A Constraint Programming Approach for the Traveling Purchaser Problem

Hadrien Cambazard, Bernard Penz

G-SCOP
Université de Grenoble / Grenoble-INP / UJF-Grenoble 1 / CNRS
{hadrien.cambazard|bernard.penz}@grenoble-inp.fr

Abstract. We present a novel approach to the *Traveling Purchaser Problem* (TPP), based on constraint programming and Lagrangean relaxation. The TPP is a generalization of the Traveling Salesman Problem involved in many real-world applications. Given a set of markets providing products at different prices and a list of products to be purchased, the problem is to determine the route minimizing the sum of the traveling and purchasing costs. We propose in this paper an efficient approach when the number of markets visited in an optimal solution is low. We believe that the real-world applications of this problem often assume a bounded number of visits when they involve a physical routing. It is an actual requirement from our industrial partner which is developing a web application to help their customers' shopping planning. The approach is tested on academic benchmarks. It proves to be competitive with a state of the art branch-and-cut algorithm and can provide in some cases new optimal solutions for instances with up to 250 markets and 200 products.

1 Introduction and industrial context

The *Traveling Purchaser Problem* (TPP) introduced by Ramesh [18], is a generalization of the *Traveling Salesman Problem* (TSP) and occurs in many real-world applications related to routing, wharehousing and scheduling [23]. Given a hometown for the traveler, a set of markets providing products at different prices and a list of products to be purchased, the problem is to determine the route minimizing the sum of the traveling and purchasing costs. The TPP was brought to our attention by a startup ("Le Bon Côté des Choses")[1] developing a web application to help their customers' shopping planning. A customer enters his location, a list of products, a maximum number of markets to visit in the application and is told the most profitable shopping plan. The original question faced by the startup was therefore a TPP with a side constraint bounding the number of markets in the route.

Such an application requires very short response times and a heuristic was previously designed to cope with this requirement. It constructs a feasible solution by greedily adding markets, then attempts to improve it using a two-opt technique. The startup is now in the process of gathering data and extending their approach with additional features. Two main extensions (not revealed here for confidentiality reasons) currently

[1] http://www.leboncotedeschoses.fr/

figure on top of their priorities. We note here that a classical extension of the TPP found in the literature [15] is to consider a limited supply of products in each market. However, this feature is not considered by the startup for the moment since the stock levels are not available online (unlike the catalogues of products). To handle the two extensions mentioned, the initial heuristic must now be deeply restructured. The need for flexibility in extending and maintaining the solver lead us to consider in parallel the development of a constraint programming approach.

The purpose of this paper is to propose a new exact algorithm, based on Constraint Programming (CP), for the TPP with a bounded number of visits in the tour. Our approach takes advantage of three key sub-problems of the TPP, and the propagation algorithms are based on dynamic programming and Lagrangean relaxation. Due to the lack of mature industrial benchmark at this stage, we tackle an academic benchmark and compare to a state of the art exact algorithm [15]. Although the approach was initially designed for a small number of visits, it proves to be surprisingly competitive when applied in the unbounded case.

The rest of the paper is organized as follows. Section 2 precisely defines the Traveling Purchaser Problem and briefly presents the literature, focusing on exact approaches. Section 3 describes the constraint programming model. The details of the main constraints are given in the following sections (4, 5 and 6). The branching strategy is detailed in section 7. Finally computational experiments are reported in section 8.

2 Problem definition and state of the art

Notations are similar to the ones of [15]. Let $K = \{p_1, \ldots, p_m\}$ be the set of products, $M = \{v_1, \ldots, v_n\}$ the set of markets and $M_k \subseteq M$ the set of markets where the product p_k is available. The problem is to determine the route starting from a depot v_0 (purchaser's hometown), and minimizing the sum of the traveling and purchasing costs to acquire the products of K. The price of product p_k in market v_i ($v_i \in M_k$) is z_{ki} and the traveling cost between two nodes v_i and v_j of $V = \{v_0\} \cup M$ is c_{ij} (we assume that the costs satisfy the triangular inequality). Moreover, each p_k has to be bought in a specific amount and this demand is denoted d_k. In this paper, we deal with the unrestricted TPP *i.e* the problem where the supply in each shop is unlimited. The unrestricted TPP was extended in [15] by bounding the amount of p_k available in each market. In an optimal solution of the unrestricted TPP, all the demand of a product p_k is bought in a single shop. Therefore, we simplify our notations by introducing b_{ki}, the cost for buying all the demand of p_k in market v_i: $b_{ki} = d_k z_{ki}$. Finally, we add a parameter, B, to bound the number of markets visited. This last constraint was also considered in [11] and is often a reasonable assumption made in routing problems as discussed in [5].

The TPP [18] is NP-hard in the strong sense since it generalizes two classical strongly NP-hard problems: the Uncapacited Facility Location Problem (UFLP) and the Traveling Salesman Problem (TSP) [8]. Any TSP can indeed be seen as a TPP where each product (one per market) is available in only one market (so that all markets have to be visited). The UFLP can also be seen as a TPP by mapping products to clients and markets to facilities.

The Traveling Purchaser problem has been largely studied during the last two decades. Numerous heuristics were developed, starting with [10, 17] and more recently by [21]. The first exact algorithm based on a lexicographic search was proposed by [18] and was able to solve optimally problems up to 12 markets and 10 products. A branch and bound algorithm was designed later on by [23]. They used a bound based on the simple plant location problem and managed to solve efficiently problems with 20 markets and 50 products. Laporte et al. [15] developed an efficient branch-and-cut algorithm for the undirected TPP. Riera-Ledesma and Salazar-Gonzalez proposed to extend the previous branch-and-cut algorithm to solve the asymmetric case [20]. To our knowledge, this is the best known exact algorithm for the TPP, designed to handle both cases of unlimited and limited supply. It seems reasonable since branch-and-cut is the state of art method for tackling TSP and the best known exact algorithms for facility locations problems are based on linear programming. In Laporte et al. [15], valid inequalities are identified based on the *cycle* (generalization of the TSP where only a subset of vertices must be visited) and the *set-covering* polytopes. The branch-and-cut algorithm generates four types of constraints (four separation procedures) but also variables (pricing procedure) to keep the size of the model reasonable. Finally it uses a primal heuristic at each node (including a 2-opt mechanism for the TSP) to improve the upper bound. It is able to solve optimally instances up to 250 markets and 200 products.

3 CP model

Our constraint programming approach is built on three core sub-problems at the heart of the TPP: the *Traveling Salesman Problem* (TSP), the *P-Median* problem and the *Hitting Set* problem. It strongly relies on the fact that the number of visited markets is bounded (by B) so that the following observations make sense:

- whether all products can be bought in less than B markets is a minimum *Hitting Set* problem (one set per product p_k containing the markets where p_k is available);
- finding the cheapest way to buy all products in less than B markets is a *P-Median* problem where each facility is a market, each client is a product and the cost of connecting a client to a facility is the cost of buying the corresponding product in the given market;
- once the markets visited are known, we are left with a TSP problem on the corresponding set of markets.

We will derive propagation mechanisms taking advantage of these three core sub-problems and achieve a strong level of consistency. One key idea of our model is to exclude the routing problem (TSP) from the search space by performing exponential time propagation in B *i.e* by encapsulating the TSP inside a constraint.

Variables. We use the variables $Ct \geq 0$ and $Cs \geq 0$ to respectively denote the total traveling and shopping cost. Variables $Cs_k \geq 0$ represent the cost of buying each product p_k. Boolean variables $y_i \in \{0, 1\}$ indicates whether market v_i is visited in the tour of the purchaser. The finite domain variables $s_k \in \{i|v_i \in M_k\}$ give the market where product k is bought. Finally $Nvisit \in \{1, \ldots, B\}$ represents the number of markets visited in the tour.

Model. The model is written as follows:

$$
\begin{array}{lll}
 & \text{Minimize } Ct + Cs & \\
(1) & Cs = \sum_{k=1}^{m} Cs_k & \\
(2) & Cs_k = \text{ELEMENT}([b_{k1}, \dots, b_{ki}, \dots, b_{kn}], s_k) & (\forall\, p_k \in K) \\
(3) & \text{OCCURRENCE}(i, [s_1, \dots, s_m]) \geq 1 \Leftrightarrow y_i = 1 & (\forall\, v_i \in M) \\
(4) & Nvisit = \sum_{v_i \in M} y_i & \\
(5) & \text{NVALUE}([s_1, \dots, s_m], Nvisit) & \\
(6) & \text{TSP}([y_1, \dots, y_n], Ct, Nvisit, \{c_{ij} | v_i, v_j \in M\}) & \\
(7) & \text{PMEDIAN}([y_1, \dots, y_n], [s_1, \dots, s_m], Cs, Nvisit, & \\
 & \qquad \{b_{ki} | p_k \in K, v_i \in M\}) & \\
 & s_k \in \{i | v_i \in M_k\} & (\forall\, p_k \in K) \\
 & Cs_k \geq 0 & (\forall\, p_k \in K) \\
 & y_i \in \{0, 1\} & (\forall\, v_i \in M) \\
 & Ct \geq 0, \;\; Cs \geq 0 &
\end{array}
$$

The domains of the variables s_k, y_i and $Nvisit$ are finite *enumerated* domains (each value is maintained in the domain representation) whereas Cs, Cs_k, Ct are represented only by their lower and upper bounds. In the following we denote by $D(x)$ the domain of variable x and by $\underline{x}$ (resp. $\overline{x}$) the lower (resp. upper) bound of x so that x takes a value from $D(x) = [\underline{x}, \dots, \overline{x}]$.

Constraints. Constraints (2) relate the shopping cost of a product to the market where it is bought. It states that $Cs_k = b_{ks_k}$. The ELEMENT constraint allows to index a table of values by an integer variable. The lower bound of Cs_k is thus maintained as the minimum price of product p_k among all currently possible markets: $\underline{Cs_k} = \min_{v_i | i \in D(s_k)} b_{ki}$.

Constraints (3) are channeling constraints linking y_i and s_k variables. Note that at least one product must be bought in a visited market. Our solver does not support this constraint directly but it can be easily decomposed or implemented directly.

Constraint (4) links the number of visits ($Nvisit$) to the y_i variables.

Constraint (5) is a redundant constraint enforcing the number of visits to be equal to the number of different values of the s_k variables ($Nvisit = |\{s_k | 1 \leq k \leq m\}|$). Our solver only provides ATMOSTNVALUE whose focus is on the lower bound of $Nvisit$: the minimum number of markets that must be visited to get all products *i.e* is the aforementioned *Hitting Set* problem. Typically, this constraint efficiently detects an unfeasible problem where B is too restrictive compared to the domains of the s_k (for example in case of rare products). This constraint is NP-hard and is propagated with a greedy algorithm [3].

Constraint (6) is a dedicated global constraint enforcing Ct to be equal to the optimal tour visiting all the markets i such as $y_i = 1$. SM denotes the set of sure markets *i.e* $SM = \{v_i \mid y_i = 1\}$. The scope of (6) encompasses $Nvisit$ because it can be used to derive a stronger lower bound on Ct. Indeed, $\underline{Nvisit}$ might be greater than $|SM|$ in which case the problem of propagating the constraint is known as the *k-TSP*: find the optimal tour visiting k (in our case $\underline{Nvisit}$) cities out of the set of original cities (in our case the set of markets associated to y_i variables not yet fixed to 0).

Constraint (7) is redundant with constraints (1) and (2) together. It propagates a lower bound of Cs by solving the corresponding *P-Median* problem by Lagrangian relaxation.

The three core constraints (5), (6) and (7) are presented in details in the next three sections but we outline now a few elements of this model.

Firstly, note that the exact route followed by the purchaser is not explicitly represented in the model but is present as a support of $\underline{Ct}$ in the TSP global constraint. This has two main drawbacks for a user: the solution is not readily available in the variables and side constraints on the tour are difficult to add. Typically, side constraints to enforce a partial order for visiting the markets or a maximum distance (a resource constraint), must be defined as new parameters of the TSP constraint. The model can also be extended to explicitly represent the route similarly to [5].

Secondly, observe that once the markets are known, it is only a matter of buying each product at the cheapest price among all the markets in the tour so that the s_k variables can be excluded from the search space. Dominance and search strategy are described in section 7.

Finally, the NVALUE and PMEDIAN constraints are redundant. They help strengthening the connection with the TSP sub-problem by increasing $\underline{Nvisit}$ which in turn helps the TSP constraint to derive a sharper $\underline{Ct}$. Similarly, NVALUE and TSP contribute to the decrease of $\overline{Nvisit}$ which helps the PMEDIAN to derive a sharper $\underline{Cs}$. The use of *P-Median* is similar to [23] whose bound is based the *Simple Plant Location Problem*. Our approach is different since the two bounds obtained for Cs and Ct can be added to give the global lower bound. The PMEDIAN provides a very strong filtering but at a high cost. Since it is redundant it can be easily removed from the model when very fast response times are needed for low B.

4 The TSP global constraint

We start the description of the TSP global constraint by recalling its scope:

$$\text{TSP}([y_1, \dots, y_n], Ct, Nvisit, \{c_{ij} | v_i, v_j \in M\})$$

It is a dedicated global constraint enforcing Ct to be equal to the optimal tour visiting all the selected markets (a market i such that $y_i = 1$). We recall that $SM = \{v_i \,|\, y_i = 1\}$ is the set of currently sure markets and denote $PM = \{v_i | y_i \, \text{unknown}\}$ the set of potential markets that can still be included in the tour. We recall that $|SM|$ is assumed to be bounded by a small constant B. Efficient approaches for solving small TSPs are based on dynamic programming or constraint programming [5]. Solving the optimal TSP by dynamic programming can be done in $O(B^2 \times 2^B)$ [13]. The recursive formulation of the TSP is easily written with $f^*(S, x)$, the value of the optimal path starting from the hometown of the traveler, visiting all cities/markets in S and finishing in city x ($x \in S$):

$$f^*(S, x) = \min_{y \in S}(f^*(S - \{x\}, y) + d(y, x))$$

We denote by $optSM$ the value of the optimal tour of the sure markets *i.e* $optSM = f^*(SM, v_0)$. Dynamic programming can be used to compute efficiently the optimal tour for instances with around 15 cities. It still fits in memory for 21-22 cities but requires several seconds.

Lower bound of Ct. The lower bound of Ct can be updated to $optSM$ since the costs satisfy the triangular inequality (adding markets in the set SM can not introduce short-cuts and thus only increase the traveling cost). This lower bound can be refined by taking into account $\underline{Nvisit}$. A minimum of $k = \underline{Nvisit} - |SM|$ number of additional markets must be part of the final tour. We describe a simple lower bound of this quantity. Let $ci(a, b, c)$ be the increase of cost for inserting market a between b and c so that $ci(a, b, c) = d_{ba} + d_{ac} - d_{bc}$ and $ci(a, S)$ the best insertion cost of a in the set of markets $S : ci(a, S) = \min_{b,c \in S \times S | b \neq c, a \neq b, a \neq c} ci(a, b, c)$. Let $< \sigma_1, \ldots, \sigma_{|PM|} >$ be the sequence of markets in PM sorted by increasing $ci(a, SM \cup PM)$ *i.e* the best insertion cost in the set $SM \cup PM$. We use the following rule for updating $\underline{Ct}$:

$$\underline{Ct} = optSM + \sum_{i=1}^{k} ci(\sigma_i, SM \cup PM)$$

The bound is summing the value of the optimal tour on the sure markets and the k best insertion costs. Alternatively the propagation of this constraint is exactly a *k-TSP* problem with a number of mandatory cities. Good approximation algorithms exist for this problem based on a primal-dual scheme initially described in [9]. A more recent paper [1] presents such a scheme with the presence of mandatory cities. A classical linear formulation with an exponential number of constraints (the sub-tour constraints) of the *k-TSP* is given. Its dual has an exponential number of variables but a feasible solution can be obtained greedily without considering all the variables explicitly and provides a lower bound. We experimented with this approach, but the bounds obtained are very weak and only improve the previous bound when k is very large (small $|SM|$ and large $\underline{\text{Nvisit}}$). Note finally that the *k-MST* (minimum spanning tree where only k nodes have to be spanned) is also NP-hard [7].

Upper bound of $Nvisit$. A simple upper bound of $Nvisit$ can be derived from the previous reasonings. Let $kmax$ be the smallest integer such that $optSM + \sum_{i=1}^{kmax} ci(\sigma_i, SM \cup PM) > \overline{Ct}$, we have:

$$\overline{Nvisit} = |SM| + kmax - 1$$

Filtering of unreachable markets. All markets $v_i \in PM$ such that $optSM + ci(v_i, SM) > \overline{Ct}$ can be eliminated from the tour by setting y_i to 0.

Scaling up with B. To ensure that the constraint can still be applied when $|SM|$ is not bounded, the classical Held and Karp bound [14] based on Lagrangian relaxation can be used instead of dynamic programming. This bound is often extremely close to the optimal value especially for small TSPs and can turn out to be faster than dynamic programming.

Implementation. The propagator of the constraint is finally implemented as follows. At any update of a domain (mainly $\underline{Ct}$) a contradiction might be raised if it leads to an inconsistency (typically by overloading $\overline{Ct}$) immediately interrupting the propagation:

- Compute a minimum spanning tree of SM and update $\underline{Ct}$ accordingly.
- If $|SM| < 15$, use dynamic programming to solve optimally the corresponding TSP and get $optSM$. Otherwise, set $optSM$ to the value of the Held and Karp bound. Update $\underline{Ct}$ accordingly.
- Compute $ci(x, SM \cup PM)$ for all $x \in \underline{PM}$. Update $\underline{Ct}$ to $optSM + \sum_{i=1}^{k} ci(\sigma_i, SM \cup PM)$ as explained above along with $\overline{Nvisit}$.
- If $PM = \emptyset$ we need to solve the TSP optimally to instantiate Ct. This has already been done if $|SM| < 15$. Otherwise, an exact TSP solver is called. We use for this purpose a CP model designed on top of the Held and Karp bound such as the one of [5] and also inspired by [2].

5 The PMEDIAN global constraint

The idea of using Lagrangean relaxation for *P-Median* problem in order to perform variables fixings (as a pre-processing) and reduce the size of a linear programming model was proposed in [4]. We intend to go a step further and design a PMEDIAN global constraint to achieve propagation during search. We recall its scope :

$$\text{PMEDIAN}([y_1, \ldots, y_n], [s_1, \ldots, s_m], Cs, Nvisit, \{b_{ki} | p_k \in K, v_i \in M\})$$

The lower bound of Cs obtained from the propagation of the ELEMENT constraints does not take advantage of the limitation enforced by $Nvisit$. Computing a sharp lower bound on Cs by using the information from $Nvisit$ is an NP-hard problem, it is exactly a *P-Median* problem where p can be set to $\overline{Nvisit}$ (visiting more markets is always cheaper so using the upper bound of $Nvisit$ ensures a lower bound). The classical formulation is the following:

$$\text{Minimize} \qquad Cs = \sum_{i=1}^{n} \sum_{k=1}^{m} b_{ki} x_{ki}$$

$$\begin{array}{llll}
(1) & \sum_{i=1}^{n} x_{ki} & = 1 & (\forall\, p_k \in K) \\
(2) & \sum_{i=1}^{n} y_i & = \overline{Nvisit} & \\
(3) & x_{ki} - y_i & \leq 0 & (\forall\, v_i \in M, p_k \in K) \\
(4) & x_{ki} \in \{0,1\} & & (\forall\, v_i \in M, p_k \in K) \\
(5) & y_i \in \{0,1\} & & (\forall\, v_i \in M)
\end{array}$$

A traditional technique to compute a lower bound $\underline{Cs}$ is to use Lagrangian relaxation [16]. Lagrangian relaxation [25] is a technique that moves the "complicating constraints" into the objective function with a multiplier, $\lambda \in \mathbb{R}$, to penalize their violation. For a given value of λ, the resulting problem is the Lagrangian sub-problem and, in the context of minimization, provides a lower bound on the objective of the original problem. The Lagrangian dual is to find the set of multipliers that provide the best possible lower bound.

A lower bound $\underline{Cs}$ can be computed by relaxation of the assignment constraints (1) [16]. Relaxations based on constraints (3) (see [6]) or both constraints (1) and (2) (see [12]) are also possible. The latter seems to be a "standard" relaxation for *P-Median*. We chose to relax (1) since keeping constraints (2) does not make the sub-problem much harder (it only adds a log factor) and seems in practice to greatly improve the convergence. Thus our Lagrangian sub-problem is (λ is unrestricted in sign since we are relaxing an equality constraint) :

$$\begin{aligned} \text{Minimize } w_\lambda(x,y) &= \textstyle\sum_{i=1}^{n}\sum_{k=1}^{m} b_{ki}x_{ki} + \sum_{k=1}^{m}(\lambda_k(1-\sum_{i=1}^{n} x_{ki})) \\ &= \textstyle\sum_{i=1}^{n}\sum_{k=1}^{m}(b_{ki}-\lambda_k)x_{ki} + \sum_{k=1}^{m}\lambda_k \end{aligned}$$

$$\begin{aligned} &\text{subject to } (2) - (5) \\ &\lambda_k \in \mathbb{R} \qquad (\forall\, p_k \in K) \end{aligned}$$

The objective function basically amounts to minimizing $\sum_{i=1}^{n}\sum_{k=1}^{m}(b_{ki}-\lambda_k)x_{ki}$ and the Lagrangian dual is: $\max_\lambda(\min_{x,y} w_\lambda(x,y))$. The Lagrangian sub-problem can be solved directly by inspection:

- For each market v_i, we evaluate the change of the objective function when setting y_i to 1 by computing
$$\alpha(i) = \sum_{k=1}^{m} min(b_{ki}-\lambda_k, 0)$$
- The optimal solution is made of the $\overline{Nvisit}$ markets with smallest $\alpha(i)$. More formally, let $<\delta_1,\ldots,\delta_m>$ be the sequence of markets sorted by increasing value of alpha so that $\alpha(\delta_1) \leq \alpha(\delta_2) \leq \ldots \leq \alpha(\delta_m)$. We have
$$w^*_\lambda(x,y) = \sum_{k=1}^{m}\lambda_k + \sum_{i=1}^{\overline{Nvisit}} \alpha(\delta_i)$$

Solving the sub-problem therefore takes $O(nm + nlog(n))$ if we solve the second steps by sorting. Note that relaxing constraint (2) would not change the quality of the bound of the relaxation since both have the integrality property (so the global bound is the one of the linear relaxation of formulation (2)). It would make the sub-problem easier but we find that it significantly increases the number of iterations in practice and does not pay off.

Solving the Lagrangian Dual. We followed the classical approach and used the subgradient method. The algorithm iteratively solves w_λ for different values of λ, initialised to 0 at the first iteration. The values of λ are updated by following the direction of a supergradient of w at the current value λ for a given step length μ.
A supergradient is given by the violation of the assignment constraints so that:

$$\lambda_k^{t+1} = \lambda_k^t + \mu_t(1 - \sum_{i=1}^{n} x_{ki})$$

The step lengths have to be chosen to guarantee convergence. In particular μ_t must converge toward 0 and not too quickly. We used $\mu_t = \mu_0 \times \epsilon^t$ with $\epsilon < 1$ ($\epsilon = 0.99$) and $\mu_0 = 10^5$. We refer the reader to [25] for more details.

Filtering. If a value is proven inconsistent in at least one Lagrangian sub-problem then it is inconsistent in the original problem [22]. We therefore try to identify infeasible values at each iteration of the algorithm. Let us consider a value i in the domain of s_k such that $i \notin [\delta_1, \ldots, \delta_{\overline{Nvisit}}]$. To establish the feasibility of i we replace $\delta_{\overline{Nvisit}}$ (the least profitable market) by i and recompute the bound by enforcing the assignment x_{ki} to 1. This is done immediately using the benefits computed previously and value i is pruned if:

$$w^*_\lambda(x, y) - \alpha(\delta_{\overline{Nvisit}}) + \alpha(\delta_i) + \max(b_{ki} - \lambda_k, 0) > \overline{Cs}$$

We can notice that $b_{ki} - \lambda_k$ is already counted in $\alpha(\delta_i)$ if it is negative thus the term $\max(b_{ki} - \lambda_k, 0)$. This Lagrangian filtering is performed in $O(nm)$. Note that infeasible markets (y_i that must be set to 0) are detected as a result of the previous filtering if all products are removed from their domain. Such markets i would indeed satisfy:

$$w^*_\lambda(x, y) - \alpha(\delta_{\overline{Nvisit}}) + \alpha(\delta_i) > \overline{Cs}$$

A market i can be proved mandatory by considering the next most beneficial market $\delta_{\overline{Nvisit+1}}$. For all $i \in [\delta_1, \ldots, \delta_{\overline{Nvisit}}]$, we can set y_i to 1 if:

$$w^*_\lambda(x, y) + \alpha(\delta_{\overline{Nvisit+1}}) - \alpha(\delta_i) > \overline{Cs}$$

Finally, a simple lower bound of $Nvisit$ can be derived from the previous reasonings. Let $kmax$ be the largest integer such that $\sum_{k=1}^{m} \lambda_k + \sum_{i=1}^{i=kmax} \alpha(\delta_i) > \overline{Cs}$, we have:

$$\underline{Nvisit} = kmax + 1$$

Implementation. We report here a number of observations that we believe are important when implementing the global constraint.

Optimal values of the dual variables λ are stored after resolution at a given node and restored upon backtracking. When going down in the tree the optimal λ found at the father node are used to initialize the new λ. When going up, the optimal λ previously found at this node are re-used to start the algorithm.

It is also important to stop the algorithm as soon as the lower bound becomes greater than $\overline{Cs}$. This can save many calls to the sub-problem.

All previous reasonings have to be adjusted to take into account the current domains of the y and s variables. This quickly reduces the size of the sub-problem as one moves down in the search tree.

Two pre-conditions are used to avoid starting the costly computations of the relaxation. They provide necessary conditions for any improvement of $\underline{Cs}$ by the use of Lagrangean relaxation compared to the bound given by the ELEMENT constraints. The first one is very cheap to compute and very simple:

$$\overline{Nvisit} < |\{v_i | y_i \neq 0\}|$$

The second one is based on the greedy resolution of the following *Hitting Set* problem. A set is associated to each market v_i and contains the products p_k that can be bought in v_i at their current overall best possible price given by $\underline{Cs_k}$. If a hitting set of cardinality less than $\overline{Nvisit}$ exists then it is a support of the current lower bound of Cs. In this case, we know the bound will not increase by solving the relaxation. The set provides a solution where each product can be bought at its minimum cost. In any of these two cases we do not call the relaxation since it would not lead to any improvement of $\underline{Cs}$ (note that filtering on y and s_k can be lost nonetheless, but this is marginal compared to increasing $\underline{Cs}$).

6 The ATMOSTNVALUE global constraint: *Hitting Set*

This constraint is NP-hard and is propagated with a greedy algorithm described in [3]. Bessiere and al. [3] also shows that the best bound for this constraint is the linear relaxation of the Linear Programming formulation of the problem. The formulation is based on variables y_i to know whether value i (market v_i) is included in the set:

$$\text{Minimize } \sum_{i=1}^{n} y_i$$

$$\begin{array}{lll} (1) & \sum_{i \in M_k} y_i \geq 1 & (\forall\, p_k \in K) \\ (2) & y_i \in \{0,1\} & (\forall\, v_i \in M) \end{array}$$

Similarly to the *P-Median* problem, we can use Lagrangian relaxation to obtain the bound of this linear program by relaxing the covering constraints (1):

$$\begin{aligned} \text{Minimize } w_\lambda(x,y) &= \sum_{i=1}^{n} y_i + \sum_{k=1}^{m} (\lambda_k (1 - \sum_{i \in M_k} y_i)) \\ &= \sum_{i=1}^{n} y_i (1 - \sum_{k | i \in M_k} \lambda_k) + \sum_{k=1}^{m} \lambda_k \end{aligned}$$

$$\begin{array}{ll} y_i \in \{0,1\} & (\forall\, v_i \in M) \\ \lambda_k \geq 0 & (\forall\, p_k \in K) \end{array}$$

The objective function basically amounts at minimizing $\sum_{i=1}^{n} y_i (1 - \sum_{k|i \in M_k} \lambda_k)$. The Lagrangian sub-problem can again be solved directly by inspection similarly to section 5. This approach could be used to strengthen the propagation of our ATMOSTNVALUE constraint without the need of a simplex algorithm. We intend to do so in the future and only use it in this current paper to get an initial lower bound of $Nvisit$.

7 Dominance and branching

We end the description of the CP model with an observation about dominance and the search strategy.

Dominance. A simple form of dominance can be added to enforce buying the products in the cheapest market of the tour. Let's consider a product p_k, when a market t is added to the tour (when y_t is set to one), all values $j \in s_k$ such that $j \neq t$ and $b_{kj} \geq b_{kt}$ can be removed from the domain of s_k by dominance. The reasoning is implemented in a dedicated constraint and stated for each product p_k.

Search. Our branching strategy proceeds in two steps, it branches first on $Nvisit$ and then operates on the y_i variables. Once all y_i are instantiated, the dominance ensures that all s_k variables are also grounded and a solution is reached. Let h^* refers to the lower bound of $Nvisit$ obtained by solving the minimum *Hitting Set* problem by Lagrangean relaxation described in section 6. Four branches start from the root node, constraining $Nvisit$ to be within the following intervals (from the left branch to the right branch) : $[h^*, h^*+1]$, $[h^*+2, 15]$, $[16, 25]$, $[26, m]$. The rational behind this branching is to ensure that good upper bounds involving small number of markets are obtained early in the search (this can be seen as a form of iterative deepening) and before facing large TSP problems. The hope is that we will be able to rule them out without having to solve them explicitly once good upper bounds are known. The values 15 and 25 are chosen simply because they correspond to the limit of efficient solving of the TSP by dynamic programming and our CP complete solver respectively. The branching continues on the y_i variables using *Impact Based Search* [19].

8 Experimental results

Benchmark. Our approach is tested on the benchmark generated by [15]. We are using their "class 3" instances where the n markets are generated randomly in the plan (x and y coordinates are taken with a uniform distribution in $[0, 1000] \times [0, 1000]$), each product is available in a number of markets randomly chosen in $[1, n]$ and product prices are generated in $[1, 500]$ with a uniform discrete distribution. 5 instances are generated for each $n \in \{50, 100, 150, 200, 250\}$ and $m \in \{50, 100, 150, 200\}$ leading to 100 instances in total (20 instances for each value of n) and each instance is identified by $n.m.z$ where $z \in [1, 5]$. We chose this benchmark because the number of markets visited in the optimal solutions are relatively small (up to 28 markets among the known optima) and because it was the hardest benchmark (with unlimited supply) for the branch-and-cut[2].

Set-up. The experiments ran as a single thread on a Dual Quad Core Xeon CPU, 2.66GHz with 12MB of L2 cache per processor and 16GB of RAM overall, running Linux 2.6.25 x64. A time limit of 2 hours was used for each run.

Algorithms. We evaluate two algorithms: CP refers to the algorithm without the redundant PMEDIAN constraint whereas CP+PM is the version including it. In both case, we start by computing the lower bound of $Nvisit$ as explained in section 6. We also apply a heuristic at the root node before starting the search to get an upper bound (we recall that

[2] The benchmark and the detailed results of [15] can be found online: http://webpages.ull.es/users/jriera/TPP.htm

[15] uses a similar heuristic at each node). It builds a feasible solution and improves it by inserting, removing or swapping markets in the pool of visited markets until a local optimum is reached. Its performances are indicated in Table 1 with the average gap to the best known value (in percentage), the number of optimal values identified (column $\#Opt$) and its CPU times.

Table 1. Performances of the heuristic applied at the root node for the 100 instances.

		Time (s)			
Average gap to opt (%)	#Opt	Avg	Median	Min	Max
10,8	14	2,15	1,3	0	13,03

Results. The aim of these experiments is threefold:

Firstly, we show the interest of the PMEDIAN global constraint in Table 2. It reports CPU times (average and median computed only on instances that did not reach the time limit) for each value of n (20 instances in each sub-class), the number of problems where an algorithm fails to prove optimality (reported in column #F) and the average number of backtracks (column Avg Back). CP+PM is clearly more efficient. It fails to prove optimality on 10 instances whereas CP fails on 18 instances.

Table 2. Results of the three approaches on each class of problem (20 instances per class)

n	Branch-and-Cut			CP+PM				CP			
	Time (s)			Time (s)				Time (s)			
	Avg	Median	#F	Avg (s)	Median (s)	#F	Avg Back	Avg (s)	Median (s)	#F	Avg Back
50	23,9	17,5	0	7,1	3,9	0	574,4	11,9	2,7	0	1162,4
100	299,8	295,0	0	187,3	16,3	2	34327,6	214,2	7,0	3	57579,9
150	1734,5	1515,5	0	997,7	231,9	0	41563,6	778,7	110,3	4	108435,5
200	4983,2	3275,0	2	518,5	89,1	6	83670,8	1329,3	43,3	7	201362,0
250	9720,2	10211,0	9	728,2	266,9	2	53741,6	1057,5	357,0	4	145697,2

Secondly, we compare the results with the algorithm of [15]. The branch and cut was run with a timelimit of 5 hours (18000 seconds) on a very old machine (Pentium 500 Mhz with CPLEX 6.0) and the code is not available anymore. Therefore, we do not perform a direct comparison with the CPU times reported for [15] in Tables 2 and 4. They are indicative and only serve to draw general trends. Table 4 gives detailed results for each instance (N and Obj are respectively the number of visits and the value of the objective function in the best solution found). In any case, the branch and cut is more robust since it can handle efficiently large TSP problems. CP+PM fails on 10 instances whose optimal number of visits is between 19 and 28 (instance 200.100.5). Surprisingly, it can find solutions including up to 25 markets (150.200.3) and prove their optimality. It also shows a very good complementarity to the branch-and-cut (see for example 250.50.5 where it needs less than 1s when the branch and cut nearly reaches the five hours). Overall it improves 10 solutions (in bold in the table) and manage to close 8 instances among the 11 that were still open.

Thirdly, Table 3 shows the results obtained when bounding B to 5 and 10. 80 instances are proved infeasible for $B = 5$ and 9 for $B = 10$. All instances are solved to optimality and CP+PM clearly outperforms CP. It shows that the approach can be very effective for low B. The CPU times reported include the proof of optimality which is irrelevant in the industrial case. We plan to analyze the quality of the solution found within 1s of time limit and the time to obtain the best solution without time limit to get more insights on the possibility of using this algorithm in the industrial case.

Table 3. Cpu times of the two algorithms (CP and CP+PM) on the 100 instances of the benchmark and for $B \in \{5, 10\}$

	CP+PM (B = 5)	CP (B = 5)	CP+PM (B = 10)	CP (B = 10)
NbInfeasible	80	80	9	9
Avg Time (s)	0,68	1,22	17,31	122,37
Median Time (s)	0,50	0,48	3,96	1,99
StDev on Time (s)	0,83	4,48	29,23	674,00
Min Time (s)	0,06	0,06	0,14	0,07
Max Time (s)	5,55	43,88	156,83	5496,67

9 Conclusions and future works

We proposed a new exact algorithm based on Constraint Programming for the Traveling Purchaser Problem. The TSP and PMEDIAN global constraints are introduced to efficiently handle two core structures of the TPP and rely on Lagrangean relaxation. The proposed algorithm is designed for problems involving a bounded number of visited markets (around 5 in the industrial application) but is robust enough to be applied on academic benchmark in the unbounded case. It proves to be very complementary to the state of the art exact algorithm and manages to close 8 instances out the 11 open on this particular benchmark.

We intend to investigate further how propagation could be strengthened. Firstly, by using Lagrangean relaxation to propagate and filter the ATMOSTNVALUE constraint. Secondly by adding the ATLEASTNVALUE constraint which would propagate an upper bound of $Nvisit$ based on a maximum matching. We plan to investigate further how to efficiently implement global constraints with Lagrangean relaxation. In particular the work of [2] for the TSP and [4] for the *P-Median* are very good starting points. Finally the use of a state of the art TSP solver might allow the approach to scale further and overcome its current limitation.

Table 4. Details of the results on the class 3 instances of [15]

Instance	Branch-and-cut			CP+PM				Instance	Branch-and-cut			CP+PM			
name	N	Obj	Time(s)	N	Obj	Time(s)	back	name	N	Obj	Time(s)	N	Obj	Time(s)	Back
50.50.1	8	1856	3	8	1856	0.54	16	50.50.2	8	1070	0	8	1070	0.48	33
50.50.3	9	1553	9	9	1553	0.41	192	50.50.4	6	1394	10	5	1394	0.08	8
50.50.5	2	1536	10	2	1536	0.09	5	50.100.1	12	2397	15	12	2397	4.31	289
50.100.2	11	2138	23	11	2138	4.39	562	50.100.3	10	1852	10	10	1852	0.43	8
50.100.4	15	3093	39	15	3093	4.73	917	50.100.5	12	2603	21	12	2603	1.77	159
50.150.1	15	2784	32	15	2784	16.7	1465	50.150.2	11	2137	12	11	2137	0.64	74
50.150.3	14	2308	49	14	2308	7.3	570	50.150.4	14	2524	15	14	2524	3.73	327
50.150.5	16	3150	53	16	3150	26.56	2411	50.200.1	19	3354	18	20	3354	28.28	1260
50.200.2	15	2397	17	15	2397	8.17	604	50.200.3	11	2319	62	11	2319	1.87	103
50.200.4	16	2858	27	16	2858	4.01	252	50.200.5	17	3575	53	17	3575	28.27	2232
100.50.1	9	1468	139	9	1468	1.17	316	100.50.2	7	971	66	7	971	1.67	306
100.50.3	10	1623	159	10	1623	0.92	206	100.50.4	10	1718	80	10	1718	1.68	349
100.50.5	11	2494	168	10	2494	1.31	695	100.100.1	16	2121	130	15	2121	8.74	1258
100.100.2	13	1906	405	13	1906	17.0	2926	100.100.3	13	1822	199	13	1822	15.28	1037
100.100.4	9	1649	55	9	1649	0.73	170	100.100.5	15	2925	758	14	2925	75.39	16289
100.150.1	14	2195	534	14	2195	132.73	11660	100.150.2	16	2806	423	16	2806	141.41	7148
100.150.3	18	2257	440	18	2257	98.84	7974	100.150.4	19	2625	234	18	2625	118.09	9820
100.150.5	18	3150	484	18	3150	390.29	21056	100.200.1	10	1883	166	10	1883	15.64	680
100.200.2	24	3077	369	24	3087	> 7200s	182021	100.200.3	23	2791	356	23	2791	2284.13	76395
100.200.4	19	3409	455	19	3409	> 7200s	342591	100.200.5	18	2732	376	17	2732	66.64	3655
150.50.1	12	1658	498	12	1658	7.58	335	150.50.2	7	1383	512	7	1383	0.27	25
150.50.3	7	821	516	7	821	1.22	308	150.50.4	10	1676	1612	10	1676	13.75	5045
150.50.5	12	1823	1647	12	1823	8.34	1408	150.100.1	17	1717	1243	17	1717	512.73	38148
150.100.2	11	1798	1419	11	1798	27.45	3259	150.100.3	16	1959	2304	15	1959	258.08	32622
150.100.4	14	1609	3408	14	1609	34.42	4179	150.100.5	14	1585	1216	14	1585	275.86	33068
150.150.1	19	1669	367	19	1669	117.76	4264	150.150.2	24	2526	1801	22	2526	2218.69	99598
150.150.3	19	2456	3092	19	2456	5037.65	245597	150.150.4	16	1761	1268	15	1761	205.76	11488
150.150.5	18	2355	3155	16	2355	1152.24	87066	150.200.1	19	1760	365	19	1760	331.94	8090
150.200.2	24	2312	1732	22	2312	1037.48	28630	150.200.3	25	2594	1317	25	2594	5677.0	122773
150.200.4	15	1889	3431	15	1889	154.14	17961	150.200.5	22	2472	3787	22	2472	2881.27	87408
200.50.1	13	1102	1644	12	1102	22.3	5631	200.50.2	6	607	337	6	607	0.45	23
200.50.3	6	530	550	6	530	0.5	71	200.50.4	9	908	849	9	908	0.63	158
200.50.5	11	1067	2248	11	1067	2.78	495	200.100.1	12	949	490	12	949	4.03	237
200.100.2	18	2271	8188	16	2271	381.32	39562	200.100.3	13	1611	2515	13	1611	37.36	4420
200.100.4	17	1799	4697	17	1799	176.97	18758	200.100.5	28	3161	8599	23	3178	> 7200s	310494
200.150.1	16	1730	1574	15	1730	140.87	16012	200.150.2	27	2745	15951	15	2790	> 7200s	212333
200.150.3	20	1861	2613	20	1861	682.82	17643	200.150.4	23	2460	18024	15	**2441**	> 7200s	238821
200.150.5	18	2079	11505	18	2079	446.03	48271	200.200.1	18	1736	3937	18	1736	578.88	21584
200.200.2	22	2352	10647	22	2359	> 7200s	156492	200.200.3	23	2505	7873	22	2505	> 7200s	179516
200.200.4	20	3314	18053	21	**2344**	4783.84	173311	200.200.5	23	2427	5481	23	2462	> 7200s	229584
250.50.1	6	533	556	6	533	0.98	145	250.50.2	10	1103	5451	10	1103	15.67	1316
250.50.3	12	1295	14030	12	1295	12.75	1349	250.50.4	13	1553	15487	12	1553	16.43	3377
250.50.5	5	1142	17399	5	1142	0.66	45	250.100.1	11	2447	18057	15	**1301**	257.56	30983
250.100.2	12	932	2771	12	932	17.22	382	250.100.3	17	1361	16376	17	1361	111.45	10874
250.100.4	16	1759	18029	16	**1673**	284.68	43921	250.100.5	14	1708	18059	15	**1641**	256.96	17568
250.150.1	15	1168	1453	15	1168	367.23	15596	250.150.2	22	2205	7999	22	2205	2036.28	169070
250.150.3	15	1582	10211	15	1582	276.17	11033	250.150.4	18	2636	18078	17	**1836**	1203.97	129473
250.150.5	15	2121	18074	18	**1531**	417.35	38506	250.200.1	20	1677	15189	20	1677	1455.16	61277
250.200.2	25	2787	18019	25	2856	> 7200s	149248	250.200.3	25	3555	18065	20	**1924**	5665.07	140904
250.200.4	17	2432	18045	15	**2139**	> 7200s	230347	250.200.5	18	2771	18071	17	**1797**	712.4	19417

References

1. Sanjeev Arora and George Karakostas. A 2+epsilon approximation algorithm for the k-mst problem. In David B. Shmoys, editor, *SODA*, pages 754–759. ACM/SIAM, 2000.
2. Pascal Benchimol, Jean-Charles Régin, Louis-Martin Rousseau, Michel Rueher, and Willem-Jan van Hoeve. Improving the held and karp approach with constraint programming. In *Proceedings of the 7th international conference on Integration of AI and OR Techniques*

in Constraint Programming for Combinatorial Optimization Problems, CPAIOR'10, pages 40–44, Berlin, Heidelberg, 2010. Springer-Verlag.

3. Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, and Toby Walsh. Filtering algorithms for the nvalue constraint. *Constraints*, 11:271–293, December 2006.
4. Olivier Briant and Denis Naddef. The optimal diversity management problem. *Operations Research*, 52(4):515–526, 2004.
5. Yves Caseau and François Laburthe. Solving small tsps with constraints. In *ICLP*, pages 316–330, 1997.
6. Nicos Christofides and John E. Beasley. A tree search algorithm for the p-median problem. *European Journal of Operational Research*, 10(2):196 – 204, 1982.
7. Matteo Fischetti, Horst W. Hamacher, Kurt Jørnsten, and Francesco Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994.
8. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
9. Michel Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1992.
10. Bruce Golden, Larry Levy, and Roy Dahl. Two generalizations of the traveling salesman problem. *Omega*, 9(4):439 – 441, 1981.
11. Luis Gouveia, Ana Paias, and Stefan Voí. Models for a traveling purchaser problem with additional side-constraints. *"Computers and Operations Research*, 38:550–558, 2011.
12. Pierre Hanjoul and Dominique Peeters. A comparison of two dual-based procedures for solving the p-median problem. *European Journal of Operational Research*, 20(3):387 – 396, 1985.
13. Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. In *Proceedings of the 1961 16th ACM national meeting*, ACM '61, pages 71.201–71.204, New York, NY, USA, 1961. ACM.
14. Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical Programming*, 1:6–25, 1971.
15. Gilbert Laporte, Jorge Riera-Ledesma, and Juan-José Salazar-González. A branch-and-cut algorithm for the undirected traveling purchaser problem. *Operations Research*, 51:940–951, 2003.
16. Subhash C. Narula, Ugonnaya I. Ogbu, and Haakon M. Samuelsson. An algorithm for the p-median problem. *Operations Research*, 25(4):pp. 709–713, 1977.
17. Wen L. Pearn and R.C. Chien. Improved solutions for the traveling purchaser problem. *Computers and Operations Research*, 25(11):879 – 885, 1998.
18. T. Ramesh. Travelling purchaser problem. *Opsearch*, 2(18):78 – 91, 1981.
19. Philippe Refalo. Impact-based search strategies for constraint programming. In Wallace [24], pages 557–571.
20. Jorge Riera-Ledesma and Juan-Jose Salazar-Gonzalez. Solving the asymmetric traveling purchaser problem. *Annals of Operations Research*, 144(1):83–97, 2006.
21. Jorge Riera-Ledesma and Juan José Salazar-González. A heuristic approach for the travelling purchaser problem. *European Journal of Operational Research*, 162(1):142 – 152, 2005.
22. Meinolf Sellmann. Theoretical foundations of cp-based lagrangian relaxation. In Wallace [24], pages 634–647.
23. Kashi N. Singh and Dirk L. van Oudheusden. A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 97(3):571 – 579, 1997.
24. Mark Wallace, editor. *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*. Springer, 2004.
25. Laurence A. Wolsey. *Integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1998.