OPERATIONS RESEARCH
Vol. 52, No. 4, July–August 2004, pp. 515–526
ISSN 0030-364X | EISSN 1526-5463 | 04 | 5204 | 0515

DOI 10.1287/opre.1040.0108

The Optimal Diversity Management Problem

Olivier Briant
Laboratoire Mathématiques Appliquées de Bordeaux, 351 cours de la libération, 33405 Talence, France,
olivier.briant@math.u-bordeaux.fr

Denis Naddef
Laboratoire Informatique et Distribution—IMAG, Institut National Polythechnique de Grenoble, 51 avenue Jean Kuntzmann,
38330 Montbonnot, France, denis.naddef@imag.fr

In some industries, a certain part can be needed in a very large number of different configurations. This is the case, e.g., for the electrical wirings in European car factories. A given configuration can be replaced by a more complete, therefore more expensive, one. The diversity management problem consists of choosing an optimal set of some given number k of configurations that will be produced, any nonproduced configuration being replaced by the cheapest produced one that is compatible with it. We model the problem as an integer linear program. Our aim is to solve those problems to optimality. The large-scale instances we are interested in lead to difficult LP relaxations, which seem to be intractable by the best direct methods currently available. Most of this paper deals with the use of Lagrangean relaxation to reduce the size of the problem in order to be able, subsequently, to solve it to optimality via classical integer optimization.

Subject classifications: large-scale integer programming and relaxations: Lagrangean relaxation of a large linear integer program arising from an application; production planning: choice of production.
Area of review: Optimization.
History: Received May 2000; revisions received May 2002, January 2003; accepted June 2003.

Introduction

In some industries, the number of different configurations of a given part exceeds by far the acceptable number on the assembly line. This is the case, e.g., for the electrical wirings in the European car factories. Due to a large range of options, European manufacturers consider up to 7,000 different wiring designs. They produce, however, only a much smaller number of $k = 6$ up to $k = 40$ configurations, depending on the manufacturer. If a wiring that is not manufactured is needed, then it is replaced by a compatible one of minimum cost among those produced, that is, one that has all the options required plus some others. This implies an overcost.

The *Optimal Diversity Management Problem* (*ODMP*) is that of choosing optimally the k models that will be manufactured, i.e., the choice that minimizes the total overcost. This problem is *NP*-hard (see Briant 2000).

A problem similar to the ODMP has been studied by Thonemann and Brandeau (2000). They consider a much smaller number of possible models (200), but also a much more complex cost function that takes into account inventory costs, set-up costs, and a factor of complexity of the chosen models.

In our case it seems realistic to only consider minimizing the total overcost in manufacturing.

Our approach is based on a classical integer linear programming model for the p-median problem, which we give in §1. It turns out that the large-scale instances we are interested in lead to difficult LP relaxations, which seem to be intractable by the best direct methods currently available.

The main purpose of this paper is to try to drastically reduce the problem size in order to be able to solve to optimality large instances of the problem. Problem reduction is a very common technique in integer programming; see, for example, Andersen and Andersen (1995), Crowder et al. (1983), Martin (1998), Hoffman and Padberg (1991), Suhl and Szymanski (1994), and Martin (2001).

We will perform problem reduction using a lower bound obtained via Lagrangean relaxation. This reduction will be obtained by variable fixing, either by a reduced cost argument or by logical implications. Beasley (1987) applied similar techniques for the set-covering problem.

The main ideas are exposed in §2. The list of all the variable fixings is then given in §3. Some of these variable fixings are more efficient if one knows the value of a very good feasible solution. In §3.6, we give heuristics that aim to obtain such solutions. These heuristics are driven by the Lagrangean relaxation solution.

In §4 we will see that the problem reduction is so successful that it either solves the problem or prepares it for a subsequent successful application of a state-of-the-art ILP solver such as CPLEX MIP. We also comment on alternative solution approaches.

1. The Model

The ODMP is a particular case of several well-known difficult problems. For example, it can be seen as a k-median or k-center problem on the graph of a partial order described

below. Another approach would be to consider the problem as a set-partitioning problem, that is, to partition the set of models into k subsets, each containing an element that can replace all the other elements of the subset. This approach leads naturally towards a column-generation solution technique, which may be another way of addressing this problem. One could, for example, try to use techniques similar to those used by Caprara et al. (1996) and by Ceria et al. (1998) for large set-covering problems in relation to railway timetable scheduling.

1.1. General Remarks and Notation

An instance of ODMP is defined by a given number of models $N = \{1, \ldots, i, \ldots, n\}$, their costs $\{c_1, \ldots, c_i, \ldots, c_n\}$, and their demands $\{d_1, \ldots, d_i, \ldots, d_n\}$, together with a partial order on the set of models $\{1, \ldots, i, \ldots, n\}$, where $i \prec j$ stands for i can be replaced by j and $i \neq j$. We let $i \preceq j$ stand for $i \prec j$ or $i = j$. If $i \prec j$, then $c_i < c_j$. In our wiring application, a model has p possible options that can be described by a p-string of 0 and 1, where a 1 in position t means that option t is present. It is then easy to check if $i \prec j$; it can be done by bit comparisons. The string corresponding to j must have a 1 in every position where i does.

Later, we will take the k smallest entries of some sets of values $\{a_j \in \mathbb{R}: j \in B \subset N\}$. We will denote the indices of these values by $\arg\min_{j \in B}^{k} a_j$, breaking ties arbitrarily.

1.2. A Graph Associated with ODMP

It is quite natural to associate with the problem the directed graph $G = (N, A)$, with nodeset $N = \{1, \ldots, i, \ldots, n\}$ representing the models, and arc set $A = \{(i, j): i \prec j\}$. From now on, we will not distinguish between the nodeset of this graph and the set of models; this is the reason we represent them by the same symbol N.

Some models have a zero demand, but may be useful because they can replace many other ones. We let $N^\star$ represent the set of models with non-zero demands.

We will also be interested in solutions that necessarily contain the models in some set N_1, and do not contain the models in some set N_0. Note that N_1 will always contain, among other models, those models with nonzero demand that cannot be replaced by any other one. We call these models *maximal* or *terminal*. These are the maximal elements of the partial order.

Similarly, we will be interested in solutions in which substitutions in some set A_1 are forced, and in which substitutions in some set A_0 are forbidden. We assume that these sets are consistent; that is, if $(i, j) \in A_1$, then $j \in N_1$. From the start $N_1 \neq \varnothing$, since all maximal models must be produced. If there is more than one maximal model, then also $A_0 \neq \varnothing$, as will become clear in §3.

In order to efficiently perform most of the operations described in this paper, it is necessary to implement the data optimally. The graph is implemented as a list of nodes representing the models with a linked list of successors and another linked list of predecessors. It is then important that both lists be sorted from the cheapest element to the most expensive one.

1.3. The Integer Programming Model

For $i \prec j$, let $x_{ij} = 1$ if model i is replaced by model j and 0 if not. When $i = j$, $x_{ii} = 1$ will mean that the model i is produced. The cost of replacing model i by model j is $d_i c_j$, and the overcost is $d_i(c_j - c_i)$.

Minimizing the total cost or the total overcost is equivalent; these two values only differ by a large constant. For notational simplicity, we will be minimizing the total cost.

The problem can then be modeled by the following Integer Linear Program, which we denote by *Choice ILP*:

$$\text{minimize} \quad \sum_{j \in N} \sum_{i \preceq j} c_j d_i x_{ij} \tag{1.1}$$

such that

$$\sum_{j \succeq i} x_{ij} = 1 \quad \forall\, i \in N^\star \tag{1.2}$$

$$\sum_{i \in N} x_{ii} = k \tag{1.3}$$

$$x_{ij} \leqslant x_{jj} \quad \forall\, i, j \in N,\ i \prec j \tag{1.4}$$

$$x_{ij} \geqslant 0 \quad \forall\, i, j \in N,\ i \prec j \tag{1.5}$$

$$x_{jj} \in \{0, 1\} \quad \forall\, j \in N \tag{1.6}$$

Equations (1.2) just state that a model is either produced or replaced by exactly one other. Equation (1.3) says that exactly k models must be produced; Inequalities (1.4) state that in order to replace model i by model j, model j has to be produced. The last conditions are just the integrality requirements on the variables, which are only necessary for the x_{jj} variables.

Inequalities (1.4) are often referred to in the literature as *variable upper bounds*. The number of these inequalities can be very large, although bounded by $n(n-1)/2$.

Note that we could have added $x_{jj} = 1$ if model j is maximal. We will come back to this later on, since one can in general do even more.

The *linear relaxation* of the previous integer program is the *linear program* obtained by replacing Conditions (1.6) by $x_{jj} \geqslant 0$. Note that because of Conditions (1.2) and the nonnegativity of the variables, we have $0 \leqslant x_{ij} \leqslant 1$ for all $i \preceq j$.

The classical way of solving this linear program (see Chapter II.4 of Nemhauser and Wolsey 1988) is to first solve the linear program obtained by deleting all the variable upper-bound inequalities (1.4). Let $x^\star$ be the optimal solution of that linear program. Iterate the following steps as long as necessary:

- Determine all, or a subset of all, variable upper-bound inequalities violated by $x^\star$. If none, stop.
- Add these inequalities to the current linear program.

• Solve the new linear program to obtain the new optimal solution $x^\star$.

None of the three steps poses a problem. Nevertheless, this approach fails in our case. The reason is that a very high percentage of the variable upper-bound constraints will have to be added to the linear program. In order to keep the linear program of manageable size, we occasionally have to remove those inequalities that are no longer active. We then have the phenomenon that some inequalities enter (and exit) the linear program several times, and convergence of the process is never reached for large instances.

The failure of this procedure is the reason for the following developments.

1.4. A Hypergraph Associated with ODMP

This hypergraph will only be used in the problem reduction part. It could be the basis for a solution method based on set partitioning by methods similar to those developed by Caprara et al. (1996) and by Ceria et al. (1998). With an instance we associate a hypergraph. The nodes of the hypergraph represent the models. For each model i, we generate an edge E_i of the hypergraph that corresponds to those nodes that can be used to replace i. Note that we have $i \in E_i$. A *transversal* T is a set of nodes that intersects all the edges. The diversity management problem reduces to finding a transversal of cardinality k of minimum cost. Since edges containing other edges are useless in this context, we keep only minimal edges (by inclusion), and remove the other ones. By removing an edge, we mean deleting it from the edge set; the removal does not affect the node set. If several edges are identical, we only keep one.

If we are interested in solutions that contain all models in a set N_1, we delete from this hypergraph all edges containing a node that corresponds to a model in N_1. Note that an edge consisting of a unique node has to appear in all solutions and therefore the corresponding model can be added to N_1, so we assume that there are no such edges. We call this hypergraph the *free-node hypergraph*. Note that any feasible solution contains at most $k - |N_1|$ nodes of this hypergraph. The reason that not all $k - |N_1|$ nodes must be chosen is that we removed all nonminimal edges, and therefore there are nodes that may be chosen that do not appear in the hypergraph.

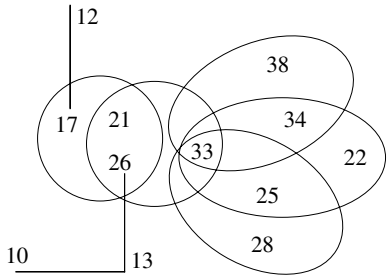
Figure 1 shows an example of a free-node hypergraph with $k = 10$ and $|N_1| = 6$, and therefore we are left with four models from which to choose. In that drawing, edges with two nodes are drawn as edges of a graph.

The free-node hypergraph will be used to reduce the size of the problem in several ways. It is also crucial in driving a branch-and-bound or branch-and-cut algorithm.

2. Our Lagrangean Relaxation Algorithm

Let us relax in a Lagrangean way the Equations (1.2), which are those that make the problem difficult, as we will see. Let u_i represent the multiplier associated with Equations (1.2) corresponding to i.

Figure 1. Example of a free node hypergraph.



For notational convenience, we will assume that a multiplier u_i exists for $i \in N \backslash N^\star$ (i.e., such that $d_i = 0$), which can only take the value 0.

The objective function becomes:

$$\text{minimize} \quad \sum_{j\in N}\sum_{i\preceq j} c_j d_i x_{ij} + \sum_{i\in N^\star} u_i\left(1 - \sum_{j\succeq i} x_{ij}\right).$$

Rearranging the terms in that objective function we get:

$$\text{minimize} \quad \sum_{i\in N^\star} u_i + \sum_{j\in N}\sum_{i\preceq j}(c_j d_i - u_i)x_{ij}.$$

Let us consider a more restricted problem, which in fact will be the one we will need. Let N_1, N_0, A_1, and A_0 be as defined earlier.

The Lagrangean relaxation we have to solve, then, is the following:

$$\text{minimize} \quad \sum_{i\in N^\star} u_i + \sum_{j\in N}\sum_{i\preceq j}(c_j d_i - u_i)x_{ij} \tag{2.1}$$

such that

$$\sum_{i\in N} x_{ii} = k \tag{2.2}$$

$$x_{ij} \leqslant x_{jj} \quad \forall\, i, j \in N,\ i \prec j \tag{2.3}$$

$$x_{ij} \geqslant 0 \quad \forall\, i, j \in N,\ i \prec j \tag{2.4}$$

$$x_{jj} \in \{0, 1\} \quad \forall\, j \in N \tag{2.5}$$

$$x_{jj} = 0 \quad \forall\, j \in N_0 \tag{2.6}$$

$$x_{jj} = 1 \quad \forall\, j \in N_1 \tag{2.7}$$

$$x_{ij} = 0 \quad \forall\, (i, j) \in A_0 \tag{2.8}$$

$$x_{ij} = 1 \quad \forall\, (i, j) \in A_1 \tag{2.9}$$

For all $j \in N$ and $i \preceq j$, we will call $\gamma_{ij}(u) = d_i c_j - u_i$ the *reduced cost* of variable x_{ij} (which is the reduced cost of arc (i, j), if $i \neq j$, referring to the arc of the underlying graph).

Given values $\bar{u}_i$ to the multipliers u_i for $i \in N^\star$, we now solve the previous Lagrangean relaxation.

Let

$$S_j(\bar{u}) = \gamma_{jj}(\bar{u}) + \sum_{i\prec j,\,(i,j)\notin A_0\cup A_1} \min(0, \gamma_{ij}(\bar{u})) + \sum_{i\prec j,\,(i,j)\in A_1} \gamma_{ij}(\bar{u}),$$

and let $K(\bar{u})$ be the set N_1 completed by $k - |N_1|$ models $j \in N \setminus (N_0 \cup N_1)$ having the smallest values $S_j(\bar{u})$, i.e.,

$$K(\bar{u}) = N_1 \cup \arg\min_{j \notin N_1 \cup N_0}^{k-|N_1|} S_j(\bar{u}),$$

then

$$L(\bar{u}) = \sum_{j \in N^\star} \bar{u}_j + \sum_{j \in K(\bar{u})} S_j(\bar{u}).$$

In the computation of $S_j(\bar{u})$ only the incoming arcs that are forced (in A_1), or those with a negative reduced cost and acceptable (not in A_0), contribute to $S_j(\bar{u})$.

The associated optimal solution $\bar{x}$ of the Lagrangean relaxation is defined by setting $\bar{x}_{jj} = 1$ for $j \in K(\bar{u})$, $\bar{x}_{jj} = 0$ otherwise, and $\bar{x}_{ij} = 1$ if and only if $(i, j) \in A_1$, or $j \in K(\bar{u})$ and $i \prec j$, $(i, j) \notin A_0$ with $\gamma_{ij}(\bar{u}) < 0$.

Note that the solution $\bar{x}$ is always integral. Therefore, the maximum value over all possible multipliers of (2.1) subject to Constraints (2.2)–(2.9) theoretically cannot be a better lower bound than the one we would obtain solving the linear relaxation (see Geoffrion 1974). However, because variable fixing changes the space we deal with, we will see that we will very often do better than the LP bound associated with the *original* choice ILP (1.1)–(1.6).

We use the following algorithm to solve the Lagrangean relaxation (2.1)–(2.9); i.e., to maximize the bound $L(\bar{u})$ and simultaneously reduce the problem as much as possible.

Step 0. Initialize the multipliers and the sets N_0, N_1, A_0, and A_1.

Step 1. Repeat the following steps until a stopping criterion is reached:

(a) Solve the Lagrangean relaxation to obtain $\bar{x}$ and $L(\bar{u})$. Eventually, update the global lower bound GLB.

(b) Perform problem reduction. If the reduction changes the primal solution $\bar{x}$, go back to (a).

(c) Perform a primal bounding. Eventually update the global upper bound GUB.

(d) Change the multipliers.

In the remainder of this article, we develop the individual parts of this algorithm. We develop the problem reduction in §3. It aims at increasing as much as possible the sets N_0, N_1, A_0, and A_1. The primal bounding will be developed in §3.6 and aims at finding better feasible solutions, that is, at reducing GUB. The better GUB, the better some of the variable fixings are.

The stopping criterion is either that $L(\bar{u})$ rounded up equals GUB or a maximum number of iterations has been reached.

The change of multipliers is done by classical subgradient optimization and will not be developed here (see Beasley 1993b). We have also tested the bundle method of Lemaréchal (see Lemaréchal and Sagastizábal 1997, Hiriart-Urruty and Lemaréchal 1993). The results with this method can be found in Briant (2000).

3. Problem Reduction or Variable Fixing

3.1. Node and Arc Fixing

A variable x_{ij} for $i \preceq j$ can be *fixed* to the value $\delta \in \{0, 1\}$, if by doing so one does not lose all optimal solutions. Most of the time, in this paper, the variables will be fixed to $\delta \in \{0, 1\}$ if *all* optimal solutions have the variable at that value. The sets N_1, N_0, A_1, and A_0 correspond, respectively, to the set of variables x_{jj} fixed to 1 and 0, and the set of variables x_{ij} for $i \prec j$ fixed to 1 and 0. Note that fixing a variable x_{jj} to 0 does not eliminate the corresponding model from the problem because it has to be replaced by another model.

The variable fixings are of two types. The ones that we call *fixing by logical implications* are consequences of former fixings; these generally make use of the fact that $c_i < c_j$ when $i \prec j$. These create a snowball effect in problem reduction and have to be applied recursively. Also, as will become clear very soon, in order for these to be efficient, node fixing must precede arc fixing in these procedures. The other variable fixings that we call *fixing by reduced costs* exploit the current Lagrangean relaxation.

3.2. Fixing by Logical Implications

We first give a list of basic fixings by logical implications.

For all $i \in N$, let $\delta^+(i) = \{(i, j) \in A\} = \delta_1^+(i) \cup \delta_0^+(i) \cup \delta_{nf}^+(i)$ be the set of arcs going out of i, partitioned into the sets of the outgoing arcs fixed to 1, fixed to 0, and not fixed, respectively. We also define $\delta^-(i) = \{(j, i) \in A\} = \delta_0^-(i) \cup \delta_1^-(i) \cup \delta_{nf}^-(i)$ as the partition of the incoming arcs to i.

FLI 1: **If** $\exists\, i \in N_0$,
Then $A_0 \leftarrow A_0 \cup \delta_{nf}^-(i)$

FLI 2: **If** $\exists\, i \in N_1$,
Then $A_0 \leftarrow A_0 \cup \delta_{nf}^+(i)$

FLI 3: **If** $\exists\, (i, j) \in A_1$,
Then
(a) $N_1 \leftarrow N_1 \cup \{j\}$
(b) $N_0 \leftarrow N_0 \cup \{i\}$
(c) $A_0 \leftarrow A_0 \cup \delta_{nf}^+(i)$
(d) $N_0 \leftarrow N_0 \cup \{t \succ i\colon c_t < c_j\}$

FLI 4: **If** $\exists\, i \notin N_1\colon \delta^+(i) \subseteq A_0$,
Then $N_1 \leftarrow N_1 \cup \{i\}$

FLI 5: **If** $\exists\, i \in N_0,\ \exists\, j \succ i\colon \delta_{nf}^+(i) = \{(i, j)\} \wedge \delta_1^+(i) = \varnothing$
Then $A_1 \leftarrow A_1 \cup \{(i, j)\}$

FLI 6: **If** $\exists\, i \notin N_1, \exists\, j \succ i\colon (i, j) \in \delta_{nf}^+(i) \wedge j \in N_1$
Then $A_0 \leftarrow A_0 \cup \{(i, t) \in \delta_{nf}^+(i)\colon c_t > c_j\}$

THEOREM 3.1. *The variable fixings* FLI 1 *to* FLI 6 *are legal.*

PROOF. All these fixings are easily understood. We explain the least trivial one only, that is, FLI 3 (d). If an optimal solution, with j chosen, contained as chosen a model t such

that $t \succ i$ and $c_t < c_j$, then i would be replaced by t and not by j, which contradicts the fact that (i, j) is in A_1. □

REMARK 3.2. Generally, there are several maximal models with nonzero demands. By rule FLI 6, the problem can be reduced from the very beginning by generating only one arc from any node to those maximal ones—namely, the arc going to the cheapest one. In case of ties, it is not important which one is kept.

There are a few less basic fixings by logical implications:

FLI 7: **If** $\exists\, i \notin N_1$: $|\delta^+_{nf}(i)| > |N| - k$
Then $A_0 \leftarrow A_0 \cup A'$
where $A' = \delta^+_{nf}(i) \backslash \underset{j \in \delta^+_{nf}(i)}{\arg\min^{|N|-k}} c_j$

FLI 8: **If** $i \prec j \prec \ell$
(a) **If** $(i, \ell) \in A_1$
Then $A_1 \leftarrow A_1 \cup \{(j, \ell)\}$
(b) **If** $(j, \ell) \in A_0$
Then $A_0 \leftarrow A_0 \cup \{(i, \ell)\}$

FLI 9: **If** $\exists\, i \in N_0,\ \exists\, j^\star \succ i$: $\{(i, j) \in A$: $c_j < c_{j^\star}\} \subseteq \delta^+_0(i)$
Then $N_0 \leftarrow N_0 \cup \{j \succ i$: $c_j < c_{j^\star}\}$

FLI 10: **If** $\exists\, i \in N_0, \exists\, j^\star \succ i$: $\{(i, j) \in A$: $c_j < c_{j^\star}\}$
$\subseteq \delta^+_0(i) \wedge j^\star \in N_1$
Then $A_1 \leftarrow A_1 \cup \{(i, j^\star)\}$

THEOREM 3.3. *The variable fixings* FLI 7 *to* FLI 10 *are legal.*

PROOF. The reason for FLI 7 is that if i was replaced by one of the $|\delta^+_{nf}(i)| - |N| + k$ most expensive successors, then by the basic variable fixings FLI 3 (b) and FLI 3 (d), i and the $|N| - k$ cheaper nodes $j \succ i$ would be fixed to 0, and there would be less than k left. FLI 8 is less obvious. If i is replaced by ℓ, then j, which is more expensive, also should be. If not, j would be replaced by something cheaper than ℓ, and by transitivity, then so should i be since it is possible. The two last fixings deal with the replacement of a model i, we know is not chosen in any optimal solution. The arcs to all the cheapest replacements are fixed to 0, i.e., those cheaper than $j^\star$. If any one of these models appears in an optimal solution, this would contradict the fixing of the arc (i, j) to 0 therefore these models appear in no optimal solution. Then if the cheapest possible replacement of i is fixed to 1, we can fix to 1 the arc joining i to it. Note that if there are other models of the same cost, there may be other optimal solutions in which i is replaced by another model. This is an exception to the fact that we fix only if all optimal solutions have the variable at the corresponding value. □

3.3. Fixing by Reduced Costs

Fixing by reduced costs consists of setting a not-yet-fixed variable x_{ij} to the value $\delta \in \{0, 1\}$, and of computing the optimal value $L(\bar{u})|_{x_{ij}=\delta}$ of the Lagrangean relaxation subject to the additional restriction $x_{ij} = \delta$. If $L(\bar{u})|_{x_{ij}=\delta} > GUB$, which means that there is no optimal solution of our problem having x_{ij} equal to δ, then we can fix x_{ij} to the opposite value $1 - \delta$.

Variable fixing in relation to Lagrangean relaxation can be found in Mulvey and Crowder (1979) and Beasley (1993a).

Given $j \in N$ and $i \preceq j$, denote further by

$$L(\bar{u})|_{ij} := L(\bar{u})|_{x_{ij}=1} \quad \text{and} \quad L(\bar{u})|_{\neg ij} := L(\bar{u})|_{x_{ij}=0}$$

the value of the Lagrangean relaxation subject to the additional restriction that i is replaced by j (or i is produced if $j = i$), and that i is not replaced by j (or i is not produced if $j = i$), respectively.

Let $S^{\uparrow 1}(\bar{u}) = \max\{S_j(\bar{u})$: $\bar{x}_{jj} = 1$ and $j \notin N_1\}$, and $S^{\downarrow 0}(\bar{u}) = \min\{S_j(\bar{u})$: $\bar{x}_{jj} = 0$ and $j \notin N_0\}$. In other words, $S^{\uparrow 1}(\bar{u})$ is the largest value of a $S_j(\bar{u})$ for a j that appears in the current Lagrangean solution and that is not fixed to 1, and $S^{\downarrow 0}(\bar{u})$ is the smallest value of a $S_j(\bar{u})$ for a j that does not appear in the current Lagrangean solution and that is not fixed to 0.

The different fixings will be split into three groups that represent the amount of computational effort required to recompute the value of $L(\bar{u})$, subject to some fixing. The first ones require only the computation of simple formulas; the last one is quite complex. One first performs the operations of the first group; when none succeed we go to those of the second group and to the most complex ones after that and, at the end, finish with variable probing, which is the extreme case.

Basic Variable Fixings.

FRC 1: **If** $\exists\, i \notin N_1$: $\bar{x}_{ii} = 1 \wedge L(\bar{u})|_{\neg ii} > GUB$
with $L(\bar{u})|_{\neg ii} = L(\bar{u}) - S_i(\bar{u}) + S^{\downarrow 0}(\bar{u})$
Then $N_1 \leftarrow N_1 \cup \{i\}$

FRC 2: **If** $\exists\, i \notin N_0$: $\bar{x}_{ii} = 0 \wedge L(\bar{u})|_{ii} > GUB$
with $L(\bar{u})|_{ii} = L(\bar{u}) + S_i(\bar{u}) - S^{\uparrow 1}(\bar{u})$
Then $N_0 \leftarrow N_0 \cup \{i\}$

FRC 3: **If** $\exists\, j \in N_1, \exists\, (i, j) \notin A_1$: $\bar{x}_{ij} = 1 \wedge L(\bar{u})|_{\neg ij} > GUB$
with $L(\bar{u})|_{\neg ij} = L(\bar{u}) - \gamma_{ij}(\bar{u})$
Then $A_1 \leftarrow A_1 \cup \{(i, j)\}$

FRC 4: **If** $\exists\, (i, j) \notin A_0$: $\bar{x}_{jj} = 1 \wedge \bar{x}_{ij} = 0 \wedge L(\bar{u})|_{ij} > GUB$
with $L(\bar{u})|_{ij} = L(\bar{u}) + \gamma_{ij}(\bar{u})$
Then $A_0 \leftarrow A_0 \cup \{(i, j)\}$

Advanced Variable Fixings.

FRC 5: **If** $\exists\, (i, j) \notin A_0$: $\bar{x}_{jj} = 0 \wedge \gamma_{ij}(\bar{u}) > 0$
$\wedge\, L(\bar{u})|_{jj,\,ij} > GUB$
with $L(\bar{u})|_{jj,\,ij} = L(\bar{u}) + S_j(\bar{u}) + \gamma_{ij}(\bar{u}) - S^{\uparrow 1}(\bar{u})$
Then $A_0 \leftarrow A_0 \cup \{(i, j)\}$

FRC 6: **If** $\exists\, i \notin N_0$: $\bar{x}_{ii} = 1$
$\wedge \sum_{j \succ i} \bar{x}_{ij} \geqslant 1$
$\wedge \bigwedge_{j \succ i:\ \bar{x}_{ij} = 1 \wedge j \notin N_1} S_j(\bar{u}) - \gamma_{ij}(\bar{u}) < S^{\downarrow 0}(\bar{u})$
$\wedge\, L(\bar{u})|_{ii, \neg ij\, \forall\, j \succ i} > GUB$
with $L(\bar{u})|_{ii, \neg ij\, \forall\, j \succ i}$
$= L(\bar{u}) - \sum\{\gamma_{ij}(\bar{u})$: $\bar{x}_{ij} = 1, j \succ i\}$
Then $N_0 \leftarrow N_0 \cup \{i\}$

The following advanced fixing may represent the maximum level of complexity that one can accept for an advanced fixing, and beyond which it is preferable to use the fixing by probing exposed in the next section.

FRC 7: **If** $\exists\, i \in N_0\ \exists\,(i,j) \notin A_1$: $\bar{x}(\delta^+(i)) = \bar{x}_{ij} = 1$
$\wedge \min_{t \succ i, t \neq j, (i,t) \notin A_0} L(\bar{u})|_{it,\, tt,\, \neg ij} > GUB$
Then $A_1 \leftarrow A_1 \cup \{(i,j)\}$ and $N_1 \leftarrow N_1 \cup \{j\}$
if not yet $j \in N_1$.

Given $L(\bar{u})$, the computation of the value $L(\bar{u})|_{it,\, tt,\, \neg ij}$ involves only changes of the quantities $S_j(\bar{u})$ and $S_t(\bar{u})$, which can be implemented in terms of simple formulas, namely:

$$S_j(\bar{u})|_{\neg ij} = S_j(\bar{u}) - \gamma_{ij}(\bar{u})$$

$$L(\bar{u})|_{\neg ij} = \begin{cases} L(\bar{u}) - \gamma_{ij}(\bar{u}) & \text{if } S_j(\bar{u})|_{\neg ij} \leqslant S^{\downarrow 0}(\bar{u}) \text{ or } j \in N_1 \\ L(\bar{u}) - S_j(\bar{u}) + S^{\downarrow 0}(\bar{u}) & \text{otherwise} \end{cases}$$

$$S^{\uparrow 1}(\bar{u})|_{\neg ij} = \begin{cases} S^{\uparrow 1}(\bar{u}) & \text{if } S_j(\bar{u})|_{\neg ij} \leqslant S^{\uparrow 1}(\bar{u}) \text{ or } j \in N_1 \\ S_j(\bar{u})|_{\neg ij} & \text{if } S^{\uparrow 1}(\bar{u}) < S_j(\bar{u})|_{\neg ij} \leqslant S^{\downarrow 0}(\bar{u}) \\ & \quad \text{and } j \notin N_1 \\ S^{\downarrow 0}(\bar{u}) & \text{if } S^{\downarrow 0}(\bar{u}) < S_j(\bar{u})|_{\neg ij} \text{ and } j \notin N_1 \end{cases}$$

and for all $t \succ i$, $(i,t) \notin A_0$, because $S_t(\bar{u})|_{\neg ij} = S_t(\bar{u})$, we have

$$L(\bar{u})|_{tt,\, \neg ij} = \begin{cases} L(\bar{u})|_{\neg ij} & \text{if } S_t(\bar{u}) \leqslant S^{\uparrow 1}(\bar{u})|_{\neg ij} \text{ or } t \in N_1 \\ L(\bar{u})|_{\neg ij} + S_t(\bar{u}) - S^{\uparrow 1}(\bar{u})|_{\neg ij} & \text{otherwise,} \end{cases}$$

and finally $L(\bar{u})|_{it,\, tt,\, \neg ij} = L(\bar{u})|_{tt,\, \neg ij} + \max(0, \gamma_{it}(\bar{u}))$.

THEOREM 3.4. *All variable fixings* FRC 1–FRC 7 *are legal.*

PROOF.

FRC 1: If $\bar{x}_{ii} = 1$, consider the new problem with $x_{ii} = 0$. The optimal solution to the Lagrangean consists of those models already chosen, except i, which is replaced by the model $j^\star$, not fixed to 0 with $S_{j^\star}(\bar{u}) = S^{\downarrow 0}(\bar{u})$. The Lagrangean value of this solution is $L(\bar{u}) - S_i(\bar{u}) + S^{\downarrow 0}(\bar{u})$. If this value is higher than GUB, the value of the best-known feasible solution, there is no optimal solution of "Choice ILP" that does not contain model i; therefore, we can fix $x_{ii} = 1$.

FRC 2: If $\bar{x}_{ii} = 0$, the argument is the same as in the previous case except that i must be chosen, so it replaces the worst of the chosen ones that are not fixed in the solution, say $j^\star$, with $S_{j^\star}(\bar{u}) = S^{\uparrow 1}(\bar{u})$. The Lagrangean value of this solution is $L(\bar{u}) + S_i(\bar{u}) - S^{\uparrow 1}(\bar{u})$. If this value is greater than GUB there is no optimal solution of "Choice ILP" that contains model i; therefore, we can fix $x_{ii} = 0$.

FRC 3: If $j \in N_1$ and $\bar{x}_{ij} = 1$, consider the new problem with $x_{ij} = 0$. The value of $S_j(\bar{u})$ in the new Lagrangean relaxation has increased to $S_j(\bar{u}) - \gamma_{ij}(\bar{u})$, but since $j \in N_1$, the chosen models in the Lagrangean relaxation remain the same and the value of that relaxation is increased by $-\gamma_{ij}(\bar{u})$. If this increase brings the value over the value of the best-known feasible solution, there is no optimal solution of "Choice ILP" that has $x_{ij} = 0$, and therefore we can fix $x_{ij} = 1$. Note that we could have $j \notin N_1$, $\bar{x}_{ij} = 1$, and $S_j(\bar{u}) - \gamma_{ij}(\bar{u}) < S^{\downarrow 0}(\bar{u})$, and the argument would work; but then if we could fix $x_{ij} = 1$, we could have fixed before $x_{jj} = 1$ and, as already stated, node fixing is always done before arc fixing.

FRC 4: If $\bar{x}_{jj} = 1$ and $\bar{x}_{ij} = 0$, the argument is the same as in the previous case except that one must replace $-\gamma_{ij}(\bar{u})$ by $+\gamma_{ij}(\bar{u})$ everywhere and, of course, invert the roles of $x_{ij} = 0$ and $x_{ij} = 1$.

FRC 5: Let $(i,j) \notin A_0$ with $\bar{x}_{ij} = 0$ and $\gamma_{ij}(\bar{u}) > 0$. The case where $\bar{x}_{jj} = 1$ has been already treated; therefore, we need only consider the case $\bar{x}_{jj} = 0$. If an optimal solution has $x_{ij} = 1$, then it should have $x_{jj} = 1$, so model j enters the solution and one must leave it. Therefore, if $L(\bar{u}) + S_j(\bar{u}) + \gamma_{ij}(\bar{u}) - S^{\uparrow 1}(\bar{u}) > \text{GUB}$, then we can fix $x_{ij} = 0$.

Notice that we do not have to treat the case $\gamma_{ij}(\bar{u}) \leqslant 0$, because in this case, the value $\gamma_{ij}(\bar{u})$ is already a part of $S_j(\bar{u})$, and if $L(\bar{u}) + S_j(\bar{u}) - S^{\uparrow 1}(\bar{u}) > \text{GUB}$, we could fix x_{jj} to 0 by FRC1, and x_{ij} to 0 by FLI 1.

FRC 6: If $\bar{x}_{ii} = 1$, $i \notin N_1$, and $\sum_{j \succ i} \bar{x}_{ij} \geqslant 1$, there are arcs of value 1 "leaving" i in the Lagrangean solution. Let us assume i is in an optimal solution, then $x_{ij} = 0$ for all arcs leaving i. Generally, this will change the Lagrangean solution. It does not if for all $j \succ i$ with $\bar{x}_{ij} = 1$, either $j \in N_1$ or $S_j(\bar{u}) - \gamma_{ij}(\bar{u}) < S^{\downarrow 0}(\bar{u})$. Therefore, if $L(\bar{u}) - \sum\{\gamma_{ij}(\bar{u})\colon \bar{x}_{ij} = 1, j \succ i\} > \text{GUB}$, we can fix x_{ii} to 0. Note that in this case x_{ii} is fixed to the opposite value that it has in the current Lagrangean solution.

FRC 7: Let $i \in N_0$, $\bar{x}_{ij} = 1$, $(i,j) \notin A_1$, $\bar{x}_{it} = 0$ for all $t \neq j$. Let us look at all solutions in which $x_{ij} = 0$. Since $i \in N_0$, i must be replaced by another model. We first look at what happens to j. If $j \in N_1$, the model remains in the Lagrangean optimal solution, if not, and $S_j(\bar{u}) - \gamma_{ij}(\bar{u}) > S^{\downarrow 0}(\bar{u})$, it will have to leave, else if $S_j(\bar{u}) - \gamma_{ij}(\bar{u}) > S^{\uparrow 1}(\bar{u})$, it becomes the new value of $S^{\uparrow 1}(\bar{u})$. We now compute the best replacement for model i. For this we look at all $t \succ i$, $t \neq j$, $(i,t) \notin A_0$ and compute the value of the solution when t replaces i. This is easily done. If $\bar{x}_{tt} = 1$, then the value is $L(\bar{u}) + \gamma_{it}(\bar{u}) - \gamma_{ij}(\bar{u})$ if j need not go out, else it is $L(\bar{u}) + \gamma_{it}(\bar{u}) - S_j(\bar{u}) + S^{\downarrow 0}(\bar{u})$. If $\bar{x}_{tt} = 0$, then t must be chosen. If j must leave because $S_j(\bar{u}) - \gamma_{ij}(\bar{u}) > S^{\uparrow 1}(\bar{u})$, the solution value becomes: $L(\bar{u}) - S_j(\bar{u}) + S_t(\bar{u}) + \max(0, \gamma_{it}(\bar{u}))$. If j does not leave, the new solution value is $L(\bar{u}) - S^{\uparrow 1}(\bar{u}) + S_t(\bar{u}) + \max(0, \gamma_{it}(\bar{u})) - \gamma_{ij}(\bar{u})$. The $\max(0, \gamma_{it}(\bar{u}))$ accounts for the fact that $\gamma_{it}(\bar{u})$ already contributes to $S_t(\bar{u})$ if it is negative. If the minimum over all t of all these values is greater than GUB, we can fix $x_{ij} = 1$, and if not already done, $x_{jj} = 1$. □

3.3.1. Variable Fixing via Variable Probing. Once we have managed to fix a high percentage of the variables, instead of designing more and more complex variable-fixing rules, we turn to the following fixing by variable probing.

As seen in §3.2, variable fixing has a snowball effect; one fixing implies many others. To obtain the set of all implied fixings, one must recursively use the fixings by logical implication. It is not rare to see up to 10 recursive calls. Let $FLI(ij)$ and $FLI(\neg ij)$ be the set of variable fixings implied, respectively, by fixing $x_{ij} = 1$ and $x_{ij} = 0$.

FP 1: **If** $\exists i \notin N_1$: $L(\bar{u})|_{\neg ii,\, FLI(\neg ii)} > GUB$
Then $N_1 \leftarrow N_1 \cup \{i\}$

FP 2: **If** $\exists i \notin N_0$: $L(\bar{u})|_{ii,\, FLI(ii)} > GUB$
Then $N_0 \leftarrow N_0 \cup \{i\}$

FP 3: **If** $\exists (i, j) \notin A_1$: $L(\bar{u})|_{\neg ij,\, FLI(\neg ij)} > GUB$
Then $A_1 \leftarrow A_1 \cup \{(i, j)\}$

FP 4: **If** $\exists (i, j) \notin A_1$: $L(\bar{u})|_{ij,\, FLI(ij)} > GUB$
Then $A_0 \leftarrow A_0 \cup \{(i, j)\}$

This brutal fixing may seem time consuming, but pays off for very many instances. We have been able to prove optimality for many instances for which it was not the case without this strategy.

3.4. Variable Fixing in Relation to the Free-Node Hypergraph

The free-node hypergraph was defined in §1.4.

3.4.1. Variable Fixing While Building the Free-Node Hypergraph. When building the free-node hypergraph, we delete edges containing other edges. Before doing so, one may fix some variables:

FH 1: **If** $\exists E_i \subset E_j, t \in E_j \backslash E_i$: $i \prec j \vee i \prec t \vee \bigwedge_{\ell \in E_i} c_t \geqslant c_\ell$
Then $A_0 \leftarrow A_0 \cup \{(j, t)\}$

THEOREM 3.5. *Variable fixing* FH 1 *is legal.*

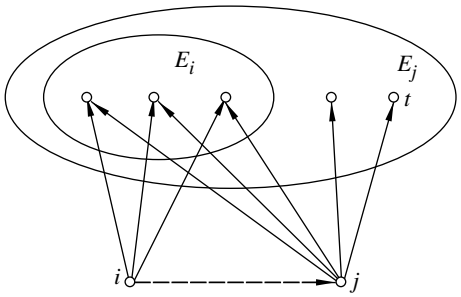
PROOF. If $i \prec j$, we know that $i \notin E_i$ so $i \in N_0$, and there exists $\ell \in E_i$ such that $x_{i,\ell} = 1$ in an optimal solution, but we know by FLI 8 (a) that we then have $x_{j,\ell} = 1$, and therefore $x_{j,t} = 0$ for all $t \in E_j \backslash E_i$. If $i \prec t$, necessarily $x_{i,t}$ has been fixed to 0, else t would be in E_i. Let ℓ be as just defined, if j is to be replaced by t, then $c_t \leqslant c_\ell$, but then it contradicts the fixing of $x_{i,t}$ to 0. The last one is obvious; since t is more expensive than anyone in E_i, it will never be used to replace j because we have to choose an element in E_i that can do the job. □

Figure 2 shows an example in which the two nodes i and j are not in the corresponding edges, which means that they both belong to N_0. Also, all the arcs (i, t) with $t \in E_j \backslash E_i$ belong necessarily to A_0.

Note that when the first case is true, then so is the second, but if we detect the first case, we can go on and do the fixing for all $t \in E_j \backslash E_i$. Also note that if $i \prec j$, then $i \notin E_i$.

3.4.2. Variable Fixing Using the Free-Node Hypergraph. Let us go back to the example of Figure 1. Assuming that 33 is not in the optimal solution, we must then find a transversal of cardinality of at most 4 in the hypergraph of Figure 3, which is clearly impossible since

Figure 2. Example for fixing in the free-node hypergraph.



we have a matching (set of disjoint edges) of cardinality 5. Therefore, we can set $N_1 = N_1 + \{33\}$; that is, fix 33 in the solution.

Let us formalize this fixing. Let $\mathcal{H}|_{ij}$ and $\mathcal{H}|_{\neg ij}$ be the free-node hypergraphs obtained by setting x_{ij}, respectively, to 1 and 0. Let $\alpha(\mathcal{H})$ represent the minimum cardinality of a transversal of a hypergraph $\mathcal{H}$.

FH 2: **If** $\exists i \notin N_1$: $\alpha(\mathcal{H}|_{\neg ii}) > k - |N_1|$
Then $N_1 \leftarrow N_1 \cup \{i\}$

FH 3: **If** $\exists i \notin N_0$: $\alpha(\mathcal{H}|_{ii}) > k - |N_1|$
Then $N_0 \leftarrow N_0 \cup \{i\}$

FH 4: **If** $\exists (i, j) \notin A_1$: $\alpha(\mathcal{H}|_{\neg ij}) > k - |N_1|$
Then $A_1 \leftarrow A_1 \cup \{(i, j)\}$

FH 5: **If** $\exists (i, j) \notin A_0$: $\alpha(\mathcal{H}|_{ij}) > k - |N_1|$
Then $A_0 \leftarrow A_0 \cup \{(i, j)\}$

THEOREM 3.6. *The variable fixings* FH 2 *to* FH 5 *are legal.*

PROOF. The minimum cardinality of a transversal of these hypergraphs is the minimum number of variables x_{ii}, $i \notin N_1$, that we have to fix to 1 to obtain a feasible solution. □

Note that to find the value α is itself *NP*-hard. We used a heuristic to find a matching (set of disjoint edges) of maximum cardinality, and use the fact that the cardinality of any matching is a lower bound on α.

3.4.3. Reducing the Size of the Problem Without Fixing. In this section, we try to reduce the number of variables even if we were not able to fix them.

Figure 3. The free-node hypergraph of Figure 1 with $x_{33,33} = 0$.

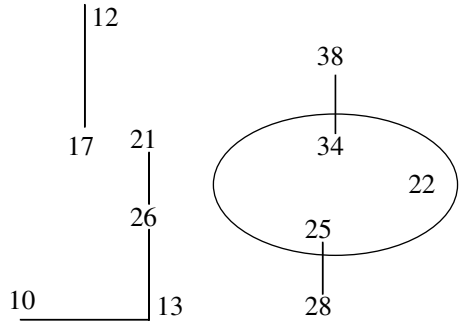
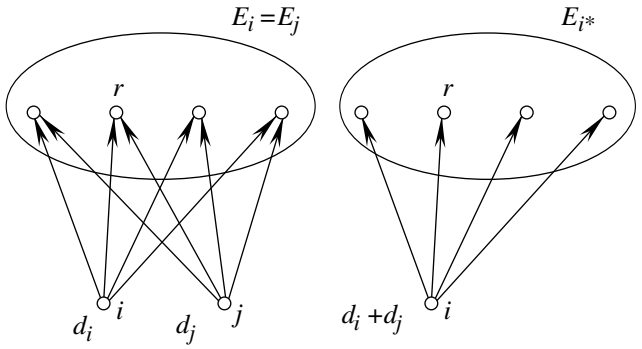


Figure 4. Node and arc agglomeration.



Let $i \neq j$ be such that $E_i = E_j$. At most one, say i, can belong to $E_i = E_j$ (else we would have $i \prec j$ and $j \prec i$); therefore j is fixed to 0 ($j \in N_0$). Figure 4 is drawn with both i and j fixed to 0. Since, in an optimal solution a model $r \in E_i$ will replace i (r may be i if i is not fixed to 0), there exists an optimal solution in which this model r will replace j. We can identify i and j to a single node that we will call $i^\star$, and set $d_{i^\star} = d_i + d_j$ and $u_{i^\star} = u_i + u_j$. If i is not fixed to 0, we let $c_{i^\star} = c_i$, and $i^\star$ is not fixed. If both i and j are fixed to 0, then so is $i^\star$.

The value $S_r(\bar{u})$ for all $r \in E_i$ may increase because $\min(0, c_r(d_i + d_j) - (u_i + u_j)) \geqslant \min(0, c_r d_i - u_i) + \min(0, c_r d_j - u_j)$ if $r \neq i$, and because $c_r d_j - u_j \geqslant \min(0, c_r d_j - u_j)$ if $r = i$. In this case, we have to recompute the optimal solution of the Lagrangean relaxation.

In their work on the p-median problem, Avella et al. (2003) report that in their code also this is a very successful case of problem reduction for our type of problems.

3.4.4. Further Use of the Free-Node Hypergraph. The free-node hypergraph can also be used in selecting a branching variable in a branch-and-cut or branch-and-bound procedure. We have successfully tested the following branching scheme:

Choose an element of the current best-known solution that belongs to the smallest cardinality edge. In the case of ties, choose the one that belongs to the largest number of edges; if ties remain, choose the one such that the total number of nodes of these edges is the smallest. For more details, see Briant (2000).

3.5. Influence of Variable Fixing on the Lower Bound

As mentioned earlier, since our relaxation meets the integrality condition we cannot hope to get a better lower bound than that given by the optimal value of the linear relaxation of our integer program by using basic Lagrangean relaxation. This is no longer true when variable fixing is performed. We show this in a small instance of 535 models for which the linear relaxation is easily solved. We report for $k = 8$ to $k = 15$ the value of the linear relaxation and of the bound we obtain by Lagrangean relaxation with variable fixing; the last column reports the difference. In the two cases in which the difference is 0, the linear bound is the value of the integer optimum. The reported values are those of the overcosts, that is, total cost minus cost of the maximum diversity.

k	Linear relax	Lagrangean bound	Difference
8	148,952,746	148,958,420	5,674
9	130,874,135	131,355,909	481,774
10	117,778,654	118,260,428	481,774
11	106,844,683	106,844,683	0
12	99,544,965	99,544,965	0
13	92,979,799	93,251,020	271,221
14	87,217,013	87,304,002	86,989
15	82,326,415	82,429,734	103,319

3.6. Upper Bounding Heuristic

Fixing by reduced cost is very dependent on the best value of a known feasible solution. Therefore, some effort is carried out in that direction.

3.6.1. Initial Upper Bound. Before starting to solve the problem, one first gets an upper bound by a greedy heuristic. That is, first choose all maximal models with positive demands (it happens that maximal models have a demand of 0). From there on, iteratively choose the model which added to the previously chosen ones decreases the cost the most until the right number is reached. This can be followed by a 2-OPT procedure that consists of trying to exchange any removable model in the solution with an addable one not in the solution, and effectively performing the exchange if it improves the solution. This is done until no improving exchange is found. By removable we mean, at this point, not maximal with nonzero demand; later on, not in N_1, by addable we mean one not in N_0.

3.6.2. Subsequent Upper Bounding. During the solution process, we try to use information coming from the Lagrangean relaxation solutions to get a better upper bound.

We start with the solution of the k models that have been most frequently present in the solutions that have improved the global lower bound, and then use a 2-OPT procedure on it.

Using the k models in the solution of the current Lagrangean relaxation and improving it by a 2-OPT procedure gives, in general, results that are not as good.

The 2-OPT procedure used in the upper bounding becomes the bottleneck when the size of the problem increases. We have implemented a fast restricted 2-OPT procedure that makes use of the $S_\cdot(\bar{u})$ values. In that procedure, the m elements of the current solutions not in N_1 with the largest value of $S_\cdot(\bar{u})$ are considered as removable, the p elements not in the current solutions not in N_0 with the smallest value of $S_\cdot(\bar{u})$ as addable. We use $m = p = 50$. This procedure is very time efficient and gives good results.

Table 1. Instance with 535 nodes and 21,003 arcs.

	Variable fixing			Bounds			
k	Nodes	Arcs	Not fixed	Lower	Upper	Guarantee	Time
5	535	21,003	0	233,812,088	233,812,088	optimal	2.9
6	535	21,003	0	196,994,120	196,994,120	optimal	3.4
7	535	21,003	0	167,558,413	167,558,413	optimal	3.5
8	535	21,003	0	148,958,420	148,958,420	optimal	5.2
9	535	21,003	0	131,355,909	131,355,909	optimal	5.1
10	535	21,003	0	118,260,428	118,260,428	optimal	6.5
11	535	21,003	0	106,844,683	106,844,683	optimal	5.4
12	535	21,003	0	99,544,965	99,544,965	optimal	5.8
13	535	21,003	0	93,251,020	93,251,020	optimal	11.9
14	535	21,003	0	87,304,002	87,304,002	optimal	10.3
15	494	20,473	571	82,429,734	82,645,076	0.260%	24.2
16	461	19,655	1,422	78,019,486	78,451,890	0.551%	39.7
17	505	20,591	442	74,077,990	74,257,796	0.242%	26.2
18	535	21,003	0	70,487,622	70,487,622	optimal	27.1
19	535	21,003	0	66,758,372	66,758,372	optimal	17.0
20	535	21,003	0	63,370,554	63,370,554	optimal	25.6

4. Computational Results

The following three tables give the results of our Lagrangean algorithm for three real-world instances. These instances all come from the automobile industry. These instances have, respectively, 535 nodes and 21,003 arcs (Table 1); 1,284 nodes and 88,542 arcs (Table 2); 5,535 nodes and 666,639 arcs (Table 3). The three first columns contain the results on the variable fixing. The next columns indicate the lower and upper bounds of the overcost, the guarantee on the quality of these bounds, and the execution CPU time in seconds. These tests were executed on a UltraSparc-II, 295 MHz, 512 Mb. The program is written in C++, compiled with gcc-2.95.1, option -O3.

Let $\mathtt{CMD} = \sum_{j \in V} c_j d_j$ represent cost of *maximal diversity*, that is, the cost of the solution that consists of producing all the models. This cost represents a very significant share of the value of GUB. The overcost is $\mathtt{GUB} - \mathtt{CMD}$. To be honest, we compute the guarantee relative to the overcost; i.e.:

$$\textit{guarantee} = \frac{\mathtt{GUB} - \mathtt{GLB}}{\mathtt{GUB} - \mathtt{CMD}}.$$

For these three instances, the percentage of fixed variables is always higher than 90%, and for many values of k, it is even 100%; i.e., all the variables were fixed, and we have the proof that the feasible solution provided by our primal heuristic is optimal.

For the other values of k, the guarantee on the value of the solution remains excellent: always lower than 1%. Moreover, the number of variables at the end of this algorithm is very reduced: This number is approximately a few thousand for the first two instances, and roughly a few tens

Table 2. Instance with 1,284 nodes and 88,542 arcs.

	Variable fixing			Bounds			
k	Nodes	Arcs	Not fixed	Lower	Upper	Guarantee	Time
5	1,284	88,542	0	2,598,461,620	2,598,461,620	optimal	28.7
6	1,284	88,542	0	2,270,141,628	2,270,141,628	optimal	50.9
7	1,284	88,542	0	1,956,636,740	1,956,636,740	optimal	150.9
8	1,284	88,542	0	1,667,587,616	1,667,587,616	optimal	30.7
9	1,284	88,542	0	1,496,819,524	1,496,819,524	optimal	39.2
10	1,284	88,542	0	1,362,417,844	1,362,417,844	optimal	77.6
11	1,191	84,078	4,557	1,234,343,666	1,245,001,468	0.856%	306.4
12	1,194	84,451	4,181	1,112,242,267	1,123,423,904	0.995%	301.2
13	1,206	85,255	3,365	1,006,237,066	1,015,984,652	0.959%	168.9
14	1,284	88,542	0	922,044,732	922,044,732	optimal	83.8
15	1,284	88,542	0	853,729,860	853,729,860	optimal	113.7
16	1,284	88,542	0	795,580,108	795,580,108	optimal	95.7
17	1,284	88,542	0	741,200,748	741,200,748	optimal	75.8
18	1,284	88,542	0	700,132,088	700,132,088	optimal	87.6
19	1,284	88,542	0	662,151,404	662,151,404	optimal	92.6
20	1,284	88,542	0	624,674,820	624,674,820	optimal	75.4

Table 3. Instance with 5,535 nodes and 666,639 arcs.

	Variable fixing			Bounds			
k	Nodes	Arcs	Not fixed	Lower	Upper	Guarantee	Time
5	5,535	666,639	0	5,141,129,908	5,141,129,908	optimal	421.8
6	5,535	666,639	0	4,123,104,419	4,123,104,419	optimal	538.7
7	5,535	666,639	0	3,523,219,454	3,523,219,454	optimal	499.5
8	5,535	666,639	0	3,256,206,050	3,256,206,050	optimal	872.4
9	5,535	666,639	0	3,004,095,791	3,004,095,791	optimal	1536.0
10	5,535	666,639	0	2,787,108,865	2,787,108,865	optimal	1226.7
11	5,535	666,639	0	2,612,835,889	2,612,835,889	optimal	686.7
12	5,535	666,639	0	2,463,416,667	2,463,416,667	optimal	626.0
13	5,535	666,639	0	2,349,025,789	2,349,025,789	optimal	1093.2
14	5,437	654,444	12,293	2,245,675,976	2,252,318,645	0.295%	3987.2
15	5,444	656,458	10,272	2,153,203,940	2,159,782,405	0.305%	3491.1
16	5,316	634,972	31,886	2,064,761,593	2,076,980,195	0.588%	3357.4
17	5,433	654,132	12,609	1,984,165,324	1,989,980,398	0.292%	3401.7
18	5,535	666,639	0	1,908,162,840	1,908,162,840	optimal	4203.8
19	5,371	645,472	21,331	1,826,739,733	1,834,549,757	0.426%	8214.2
20	5,272	631,104	35,798	1,757,906,432	1,768,209,494	0.583%	6608.1

of thousands for the last one. Taking into account the reduction in size of the problem, we can now use an MIP solver to compute the optimal solution of the problem. We will discuss this shortly.

For instances of average sizes, such as Inst535 or Inst1284, the execution times are relatively small, from a few minutes down, sometimes, to a few seconds. However, for instances of large size, such as Inst5535, they range from a few minutes to several hours.

In Table 4, we present the results obtained for Inst5535 with the MIP solver CPLEX 6.0 for the values of k for which the Lagrangean algorithm failed to prove the optimality of the solution. This is now possible due to the drastic decrease in size of the problem in the Lagrangean relaxation phase. We enter at once all the nonfixed variables and therefore the corresponding variable upper-bound constraints. We present in Table 4 the results obtained, after this reduction, with CPLEX 6.0. The times have to be added to the time spent in the Lagrangean relaxation. In this table we indicate, for each instance, the values of k for which we did not have the proof of the optimality of the solution with the Lagrangean algorithm, the number of nodes in the enumeration tree, and the CPU time in seconds. In fact, we proved that the value of GUB provided by the Lagrangean algorithm was always optimal, except for the Inst5535 instance, with $k = 16$ for which the optimal value is 2,072,782,608 instead of 2,076,980,195, i.e., the solution provided by our primal heuristic in the Lagrangean relaxation algorithm was 0.2% greater than the optimal value. Note that because of the initial number of variables and the behavior of the classical methods described in the beginning of this paper, these optimal solutions could not be reached without the preprocessing via the Lagrangean algorithm, in particular for the instance Inst5535.

We have also compared our Lagrangean algorithm against a simple tabu search method. The neighborhood of a solution is that of a 2-OPT move, that is, any solution differing from it by the exchange of an element in the solution with one out of it. We tested two tabu lists, one in which

Table 4. Results with CPLEX after the Lagrangean reduction.

Instance	k	# Nodes	Time
Inst535	15	19	0.33
	16	16	5.49
	17	6	0.16
Inst1284	11	24	29.06
	12	10	17.79
	13	14	13.10
Inst5535	14	7	63.38
	15	3	44.65
	16	4	383.61
	17	3	76.85
	19	9	326.96
	20	12	1066.48

Table 5. Tabu search results for Inst3773 (see Table 7).

k	Greedy	2-OPT	Tabu	Soln at it.	Time
5	734,420,974	729,300,375	726,954,998	6	4,059
6	692,006,441	691,996,489	686,527,844	130	5,335
7	658,175,749	654,008,413	651,930,471	16	6,518
8	631,457,381	631,457,381	622,367,728	35	7,688
9	600,296,729	596,261,326	596,261,326	0	8,648
10	578,037,314	577,041,916	575,494,627	3	10,036
11	558,446,864	557,555,562	555,660,599	16	11,459
12	541,131,116	538,700,803	538,086,796	23	13,208
13	530,670,614	526,812,352	524,118,173	42	10,662
14	514,496,059	514,496,059	509,226,102	65	11,674
15	499,804,112	496,168,918	495,365,155	173	12,829
16	487,707,804	486,527,197	483,534,414	71	13,686

Table 6. Tabu search results for Inst5535 (see Table 3).

k	Greedy	2-OPT	Tabu	Soln at it.	Time
16	2,226,151,310	2,185,733,408	2,084,896,310	13	21,786
17	2,017,986,650	2,007,827,422	2,003,078,752	3	22,885
18	2,045,843,868	2,006,141,829	1,915,364,871	192	25,229

Table 7. Instance with 3,773 nodes and 349,524 arcs.

	Variable fixing			Bounds			
k	Nodes	Arcs	Not fixed	Lower	Upper	Guarantee (%)	Time
5	3,321	201,173	148,803	715,763,565	726,954,998	1.539	2059
6	3,015	160,962	189,320	673,300,416	685,812,258	1.824	2651
7	2,659	135,702	214,936	636,548,725	651,930,471	2.359	3071
8	2,389	110,150	240,758	606,578,332	621,875,275	2.459	3530
9	1,916	68,335	283,046	581,010,042	595,955,799	2.507	4398
10	897	19,422	332,978	559,082,156	574,634,207	2.706	4204
11	401	7,575	345,321	539,800,081	555,210,688	2.775	4250
12	1	3,068	350,228	522,600,933	538,648,580	2.979	4551
13	1	3,749	349,547	507,124,143	522,028,357	2.855	4855
14	1	3,814	349,482	493,041,538	508,365,823	3.014	5168

we forbid a *leaving* element to come back into the solution for the next `nbtabu1` iterations; in the other we forbid the reverse 2-OPT move for some time. That is, if i was removed from the list of produced items and j added to it, we forbid i to be added and j removed for the next `nbtabu2` iterations, with `nbtabu2` much larger than `nbtabu1`. The first tabu list gave much better results, was much easier to implement, and was much faster since we did not have to examine tabu lists to check whether a move is or is not tabu. This can be done by marking the tabu elements. Aspiration has been taken into account, but not search diversification based on recency. It is difficult to catch the consequences of a 2-OPT move on the other such moves; therefore, at every iteration, the whole neighborhood has to be searched. This leads to high computing times.

The following Tables 5 and 6, give the results for Inst3773 (that will be presented further down; see Table 7) and for Inst5535 for a few values of k. We give the greedy value, the value after the 2-OPT procedure on the greedy solution, the solution of the first implementation of tabu search together with the iteration at which the best solution was attained. The number of tabu iterations was limited to 200, since each iteration is costly. All the values are overcosts. The results of the tabu search are not bad in relative values, but do not reach the values we obtained by our algorithm and, because of the large numbers involved, the difference may amount to a significant amount of money. The times of the tabu search are very high and increase very quickly with the number k of items to choose. The tabu algorithm does not provide us with a guarantee on the quality of the solution. Note that even in the framework of local search, one can benefit a lot from a preprocessing by Lagrangean relaxation, since the information we get can help. The neighborhood of each solution is considerably narrowed when we take into account the status of some variables obtained from the variable fixing. This enables these heuristics to search in the right direction while considerably decreasing the computing times.

To conclude this part, we present an instance illustrating the limit of our Lagrangean algorithm. In this instance, we have 3,773 nodes and 349,524 arcs; i.e., this instance is much smaller than Inst5535. However, as Table 7 shows, except for $k = 5$, we never succeeded in fixing more than half of the variables of this instance. From $k = 12$ and up, the percentage of fixed variables is about 1%, i.e., negligible. We suppose that these bad results are due to the presence of several optimal solutions. In fact, the unit costs of production are not very differentiated here: There are very many nodes (incomparable but being able to replace the same one) having identical costs. It is clear that our method of variable fixing gives good results only if these costs really are differentiated. Avella et al. (2003) report better results with this instance.

Acknowledgments

The authors thank Abilio Lucena for his help with the Lagrangean relaxation and Yves Pochet for very helpful discussions on the problem. They also thank an anonymous referee for his time and effort in helping them improve the presentation of this paper.

References

Andersen, E. D., K. D. Andersen. 1995. Presolving in linear programming. *Math. Programming* **71** 221–245.

Avella, P., A. Sassano, I. Vasiliev. 2003. Computational study of large-scale p-median problems. Technical report 08-03 DIS—Universita di Roma "La Sapienza," Rome, Italy.

Beasley, J. E. 1987. An algorithm for set covering problems. *Eur. J. Oper. Res.* **31** 85–93.

Beasley, J. E. 1993a. Lagrangean heuristics for location problems. *Eur. J. Oper. Res.* **65**(4) 383–399.

Beasley, J. E. 1993b. Lagrangean relaxation. C. Reeves, ed. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, Oxford, U.K., 243–303.

Briant, O. 2000. Etude théorique et numérique du problème de la gestion de la diversité. Ph.D. thesis, INP Grenoble, France. ftp://ftp.imag.fr/pub/bibliotheque/theses/2000/Briant.Olivier/.

Caprara, A., M. Fischetti, P. Toth. 1996. A heuristic algorithm for the set covering problem. *IPCO: 5th Integer Programming and Combin. Optim. Conf.*, Vancouver, British Columbia, Canada.

Ceria., S., P. Nobilli, A. Sassano. 1998. A Lagrangian-based heuristic for large-scale set covering problems. *Math. Programming* **81** 215–228.

Crowder, H., E. Johnson, M. W. Padberg. 1983. Solving large-scale zero-one linear programming problems. *Oper. Res.* **31** 803–834.

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

Hiriart-Urruty, J., C. Lemaréchal. 1993. *Convex Analysis and Minimization Algorithms II*. Springer, Berlin, Heidelberg, Germany.

Hoffman, K. L., M. W. Padberg. 1991. Improved LP-representations of zero-one linear programs for branch-and-cut. *ORSA J. Comput.* **3** 121–134.

Lemaréchal, C., C. Sagastizábal. 1997. Variable metric bundle methods: From conceptual to implementable forms. *Math. Programming* **76**(3) 393–410.

Martin, A. 1998. *Integer Programs with Block Structure*. Technische Universität Berlin, Habilitations-Schrift, Berlin, Germany.

Martin, A. 2001. *General Mixed Integer Programming*. Springer Lecture Notes, LNCS2241. Springer, Berlin, Germany.

Mulvey, J., H. Crowder. 1979. Cluster analysis: An application of Lagrangian relaxation. *Management Sci.* **25**(4) 329–340.

Nemhauser, G., L. Wolsey. 1988. *Integer and Combinatorial Optimization.* Wiley-Interscience, New York.

Suhl, U. H., R. Szymanski. 1994. Supernode processing of mixed-integer models. *Comput. Optim. Appl.* **3** 317–331.

Thonemann, U. W., M. L. Brandeau. 2000. Optimal commonality in component design. *Oper. Res.* **48** 1–19.