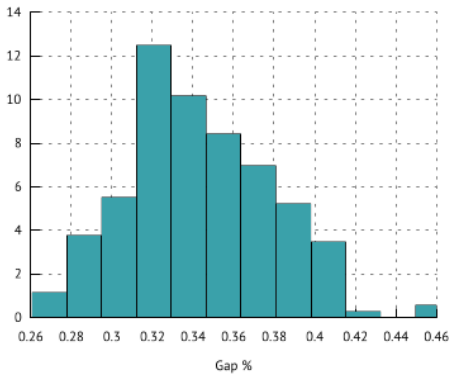

Figure 12. Histogram of % gap for 100 random product orders for problems of size 6

have a low dispersion, and there are now clear outliers. This indicates the order of the products does not have a significant impact in the solution gap.

6.4 Probing subgradient performance

The efficiency of the improved subgradient method is assessed by comparison of its application to the traditional subgradient and to the three decomposition schemes analyzed: temporal decomposition, product decomposition, and simultaneous product and temporal decomposition. The results of the first 10 iterations for problems of size 6 is shown in Fig. 13 A direct comparison can be made looking closely

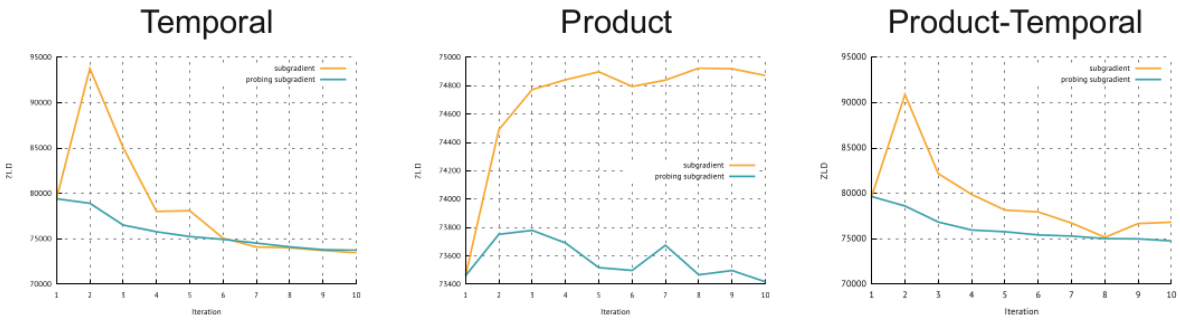

Figure 13. Lagrange dual of the first 10 iterations of problems of size 6 using probing subgradient and traditional subgradient

at the results of iteration 2. For the case of temporal decomposition, the candidate step resulted in a large increase in the Lagrange dual. By taking a quarter step it was possible to find a point were the Lagrange dual decreased. For product decomposition the candidate step also produces an increase in the Lagrange dual. However, in this case the quarter step also did not result in a decrease of the Lagrange dual. Nevertheless, the effect of picking an ascending value was mitigated.

Both the temporal decomposition, and the product-temporal decomposition show that the traditional subgradient can recover from diverging, but it takes several iterations to catch up with the bounds obtained using the probing subgradient. In the product decomposition, the difference is even clearer. The Lagrange duals obtained after 10 iterations are completely different. These two results combined indicate that the probing subgradient is a good choice in presence of a low budget of iterations.

7 Conclusions

Product decomposition is a novel decomposition scheme that provides additional options for modelers to find the most efficient optimization algorithms. The idea of decomposing was motivated by the large

number of products in some supply chain planning problems. However, the derivation can be applied to other problems with sets sharing resource in knapsack-like constraints. The results are also useful to design algorithms similar to rolling horizon, in which the problem is solved one product at a time. The results of temporal decomposition applied to case study resulted better than the results of product decomposition. However, the sensitivity analysis indicates that product decomposition can lead to a smaller gap when capacity is larger, demonstrating that the results depend on the parameters of the model. Furthermore, both approaches are mathematically equivalent. The only difference is the effect of the relaxation obtained from the Lagrange dual in the objective function. This effect is problem-specific and depends on the modeling parameters. The choice between product or temporal decomposition depends on the number of elements on each set (products and time periods) and the structure of the objective function. Modelers are encouraged to experiment with both approaches when facing a new supply chain planning problem. In most applications, the number of products is larger than the number of time periods, making product decomposition better suited to take advantage of parallelization.

The results of the simultaneous product and temporal decomposition show the tradeoff between number of subproblems (or multipliers), and the difficulty to converge to the optimal Lagrange dual. A larger number of subproblems requires a larger number of multipliers, with a lower average iteration time, and a resulting algorithm with a lower probability to converge to the optimal Lagrange dual. The simultaneous decomposition is recommended only for cases in which decomposing only by time periods or products still results in subproblems that are too expensive to solve, with large average iteration times. Aggregation of linking constraints should also be considered in a case by case basis, as it can be beneficial in some applications, but not in others. The results indicate that it can lead to reductions in the number of multipliers while maintaining the quality of the bounds. Together with the decomposition schemes it represents another lever for modelers to design the most efficient algorithms. On the other hand, the order of the products did not have much impact on the results.

Finally, the probing subgradient method resulted in a more stable Lagrange dual method, preventing the Lagrange dual to take large ascending steps with make the algorithm diverge in the first iterations. The method require the solution of more optimization problems per iteration. therefore, it should only be used for expensive subproblems, i.e. a large average iteration time.

In summary, product decomposition is a novel approach that provides modelers with more options to design the most efficient decomposition algorithms and obtain better results.

Decomposing the problem by products is not consistently better or worse than decomposing by time periods. This is explained by the mathematical equivalence of both reformulations. This decomposition scheme is presented as an alternative to modelers to experiment during the implementation of Lagrangean decomposition. Simultaneous product and temporal decomposition allows to generate a larger number of subproblems and take greater advantage of parallel computing. The number of subproblems used is also a matter of experimentation in the implementation phase of the algorithm. Aggregating linking constraints to reduce the number of multipliers can also lead to improved performance and quality of the bounds obtained. This offers additional options to optimize the performance of the decomposition algorithms. The order of products is another lever that can be adjusted to improve the obtained results.

8 Acknowledgments

The authors acknowledge the financial support from the Center for Advanced Process Decision-making (CAPD) from Carnegie Mellon University, and the Government of Chile through its Becas Chile program.

References

Álvarez, J., Redondo, J., Camponogara, E., Normey-Rico, J., Berenguel, M., and Ortigosa, P. (2013). Optimizing building comfort temperature regulation via model predictive control. *Energy and Buildings*, 57:361–372.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122.

Calfa, B. A., Agarwal, A., Grossmann, I. E., and Wassick, J. M. (2013). Hybrid Bilevel-Lagrangean decomposition scheme for the integration of planning and scheduling of a network of batch plants. *Industrial & Engineering Chemistry Research*, 52(5):2152–2167.

Chen, B. and Guignard, M. (1998). Polyhedral analysis and decompositions for capacitated plant location-type problems. *Discrete Applied Mathematics*, 82(1-3):79–91.

Dunning, I., Huchette, J., and Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009.

Fisher, M. L. (1985). An applications oriented guide to Lagrangian relaxation. *Interfaces*, 15(2):10–21.

Geoffrion, A. M. (1974). Lagrangean relaxation for integer programming. In *Approaches to integer programming*, pages 82–114. Springer.

Granada, M., Rider, M. J., Mantovani, J., and Shahidehpour, M. (2012). A decentralized approach for optimal reactive power dispatch using a Lagrangian decomposition method. *Electric Power Systems Research*, 89:148–156.

Graves, S. C. (1982). Using Lagrangean techniques to solve hierarchical production planning problems. *Management Science*, 28(3):260–275.

Guignard, M. (2003). Lagrangean relaxation. *Top*, 11(2):151–200.

Guignard, M. and Kim, S. (1987). Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2):215–228.

Gupta, A. and Maranas, C. D. (1999). A hierarchical Lagrangean relaxation procedure for solving midterm planning problems. *Industrial & Engineering Chemistry Research*, 38(5):1937–1947.

Gurobi Optimization, I. (2015). Gurobi optimizer reference manual. *URL http://www. gurobi. com*.

Held, M. and Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162.

Jackson, J. R. and Grossmann, I. E. (2003). Temporal decomposition scheme for nonlinear multisite production planning and distribution models. *Industrial & engineering chemistry research*, 42(13):3045–3055.

Kearney, A. (2004). The complexity challenge: A survey on complexity management across the supply chain. *AT Kearney Publications, Available at: https://www. atkearney. de/content/misc/wrapper. php/id/49230/name/pdf_complexity_management_s_1096541460ee67. pdf*.

Kelley, Jr, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712.

Kelly, J. D. and Zyngier, D. (2008). Hierarchical decomposition heuristic for scheduling: Coordinated reasoning for decentralized and distributed decision-making problems. *Computers & Chemical Engineering*, 32(11):2684–2705.

Lemarechal, C., Strodiot, J.-J., and Bihain, A. (1981). On a bundle algorithm for nonsmooth optimization. In *Nonlinear programming 4*, pages 245–282. Elsevier.

Mouret, S., Grossmann, I. E., and Pestiaux, P. (2011). A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Computers & Chemical Engineering*, 35(12):2750–2766.

Oliveira, F., Gupta, V., Hamacher, S., and Grossmann, I. E. (2013). A Lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations. *Computers & Chemical Engineering*, 50:184–195.

Terrazas-Moreno, S., Trotter, P. A., and Grossmann, I. E. (2011). Temporal and spatial Lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers. *Computers & Chemical Engineering*, 35(12):2913–2928.

Van Elzakker, M., Zondervan, E., Raikar, N., Hoogland, H., and Grossmann, I. E. (2014). An SKU decomposition algorithm for the tactical planning in the FMCG industry. *Computers & Chemical Engineering*, 62:80–95.

Wang, S.-H. (2003). An improved stepsize of the subgradient algorithm for solving the Lagrangian relaxation problem. *Computers & Electrical Engineering*, 29(1):245–249.

Wu, D. and Ierapetritou, M. (2006). Lagrangean decomposition using an improved Nelder–Mead approach for lagrangean multiplier update. *Computers & chemical engineering*, 30(5):778–789.

Yongheng, J., Rodriguez, M. A., Harjunkoski, I., and Grossmann, I. E. (2014). Optimal supply chain design and management over a multi-period horizon under demand uncertainty. Part II: A Lagrangean decomposition algorithm. *Computers & Chemical Engineering*, 62:211–224.

Contents