Annals of Operations Research manuscript No.
(will be inserted by the editor)

Preprocessing for a map sectorization problem by means of mathematical programming

Xin Tang · Ameur Soukhal · Vincent T'kindt

Received: date / Accepted: date

Abstract The sectorization problem is a particular case of partition problems occurring in cartography. The aim is to partition a territory into sectors such that the statistical activity measure of each sector is as close as possible to a given target value. We model this as a problem of minimizing the maximum deviation among all the sectors between their activity measure and their target value. We propose a mathematical programming formulation for the problem, we add some valid inequalities to restrict the solution space and develop a preprocessing procedure to reduce the number of variables. Computational results on different maps highlight the strong efficiency of this reduction procedure.

Keywords sectorization · preprocessing · mathematical programming

1 Introduction

The sectorization problem, in cartography, is a particular case of partition problems. The aim is to partition a territory, which is composed by a set V of n small geographic basic areas, into sectors. Each sector can be viewed as a set

X. Tang
University François Rabelais Tours, Laboratory of Computer Science, Team Scheduling and Control (ERL CNRS 6305), 64 Av. J. Portalis, 37200 Tours, France
E-mail: xin.tang@etu.univ-tours.fr
Groupe Articque Solutions, 149 Av. G. D. Gaulle, 37230 Fondettes, France
E-mail: txin@articque.com

A. Soukhal
University François Rabelais Tours, Laboratory of Computer Science, Team Scheduling and Control (ERL CNRS 6305), 64 Av. J. Portalis, 37200 Tours, France
E-mail: ameur.soukhal@univ-tours.fr

V. T'kindt
University François Rabelais Tours, Laboratory of Computer Science, Team Scheduling and Control (ERL CNRS 6305), 64 Av. J. Portalis, 37200 Tours, France
E-mail: tkindt@univ-tours.fr

of connected basic areas in the territory. The basic areas are generally given as polygons on the map, and usually one or more statistical activity measure information are attached to each basic area. Typical examples are surface, population, number of customers within the area, etc. In this study, we are given for each basic area $v_i \in V$, a single statistical information $w_i \in \Re_+$, the number of sectors to build p and the target value e_j for each sector s_j ($j \in \{1, ..., p\}$). Our aim is to have, for each sector s_j, its statistical activity measure f_j as close as possible to its target value e_j. The statistical activity measure of sector s_j is the total statistical activity measure of the contained basic areas, *i.e.*, $f_j = \sum_{v_i \in s_j} w_i$. We formulate the problem as minimizing the maximum deviation among all the sectors, *i.e.* criterion $Z = \max_j |f_j - e_j|$ has to be minimized. Besides, sectors have to be geographically connected. Sometimes, a center may be associated with each sector. A center may be a predefined basic area, or a basic area with important activity measure, or simply a geographical center. In this study, we assume that no center is given. The above model of sectorization has been elaborated in collaboration with the Articque company which provides softwares for statistical cartography. The sectorization problem can be shown to be NP-hard since the subproblem for which $p = 2$ is so (straightforwardly by reduction from the 2-PARTITION problem).

The sectorization problem is closely related to political districting problems. In political districting problems, the aim is to partition a territory into electoral districts, subject to some side constraints including population equality, contiguity and compactness (Bozkaya et al. 2003): the population of each district must be almost the same with the average district population; the districts need to be connected and it is desirable that the shapes of the districts are compact, like square or circle shapes. The first mathematical programming formulation of political districting problems was proposed by Hess et al. (1965). They considered compactness as an objective to achieve while population equality was taken into account by constraints. But their formulation does not guarantee contiguity. The second mathematical programming formulation was proposed by Garfinkel and Nemhauser (1970). They proposed an exact algorithm based on the set-covering problem: in a first phase, all feasible candidate districts, being contiguous, compact and having population deviation within the tolerance are generated; in a second phase, districts are selected from the set of candidates to minimize the maximum deviation of any district from the average district population. In their approach, the number of feasible candidate districts is huge, which makes the algorithm inefficient to solve even median-size instances. Mehrotra et al. (1998) picked up this model and developed a column generation algorithm. They applied it to districting problems with up to 50 basic areas. But their method remains a heuristic since the problem of generating additional columns itself is NP-hard. For more background information on political districting, the reader is referred to Grilli di Cortona et al. (1999) and Bozkaya et al. (2005). There are several differences between the sectorization problem and political districting problems. In the sectorization problem, the objective is to minimize the maximum deviation between the activity measure of sectors and their target value. Therefore, it is more general than considering equality. But in political districting problems, equality is always taken into account as a constraint imposed on the population. The second difference is that there is no need to consider compactness in the sectorization problem.

The sectorization problem is also related to location-allocation problems (Daskin 1995; Drezner and Hamacher 2004; Eiselt and Sandblom 2004), and more especially to the capacitated p-median (or p-center) problem (van Roy 1986; Hansen and Jaumard 1997) and to location problems with equitable loads (Berman et al. 2009; Kalcsics et al. 2010). In location-allocation problems, the aim is to determine an optimal location for one or more facilities that will serve demands of customers, and to assign these demands to the facilities, taking into account factors such as the number of available facilities, the cost, the capacities of facilities, etc. Applying the general location-allocation model to the sectorization problem, customers correspond to basic areas and their demands correspond to the statistical information of basic areas. The facilities to be located can be viewed as centers of sectors. When solving the model, simultaneously a center for each sector will be located, and basic areas will be allocated to a center. Particularly, the p-median problem aims at finding the locations of p facilities that minimize the demand-weighted total distance between demand points and the facilities to which they are assigned. In the capacitated p-median problem, each candidate facility has a fixed capacity, and the loads of demand points at each facility cannot exceed its capacity. Applying the capacitated p-median model to the sectorization problem, we can limit the capacity of each facility with a value depending on the target value of each sector. In general, the solution derived from this model is "good" in terms of objective function value for the sectorization problem, but unfortunately, there is no guarantee that the sectors will be connected, although this will be favored by the objective function of the capacitated p-median model. The location problem with equitable loads is to find the locations of p facilities such that the demand-weights attached to each facility will be as close as possible to one another, and each demand will be assigned to the closest facility. Applying this model to the sectorization problem, the assignment rule will guarantee each sector to be a connected and convex (well-shaped) region. But this rule is too strong for the sectorization problem, consequently the optimal solution of this location problem may be not optimal for the sectorization problem.

Constrained graph partitioning problems (Mehrotra 1992; Schloegel et al. 2000) are also closely related problems. Let be a graph $G = (V, E)$, where V is the set of nodes, E is the set of edges and both nodes and edges are weighted. A constrained graph partitioning problem consists in dividing G into p disjoint subdomains. The objective is to minimize the total number of edges crossing the subdomains, and at the same time, to reduce the weights imbalance of subdomains. The weight of a subdomain is the sum of weights of nodes allocated to it. The sectorization problem can be modeled as a graph partitioning problem. We model each basic area on the map as a node in the graph. We connect two nodes by an edge if the corresponding basic areas are geographical neighbors on the map. The weight of a node is equal to the statistical measure of the corresponding basic area. A map sectorization can be represented by a partitioning of the nodes such that the nodes in each subdomain induce a connected subgraph and the weight of each subdomain is close to a target value. The difference between these two problems is that in the sectorization problem, the number of cuts in edges does not need to be taken into account and the weights of subdomains are measured by an objective function. However, existing heuristic solution techniques like geometric techniques (Ou et al. 1996; Patra and Kim 1998) and multilevel schemes techniques (Kernighan and Lin 1970; Karypis and Kumar 1998), can serve as a guide for a heuristic solution of the sectorization problem.

To the best of our knowledge, there does not exist any problem in the literature which is the same than the sectorization problem. The purpose of this paper is to propose a mathematical programming formulation for the sectorization problem, to generate some valid inequalities to restrict the solution space of the problem and to see the efficiency of the preprocessing algorithm which optimally fixes variables in the mathematical model. The remainder is organized as follows. In Section 2 we model the sectorization problem and present some valid inequalities. In Section 3 we propose a specific heuristic to calculate a feasible solution which gives us an initial upper bound on the optimal solution value. In Section 4 we present a preprocessing algorithm based on some variable-fixing techniques in order to make faster the exact solution of the problem. This is followed by computational results in Section 5, and by conclusions and future research in Section 6.

2 Mathematical programming formulation and valid inequalities

2.1 Problem definition

The sectorization problem can be formally defined as follows. Let $G = (V, E)$ be an undirected connected graph with node set $V = \{v_1, ..., v_n\}$ and edge set E. A node $v_i \in V$ in the graph corresponds to a basic area on the map, an edge $(v_i,\, v_k)$ exists if only if the corresponding basic areas are geographical neighbors. Each node v_i is associated with a nonnegative demand weight w_i. For any pair of nodes v_i and v_k, let a_{ik} be a binary index equal to 1 if there is an edge between v_i and v_k. The number of sectors to be created is given and equal to p. The target value of sector s_j is given and denoted by e_j. To guarantee connectedness, there must be a spanning tree corresponding to each sector. Let x_{ij} and y_{ijk} be binary variables: x_{ij} equal to 1 if and only if v_i is assigned to sector s_j and y_{ijk} equal to 1 if and only if edge $(v_i,\, v_k)$ is selected to construct the spanning tree associated with sector s_j. To exclude subtours, extra variables u_{ij} are used. We denote by $I = \{1, ..., n\}$ and $J = \{1, ..., p\}$ the index sets of areas and sectors, respectively. Additionally, in the remainder, we use $f_j = \sum_{i=1}^{n} w_i x_{ij}$ to refer to the total statistical activity of sector s_j. The problem is formulated as an assignment problem with connectedness constraints and the corresponding Mixed Integer Program (MIP) is as follows.

$$
\begin{aligned}
&min \quad && Z = \max_{1\le j\le p} |f_j - e_j| && && (1)\\
&st. && \textstyle\sum_{j=1}^{p} x_{ij} = 1, && \forall i \in I && (2)\\
& && y_{ijk} \le x_{ij}, y_{ijk} \le x_{kj}, y_{ijk} \le a_{ik}, && \forall i,k \in I \quad \forall j \in J && (3)\\
& && \textstyle\sum_{k=1}^{n} y_{kji} \le x_{ij}, && \forall i, \quad \forall j && (4)\\
& && \textstyle\sum_{i=1}^{n}\sum_{k=1}^{n} y_{kji} = \sum_{i=1}^{n} x_{ij} - 1, && \forall j && (5)\\
& && n y_{ijk} + u_{ij} \le u_{kj} + (n-1), && \forall i,k, \quad \forall j && (6)\\
& && u_{ij} \le n x_{ij}, u_{ij} \ge x_{ij}, && \forall i, \quad \forall j && (7)\\
& && x_{ij}, y_{ijk} \in \{0,1\}, && \forall i,k, \quad \forall j && (8)
\end{aligned}
$$

The objective function (1) is to minimize the maximum deviation of sector activities. Constraints (2) are assignment constraints and they ensure that each basic area is assigned to exactly one sector. Constraints (3) state that an edge (v_i, v_k) cannot be in the spanning tree of sector s_j if v_i and v_k are not assigned to sector s_j or v_i and v_k are not neighbors. Constraints (4) state that there cannot exist more than one incoming edge for each node in a spanning tree. Constraints (5) state that the number of edges is equal to the number of nodes minus 1 for each spanning tree. Constraints (6) and (7) are the subtour-elimination conditions derived from the traveling salesman problem by Miller, Tucker and Zemlin (1960). Constraints (8) are the integrality conditions.

We give a small example of the sectorization problem, illustrated in Figure 1. Consider $V = \{a, b, c, d, e, f\}$ six basic areas (nodes), with their statistical activity measures $w_a = 9$, $w_b = 2$, $w_c = 6$, $w_d = 7$, $w_e = 2$, $w_f = 2$. Let $p = 2$ be the number of sectors to be created, and let $e_1 = 13$, $e_2 = 15$ be their target values. The dotted line in Figure 1 separates two possible sectors: the first sector s_1 contains nodes a and b, while the second s_2 contains nodes c, d, e and f. The activity measure of s_1 is $f_1 = 9 + 2 = 11$, the second is $f_2 = 6 + 7 + 2 + 2 = 17$. The objective value of this solution is 2.

2.2 Valid inequalities

In this section we present a series of valid inequalities which can be used to restrict the solution space of the sectorization problem and to improve the efficiency of the preprocessing algorithm (see Section 4). These inequalities are derived by an analysis of structural properties of the problem. Assume that we have an admissible solution which is determined by a specific heuristic (see Section 3). This solution gives us an initial upper bound on the optimal

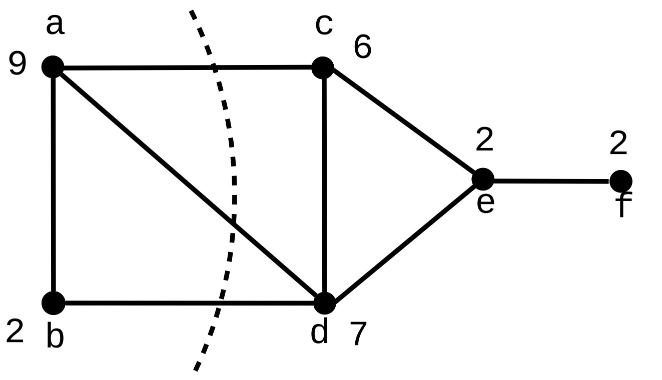

Fig. 1 A small example of the sectorization problem

solution value, denoted by UB. With this upper bound, we add several new constraints to the original formulation of the problem. These constraints are redundant for the original problem formulated as a MIP, but they are not redundant for its continuous relaxation referred to as LP. They may improve (increase in our case) the optimal solution value of the LP and by the way accelerate the solution of the MIP by exact algorithms like branch-and-bound or branch-and-cut algorithms. In the following, we establish some features of optimal solutions.

Lemma 1 *For any sector s_j, its statistical activity measure f_j belong to $[e_j - UB, e_j + UB]$.*

Proof. From the objective function, we have: $\forall j \in J$, $Z \geq f_j - e_j$ and $Z \geq e_j - f_j$. As $Z \leq UB$, this yields $UB \geq f_j - e_j$ and $UB \geq e_j - f_j$. Thus we have $e_j - UB \leq f_j \leq e_j + UB$, $\forall j \in J$. $\square$

Now consider, without loss of generality, that the target values e_j are indexed in a non-decreasing order. Then, it can be shown that there exists an optimal solution in which all f_j's are also sorted in non-decreasing order.

Proposition 1 *Under the assumption that $e_j \leq e_{j+1}$, $\forall j = 1, ..., p-1$, there exists an optimal solution to the MIP for which $f_j \leq f_{j+1}$, $\forall j = 1, ..., p-1$.*

Proof. Assume that S is an optimal solution with two sectors s_{j_1}, s_{j_2} $(j_1, j_2 \in J)$ such that $e_{j_1} \leq e_{j_2}$, $f_{j_1} > f_{j_2}$. We construct a new solution S' based on S with just a simple permutation between s_{j_1} and s_{j_2}:
$s'_{j_1} = s_{j_2}$, $s'_{j_2} = s_{j_1}$ and, $\forall j \in J \backslash \{j_1, j_2\}$, $s'_j = s_j$.
In the following, we prove that S is dominated by S'. The objective values of S and S' are:
$Z_S = max(max(|f_{j_1} - e_{j_1}|, |f_{j_2} - e_{j_2}|), max_{j \in J \backslash \{j_1, j_2\}}(|f_j - e_j|))$,
and,
$Z_{S'} = max(max(|f_{j_2} - e_{j_1}|, |f_{j_1} - e_{j_2}|), max_{j \in J \backslash \{j_1, j_2\}}(|f_j - e_j|))$.

The second parts of Z_S and $Z_{S'}$ are the same, so we will focus on the first parts since
$max(|f_{j_1} - e_{j_1}|, |f_{j_2} - e_{j_2}|) \geq max(|f_{j_2} - e_{j_1}|, |f_{j_1} - e_{j_2}|)$
$\Rightarrow Z_S \geq Z_{S'}$.
Let be $Z'_S = max(|f_{j_1} - e_{j_1}|, |f_{j_2} - e_{j_2}|)$ and $Z'_{S'} = max(|f_{j_2} - e_{j_1}|, |f_{j_1} - e_{j_2}|)$.
We divide the relations between e_{j_1} and f_{j_1}, e_{j_2} and f_{j_2} into 4 exhaustive cases.
Case 1: $e_{j_1} \geq f_{j_1}$; $e_{j_2} \geq f_{j_2}$
This case implies that $e_{j_2} \geq e_{j_1} \geq f_{j_1} > f_{j_2}$. So, we have
$Z'_S = max(e_{j_1} - f_{j_1}, e_{j_2} - f_{j_2}) = e_{j_2} - f_{j_2}$, $Z'_{S'} = max(e_{j_1} - f_{j_2}, e_{j_2} - f_{j_1})$.
Since $e_{j_1} - f_{j_2} \leq e_{j_2} - f_{j_2}$ and $e_{j_2} - f_{j_1} < e_{j_2} - f_{j_2}$, we have $Z'_S \geq Z'_{S'}$.
Case 2: $e_{j_1} < f_{j_1}$; $e_{j_2} < f_{j_2}$
This case implies that $f_{j_1} > f_{j_2} > e_{j_2} \geq e_{j_1}$. So, we have
$Z'_S = max(f_{j_1} - e_{j_1}, f_{j_2} - e_{j_2}) = f_{j_1} - e_{j_1}$, $Z'_{S'} = max(f_{j_2} - e_{j_1}, f_{j_1} - e_{j_2})$.
Since $f_{j_2} - e_{j_1} < f_{j_1} - e_{j_1}$ and $f_{j_1} - e_{j_2} \leq f_{j_1} - e_{j_1}$, we have $Z'_S \geq Z'_{S'}$.
Case 3: $e_{j_1} \geq f_{j_1}$; $e_{j_2} < f_{j_2}$
This case implies that $f_{j_2} > e_{j_2} \geq e_{j_1} \geq f_{j_1}$ which results a contradiction with the assumption that $f_{j_1} > f_{j_2}$. So, this case does not exist.
Case 4: $e_{j_1} < f_{j_1}$; $e_{j_2} \geq f_{j_2}$
In this case, $Z'_S = max(f_{j_1} - e_{j_1}, e_{j_2} - f_{j_2})$, and we consider 4 subcases.

Case 4.1: $f_{j_2} \geq e_{j_1}$; $f_{j_1} \geq e_{j_2}$
In this case we have $Z'_{S'} = max(f_{j_2} - e_{j_1}, f_{j_1} - e_{j_2})$. Since $f_{j_1} - e_{j_1} > f_{j_2} - e_{j_1}$ and $f_{j_1} - e_{j_1} \geq f_{j_1} - e_{j_2}$, we have $Z'_S \geq Z'_{S'}$.
Case 4.2: $f_{j_2} \geq e_{j_1}$; $f_{j_1} < e_{j_2}$
In this case, $Z'_{S'} = max(f_{j_2} - e_{j_1}, e_{j_2} - f_{j_1})$. Since $f_{j_1} - e_{j_1} > f_{j_2} - e_{j_1}$ and $e_{j_2} - f_{j_2} > e_{j_2} - f_{j_1}$, we have $Z'_S \geq Z'_{S'}$.
Case 4.3: $f_{j_2} < e_{j_1}$; $f_{j_1} \geq e_{j_2}$
In this case, $Z'_{S'} = max(e_{j_1} - f_{j_2}, f_{j_1} - e_{j_2})$. Since $e_{j_2} - f_{j_2} \geq e_{j_1} - f_{j_2}$ and $f_{j_1} - e_{j_1} \geq f_{j_1} - e_{j_2}$, we have $Z'_S \geq Z'_{S'}$.
Case 4.4: $f_{j_2} < e_{j_1}$; $f_{j_1} < e_{j_2}$
In this case, $Z'_{S'} = max(e_{j_1} - f_{j_2}, e_{j_2} - f_{j_1})$. Since $e_{j_2} - f_{j_2} \geq e_{j_1} - f_{j_2}$ and $e_{j_2} - f_{j_2} > e_{j_2} - f_{j_1}$, we have $Z'_S \geq Z'_{S'}$.

In all cases, S is dominated by S'. □

We can use Lemma 1 and the connectedness constraint to generate other valid inequalities. For a basic area v_i, we denote the set of its neighbors by $M_{\{v_i\}} \subseteq V$. We can see that if its statistical activity measure w_i is not big enough for sector s_j (*i.e.* $w_i < e_j - UB$), then it can be in sector s_j under the condition that at least one of its neighbors also belongs to s_j.

Lemma 2 *$\forall i \in I, \forall j \in J$, if $w_i < e_j - UB$, then we have: $x_{ij} \leq \sum_{k \in M_{\{v_i\}}} x_{kj}$.*

Proof. Assume that there is a solution S such that a small basic area v_i with $w_i < e_j - UB$, defines alone a sector s_j, then $f_j = w_i < e_j - UB$ and solution S cannot be optimal. □

Consider again the example of Figure 1. The illustrated solution gives us an upper bound $UB = 2$. Besides, node b is a small area for s_1 since $w_b < e_1 - UB = 11$. Its neighbor set is denoted by $M_{\{b\}} = \{a, d\}$. Lemma 2 states that in an optimal solution, node b can be in s_1 under the condition that node a or node d is in s_1. This can be expressed by $x_{b1} \leq x_{a1} + x_{d1}$.

We can extend Lemma 2 to some specific set of nodes, denoted by $R \subseteq V$. If the statistical activity measure of R, denoted by $f_R = \sum_{v_i \in R} w_i$, is not big enough for sector s_j (*i.e.* $f_R < e_j - UB$), then there must exist at least one other basic area, which does not belong to R, in sector s_j. Similarly, if the statistical activity measure of R is too big for sector s_j (*i.e.* $f_R > e_j + UB$), then the elements of R cannot be all in sector s_j.

Lemma 3 *$\forall R \subseteq V$, $\forall j \in J$, if $f_R < e_j - UB$, then we have: $\sum_{v_i \notin R} x_{ij} \geq 1$.*

Lemma 4 *$\forall R \subseteq V$, $\forall j \in J$, if $f_R > e_j + UB$, then we have: $\sum_{v_i \in R} x_{ij} \leq |R| - 1$.*

Lemmas 3 and 4 can be proved by the same reasoning than in the proof of Lemma 2. In Lemma 3, we may have intuition that instead of considering basic areas $v_i \notin R$, we can focus on basic areas $v_i \in M_R$, with M_R the neighborhood of R: $M_R = \{v_i | v_i \in V$ and v_i is neighbor of an element in $R\}$. But this is true only under the condition that R is already assigned to s_j. This would imply adding extra binary variables and then result, at the end, in weak inequalities due to the LP relaxation.

Consider again the example of Figure 1 to illustrate these two lemmas. For Lemma 3, we can find a maximal cardinality set $R = \{b, c, e, f\}$ for sector s_2 such that $f_R = w_b + w_c + w_e + w_f = 12 < e_2 - UB = 13$. According to Lemma 3, there must exist at least one element in the complement of R assigned to sector s_2. This can be expressed by $x_{a2} + x_{d2} \geq 1$. For Lemma 4, we can find a minimal cardinality set $R = \{a, d\}$ for sector s_1 such that $f_R = w_a + w_d = 16 > e_1 + UB = 15$. According to Lemma 4, these two elements cannot be assigned to sector s_1 at the same time. This can be expressed by $x_{a1} + x_{d1} \leq 1$.

In what follows, we introduce the Steiner tree problem and the notion of node separator in graph theory. These are used to strengthen the valid inequalities of Lemma 4. Given an undirected graph $G = (V, E)$ represented by a set V of n nodes and a set of weighted edges E, the Steiner tree problem consists in connecting a given subset R of k nodes, called terminals, by means of a minimum cost tree (Hakimi 1971). This is one of the first problems shown to be NP-hard by Karp (1972). However, some algorithms with exponential running time have been considered to be acceptable if the size of the problem is not too large. A well-known exact algorithm for the Steiner tree problem is the dynamic programming method presented by Dreyfus and Wagner (1972). This one has been improved latter by Niedermeier (2008). We recall this dynamic

programming algorithm. Let be $X \subseteq R$, and $v \in V \backslash X$. Let be $F(X \cup \{v\})$ denotes the weight of a minimum Steiner tree connecting all nodes from $X \cup \{v\}$ in graph G and let $d(u, v)$ be the total weight of the shortest path in terms of weights between nodes u and v. Let be $X \neq \emptyset$, $X \subseteq R$, and $v \in V \backslash X$. Then, $F(X \cup \{v\}) = min\{min_{u \in X}\{F(X) + d(u, v)\}, min_{u \in V \backslash X}\{F_u(X \cup \{u\}) + d(u, v)\}\}$ with
$F_u(X \cup \{u\}) = min_{X' \neq \emptyset, X' \subset X}\{F(X' \cup \{u\}) + F((X \backslash X') \cup \{u\})\}$.
This algorithm requires time proportional to $O(k^3 n + k^2 n^2 + n^2 logn + nm)$. Noteworthy, the distance $d(u, v)$ can be computed by any shortest path algorithm taking as distances on edges the weights. The directed version of the Steiner tree problem is an extension of the undirected version. According to Charikar et al. (1999) the directed version of the Steiner tree problem is defined as follows. Given a directed graph $G = (V, A)$, a specified root $r \in V$, and a set of terminals $R \subseteq V$ ($|R| = k$), the objective is to find the minimum cost arborescence of root r and spanning all the vertices in R. Hopefully, the dynamic programming algorithm of Niedermeier can be easily adapted to solve the directed version of the Steiner tree problem. This is mainly done by taking account in the computation of $d(u, v)$ that we deal with arcs and not edges. The root of the arborescence is taken into account in setting the initial condition that $X = \{r\}$.

In the Steiner tree problem, weights are assigned to edges, but in the sectorization problem, weights are assigned to nodes. In order to use Niedermeier's algorithm to solve the node-weighted Steiner tree problem, we need to transform a node-weighted graph (original graph G) into an arc-weighted directed graph (transformed graph G'): for each node v_i in G, we create two corresponding nodes v_i^I (node that receives arcs) and v_i^O (node that emits arcs) in G'. We build an arc from v_i^I to v_i^O with a weight of w_i (the weight of v_i in G). For each pair of nodes in G, v_i and v_k, if there exists an edge from v_i to v_k, we build an arc from v_i^O to v_k^I with a weight of 0 in G'. This transformation procedure is illustrated in Figure 2. It is easy to show that the obtained graph G' is strongly connected. Now, we illustrate why a minimal Steiner arborescence of root r in G' can be associated to a minimal Steiner tree in G. First, if R is the set of terminals in G, then the set R' of terminals in G' comprises all nodes v_i^O and v_k^I of G' associated to nodes $v_i \in R$. The root r is chosen in R' and it corresponds to a node that receive arcs, i.e. of type x^I. It is clear that for each arborescence ST of root r and spanning all the nodes in R', the arcs of type (x^I, x^O) are in ST (because by construction x^O is the only direct successor of x^I in G'). Consequently, a minimal cost arborescence of root r contains all arcs with weights associated to nodes in R, this is true whatever $r \in R'$ of type x^I. Besides, all arcs of weight 0 in the arborescence correspond to an edge in G. By the way, a minimal cost arborescence as computed above can be easily transformed to a minimal Steiner tree with weights on the nodes.

We can strengthen Lemma 4 by solving Steiner tree problems. For a set of nodes (terminals) R, if the cost of the corresponding Steiner tree, denoted by

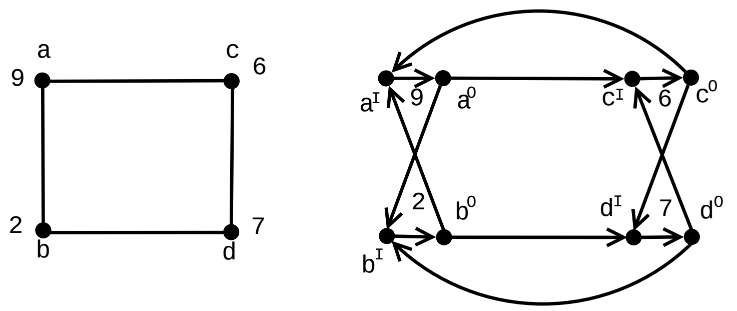

Fig. 2 A small example of graph transformation

t_R, is too big for sector s_j (*i.e.* $t_R > e_j + UB$), then the nodes of R cannot be all in sector s_j.

Proposition 2 *$\forall R \subseteq V$, $\forall j \in J$, if $t_R > e_j + UB$, then we have: $\sum_{v_i \in R} x_{ij} \leq |R| - 1$.*

This proposition dominates Lemma 4 since for any set of nodes $R \subseteq V$, we always have $f_R \leq t_R$. For example, in Figure 1, we have a terminal set $R = \{a, f\}$. The corresponding Steiner tree is $a \to c \to e \to f$ with $t_R = 9+6+2+2 = 19 > e_1 + UB$. With Proposition 2, a and f cannot be assigned simultaneously to sector s_1 and this can be expressed by $x_{a1} + x_{f1} \leq 1$. Noteworthy, this inequality cannot be derived from Lemma 4.

A node separator is defined as a set of nodes $Sep \subseteq V$, whose removal divides the graph into K ($K \geq 2$) disconnected components, denoted by C_k ($k \in \{1, ..., K\}$). For example, in Figure 1, $\{e\}$ is a node separator since without this node the graph is divided into 2 components ($C_1 = \{f\}$; $C_2 = \{a, b, c, d\}$). The nodes set $\{c, d\}$ is also a node separator whose cardinality is 2.

Proposition 3 *For any component C_k ($k \in \{1, ..., K\}$) generated by a given node separator $Sep \subseteq V$ and $\forall j \in J$, if $f_{C_k} < e_j - UB$ then, $\forall v_i \in C_k$, we have: $x_{ij} \leq \sum_{v_{i'} \in Sep} x_{i'j}$.*

Proof. Assume that a node separator Sep and the corresponding components C_k, $\forall k \in \{1, ..., K\}$ are given. Besides, let $j \in J$ be a sector index such that $f_{C_k} < e_j - UB$. $\forall k$, C_k forms a connected subgraph of G and the nodes in Sep are the only nodes of V connected to C_k. Therefore, similarly to Lemma 2, if $v_i \in C_k$ is assigned to sector s_j, then at least one $v_i \in Sep$ must be assigned also to sector s_j. □

For example, in Figure 1, $Sep = \{e\}$ is a separator. The component composed by $\{f\}$ is small for sector s_1 ($w_f < e_1 - UB$). By Proposition 3, we have that node f can be assigned to s_1 under the condition that node e is assigned to s_1, this can be expressed by $x_{f1} \leq x_{e1}$.

3 Heuristic solution of the sectorization problem

In this section, we propose to apply a greedy algorithm and a descent algorithm to obtain a feasible solution for the sectorization problem. The latter is based on the idea of the descent algorithm given by Teitz and Bart (1968) for the p-median problem. The greedy algorithm takes as input a set of p areas, denoted by $C = \{c_1, ..., c_p\}$, considered as centers of sectors in the algorithm. It returns a feasible solution S and its ojective function value Z. Let L_j be the set of basic areas that can be assigned to sector s_j at a given iteration of the greedy algorithm: $L_j = \{v_i | v_i \in V/v_i \notin s_j, \forall j \in J$, and $\exists v_k \in s_j$ such that $(v_i, v_k) \in E\}$. The algorithm is described as follows:

```
Procedure Greedy(C)
  Step 1.  $\forall j \in J$, $s_j = \{c_j\}$. Construct candidate lists $L_j$.
  Step 2.  While $\cup_{j\in J} L_j \neq \emptyset$, Repeat
           2.1.  Get $s_j$ :=arg $max_{k\in J/L_k\neq\emptyset}(e_k - f_k)$.
           2.2.  Get $v_i$ :=arg $max_{v_l\in L_j}(w_l)$.
           2.3.  Assign $v_i$ to $s_j$, update $f_j$ ($f_j := f_j + w_i$),
                 update the $L_j$'s.
           2.4.  End While.
  Step 3.  $S = \{s_1, ..., s_p\}$, $Z = max_{1\leq j\leq p}|f_j - e_j|$. Return ($S$, $Z$).
```

By integrating this greedy algorithm, the descent algorithm can provide a better solution. Details are as follows.

```
Descent Algorithm
  Step 1.  Select at random $p$ areas $c_p$ as sector centers $C = \{c_1, ..., c_p\}$,
           get (S, Z) := Greedy (C).
  Step 2.  $\forall v_i \in C$, $\forall v_k \in V\backslash C$,
               Get new sector centers $C_{ik} = C\backslash\{v_i\} \cup \{v_k\}$,
               Get $(S_{ik}, Z_{ik})$ := Greedy ($C_{ik}$).
  Step 3.  If $min_{i,k} Z_{ik} \geq Z$, then return ($S$, $Z$); End.
  Step 4.  $(i', k')$ = arg $min(Z_{ik})$, $S = S_{i'k'}$, $Z = Z_{i'k'}$, $C = C_{i'k'}$.
           Goto Step 2.
```

4 The preprocessing phase

4.1 A general preprocessing algorithm

Given a MIP formulation of an optimization problem, preprocessing are operations that can be performed to improve or simplify the formulation. A preprocessing phase can lead to fixing values of some variables, tightening bounds on the variables, identifying infeasibility or generating new valid inequalities. It is an important phase between formulation and solution. Some preprocessing techniques are described by Savelsberg (1994) and also by Nemhauser and Wolsey (1999). In this paper, we focus on fixing values of variables by means of a variable fixing technique applied only on the binary assignment variables x_{ij}. These variables are more significant than y_{ijk} and u_{ij} to the solution of the sectorization problem. Variable fixing techniques on boolean variables have already been applied successfully to NP-hard combinatorial optimization problems like the multidimensional knapsack problem (Osorio et al. 2002) or scheduling problems (T'kindt et al. 2007, Baptiste et al. 2010). They usually exploit information from the LP relaxation such as reduced costs of non basic variable or even pseudo-costs for basic variables. However, for our sectorization problem it turns out that the reduced costs and pseudo-costs are almost all equal to 0, thus providing no information on the increase of the optimal solution value of the LP relaxation when changing some x_{ij}'s values. Consequently, the variable fixing technique used in this paper is fully based on the iterative solution of LP problems in which variables are tested to be 0 or 1. So, the main idea consists in making optimal decisions with respect to the x_{ij}'s in an integer optimal solution. The goal is then to reduce in polynomial time the size of the instance to solve by an exponential-time algorithm like a branch-and-bound algorithm.

Let LB be a lower bound on the optimal solution value of the MIP. It is calculated by solving the LP relaxation of the MIP given in Section 2.1. Now, consider any assignment variable x_{ij}. We add a constraint $x_{ij} = 1$ to the LP formulation and solve it again. If the optimal value $\overline{Z}$ of this relaxed problem is greater than a known UB on the MIP solution or there is no feasible solution (this case can also be viewed as that the solution has an infinity value), then the variable x_{ij} is necessarily to 0 in an optimal solution to the MIP. We can use the same reasoning to set x_{ij} to 1. To complement this technique, we use the same technique by considering couples of variables. Let $x_{i_1j_1}$, $x_{i_2j_2}$ be two variables, and two subproblems LP_1 and LP_2: we add two constraints $x_{i_1j_1} = 1$, $x_{i_2j_2} = 1$ to the LP relaxation to obtain LP_1. Similarly, two constraints $x_{i_1j_1} = 1$, $x_{i_2j_2} = 0$ are added to obtain LP_2. If the optimal values of LP_1 and LP_2 are both greater than UB, then the variable $x_{i_1j_1}$ must be set to 0. A similar reasoning can be derived to fix variable $x_{i_1j_1}$ to 1. Since the cardinality of variables x_{ij} is $n \times p$ which is relatively small, we first test this fixing technique on each single variable x_{ij}, and then we select two specific basic areas i_1 and i_2 by some rules: in this paper, we select two relatively high weights ares i_1 and i_2 with w_{i_1}, $w_{i_2} \geq \sum_{i \in I} w_i/n$, and i_1 and i_2 are relatively far away from each other on the map with their shortest path $d_{i_1,i_2} > 3.0 \sum_{i \in I} w_i/n$. This technique generalizes the use of reduced costs and pseudo-costs for building lower bounds on the increase of the lower bound given by solving the continuous relaxation of the MIP. In this paper, we use this variable-fixing technique. The preprocessing for fixing variables is decribed as follows.

Preprocessing Algorithm

```
Step 1. $\forall i \in I$, $\forall j \in J$, If $x_{ij}$ is not fixed, Then
    1.1. Set $x_{ij} = 0$ and solve LP, $S_0$ is the corresponding solution
         and $Z_0$ is the objective function value.
         If $Z_0 > UB$, Then $x_{ij}$ is fixed to 1;
         Update UB:
         If $S_0$ is a feasible solution and $Z_0 < UB$, Then $UB = Z_0$;
    1.2. Set $x_{ij} = 1$ and solve LP, $S_1$ is the corresponding solution
         and $Z_1$ is the objective function value.
         If $Z_1 > UB$, Then $x_{ij}$ is fixed to 0;
         Update UB:
         If $S_1$ is a feasible solution and $Z_1 < UB$, Then $UB = Z_1$;
    1.3. Update LB:
         If $min\{Z_0, Z_1\} < LB$, Then $LB = min\{Z_0, Z_1\}$.
Step 2. For selected pairs $(i_1, i_2) \in I$, $\forall j_1 \in J$, If $x_{i_1j_1}$ is not fixed, Then
    2.1. Set $x_{i_1j_1} = 0$, $\forall j_2 \in J$,
        2.1.1. Set $x_{i_2j_2} = 0$ and solve LP, $S_0$ is the corresponding
               solution and $Z_0$ is the objective function value.
               Update UB: If $S_0$ is a feasible solution
               and $Z_0 < UB$, Then $UB = Z_0$;
        2.1.2. Set $x_{i_2j_2} = 1$ and solve LP, $S_1$ is the corresponding
               solution and $Z_1$ is the objective function value.
               Update UB: If $S_1$ is a feasible solution
               and $Z_1 < UB$, Then $UB = Z_1$;
        2.1.3. If $min\{Z_0, Z_1\} > UB$, then $x_{i_1j_1}$ is fixed to 1;
    2.2. Set $x_{i_1j_1} = 1$, $\forall j_2 \in J$,
        2.2.1. Set $x_{i_2j_2} = 0$ and solve LP, $S'_0$ is the corresponding
               solution and $Z'_0$ is the objective function value.
               Update UB: If $S'_0$ is a feasible solution
               and $Z'_0 < UB$, Then $UB = Z'_0$;
        2.2.2. Set $x_{i_2j_2} = 1$ and solve LP, $S'_1$ is the corresponding
               solution and $Z'_1$ is the objective function value.
               Update UB: If $S'_1$ is a feasible solution
               and $Z'_1 < UB$, Then $UB = Z'_1$;
        2.2.3. If $min\{Z'_0, Z'_1\} > UB$, then $x_{i_1j_1}$ is fixed to 0.
```

4.2 Strengthening the preprocessing

In Section 2.2 we have presented some valid inequalities which can be used to restrict the solution space of the sectorization problem. In this section, we show how to generate the inequalities using Lemmas 3 and 4, and some details regarding Propositions 2 and 3. To have the inequalities in Lemmas 3 and 4, we need to generate the set R with maximal cardinality for Lemma 3 and minimal cardinality for Lemma 4. Noteworthy, the algorithm for generating all 1-cuts for a knapsack constraint, proposed by Osorio et al. (2002), can be used to generate the set R. In Lemma 3, for a given j, we generate all the sets

R that satisfy $\sum_{i \in R} w_i < e_j - UB$. The procedure for generating R is depicted as follows:

Maximum Set Generation Algorithm

```
Step 1.  Basic areas $v_i$ are indexed in non-decreasing order of $w_i$,
         $i.e.\ 0 \leq w_1 \leq w_2 \leq ... \leq w_n$.
Step 2.  $t := 0$; $s := 0$; $m := 1$;
         While $s + w_t < e_j - UB$, Repeat
             $s := s + w_t$; $t := t + 1$;
         End While.
Step 3.  $R_0 := \{v_1, ..., v_t\}$.
Step 4.  For $k := t$ down to 1 Do
             For $l := t + 1$ to n Do
                 If $s - w_k + w_l < e_j - UB$, Then
                 $R_m := R_0 \backslash \{v_k\} \cup \{v_l\}$; $m := m + 1$;
             End For;
         End For;
```

For Lemma 4, the set R with minimal cardinality can be generated in a similar way.

In Proposition 2, for the Steiner tree based inequalities, we generate all the terminal sets R such that $|R| \leq 3$ and the condition of Proposition 2 is answered. When the cardinality of terminal set becomes greater, the time spent on searching the Steiner tree increases quickly compared to the reduction of the time needed to solve optimally the sectorization problem.

In Proposition 3, we need to identify if a node set *Sep* is a separator or not. By the definition of the node separator, if the removal of nodes of *Sep* creates a disconnected graph, then R is separator. To check the connectedness of a graph, we use the depth-first search (DFS) which is the basis of Tarjan's algorithm (1972). The DFS algorithm is a search method in which an edge emanating from the lastly reached node which remaining unexplored edges are always chosen. This rule can be applied using an adjacency structure. DFS is a very efficient algorithm which requires time $O(|V| + |E|)$ to check the connectedness. In the computational phase, we decide to generate separator set *Sep* such that $|Sep| \leq 3$. We limit to that size because when $|Sep|$ becomes greater, the inequalities becomes less efficient.

5 Computational results

This section presents the results of the computational experiments conducted to evaluate the efficiency of the preprocessing algorithm introduced in this paper. Instances are randomly generated according to various map configurations. Each instance is defined by a planar graph generated starting with a

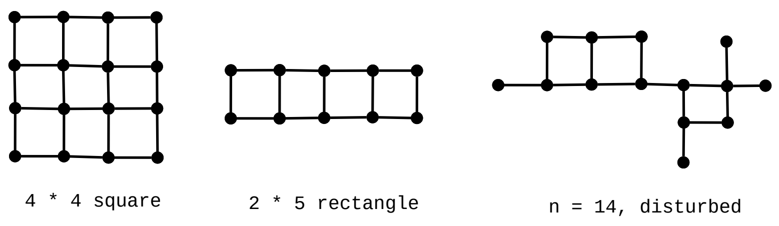

Fig. 3 Examples of the various map categories

given set of points (the basic areas) in the plane. Four different types of map instances are generated: square, rectangle, little disturbed and disturbed. A square map is a square of $m \times m$ basic areas with m given. A rectangle map is a rectangle of m rows and m' columns ($m \neq m'$), with $m \times m'$ basic areas. A little disturbed map is composed by basic areas that have various locations. And in a disturbed map basic areas have more various locations than a little disturbed map. We have some parameters to control each type of maps: the minimal width w_{min}, maximal width w_{max}, the minimal length l_{min}, maximal length l_{max}, the percentage pl that controls the possibility of a basic areas to appear at a given point on the map. For a square map, we choose at random $m \in \{4, 5\}$, $w_{min} = w_{max} = l_{min} = l_{max} = m$, and $pl = 100\%$. For a rectangle map, we choose at random $m \in \{2, 3\}$ and $m' \in \{6, 7, 8, 9\}$, $w_{min} = w_{max} = m$, $l_{min} = l_{max} = m'$, and $pl = 100\%$. For a little disturbed map, we choose at random $w_{min} \in \{3, ..., 7\}$ and $l_{min} \in \{3, ..., 7\}$, $w_{max} = 1.5w_{min}$, $l_{max} = 1.5l_{min}$, and $pl = 50\%$. For a disturbed map, we choose at random $w_{min} \in \{1, 2\}$, $w_{max} = 3w_{min}$, $l_{min} = 6w_{min}$, $l_{max} = 10w_{min}$ and $pl = 30\%$. For all kind of maps, we limit the number of basic areas n between 12 and 27, and the number of sectors p ranges between 2 and 4. For example, in Figure 3, there are a square map, a rectangle map and a disturbed map. The value of the statistical activity measure of each area is drawn at random between 1 and 100 and the target values of sector are almost balanced. For each combination between the type of the map and p, we generate 30 instances. The preprocessing algorithm has been implemented in C++ language and tested on a PC Inter Core 2 Duo processor 3.0 Ghz with 3.48 Gb of RAM. All mathematical formulations are solved by using solver CPLEX 12.2.

We evaluate the impact of the preprocessing algorithm by evaluating the gain in solving the MIP after its application. From a practical point of view, the LP relaxation used along the preprocessing is converted into a MIP model simply by changing to boolean the type of the variables not fixed by the preprocessing. Henceforth, the obtained MIP includes all valid inequalities and some fixed variables. In CPLEX solver, we set limits on the memory used by the search tree to 650 Mb and on the CPU time to 1800 seconds. CPLEX terminates if the problem is solved to optimality or one of these limits is reached.

Table 1 Comparison of the preprocessing with and without valid inequalities

Type	n_{min}	n_{max}	p	pv_{min}	pv_{avg}	pv_{max}	t_{min}	t_{avg}	t_{max}
square	16.0	25.0	2	0.0	0.0	0.0	0.1	0.2	0.4
				0.0	10.0	100.0	0.0	0.7	1.6
			3	0.0	0.0	0.0	0.3	0.7	1.3
				0.0	0.0	0.0	2.1	7.8	22.1
			4	0.0	0.0	0.0	0.5	1.3	2.5
				0.0	0.0	0.0	5.2	32.0	103.4
rectangle	12.0	27.0	2	0.0	0.0	0.0	0.0	0.2	0.5
				0.0	13.3	100.0	0.0	1.7	5.5
			3	0.0	0.0	0.0	0.1	0.7	1.9
				0.0	3.3	100.0	0.6	11.8	43.7
			4	0.0	0.0	0.0	0.3	1.5	4.4
				0.0	6.7	100.0	1.5	45.9	211.4
little disturbed	12.0	26.0	2	0.0	0.0	0.0	0.0	0.2	0.6
				0.0	26.7	100.0	0.0	1.8	6.6
			3	0.0	0.0	0.0	0.1	0.8	2.0
				0.0	13.3	100.0	0.7	12.4	52.1
			4	0.0	0.0	0.0	0.3	1.7	4.0
				0.0	13.3	100.0	2.2	57.7	190.3
disturbed	12.0	27.0	2	0.0	0.0	0.0	0.1	0.3	0.6
				100.0	100.0	100.0	0.0	0.2	0.8
			3	0.0	0.0	0.0	0.2	1.0	1.9
				0.0	76.7	100.0	0.5	9.2	40.4
			4	0.0	0.0	0.0	0.3	2.2	5.2
				0.0	31.3	100.0	1.5	52.4	310.4

In Table 1, we provide the results obtained by the preprocessing algorithm with and without the valid inequalities. For each type of map and each number of sectors p, there are two lines of results: the first line provides the results of the preprocessing without valid inequalities while the second line provides the results of the preprocessing with the valid inequalities. This table provides the minimum (n_{min}) and maximum (n_{max}) number of basic areas, the minimum (pv_{min}), average (pv_{avg}) and maximum (pv_{max}) percentage of fixed assignment variables x_{ij}, and the minimum (t_{min}), average (t_{avg}) and maximum (t_{max}) CPU times in seconds. The results provided in Table 1 show that the valid inequalities given in Section 2.2 are useful to fix variables: without them, no variable is fixed by the preprocessing. Besides, in terms of hardness, the square maps are the hardest map, the rectangle maps are second, the little disturbed maps are third, and the disturbed maps are the easiest ones. Hopefully, except on the square maps the preprocessing with valid inequalities is able to fix variables on all map configurations: the average percentage of fixed variables ranges from 3.3% for the rectangle maps and 3 sectors up to 100% for disturbed maps with 2 sectors. If we discard the square maps which are very difficult ones, the overall average percentage of fixed variables is about 32%.

Besides when p increases, the problem becomes more difficult to solve. In terms of running times, the preprocessing with valid inequalities takes more

Table 2 Comparison between original MIP model and MIP model with preprocessing (1)

Type	p	pc	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
square	2	100.0	227.0	64618.4	347444.0	0.3	45.7	330.9
		96.7	0.0	16390.9	152887.0	0.0	13.3	130.5
	3	66.7	796.0	85628.4	658183.0	1.9	144.6	1634.5
		70.0	193.0	96439.0	1058146.0	4.4	160.6	1666.0
	4	56.7	14137.0	120403.8	295361.0	72.7	238.1	496.3
		90.0	3.0	58111.4	378665.0	9.9	295.8	1355.1
rectangle	2	96.7	0.0	231867.4	1903935.0	0.0	165.6	1519.3
		100.0	0.0	39550.2	1113782.0	0.0	60.8	1676.7
	3	73.3	613.0	135976.3	718409.0	0.5	243.2	1637.0
		93.3	0.0	14778.5	298746.0	0.6	43.6	612.3
	4	60.0	1078.0	144623.3	585489.0	3.5	356.2	1708.7
		96.7	0.0	21483.8	301202.0	1.5	142.3	1197.0
little disturbed	2	83.3	66.0	50999.2	236296.0	0.3	38.9	204.9
		100.0	0.0	5494.0	84459.0	0.0	6.4	48.0
	3	63.3	730.0	180910.7	781927.0	0.6	355.0	1677.3
		93.3	0.0	44076.5	739065.0	0.7	98.4	1506.9
	4	40.0	1095.0	88410.8	392136.0	3.8	260.2	1329.2
		93.3	0.0	49009.0	697339.0	2.3	182.5	1810.8
disturbed	2	100.0	487.0	11211.3	108141.0	0.3	13.1	68.7
		100.0	0.0	0.0	0.0	0.0	0.2	0.8
	3	93.3	669.0	35621.4	226787.0	0.7	115.6	557.7
		100.0	0.0	2254.3	34946.0	0.5	13.1	93.7
	4	66.7	379.0	37444.3	179166.0	0.6	181.4	670.8
		96.7	0.0	9219.9	90276.0	1.5	79.7	431.7

time since we need to generate the valid inequalities, and some of them require significant CUP times. This is the case of the valid inequalities via the solution of Steiner tree problems (Proposition 2).

Tables 2 and 3 evaluate the impact of the preprocessing with valid inequalities on the solution of the MIP. Table 2 is arranged in the same way as Table 1: for each type of map and each p, the first line provides the results corresponding to the original MIP model while the second line corresponds to the MIP solution with an initial preprocessing. Table 2 provides the percentage of instances solved to optimality (pc) within the limits, the minimum (nd_{min}), average (nd_{avg}) and maximum (nd_{max}) number of nodes in the branch-and-cut tree explored by CPLEX, and the minimum (t_{min}), average (t_{avg}) and maximum (t_{max}) CPU times, in seconds, required to solve the problems to optimality. Noteworthy, the CPU times of the MIP plus preprocessing include the times required by the preprocessing (see Table 1). The presented results are computed on all the instances, including those for which at least one solution of a MIP fails within the imposed limits. In Table 3, the same results as in Table 2 are computed but only considering instances solved by CPLEX for the MIP with and without preprocessing.

The results provided in Table 2 and Table 3 show that generally the MIP with preprocessing can solve larger instances than the original MIP. First,

Table 3 Comparison between original MIP model and MIP model with preprocessing (2)

type	p	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
square	2	227.0	56186.7	347444.0	0.3	36.5	330.9
		0.0	16390.9	152887.0	0.0	13.3	130.5
	3	796.0	56331.0	222279.0	1.9	64.4	289.3
		193.0	34374.7	581020.0	4.4	59.9	947.0
	4	14137.0	120403.8	295361.0	72.7	238.1	496.3
		3.0	758.5	2398.0	9.9	17.1	24.5
rectangle	2	0.0	231867.4	1903935.0	0.0	165.6	1521.9
		0.0	2507.8	56640.0	0.0	5.0	45.9
	3	613.0	135976.3	718409.0	0.5	243.2	1637.0
		0.0	267.8	1357.0	0.6	8.7	26.1
	4	1078.0	144623.3	585489.0	3.5	356.2	1708.7
		0.0	482.7	2143.0	1.5	17.1	35.9
little disturbed	2	66.0	50999.2	236296.0	0.3	38.9	204.9
		0.0	5394.7	84459.0	0.0	5.3	48.0
	3	730.0	180910.7	781927.0	0.6	355.0	1677.3
		0.0	4461.7	58278.0	0.7	14.0	110.6
	4	1095.0	88410.8	392136.0	3.8	260.2	1329.2
		0.0	489.3	1485.0	2.3	12.9	45.0
disturbed	2	487.0	11211.3	108141.0	0.3	13.1	68.7
		0.0	0.0	0.0	0.0	0.2	0.8
	3	669.0	35621.4	226787.0	0.7	115.6	557.7
		0.0	8.4	134.0	0.5	8.5	48.0
	4	379.0	37292.5	179166.0	0.6	183.1	670.8
		0.0	474.9	5647.0	1.5	25.8	75.4

except on square maps with 2 sectors, the MIP with preprocessing always helped in solving more instances within the imposed limits: the most favorable scenario is obtained for little disturbed maps with 4 sectors where using the preprocessing leads to increase the percentage of solved instances from 40% to 93.3% (Table 2). Regarding the number of nodes of the branch-and-cut algorithm applied by CPLEX to solve the two MIP, Table 3 shows that for the hardest instances, *i.e.* square maps, the percentage of reduction of the average number of nodes is about 70.8%, 39.0%, 99.4% for 2, 3, 4 sectors, respectively. Therefore, the preprocessing has a strong impact on the exact solution of the MIP which is confirmed by the CPU times required by CPLEX. Table 3 shows that, again, for the square maps the percentage of reduction of the average CPU times about 63.6%, 7.0%, 92.8% for 2, 3, and 4 sectors, respectively. And for the disturbed maps, the percentages of reduction of CPU time are about 98.5%, 92.6%, 85.9% for 2, 3, and 4 sectors, respectively. The comparison of Tables 2 and 3 shows that most of the time the preprocessing enables to solve instances that CPLEX was unable to solve. Exceptions are most of the square maps and one disturbed map with 4 sectors. It turns out that even by considering those instances for which the MIP without preprocessing was unsolved by CPLEX, the average CPU time and number of nodes required to solve the MIP with preprocessing are better. This enforces the conclusion that the preprocessing, as introduced in this paper, helps significantly in solving the sectorization problem.

6 Conclusion

In this paper, we have formulated the sectorization problem as an assignment problem with connectedness constraints and we have developed a descent heuristic algorithm to find a feasible solution. This information is then used as an initial upper bound on the optimal solution value. New valid inequalities are developed to limit the solution space and are integrated into a dedicated preprocessing procedure aiming at reducing the number of variables. Computational results indicate that the new valid inequalities and the preprocessing algorithm can serve to fix variables, especially on irregular maps. Compared with the original MIP model the preprocessing is shown to be very efficient in making faster the optimal solution of the problem.

As future research we intend to develop a dedicated branch-and-bound algorithm, incorporating lower bounds, improved upper bound, stronger dominance relations and cut generation processes. This branch-and-bound algorithm will fully exploit the preprocessing technique.

References

1. Baptiste, P., Della Croce, F., Grosso, A., T'kindt, V., Sequencing a single machine with due dates and deadlines: an ILP-based approach to solve very large instances, Journal of scheduling, 13(1), 39-47. (2010)
2. Berman, O. , Drezner,Z., Tamir, A., Wesolowsky, G.O., Optimal location with equitable loads, Annals of Operations Research, 167, 307-325. (2009)
3. Bozkaya, B., Erkut, E., Laporte, G., A tabu search heuristic and adaptive memory procedure for political districting, European Journal of Operational Research, 144, 12-26. (2003)
4. Bozkaya, B., Erkut, E., Laporte, G., Neuman, S., Political Districting: Solving a Multi-Objective Problem Using Tabu Search, Kluwer, Boston (2005)
5. Charikar, M., Chekuri, C., Cheung, T., Dai, Z., Goel, A., Guha, S., Li, M., Approximation algorithms for directed Steiner problems, Journal of Algorithms, 33, 73-91. (1999)
6. Daskin, M. S., Network and Discrete Location: Models, Algorithms and Applications, JohnWiley, New York (1995)
7. Dreyfus, S.E., Wagner, R.A., The Steiner problem in graphs, Networks, 1, 195-207. (1972)
8. Drezner, Z., Hamacher, H.W., Facility Location: applications and theory. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, Berlin (2004)
9. Eiselt, H.A., Sandblom, C.-L., Decision Analysis, Location Models, and Scheduling Problems, Springer-Verlag, Berlin-Heidelberg-New York (2004)
10. Garfinkel, R.S., Nemhauser, G.L., Optimal political districting by implicit enumeration techniques, Management Science, 13, 495-508. (1970)
11. Grilli di Cortona, P., Manzi, C., Pennisi, A., Ricca, F., Simeone, B., Evaluation and optimization of electoral systems, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (1999)
12. Hakimi, S.L., Steiner's problem in graphs and its implications, Networks, 1, 113-133. (1971)
13. Hansen, P., Jaumard, B., Cluster analysis and mathematical programming, Mathematical Programming, 79, 191-215. (1997)
14. Hess, S.W., Weaver, J.B., Siegfeldt, J.N., Whelan, J.N., Zhitlau, P.A., Nonpartisan political districting by computer, Operations Research, 13, 998-1006. (1965)
15. Kalcsics, J., Nickel, S., Puerto, P., Rodríguez-Chía, A., The ordered capacitated facility location problem, An Official Journal of the Spanish Society of Statistics and Operations Research, 18, 203-222. (2010)

16. Karp, R.M., Reducibility among combinatorial problems, Complexity of Computer Computations, 85-103, Plenum, New York. (1972)
17. Karypis, G., Kumar, V., A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing, 20(1), 359-392. (1998)
18. Kernighan, B., Lin, S., An efficient heuristic procedure for partitioning graphs, The Bell System Technical Journal, 49(2), 291-307. (1970)
19. Mehrotra, A., Constrained graph partitioning: decomposition, polyhedral structure and algorithms, Ph.D, thesis, Georgia Institute of Technology, Atlanta, GA (1992)
20. Mehrotra, A., Johnson, E.L., Nemhauser, G.L., An optimization based heuristic for political districting, Management Science, 44, 1100-1114. (1998)
21. Miller, C., Tucker, A.W, Zemlin, R.A, Integer programming formulation of the travelling salesman problem, Journal of the Association for Computing Machinery 7 (1960)
22. Nemhauser, G.L., Wolsey, L.A., Integer and Combinatorial Optimization, Wiley-Interscience, New York. (1999)
23. Niedermeier, R., Invitation to Fixed-Parameter Algorithms, Oxford lecture series in mathematics and its application 31, Oxford University Press, USA. (2008)
24. Osorio, M.A., Glover, F., Hammer, P., Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions, Annals of Operations Research, 117, 71-93. (2002)
25. Ou, C., Ranka, S., Fox, G., Fast and parallel mapping algorithms for irregular and adaptive problems, Journal of Supercomputing, 10, 119-140. (1996)
26. Patra, A., Kim, D., Efficient mesh partitioning for adaptive hp finite element methods, In International Conference on Domain Decomposition Methods. (1998)
27. Savelsberg, M.W.P., Preprocessing and probing techniques for mixed integer programming problems, ORSA Journal of Computing, 6, 445-454. (1994)
28. Schloegel, k., Karypis, G., Kumar, V., Graph Partitioning for High Performance Scientific Simulations. (2000)
29. T'kindt, V., Della Croce, F., Bouquard, J.-L., Enumeration of Pareto optima for a flowshop scheduling problem with two criteria, INFORMS Journal on Computing, 19(1), 64-72. (2007)
30. Tarjan, R., Depth-first search and linear graph algorithms, SIAM J Computing, 1, 146-160. (1972)
31. Teitz, M. B., Bart, P., Heuristic methods for estimating the generalized vertex median of a weighted graph, Operations Research, 16, 955-961. (1968)
32. van Roy, T. J., A cross decomposition algorithm for capacitated facility location, Operations Research, 34, 145-163. (1986)