Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 385–392

ISBN 978-83-60810-51-4

Lagrangean Decomposition of a Lot-Sizing Problem into Facility Location and Multi-commodity Flow

Samuel DELEPLANQUE
LIMOS CNRS/UMR 6158
Blaise Pascal University,
Campus des Cézeaux,
63173 Aubière, France
Email: deleplanque@isima.fr

Safia KEDAD-SIDHOUM
LIP6, CNRS, Paris VI University,
4 Place JUSSIEU
750065 Paris, France,
Email:
Safia.Kedad-Sidhoum@lip6.fr

Alain QUILLIOT
LIMOS CNRS/UMR 6158
Blaise Pascal University,
Campus des Cézeaux,
63173 Aubière, France
Email: alain.quilliot@isima.fr.

***Abstract*—This paper describes the way a multi-item, multi-plant Lot-Sizing problem with transfer costs and capacities may be reformulated according to a multi-commodity flow formalism, and decomposed, through Lagrangean Relaxation, into a master Facility Location problem and a slave Minimal Cost Multi-commodity Flow problem. This decomposition framework gives rise in a natural way to the design of a relax/project approximate algorithm, and it ends with numerical experiments.**

I. Introduction

LOT-SIZING models are production planning models which involve clustering effects: tasks may be run together as part of lots or batches [3], [4], [8], [19], [23], [12], while sharing some kind of fixed cost resource. Those problems may be set at both operational and tactical levels. Models may be dynamic or static, and involve assembly processes, one or several plants or machines, one or several kinds of products (multi-item), due dates, non deterministic demands [7]. Performance criteria may be about production cost minimization, delay or shortage minimization [1], robustness in relation to uncertain demands.

One of the major difficulties in solving Lot-Sizing problems arises from capacity restrictions in each time period. Problems are polynomially solvable when capacities are not considered, as when as when production capacities are identical in each period of the planning horizon [3], [25]. When capacities are allowed to vary over the planning horizon, the problem is NP-Hard.

Real Lot-Sizing problems are usually constrained by tight capacity restrictions, high set-up times and specific industrial constraints. The classical multi-item capacitated Lot-Sizing problem (MCLS) is widely the most widely big-bucket model studied in the context of multi-item capacitated Lot-Sizing. Nevertheless, obtaining optimal and sometimes even feasible solutions remains challenging. It has been proved [5] that the MCLS problem is strongly NP-Hard. Also, it has been shown [25], [27], that the single-item capacitated Lot-Sizing problem is NP-Hard in the ordinary sense.

Since Lot-Sizing problems have been most often proved to be NP-Hard [19], [18], it may be convenient to deal with through greedy heuristics involving ad hoc decision rules, which are usually well adapted to dynamic contexts, through heuristic algorithmic scheme, involving local search or dynamic programming [1], or through Mathematical Programming frameworks, linked to the Linear MIP/Polyhedral framework.

Several authors made appear that, in case item transfer had to be performed, Lot-Sizing could be linked with Network Flow Theory [18], [10], [21]. Network Flow Theory [2], [22], is devoted to the algorithmic handling of problems which involve the circulation of goods, people, money, energy... It has been essentially used in order to design transportation and telecommunication systems or in order to optimize gas or electricity distribution [2], [17], [15]. It proved itself to be a very powerful link between the LIP machinery and combinatorics.

There are extensions of the basic Lot-Sizing problem that can be used to model a variety of industrial problems. We refer here to the extensive literature review of [12] for an overview of the latest modelling developments in Lot-Sizing. We deal here with an extension, to the case of the existence transfer capacities, of a multi-period, multi-plant, multi-item capacitated *Lot-Sizing* problem with inter-plant transfers, which was introduced in [21], [5], [16], [13], with application to Steel Industry. While using the same kind of Lagrangean Relaxation scheme as in [21], we first make appear that this problem may be decomposed into a master *Facility Location* problem [15], [9], [24], [6], [20], [13], and a slave *Minimum Cost Multi-commodity Flow* problem. Since both *Facility Location* and *Multi-Commodity Flow* problems have been widely studied, with many applications to the design of telecommunication networks or to VLSI conception, this kind of decomposition framework opens the way to faster generic software development. Part of the difficulty consists in designing the projection part of this

Relax/Project decomposition scheme, in such a way that it ensures the satisfaction of both capacity and demand constraints. So, the rest of the paper describes algorithms which implement this decomposition scheme and tackle this feasibility problem, and the numerical experiments which were then carried on.

II. MULTI-PLANT, MULTI-ITEM LOT-SIZING WITH STORAGE AND TRANSFER CONSTRAINTS

We consider here a product (item) set K, a plant set I, and a discrete time space $T = \{0..NT\}$. A plant i in I may be at the same time producer and demander, and may also be used as an inventory place for the transfer of some items from some producer site j_1 to some demander site j_2. For every instant t in space $T = \{0..NT\}$, every item k in K, and every plant i in I, we are provided with the demand $d^k_{i,t}$ of k by i at instant t. As in [21], production requires one time unit in order to be performed, while transfers may be considered as instantaneous: some item may be produced by some plant i during period t and arrive to any plant j at the end of the same period. In case this item has to be stored at plant j, we consider that this storage transition takes place between period t and period $t + 1$. Simultaneously planning production and transfer/storage operations means dealing with the following variables:

Variables: for every $i, j \in I$, k in K, and $t \in T$, we search:
- the quantity $x^k_{i,t}$ of item k, which will be produced at plant i at time t: depending on the context, this variable may take rational or integral values;
- the boolean indicator $z^k_{i,t}$, which tells us whether plant i is effectively going to produce some item k at time t;
- the quantity $s^k_{i,t}$ of item k, which is stored at plant i between time t and time $t+1$;
- the quantity $y^k_{i,j,t}$ of item k, which is transferred from plant i to plant j at instant t.

Constraints and criteria are mainly defined by **costs** and **capacities**: As for **costs**, we denote by:
- $P^k_{i,,t}$ the *marginal production cost* of item k at plant i during period t;
- $S^k_{i,,t}$ the *marginal inventory* (storage) *cost* of item k at plant i between t and $t+1$;
- $R^k_{i,j,\,t}$ the *marginal transfer cost* of item k from plant i to plant j at time t ;
- $C^k_{\,i,t}$ the *fixed cost* related to the production of any non null amount of item k by plant i at time t.

As for **capacities**, we use the following input data:
- $1/\alpha^k_{i,\mathrm{t}}$ = production rate for item k at plant i, time t;
- $\beta^k_{i,\mathrm{t}}$ = setup time rate for item k at plant i, time t;
- $A_{\,i,t}$ = production capacity available at plant i, time t;
- $B_{i,t}$ = storage capacity available at plant i time t;
- $\gamma^k_{\mathrm{i},t}$ = storage rate for item k at plant i and time t;
- $BT_{,j,t}$ = transfer capacity from plant i to plant j at time t;
- $\tau^k_{\mathrm{i},j,t}$ = transfer rate for k from plant i to plant j at time t.

So the *Production Capacity Constraint* at plant i, time t is expressed: $\Sigma_{\,k \in K}\, \alpha^k_{i,\mathrm{t}}\, x^k_{i,t} + \Sigma_{\,k \in K}\, \beta^k_{i,\mathrm{t}}\, z^k_{i,t} \le A_{\,i,t}$,
Also, *Storage Capacity Constraint* at plant i, time t will be: $\Sigma_{\,k \in K}\, \gamma^k_{\mathrm{i},t}\, s^k_{i,t} \le B_{i,t}$, while *Transfer Capacity Constraint* related at plants i, j, time t will be: $\Sigma_{\,k \in K}\, \tau^k_{\mathrm{i},j,t}\, y^k_{i,j,t} \le BT_{,j,t}$.

This yields the following MIP (*Mixed Integer Programming*) formulation of *MSI-ST-Lot-Sizing* (*Lot-Sizing with Storage and Transfer Constraints*):

LIP Formulation for the MSI-ST-Lot Sizing Problem:

Compute $z^k_{i,t}$: Boolean, $x^k_{i,t}$, $s^k_{i,t}$, $y^k_{i,j,t}$: Integer or Rational, $i, j \in I$, $k \in K$, $t \in T$ such that:

Constraints.
For every plant i, and period t:

$\Sigma_{\,k \in K}\, \alpha^k_{i,\mathrm{t}}\, x^k_{i,t} + \Sigma_{\,k \text{ in } K}\, \beta^k_{i,\mathrm{t}}\, z^k_{i,t} \le A_{i,t}$ (*Production Capacity Constraint*);

$\Sigma_{\,k \in K}\, \gamma^k_{\mathrm{i},t}\, s^k_{i,t} \le B_{i,t}$ (*Storage Capacity Constraint*);

For every plant i, j, and period t: $\Sigma_{\,k \text{ in } K}\, \tau^k_{\mathrm{i},j,t}\, y^k_{i,j,t} \le BT_{,j,t}$ (*Transfer Capacity Constraint*);

For every plant i, item k, and period t:

$x^k_{i,t} \le N.z^k_{i,t}$, where N is some large number (*Set-up Constraint*);

$x^k_{i,t} + s^k_{i,t-1} + \Sigma_{\,j \text{ in } I}\, y^k_{i,j,t} = d^k_{i,t} + s^k_{i,t} + \Sigma_{\,j \text{ in } I}\, y^k_{j,i,t}$ (*Item Conservation Constraint*);

Minimize: $\Sigma_{\,k \in K}\, P^k_{i,t}\, x^k_{i,t} + \Sigma_{\,k \text{ in } K}\, C^k_{i,t}\, z^k_{i,t} + \Sigma_{\,k \text{ in } K}\, S^k_{i,t}\, s^k_{i,t} + \Sigma_{\,k \text{ in } K}\, R^k_{i,j,\,t}\, .y^k_{i,j,t}$.

III. MULTI-COMMODITY FLOW REFORMULATION OF THE MSI-LOT-SIZING PROBLEM

A. Recall: Network Flows and Multi-commodity Flows

Given a network $G = (V, A)$, i.e. an oriented graph with node (vertex) set V and arc set A, we say that a $\mathbb{Q}$-valued A-indexed vector f is a *flow vector* iff: (*Kirchoff's Law*) for every node z, $\Sigma_{\,e \text{ arrives into } z} f_e = \Sigma_{\,e \text{ gets out of } z} f_e$.
We denote by $Ind(f)$ the Boolean vector which takes $Ind(f)_e$, e in A, values equal to 1 value when f_e is non null. If K is some *commodity* set, then we call *multi-commodity flow* any collection $f = (f(k), k \in K)$, where every $f(k)$, $k \in K$, is a *flow vector*.

B. The Dynamic MSI-Lot-Sizing Network

We define the node set of the *MSI-ST-Lot-Sizing Network G* = (*V*, *A*) as the set $V = \{(i, t), i \in I, t \in T\}$ of all pairs (plant, instant), augmented with two auxiliary nodes *Start* and *End.* Then the arc set *A* of *G* is defined as follows:

$A = \{[(i, t), (i, t+1)], i \in I, t \in T\} \cup \{[(i, t), (j, t)], t \in T, i, j \in I\} \cup \{[Start, (i, t)], i \in I, t \in T\} \cup \{[(i, t), End], i \in I, t \in T\} \cup \{[End, Start]\}$.

The arcs of *A-Prod* = {[*Start*, (*i*, *t*)], $i \in I, t \in T$} are called the *producer arcs*; the arcs of *A-Dem* = {[(*i*, *t*), *End*], $i \in I, t \in T$} are the *demander arcs*; the arc of *A-St* = {[(*i*, *t*), (*i*, *t*+1)], $i \in I, t \in T$} are the *storage arcs*; the arc of *A-Tr* = {[(*i*, *t*), (*j*, *t*)], $i, j \in I, t \in T$} are the *transfer arcs*; the arc {[*End*, *Start*] is the *equilibrium* arc.

Any *demander* or *producer* arc e of *G* is provided in a natural way with its *time value* $t = t(e)$.

Every arc *e* in *G* may be provided as follows with fixed costs $CF(k)_e$ and marginal costs $CM(k)_e$, $k \in K$:

- If $e = [Start, (i, t)]$, $s \in I, t \in T$, is a *producer* arc then $CF(k)_e = C^k_{i,t}$ and $CM(k)_e = P^k_{i,t}$;
- If *e* is the *equilibrium* arc or is a *demander* arc, then the related costs are null;
- If *e* is a *storage* arc {[(*i*, *t*), (*i*, *t*+1)], $i \in I, t \in T$ then $CF(k)_e = 0$ and $CM(k)_e = S^k_{i,t}$;
- If *e* is a *transfer* arc {[(*i*, *t*), (*j*, *t*)], $i, j \in I, t \in T$ then $CF(k)_e = 0$ and $CM(k)_e = R^k_{i,,j,\,t}$;

By the same way, any arc *e* in *G* may be viewed as provided with set-up capacities $CAS(k)_e$, marginal capacities $CAM(k)_e$, *k* in *K*, and global capacities CA_e, according to the following formulas :

- If $e = [Start, (i, t)]$, $i \in I, t \in T$, is a *producer* arc then $CAS(k)_e = \beta^k_{i,t}$, $CAM(k)_e = \alpha^k_{i,t}$ and $CA_e = A_{i,t}$
- If e is the *equilibrium* arc or is a *demander* arc, then the related capacities are infinite;
- If e is a *storage* arc {[(*i*, *t*), (*i*, *t*+1)], $i \in I, t \in T$ then $CAS(k)_e = 0$; $CAM(k)_e = \gamma^k_{i,t}$ and $CA_e, = B_{i,t}$;
- If *e* is a *transfer* arc {[(*i*, *t*), (*j*, *t*)], $i, j,t \in I.I.T$, then $CAS(k)_e = 0$, $CAM(k)_e = \tau^k_{i,j,t}$ and $CA_e, = BT_{,j,t}$

Every *demander* arc $e = \{[(i, t), End], i \in I, t \in T$, is provided with *demand* coefficients $D(k)_e = d^k_{i,t}$.

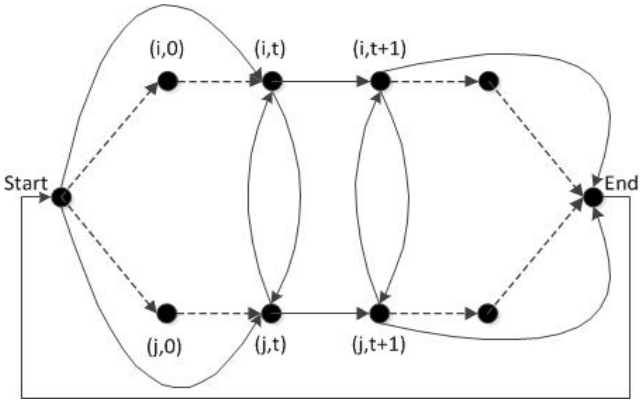

Fig. 1 The MSI-Lot-Sizing Network

C. Flow Formulation of the MSI-ST-Lot-Sizing Problem.

It comes that the *MSI-ST-Lot Sizing* Probem may be reformulated as follows:

MSI-ST-Lot Sizing-Mult-Flow: {Compute, on the *MSI-ST-Lot Sizing* dynamic graph, a multi-commodity flow $f = (f(k), k \in K) \geq 0$, such that:

- For any *demande*r arc *e*, any item *k*, $f(k)_e = D(k)_e$ (*Demand Constraints*);
- For any arc *e*, $\Sigma_{k \in K} (CAS(k)_e.Ind(f(k))_e + CAM(k)_e.f(k)_e) \leq CA_e$ (*Capacity Constraints*);
- The *cost* quantity $\Sigma_{k \in K, e \in A} (CF(k)_e.Ind(f(k))_e + CM(k)_e.f(k)_e)$ is the smallest possible}

IV. Decomposing MSI-LOT-SIZING into Facility Location and Minimum Cost Multi-Commodity Flow.

A. Recall: The Facility Location Problem

The standard *Uncapacited Facility Location* problem is defined by a set *X* (the *location* set), a function *p* from *X* to the set of the non negative rational numbers, and a *function q* from *X.X* to the set of the non negative rational numbers, such that $q(x, x) = 0$ for any *x* in *X*. The related goal is to compute some subset *Y* of *X* which minimizes the quantity $\Sigma_{x \in Y} p(x) + \Sigma_{x \in X} \text{Min}_{y \in Y} q(x, y)$. One gets an equivalent formulation by considering the unknown object as being the subset *Y*, together with, for every *location y* in *Y*, some *neighbourhood* $N(y) \subseteq X$, and subject to:

- the *neighbourhoods* $N(y)$, *y* in *Y*, define a partition of *X*;
- $\Sigma_{x \in Y} p(x) + \Sigma_{x \in N(y),\ y \in Y} q(y, x)$ is the smallest possible.

This problem was studied in an extensive way (see [9], [6], [15]). It may be handled in an exact way through LIP or Branch/Bound or in a heuristic way through the use of local search operators. For instance, one may consider the following operators, which operate on the location subset *Y*:

- *Remove*(*y*): the location *y* is removed from the subset *Y*;
- *Transfer*(*y*, *y*', *x*): *x* in *N*(*y*') is transferred into *N*(*y*);
- *Fusion*(*y*, *y*'): the location *y*' is removed from the subset *Y*, and *N*(*y*') is merged with *N*(*y*);
- *Replace*(*y*, *y*'): *y* in *Y* is replaced by *y*' in *X* – *Y*.

Then one gets a simple heuristic ***FAST-FACLOC***(*N*: Replication Number) for this problem by designing a randomized GRASP greedy + descent scheme, related to those operators, which is an implementation of the "naive" heuristics which were proposed in [6], [24]. Still, one must mention that more efficient algorithms exist [20], which provide programmers with very significant time consumption improvement.

B. A Lagrangean Relaxation Scheme

Let us denote by *A-PST* the union the *producer* arcs, the *transfer* arcs and the *storage* arcs, and by *A-ST* the difference set *A-PST* - *A-Prod*. Then any *Lagrangean Multiplier* vector is a *A-PST*-indexed vector $\lambda = (\lambda_e, e \in A\text{-}PST) \geq 0$, and the related *Lagrangean Function* $L(f, \lambda)$ comes as follows:

$$L(f, \lambda) = \Sigma_{k \text{ in } K,\, e \text{ in } A\text{-}Prod} CF(k)_e.Ind(f(k))_e + \Sigma_{k \text{ in } K,\, e \text{ in } A\text{-}PST} CM(k)_e.f(k)_e + \Sigma_{e \in A\text{-}Prod} \lambda_e.\Sigma_k CAS(k)_e.Ind(f(k))_e + \Sigma_{e \in A\text{-}PST} \lambda_e.\Sigma_k CAM(k)_e,.f(k)_e - \Sigma_{e \in A\text{-}PST} \lambda_e.CA_e.$$

Let us suppose that we are provided with some *Lagrangean Multiplier* vector $\lambda = (\lambda_e, e \text{ in } A\text{-}PST) \geq 0$. Then, for every pair of nodes (v, w) in $V - \{Start, End\}$, we may set:

- $p^k_\lambda(v) = CF(k)_{[Start,v]} + \lambda_{[Start,v]}.CAS(k)_{[Start,v]}$;
- $q^k_\lambda(v, w) = \lambda_{[Start,v]}.CAM(k)_{[Start,v]} + D(k)_{[w,End]}.Dist^k_\lambda(v, w)$, where $Dist^k_\lambda(v, w)$ is the length of a shortest path from *v* to *w* in the network *G*, computed while considering every arc *e* in *A-ST* as provided with a length $d^k_\lambda(e) = CM(k)_e + \lambda_e.\Sigma_k CAM(k)_e$.

Then, one easily checks that:

Theorem 1: Minimizing $L(f, \lambda)$ on the set of multi-commodity flows *f* which satisfy the *Demand Constraints*, means solving, for every *k* in *K*, the *Facility Location* instance FL^k_λ defined by:

- $X = V - \{Start, End\}$; $p = p^k_\lambda$; $q = q^k_\lambda$.

The related optimal value $\text{Inf}_f L(f, \lambda)$ is then equal to $W^k_\lambda - \Sigma_{e \in A\text{-}PST} \lambda_e.CA_e$, where W^k_λ is the optimal value of the *Facility Location* instance FL^k_λ.

Proof.

Minimizing $L(f, \lambda)$ on the set of multi-commodity flows *f* which satisfy the *Demand Constraints* means relaxing the capacity constraints. Then we see that, if *k* is some item in *K*, and if P_k is the set of the active producer nodes, that means the set of the nodes *v* of the network *G* which are going to bring a non null $CF(k)_{[Start,v]}$ fixed cost contribution to the quantity $L(f, \lambda)$, then every *k*-demander node *w* is going to get the whole demand $D(k)_{[w,End]}$ from its closest node *v* in P_k, in the sense of the "distance" function (it is not really a distance since $q^k_\lambda(v, w)$ is non null) $q^k_\lambda(v, w) = \lambda_{[Start,v]}.CAM(k)_{[Start,v]} + D(k)_{[w,End]}.Dist^k_\lambda(v, w)$, where $Dist^k_\lambda(v, w)$ is defined as above. More, this demand is going to be routed along a shortest path related to the $Dist^k_\lambda(v, w)$ shortest path distance. Clearly, this allows us to conclude: any feasible solution of the *Facility Location* instance FL^k_λ defined by:

- $X = V - \{Start, End\}$; $p = p^k_\lambda$; $q = q^k_\lambda$,

may be turned into a feasible solution of the relaxed multi-commodity flow problem, with same induced cost; conversely, any feasible solution of this relaxed multi-commodity flow problem may be improved by routing the $D(k)_{[w,End]}$ demands according to the smallest $q^k_\lambda(v, w)$ values, and, thus, turned into a feasible solution of the *Facility Location* instance FL^k_λ with no larger cost . *End-Theorem*.

The sub-gradient $\nabla^{f,\lambda}$ of the concave function $\lambda \geq 0 \rightarrow \text{Inf}_f L(f, \lambda)$, is defined, for every arc e in *A-PST*, by:

- if e ∈ *A-Prod*, then $\nabla^{f,\lambda}_e = \Sigma_k CAS(k)_e.Ind(f(k))_e + \Sigma_k CAM(k)_e.f(k)_e - CA_e$;
- if e ∈ *A-ST*, then $\nabla^{f,\lambda}_e = \Sigma_k CAM(k)_e.f(k)_e - CA_e$.

It comes that computing the maximal value $\text{Sup}_{\lambda \geq 0} \text{Inf}_f L(f, \lambda)$, may be done according to the following process *LAG-LOT-SIZING*:

LAG-LOT-SIZING(*N*: Integer)
Initialize λ <- 0; Not *Stop*; *W* <- 0; λ-*max* <- 0; *n* <- 0;
While Not *Stop* do
  Apply either an *exact algorithm* or an *approximate FAST-FACLOC* like procedure to the *Facility Location* instance FL^k_λ;
  Let W_λ be the obtained value;
  If $W_\lambda > W$ then W <- W_λ; λ-*max* <- λ;
  λ <– Sup(0, $\lambda + \delta_n.\nabla^{f,\lambda} / || \nabla^{f,\lambda} ||$), where $\delta = \delta_n$ is a small coefficient, and $|| \ ||$ is the Euclidean Norm;
  Update the $Dist^k_\lambda$, *k* in *K*, matrices;

If there is no improvement of the current value W during more than N iterations then *Stop*;
n <- $n + 1$;
Keep the best λ value ever obtained, denote it by λ-*max*, and denote by *A-Prod-Act* = {*A-Prod-Act*^{k}, $k \in K$} the related *producer* subset collection.

Remark 1: *Computing the incremental coefficient* $\delta = \delta_n$.
Several formulas for the δ_n coefficients exist [MIN83], which allow efficient updating of the *Lagrangean Multipliers*. Most often those coefficients define a decreasing sequence, such that $\Sigma_n\ 1/\ \delta_n = +\infty$.

Remark 2: *Incremental handling of* $FL^k{}_\lambda$ *and* $Dist^k{}_\lambda$.
Launching a *Facility Location* heuristic every time λ is updated should be avoided. Instead, a well-fitted approach consists in adapting such a heuristic, and maintaining during the whole process some collection (*population*) P of solutions of FL^k: every time λ is updated, we only perform local search on every element of P. By the same way, one should update the coefficients $Dist^k{}_\lambda(v, w)$, $v, w \in V - \{Start, End\}$, k in K, through constraint propagation. The same remark holds in case one deal with the *Facility Location* problem with an exact method.

C. A Related Projection Process

LAG-LOT-SIZING(*N*: Integer) ends while yielding some value W, together with some multiplier vector λ-*max* and some subset *A-Prod-Act*^{k} of the producer arcs set *A-Prod*. Then, turning the subset collection *A-Prod-Act* = {*A-Prod-Act*^{k}, $k \in K$} into an efficient (eventually optimal) solution of the related *MSI-ST-Lot-Sizing* instance (projecting *A-Prod-Act* onto the feasibility domain of *MSI-ST-Lot-Sizing*) means solving the following multi-commodity flow problem *Mult-Flow*$_{A\text{-}Prod\text{-}Act}$:

{Compute a multi-commodity flow $f \geq 0$, such that:
- it satisfies the *Demand Constraints*;
- for any *producer* arc e not in *A-Prod-Act*^{k}, $f(k)_e = 0$;
- for any arc $e \in$*A-Prod*,
  $\Sigma_{k \in K} CAM(k)_e.\ f(k)_e \leq$
  CA_e - $\Sigma_{k \in K \text{ such that } e \in A\text{-}Prod\text{-}Actk}\ CAS(k)_e$;
- for any arc $e \in$ *A-ST*, $\Sigma_{k \in K} CAM(k)_e.\ f(k)_e \leq CA_e$;
- the quantity $\Sigma_{k \in K,\ e \in A} CM(k)_e.\ f(k)_e$ is minimal}

It may occur that *Mult-Flow*$_{A\text{-}Prod\text{-}Act}$ does admit any feasible solution. In order to deal with this issue, we introduce an auxiliary positive vector parameter:
$\mu = (\mu_e,\ e$ in the set *A-Dem* of the *demander* arcs)
And we replace the *Mult-Flow*$_{A\text{-}Prod\text{-}Act}$ problem by the following relaxation *Mult-Flow*$^\mu{}_{A\text{-}Prod\text{-}Act}$:

{Compute a multi-commodity flow $f \leq 0$, together with a [0, 1]-valued vector $r = (r_e,\ e$ in *A-Dem*) in such a way that:
- for any *demander* arc e, any item k, $f(k)_e \geq\ r_e.D(k)_e$;
- for any *producer* arc e not in *A-Prod-Act*^{k}, $f(k)_e = 0$;
- for any arc $e \in$ *A-Prod*,
  $\Sigma_{k \in K} CAM(k)_e.\ f(k)_e \leq$
  CA_e - $\Sigma_{k \text{ in } K \text{ such that } e \in A\text{-}Prod\text{-}Actk}\ CAS(k)_e$;
- for any arc $e \in$ *A-ST*, $\Sigma_{k \in K} CAM(k)_e.\ f(k)_e \leq CA_e$;
- the quantity $\Sigma_{e \in A\text{-}Dem}\ \mu_e.r_e$, is maximal; (E1)
- the quantity $\Sigma_{k \in K,\ e \in A}\ CM(k)_e.\ f(k)_e$ is the smallest possible, (E1) being satisfied}

Then we proceed as follows:

PROJECTION(*A-Prod-ACT* = {*A-Prod-Act*^{k}, $k \in K$}: Collection of *k-producer* subsets).

Choose μ; Solve *Mult-Flow*$^\mu{}_{A\text{-}Prod\text{-}Act}$: let f be the resulting multi-commodity flow;
Compute the *residual demand* vector: for any $e \in$ *A-Dem*, $k \in K$, do $D(k)_e$ <- $D(k)_e$ - $f(k)_e$;
Kill the *A-Prod-Act* producer set: for any k in K, $e \in$ *A-Prod-Act*^{k}, do $CF(k)_e$ <- $+\infty$;
For any arc $e \in$ *A-Prod*, compute the *residual* CA_e capacity:
CA_e <- CA_e - $\Sigma_{k \in K}\ CAM(k)_e.\ f(k)_e$
- $\Sigma_{k \in K \text{ such that } e \in A\text{-}Prod\text{-}Actk}\ CAS(k)_e$;
For any arc e in *A-ST*, compute the *residual* CA_e capacity: CA_e <- CA_e - $\Sigma_{k \in K} CAM(k)_e.\ f(k)_e$.

Remark 3: *Dealing with the* *Mult-Flow*$^\mu{}_{A\text{-}Prod\text{-}Act}$ *program.*
Multi-commodity flow problems are difficult ones, even if rational numbers are involved. In such a case, the best approach consists in using a path/arc representation and implementing Dantzig-Wolfe decomposition algorithmic scheme, which involves a column generation process. In case one is required to deal with integral $f(k)_e$ values, solving *Mult-Flow*$^\mu{}_{A\text{-}Prod\text{-}Act}$ becomes NP-Hard. Still, as soon as those values are not too small, the best approach consists in first relaxing the integrality constraint, and next, in applying some ad hoc rounding procedure.

V. An Algorithm for the MSi-ST-Lot-Sizing Problem.

This algorithm works by first applying *LAG-LOT-SIZING* in order to get the *A-Prod-ACT active producer* subset collection, and by next applying *PROJECTION* in order to get some related multi-commodity flow f. Since this *PROJECTION* process may not be able to compute f in such a way the *Demand Constraint* be satisfied, *MST-ST-LOT-*

SIZE may have to enter again the global *while* main loop, while *killing* the *producer* arcs of *A-Prod-Act* and considering the *residual* demands and capacities. Once *MST-ST-LOT-SIZE* gets out of the main *while* loop, it updates the multi-commodity flow f in order to make it optimal in relation to the costs $CM(k)_e$, $e \in A$, $k \in K$.

MSI-ST-LOT-SIZE Algorithm:

D-Init, *CA-Init* and *CF-Init* are original demands, costs, and capacities; f -> 0; *Active-Prod* <- Nil; Not *Stop*;

D <- *D-Init*; *CA* <- *CA-Init*; *CF* <- *CF-Init*;

While Not *Stop* do
Apply *LAG-LOT-SIZING* while considering that *producer* arcs in *Active-Prod* are "dead" (for the related items): let *A-Prod-Act* be the resulting subset collection; (I1)
If *A-Prod-Act* = Nil then *Stop* (Fail: the *Demand Constraint* is not satisfied)
Else
Apply *PROJECTION* to *A-Prod-Act*;
If the resulting flow vector is null then *Stop* (Fail: the *Demand Constraint* is not satisfied)
Else
Insert *A-Prod-Act* in *Active-Prod*;
If *residual* demands are null then *Stop* (Success).

For any *demander* arc e, any item k, $D(k)_e$ <- *D-Init*$(k)_e$;
For any $e \in$ *A-PST*, any item k, $CA(k)_e$ <- *CA-Init*$(k)_e$;
Solve the related *Mult-Flow*$_{Active\text{-}Producers}$ program.

Remark 4: the above scheme corresponds to the case when the flow vector f takes rational values. In case f must be integral, one must refer to remark 3 of section 4.

Theorem 2: *MSI-ST-LOT-SIZE Algorithm* terminates.

Proof. Obvious, since any iteration of the main loop makes decrease the set of the node which may still be identified as produce nodes by the *Facility Location* algorithm. *End-Proof.*

Some important questions must now be raised:

First question: *what about the choice of* μ?
Clearly, what is at stake is the ability of the *MSI-ST-LOT-SIZE* Process to yield a feasible solution. So, μ must be chosen in such a way it avoids breaking the connection between *demander* arcs e and *producer* arcs e' such that $t(e) \geq t(e')$. In order to make *demander* arcs e with $t(e)$ small *time value* be prior satisfied, we make μ_e values decrease in a significant way every time $t(e)$ increases.

Second Question: *Do MSI-ST-LOT-SIZE gets a feasible solution if such a solution exists*?
Unfortunately not, even when μ is chosen as above. In order to improve the algorithm, one may compute an estimation *PL*(*A-Prod-Act*) of the ability of a subset collection *A-Prod-Act* to meet current *residual* demands, and, instead of picking up λ value which yields Sup $_\lambda$ Inf $_f$ $L(f, \lambda)$ value, select, among a package of *N1* "good" values, the one which maximizes *PL*(*A-Prod-Act*).

VI. NUMERICAL EXPERIMENTS

We performed experiments on PC AMD opteron 2.1GHz, while using gcc 4.1 compiler and dealing with the *Mult-Flow*$^{\mu}_{A\text{-}Prod\text{-}Act}$ program through the CPLEX library. Facility Location instances were alternatively handled through CPLEX (exact way) and through a *FAST-FACLOC* heuristics.

We tried several instance packages, generated as in [SAM05]. For every test, the identification parameters were (UD means here *Uniform Distribution*):

- *N-Plant* = number of plants, in {2, 3, 5, 10};
- *N-Item* = number of item, with values in {2, 3, 5, 10};
- *N-Time* = number of periods, with values in {3, 5, 10};
- *Size Instance* = triple (*N-Plant*, *N-Item*, *N-Time*);
- *MP* = *mean marginal production cost,* UD in [0,8];
- *MI* = *marginal inventory* (storage) *cost* = UD in [0, 2];
- *MT* = *marginal transfer cost*, UD in [0, 2];
- *CF* = *fixed cost* = UD in [5, 15];
- *DM* = *Mean Demand* = UD in [0, 50].

Capacities were generated as follows: a first flow f was randomly generated, in such a way it satisfies the *Demand Constraint*. A value of a *covering parameter* α was chosen in [0, 100], and CA_e values were computed in order to make possible an increase f on the *Equilibrium* arc by α%.

For every test, we computed: (all but *Ite* are percentages):

- *Ite* = iteration number of MSI-ST-LOT-SIZE;
- *Gap* = gap (W- W_{Opt}) / W_{Opt} between the value W produced by *MSI-ST-LOT-SIZE* and the optimal value W_{Opt} obtained while using the CPLEX library.
- *GLg* = gap (W- W_{Lag}) / W_{Lag} between W and the value W_{Lag} produced by the *LAG-LOT-SIZING* Process.
- *Def* = percentage of the whole demand which could not be satisfied after the 1th iteration of *MSI-ST-LOT-SIZE*.

- *GLP* = gap (W- W_{LP}) / W_{LP} between W and W_{LP} produced by the LP relaxation of *MSI-ST-Lot Sizing*.
- *GPr* = gap (W_{Opt} - W_{Opt1})/ W_{Op1} between W_{Opt} and the optimal value W_{Opt1} of the problem obtained by only taking into account production capacity constraints.
- *GFr* = gap (W_{Opt} - $W_{Opt\text{-}free}$) / $W_{Opt\text{-}free}$ between W_{Opt} and $W_{Opt\text{-}free}$ related to the problem without any capacity.

A. Experiments with Facility Location handled through CPLEX Libraries

In the case that we handled the *Facility Location* instances in an **exact way** (through CPLEX) we got:

TABLE I: GAP ANALYSIS

Size	*Ite*	*Gap*	*GLg*	*GLP*	*GPr*	*GFr*	*Def*
3*5*3	1	0	5.4	15.6	0.8	85	0
3*5*3	1	0	7.5	12.7	0.6	54	0
3*5*3	1	0	0.8	15.9	5.8	32.5	0
4*10*6	1	1.7	0.2	15.6	2.9	21.5	0
4*10*6	1	0	0.2	20.3	3.8	9.2	0
4*10*6	2	1.0	5.4	13.8	20.0	25.4	3.5
5*10*10	1	0.7	0.6	19.7	5.2	20.2	0
5*10*10	1	0.7	1.5	15.3	6.7	19.1	0
5*10*10	1	0.1	0.4	20.2	5.5	14.7	0
8*10*15	1	0	0.9	13.5	2.6	23.6	0
8*10*15	1	0.8	3.4	12.8	3.7	49.0	0
8*10*15	2	1.5	4.5	19.4	7.5	58.3	2.2
10*15*15	1	1.2	2.6	23.7	3.4	37.2	0
10*15*15	1	0	0.9	14.5	3.5	29.6	0
10*15*15	3	2.5	6.5	28.2	11.8	82.0	4.8

Comments: Lagrangean relaxation yields good approximations, if we compare it with the usual LP bound. The projection mechanism, though it most often fails in meeting demands at the end of the first iteration, still gets close to those demands and usually meets them fast.

B. Experiments with Facility Location handled through FAST-FACLOC heuristics

In the case that the size of the instance is large, or in case a LIP library is not available for the programmers, it may be necessary for them to develop their own procedure for the handling of the intermediate location problem. What is interesting to study in such a case, is the way approximation induced by the application of such a heuristic procedure is going to impact the final result. In order to understand it, we used the same instances as in table 1, and, for every instance, we computed the mean error (as a rate) performed by our ad hoc *FAST-LOC* procedure, and compared it with the final error. The *FAST-FACLOC* like heuristic which we used was a rough adaptation (with significantly worse performances) of the sophisticated algorithm of [20). Then we denoted by:

- *Gap-Approx* and *GLg-Approx* are the increase rate (in %= with relation respectively with *Gap* and *GLg* of experiment A, due *Facility Location* handling in a heuristic way;
- *Gap-Loc-Approx*: the mean error (in %) induced by the use of the *FAST-LOC* Heuristic instead of CPLEX on the resolution of the intermediate *Facility Location* sub-problem.
- *Ite-Approx* and *Def-Approx*: as in experiment A, in case *Facility Location* is handled in a heuristic way;

Then we got:

TABLE 2: SENSIBILITY ANALYSIS: HEURISTIC VERSUS EXACT HANDLING OF THE FACILITY LOCATION SUB-PROBLEM

Size	*Ite-Approx*	*Gap-Approx*	*GLg-Approx*	*Gap-Loc-Approx*	*Def-Approx*
3*5*3	2	10.8	6.7	6.5	2.3
3*5*3	1	2.0	2.0	2.1	0
3*5*3	1	12.6	12.8	12.5	0
4*10*6	2	9.9	5.1	3.8	1.8
4*10*6	1	6.8	6.7	6.4	0
4*10*6	1	0.1	1.5	1.9	0
5*10*10	2	11.0	9.6	8.5	5.2
5*10*10	2	20.7	15.0	13.4	2.1
5*10*10	1	8.4	6.9	7.7	0
8*10*15	1	5.0	5.1	4.8	0
8*10*15	1	14.5	16.4	15.2	0
8*10*15	3	10.2	4.5	3.9	6.2
10*15*15	1	12.5	14.9	10.6	0
10*15*15	1	7.0	6.8	7.3	0
10*15*15	2	3.2	7.6	5.5	2.8

Comment: We can see here the *GLg* increase is closely following the gap between optimality the result produced by this heuristic. Still, impact on the final value *Gap* was more

difficult to predict, due to the dependency of the behaviour of the projection mechanism to the characteristics of the producer set obtained at the end of the *LAG-LOT-SIZING* process.

VI. Conclusion

We just described the way *Lot-Sizing* might be decomposed, into *Facility Location* and *Network Multi-Commodity Flow*. Since both *Facility Location* and *Network Multi-Commodity Flow* problems have been widely studied, casting *Lot-Sizing* problems into such a decomposition framework opens a way to fast design of generic software components. It would be interesting to go further, and better identify those *Lot-Sizing* problems, which may be cast into such a decomposition scheme.

References

[1] N. Absi, S. Kedad-Sidhoum: "The multi-item capacitated lot sizing problem with setup times and shortage costs"; *EJOR* 185-3, p 1351-1374, 2008.

[2] R. V. Ahuja, T. L. Magnanti J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*, Prentice hall, Englewood Cliffs, N.J, 1993.

[3] A. Allahverdi, C. T. Ng, T. C. Cheng, M. Y. Kovalyov: "A survey of scheduling problems with set up times or costs"; *EJOR* 187, p 985-1032, 2008.

[4] J. J. Blasewicz, K. H. Ecker, G. Schmlidt, J. Weglarcz. *Scheduling in Computer and Manufacturing Systems*, 2 th edn, Springer-Verlag, Berlin, 1993.

[5] W. H. Chen, J. M. Thizy: "Analysis of relaxations of the multi-item capacity lot-sizing problem"; *Annals of operations Research* 26, p 29-72, 1990.

[6] G. Cornuejols, G. L. Nemhauser, L. A. Wolsey: "The uncapacited facility location problem", in P. B. Mirchandani, R. L. Francis Eds, *Discrete Location Theory*, p 119-171, 1990.

[7] A. Dolgui, J. M. Proth: *Supply Chain Engin.: Methods/Techniques*; Springer, 2010.

[8] A. Dolgui, A. Eremeev, M. Kovalyov, P. Kuznetsov: "Multi-product lot sizing/scheduling on parallel machines"; *IIE Trans*. 42, p 514-524, 2010.

[9] G. Ghiani, F. Guerrero, F. Musmanno: "The capacitated plant location problem with multiple facilities in the same site"; *Computer and Operations Research* 29-13, p 1903-12, 2002.

[10] M. Haouari, A. Gharbi. "A improved max-flow based lower bound for minimizing maximum lateness on parallel machines", *O. R Letters* 31, pp. 49-52, 2003.

[11] P. Hansen, J. Brimberg, D. Urosevic, N. Mladenovic: "Primal-dual local search for plant location", *INFORMS Jour. Comp.* 19-4, p 58-72, 2007.

[12] R. Jans, Z. Degraeve: "Modeling industrial lot sizing problems: a review"; *International Journal of Production Res.* 46-6, p 1619-1643, 2008.

[13] A. Klose, A. Drexl: "Facility location models for distribution systems"; *EJOR* 162, p 429-449, 2005.

[14] M. Minoux: *Programmation Mathématique : Théorie et Algorithmes*, Vol 1, DUNOD , 1983).

[15] P.B. Mirchandani, R.L. Francis. *Discrete Location Theory*. J.WILEY SONS, 1990.

[16] L. Ozdamar, G. Barbatroglu: "A lagrangean relaxation simulated annealing approach to multilevel capacitated lot-sizing"; *Int. Jour. Prod. Eco.* 68-3, p 319-331, 2000.

[17] P. M. Pardalos, D. Z. Du : *Network design: connectivity and facility location*, DIMACS Series 40, N.Y, American Math Society, 1998.

[18] Y. Pochet, L. A. Wolsey: *Production Planning by Mixed Integer Programming* ; Springer Series in Operat. Res. And Financial Engineering, 2006.

[19] C. N. Potts, N. Y. Kovalyov: "Scheduling with batching: a review"; *EJOR* 120, p 228-249, 2000.

[20] M. G. Resende, R. F. Werneck: "A hybrid heuristic for the p-median problem"; *Jour. Heuristics* 10, p 59-88, 2004.

[21] M. Sambivasan, S. Yahia "A Lagrangean based heuristic for multi-plant, multi-item, multi-period capacited lot sizing problems with inter-plant transfers"; *Computers and Operations Res.* 32, p 537-552, 2005.

[22] E. Tardos: "A strongly polynomial minimum cost circulation algorithm"; *Combinator.* p 247-55, 1985.

[23] W. Trigerio, L. J. Thomas, J. O. Mac Clain: "Capacited Lot Sizing with set up times"; *Management Sciences* 35-3, p 353-66, 1989.

[24] R. Whitaker: "A fast algorithm for the greedy interchange of large scale clustering and median location problems"; *INFOR* 21, p 95-108, 1983.

[25] G. Bitran, H. Yanasse: "Computational complexity of the capacitated lot size problem", *Management Sciences* 28, p 1174-1186, 1982.

[26] M. Florian, M. Klein: "Deterministic production planning with concave costs and capacity constraints"; *Management Sciences* 18, p 12-20, 1971.

[27] M. Florian, J. Lenstra, A. R. Kan: "*Deterministic production planning: algorithm and complexity*"; Management Sciences26, p 669-679, 1980.