A Simulated Annealing Algorithm for the Directed Steiner Tree Problem

Matias Siebert[1], Shabbir Ahmed[1], and George Nemhauser[1]

[1]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA

Abstract

In [29] the authors present a set of integer programs (IPs) for the Steiner tree problem, which can be used for both, the directed and the undirected setting of the problem. Each IP finds an optimal Steiner tree with a specific structure. A solution with the lowest cost, corresponds to an optimal solution to the entire problem. The authors show that the linear programming relaxation of each IP is integral and, also, that each IP is polynomial in the size of the instance, consequently, they can be solved in polynomial time. The main issue is that the number of IPs to solve grows exponentially with the number of terminal nodes, which makes this approach impractical for large instances.

In this paper, we propose a local search procedure to solve the directed Steiner tree problem using the approach presented in [29]. In order to do this, we present a dynamic programming algorithm to solve each IP efficiently. Then we provide a characterization of the neighborhood of each tree structure. Finally, we use the proposed algorithm and the neighborhood characterization to solve the problem using a simulated annealing framework. Computational experiments show that the quality of the solutions delivered by our approach is better than the ones presented in the literature for the directed Steiner tree problem.

1 Introduction

The Steiner tree problem is a classic network design problem. There are two versions, directed and undirected. In the directed version (DST), we are given a directed graph $D = (V, A)$, with non-negative arc costs c_a for all $a \in A$, a root node $r \in V$, and a set of terminal nodes $R \subset V$. In the undirected case (UST), we are given an undirected graph $G = (V, E)$, with non-negative edge costs c_e for all $e \in E$, and a set of terminal nodes $R \subseteq V$. In the latter case, one can always take the bidirected version of the graph G, pick an arbitrary terminal node $r \in R$ as a root node, and solve the problem as a directed Steiner tree. Note that we may assume without loss of generality that $c > 0$, since we can contract all the edges with cost 0.

The undirected version of the problem is known to be an NP-Hard [19], and it is even NP-Hard to find an approximate solution whose cost is within a factor of $\frac{96}{95}$ of the optimum [3, 9]. Nevertheless, there are constant factor approximation algorithms for this problem [7, 28, 36], with $\ln(4) + \epsilon$ being the best known approximation factor for the undirected problem.

On the other hand, for DST there are not any known constant factor approximation algorithms. Actually, in [17], the authors show that the directed Steiner tree problem admits no $\mathcal{O}\left(\log^{2-\epsilon}(|R|)\right)$, unless $NP \subseteq ZTIME(n^{polylog(n)})$. The best known approximation ratio for the DST problem is $\mathcal{O}\left(|R|^{\frac{1}{t}}\right)$ for $t > 1$, which corresponds to the algorithm presented in [8]. This algorithm constructs low density directed Steiner trees in polynomial time, where the density of a tree is defined as the ratio of the cost of the tree over the number of terminals of the tree.

For UST, there are several efficient algorithms to reduce the instances size, keeping at least one optimal solution. Moreover, there exists efficient algorithms to solve the problem [14, 13, 25, 24]. These algorithms

solve to optimality a large part of the problems in the SteinLib library [21]. Since all of those algorithms assume the graph is undirected, in most cases they cannot be extended to the directed problem [27].

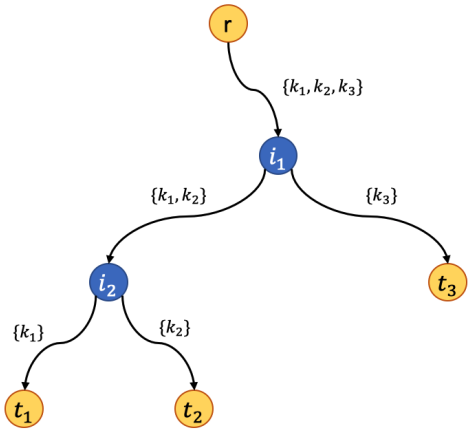
In contrast, there are not many papers in the literature related to computational experiments for the directed Steiner tree problem. In [33], the authors present two approximation algorithms, with $\mathcal{O}(|R|)$ and $\mathcal{O}(\sqrt{|R|})$ approximation ratios respectively. The authors present computational experiments, creating directed instances from over 900 undirected instances of the SteinLib library. All the directed graphs created by the authors keep at least one optimal solution of the original instance, consequently, they know in advance the value of an optimal solution. They compare their proposed algorithms against four benchmark algorithms used in practice. The authors conclude that their proposed algorithms outperform the rest of the studied algorithms in terms of solution quality, while having similar execution times.

In this paper we focus on the new approach proposed in [29], where the authors present a set of integer programs (IP) for the Steiner tree problem. The solution of an IP with lowest cost, corresponds to an optimal solution of the problem. Each IP corresponds to a specific tree structure, and its size is polynomial in the size of the instance. Moreover, the linear relaxation of each IP is integral, therefore each sub-problem can be solved in polynomial time. The issue with the proposed approach is that the number of IPs to solve grows exponentially with the number of terminal nodes. We present a dynamic programming algorithm and a simulated annealing based framework to address this issue, and we present computational experiments comparing the proposed approach to all the algorithms studied in [33].

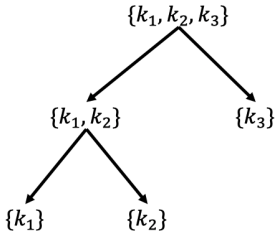
The remainder of the paper is organized as follows. Section 2 provides a detailed review of the approach presented in [29], and introduces definitions used in the paper. Section 3 presents an efficient algorithm to solve each subproblem of the approach presented in [29]. Section 4 provides a complete characterization of the neighborhood of each subproblem, which is needed since simulated annealing performs a local search at each iteration. Section 5 presents the proposed simulated annealing framework to solve the problem. This section also provides a routine to check and, potentially, improve the quality of the solutions obtained at each iteration of the simulated annealing algorithm. Moreover, this section also presents a special analysis for the case of rectilinear graphs. In Section 6, we present computational experiments obtained by solving the directed Steiner tree problem using our proposed formulation. Computational experiments show that the quality of the solutions delivered by our approach is better than the ones presented in the literature for the directed Steiner tree problem. Section 7 gives conclusions.

2 Description of LP-based approach

The directed Steiner tree problem is to construct a minimum cost directed tree, rooted at r, which spans all terminal nodes. In [29], the authors give a new approach to solve this problem. It involves solving a set of independent IPs, with the property that the solution of an IP with lowest cost corresponds to an optimal solution of the problem. Each IP solves the problem for a fixed Steiner tree structure, where each structure is defined by the way the paths from r to each terminal node share arcs. In this approach, one commodity per terminal node is created. The source node of all commodities is the root node r, and the sink is the corresponding terminal node. Figure 1 illustrates the idea.



(a) 3 terminals directed Steiner tree rooted at r.



(b) Structure of directed Steiner tree.

Figure 1: A 3 terminals directed Steiner tree rooted at r, and tree representation of its laminar structure.

Figure 1a shows a directed Steiner tree, which is composed of 5 paths. Above each path, the set of commodities that share arcs in such paths is highlighted. Figure 1b displays the tree representation of the laminar structure of the sets that are present in the solution of the tree shown in Figure 1a.

The main result in [29] is that the formulation for each structure is perfect, meaning that the linear programming (LP) relaxation of each IP corresponds to the convex hull of the set of all feasible solutions to the IP. Consequently, each IP can be solved by solving its LP relaxation. Furthermore, each IP has polynomial size with respect to the size of the input data, consequently, they can be solved efficiently, even for large instances. The main drawback of the proposed approach, is that the number of IP problems to solve grows exponentially with the number of terminal nodes, which makes this approach tractable only when the number of terminal nodes is fixed, and impractical for real-world size instances. For a detailed explanation of this approach we refer the readers to [29].

2.1 Notation

In this section, we introduce notation used throughout this paper. We assume we have a directed Steiner tree instance. Let $D = (V, A)$ be the directed input graph, where V is the set of nodes of the graph, and A is the set of arcs of the graph. We are also given a root node $r \in V$, and a set of terminal nodes $R \subseteq V \backslash \{r\}$. The nodes in $V \backslash (R \cup \{r\})$ are called Steiner nodes. For each terminal node $t_k \in R$, we define a commodity k, and we define K to be the set of all commodities. All commodities share the same source node, which is r, and for each commodity $k \in k$, its sink node is the corresponding terminal $t_k \in R$. Let b be the number of commodities, and let S be the set of all non-empty subsets of elements in K.

As discussed in section 2, the tree structure of every directed Steiner tree is defined by the sets of commodities present in such tree. This collection of sets of S forms a laminar family. Recall that a family $\mathcal{C}$ of sets is *laminar* if for every $A, B \in \mathcal{C}$ we either have that $A \subseteq B$, or $B \subseteq A$, or $A \cap B = \emptyset$. We define $\mathcal{L}_b$ to be the set of admissible laminar families that describe the structure of a Steiner tree with b commodities, where an admissible laminar family is one that contains the set of all commodities, and all the singletons. For laminar family $l \in \mathcal{L}_b$, we define $S(l)$ as the set of subsets of K that are present in laminar family l.

We say that $\hat{s} \in S(l)$ is the *parent* set of $s' \in S(l)$, if s' is one of the sets obtained when $\hat{s}$ is split according to laminar family l. Moreover, we say that s' is a *child* set of $\hat{s}$ in l. Notice that, for all $s' \in S(l)$ with $|s'| < |K|$, there is exactly one parent set of $\hat{s}$ in $S(l)$. Also, for each set $\hat{s} \in S(l)$ with $|\hat{s}| \geq 2$, there is at least two child sets of $\hat{s}$ in $S(l)$. We say that a directed Steiner tree T in D follows an (r, l) *structure*, if T is

an arborescence rooted in r that follows the splitting sequence defined by laminar family $l \in \mathcal{L}_b$.

Finally, we define $\mathcal{Z}_l$ to be the sub-problem of laminar family $l \in \mathcal{L}_b$, where we look for a lowest cost (r, l) structured Steiner tree.

3 Algorithm to solve $\mathcal{Z}_l$

Although $\mathcal{Z}_l$ can be solved by a linear programming algorithm as noted above, we will present in this section a recursive algorithm that is much more efficient.

3.1 Observations and algorithm intuition

Consider a fixed laminar family $l \in \mathcal{L}_b$, and let $s \in S(l)$ be such that $2 \leq |s| < |K|$. Since we are in a particular laminar family l, then we know the way s, and its subsets, split. Now, suppose we assume that the path of set s starts at node i, then the solution is equivalent to finding a minimum Steiner tree rooted at i, with $|s|$ commodities[1], that follows the splitting sequence for s defined in l. We will denote $l(s)$, to the "sub-laminar" family of l, when we only focus on s.

For any laminar family, the problem reduces to finding the splitting nodes, because once we have the splitting nodes, then we only need to connect the corresponding nodes by the shortest paths between them. The proposed algorithm uses this fact, and since we do not know the splitting nodes in advance, then we need to compute the shortest path between all pairs of nodes in D.

Consider the following example. Suppose we want to solve a directed Steiner tree with 4 terminals. Also, suppose we are solving the sub-problem for the laminar family shown in Figure 2

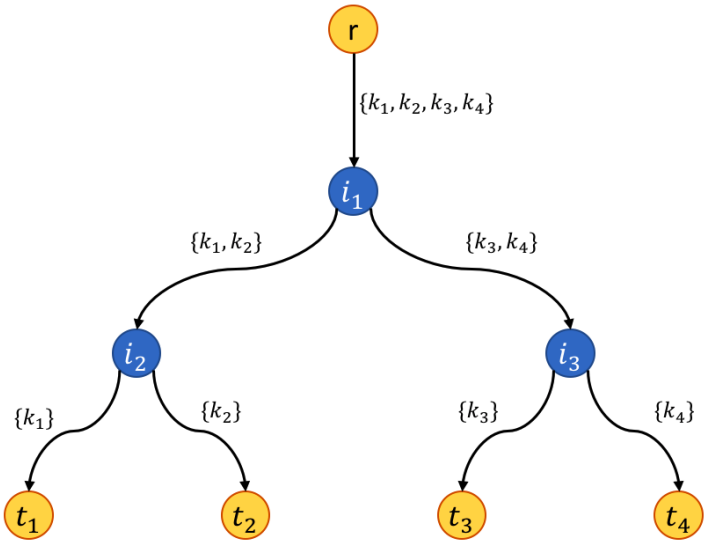


Figure 2: Laminar family representation.

We have to find nodes i_1, i_2, and i_3, shown in Figure 2. The idea of the proposed algorithm to solve $\mathcal{Z}_l$ is the

[1]This is equivalent to a directed Steiner tree whose root node is i, and that has $|s|$ terminal nodes which correspond to $\{t_k\}_{k \in s}$

following. For all sets $s \in S(l)$ with $2 \leq |s| < |K|$, and for all nodes $i \in V$, we define the minimum $(i, l(s))$ structured Steiner tree, whose optimal solution will be denoted by $x(i, s)$. Now, consider the set $\{k_1, k_2\}$ in the example shown in Figure 2. If we fix a given node i as a root for its corresponding Steiner tree, then we can compute $x(i, \{k_1, k_2\})$ in the following fashion. For all $j \in N$, we compute the sum of 3 shortest paths, which are an i-j shortest path, a j-t_1 shortest path, and a j-t_2 shortest path. Let j^* be a node such that the sum is the smallest, then $x(i, s)$ is the union of an i-j^* shortest path, a j^*-t_1 shortest path, and a j^*-t_2 shortest path. We repeat that procedure for all $i \in V$ and we will have computed all possible Steiner trees for $\{k_1, k_2\}$. This procedure is the same for $\{k_3, k_4\}$. For set $\{k_1, k_2, k_3, k_4\}$, the process is similar, since $x(i, \{k_1, k_2, k_3, k_4\})$ is going to be the union of 3 solutions, but in this case, it is the union of a shortest i-j^* path, $x(j^*, \{k_1, k_2\})$ and $x(j^*, \{k_3, k_4\})$. Note, that since in this case $K = \{k_1, k_2, k_3, k_4\}$, then we only care about $x(r, \{k_1, k_2, k_3, k_4\})$.

3.2 Proposed algorithm

Before stating the algorithm, we introduce the notation used within this section.

- $sp(i, j)$: Collection of arcs that compose a shortest i-j path in D.
- $c(sp(i, j))$: Cost of a shortest i-j path in D.
- $z(i, s)$: Cost of solution of set s rooted in i.
- $j_s^*(i)$: Node at which set s splits, for an optimal $(i, l(s))$ structured directed Steiner tree.
- $N(s)$: Nodes that can be root node for subset s. Note that $N(s) = N$ for all $s \in S(l)$, with $s \subset K$. For $s = K$ we have $N(s) = r$.

Observe that, for all $k \in K$, $z(i, k) = c(sp(i, t_k))$. Since we only focus on laminar families whose tree representation is a full binary tree (see Proposition 5 in [29]), then each set with at least 2 elements has exactly 2 children. We denote s_1 and s_2 to be the child sets of set $s \in S(l)$, $|s| \geq 2$. As an abuse of notation, we define $x(i, s)$ as follows $x(i, s) = sp(i, j) + x(j, s_1) + x(j, s_2)$, which means that the solution of the directed Steiner tree, rooted in i using sets in s, is the union of a shortest i-j path, and the solutions of its two child sets s_1 and s_2 rooted in j, where j is the splitting node of set s.

The recursion to compute $z(i, s)$ and $x(i, s)$ is the following,

$$z(i, s) = \min_{j \in N} \Big\{ c(sp(i, j)) + z(j, s_1) + z(j, s_2) \Big\} \qquad \forall i \in N(s), s \in S(l) : |s| \geq 2$$

$$j_s^*(i) \in \operatorname*{Argmin}_{j \in N} \Big\{ c(sp(i, j)) + z(j, s_1) + z(j, s_2) \Big\} \qquad \forall i \in N(s), s \in S(l) : |s| \geq 2$$

$$x(i, s) = sp(i, j_s^*(i)) + x(j_s^*(i), s_1) + x(j_s^*(i), s_2) \qquad \forall i \in N(s), s \in S(l) : |s| \geq 2$$

Note that for $s = K$, we only need to compute $z(r, K)$ and $j_K^*(r)$. Assume that $S(l)$ is ordered in increasing order of the cardinality of its elements. Then, the algorithm to solve $\mathcal{Z}_l$ is the following,

Algorithm 1 Compute $x(i,s)$ and $z(i,s)$ for $s \in S(l), |s| \geq 2$

1: Set $x(i,k) = sp(i,t_k)$ for all $k \in K, i \in V$.
2: Set $z(i,k) = c(sp(i,t_k))$ for all $k \in K, i \in V$.
3: Set $z(i,s) = +\infty$ for all $s \in S(l), |s| \geq 2, i \in N$
4: **for** $s \in S(l), |s| \geq 2$ **do**
5: **for** $i \in N(s)$ **do**
6: **for** $j \in N$ **do**
7: **if** $c(sp(i,j)) + z(j,s_1) + z(j,s_2) < z(i,s)$ **then**
8: $x(i,s) \leftarrow sp(i,j) + x(j,s_1) + x(j,s_2)$
9: $z(i,s) \leftarrow c(sp(i,j)) + z(j,s_1) + z(j,s_2)$
return $x(r,K)$ and $z(r,K)$.

First, note that since the elements in $S(l)$ are ordered in increasing order of their cardinality, then for all $s \in S(l)$ with $|s| \geq 2$ we compute $x(i,s_1)$ and $x(i,s_2)$ before computing $x(i,s)$ and its cost $z(i,s)$, since $|s_1| < |s|$ and $|s_2| < |s|$. Now, the *for loop* in step 5 is looping over all possible root nodes for set s. The *for loop* in step 6, loops over all splitting nodes for set s. In step 7, we check whether the current best solution can be improved. If the solution can be improved, then the best solution is updated. Finally, the solution is given by $x(r,K)$ whose cost if $z(r,K)$.

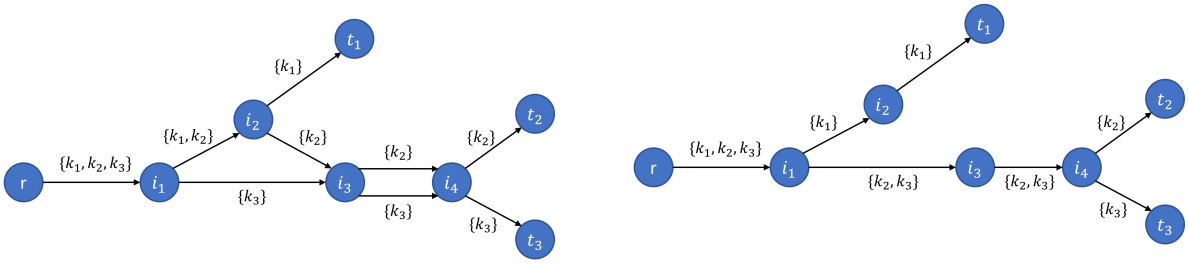
Note that the number of sets in $S(l)$, with at least 2 elements, is $b-1$, since the tree representation of l is a full binary tree. Moreover, we have to visit at most n^2 pairs of nodes for each set in $S(l)$, with cardinality of at least 2, where $n = |V|$. Consequently, if the number of terminals is fixed, the complexity of the proposed algorithm is $\mathcal{O}(n^2)$.

One can think of this algorithm as a simplified version of the algorithm proposed in [12]. The main difference is that the algorithm proposed [12] has to decide the way each set is split, while in this case, since we are in a particular laminar family, the split for every set is fixed.

3.3 Solution Improvements

For a given laminar family $l \in \mathcal{L}_b$, an optimal solution to $\mathcal{Z}_l$ may not correspond to a directed Steiner tree, but in the support of the solution we may have a directed Steiner tree, with a different structure, i.e., from a different laminar family in $\mathcal{L}_b$ (see Proposition 1 of [29]).

Figure 3 shows an example where an optimal solution of a laminar family l is not a directed Steiner tree. For the example, let $S(l) = \{\{k_1,k_2,k_3\},\{k_1,k_2\},\{k_1\},\{k_2\},\{k_3\}\}$.



(a) Optimal solution x for $\mathcal{Z}_l$.

(b) Steiner tree with different structure, contained in support of x.

Figure 3: Optimal solution x of $\mathcal{Z}_l$, which is not a directed Steiner tree, and a Steiner tree, with different structure contained in support of x.

We observe in Figure 3a, an optimal solution x of $\mathcal{Z}_l$, which is not a directed Steiner tree. Note that commodities k_2 and k_3 share arcs, but not using set $\{k_2, k_3\}$ because this set is not in l. In contrast, there is a Steiner tree, with $\hat{l}$ structure, contained in the support of x, where $S(\hat{l}) = \{\{k_1, k_2, k_3\}, \{k_2, k_3\}, \{k_1\}, \{k_2\}, \{k_3\}\}$.

Since the cost vector is positive, then for every set $s \in S(l)$, there are no cycles in any optimal solution of $\mathcal{Z}_l$. Consequently, the only way the solution is not a directed Steiner tree is when, at least, two different sets reach the same node. Following the notation of [29], if $x = (f, \hat{y}, \overline{y}, w)$ is an optimal solution for $\mathcal{Z}_l$, then there exists $i \in V$, such that for two sets $s_1, s_2 \in S(l)$, we have that $\sum_{a \in \delta^-(i)} f_a^{s_1} = 1$ and $\sum_{a \in \delta^-(i)} f_a^{s_2} = 1$. Whenever this happens, we have that the support of f vector does not correspond to a directed tree rooted in r.

By construction of the solution, there exists at least 1 path from r to t_k for all $k \in K$. Therefore, we can construct an r-arborescence using the arcs of the support of f, such that all the terminal nodes t_k are reached from r, and that all leaves of such arborescence correspond to terminal nodes. Let $A(l) = \{a \in A : \sum_{s \in S(l)} f_a^s \geq 1\}$ be the support of an optimal solution x to $\mathcal{Z}_l$, and let $T \subseteq A(l)$ be the set of arcs used by a minimum cost r-arborescence constructed in the subgraph defined by $A(l)$. If all leaves of T are terminal nodes, then T is a directed Steiner tree. On the other hand, if at least 1 leaf is a Steiner node, then we can prune such leaves and still have an r-arborescence with a path connecting r to t_k for all $k \in K$. If the tree after the first round of pruning still has leaves that are Steiner nodes, then we can prune such a tree again. After a finite number of rounds of pruning, all leaves of the r-arborescence will correspond to terminal nodes, which will correspond to a directed Steiner tree.

Once we have a rooted Steiner tree, we can identify its structure. We just have to see the way the paths from the root r to every terminal t_k, share arcs. Since we are considering admissible laminar families, then the set of all commodities and all the singletons are always present. Consequently, by identifying the nodes where a split occurs, we can determine into which sets each set is partitioned, and therefore, identify the parent-child relationship between sets.

It may happen that the laminar family of such a tree may not have a full binary tree representation, or in other words, we may have that a set has more than 2 children. In this case, there are several laminar families in $\mathcal{L}_b$ that have the constructed solution as a feasible solution. When this happens, we randomly select one of those laminar families as the laminar family of the constructed solution, using Algorithm 2. Lets call the selected laminar family $\hat{l}$. Finally, since the constructed solution may not be an optimal solution for $\hat{l}$, we solve the sub-problem $\mathcal{Z}_{\hat{l}}$ to get a new solution.

We describe the algorithm to construct a random laminar family from a laminar family, whose tree representation is not a full-binary tree. We refer to a laminar family l as the collection of sets contained in the family, denoted by $S(l)$.

Algorithm 2 Algorithm *randomLaminar*(l)

Require: Collection of sets $S(l)$.
1: Set $S' = \{s \in S(l) : s \text{ has at least 3 children}\}$
2: **while** $S' \neq \emptyset$ **do**
3: Select arbitrary $\hat{s} \in S'$
4: Let $\hat{S} = \{s \in S(l) : s \text{ is child set of } \hat{s}\}$
5: **while** $|\hat{S}| \geq 3$ **do**
6: Let s_1 and s_2 be randomly selected elements of $\hat{S}$
7: $s' \leftarrow s_1 \cup s_2$
8: $S(l) \leftarrow S(l) \cup s'$
return $S(l)$.

Algorithm 2 creates a laminar family $\hat{l}$, with a full-binary tree representation, from an admissible laminar family l, whose tree representation is not a full-binary tree. For every set $\hat{s}$ in $S(l)$ that has at least 3 children,

we take two random children s_1 and s_2 (step 6), and we create a new set from the union of s_1 and s_2, which is added to $S(l)$ (steps 7 and 8). Note that this reduces the number of child sets of $\hat{s}$ by 1, since $s_1 \cup s_2$ is now a child of $\hat{s}$, while s_1 and s_2 are now children of $s_1 \cup s_2$.

4 Laminar family neighborhood characterization

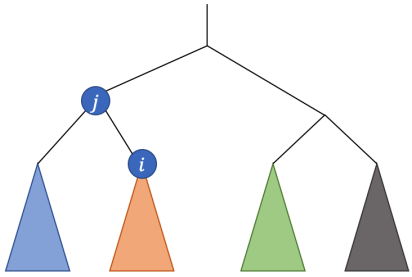
Our final goal is to propose a local search heuristic algorithm to obtain good solutions to the directed Steiner tree problem. This local search algorithm starts from a given laminar family, and then moves to a promising neighbor laminar family. This process is performed several times. Each laminar family subproblem is solved in polynomial time using the algorithm presented in Section 3.

Consequently, we need to have a characterization of the neighborhood of each laminar family. Recall that, all the laminar families considered in the proposed approach have a unique corresponding full binary tree representation. Therefore, it suffices to define the neighborhood of a laminar family, as the neighborhood of its corresponding tree representation. For the rest of this section, we define T to be a full binary tree, i.e., each node of T has a degree 3 or 1.

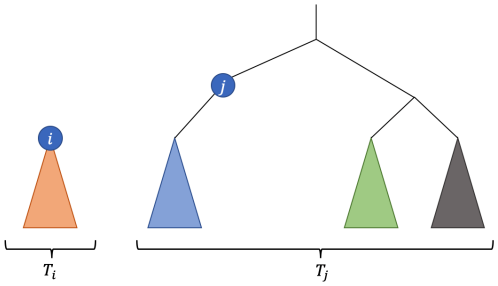
In the literature, there are several ways proposed to characterize the neighborhood of a tree. The most commonly used are the Nearest Neighbor Interchange (NNI), the Subtree Pruning and Regrafting (SPR), and the Tree Bisection Reconnection (TBR) [6, 15].

We decide to use the SPR process to construct the neighbors of a laminar family. In the studied literature, SPR was widely used over NNI and TBR. Furthermore, we consider the neighborhood size under SPR, which is $4(b-2)(b-3)$ [15], to be adequate for our case. Finally, we consider that SPR is reasonably easy to be adapted to our setting.

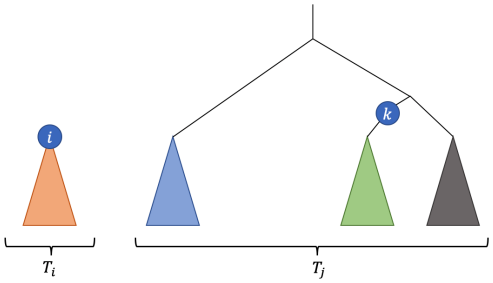
The SPR process corresponds to selecting an edge $\{i, j\}$ of tree T. The edge $\{i, j\}$ is removed from T, dividing T in two connected subtrees T_i and T_j, containing i and j respectively. For simplicity of the argument, suppose that T_i and T_j have more than 3 nodes. Note that nodes i and j will have degree 2, after removing edge $\{i, j\}$. We leave T_i as is, and we replace the two incident edges to j in T_j, by an edge connecting the two neighbors of j in T_j. Then, we select an edge of the updated T_j, and then we subdivide it creating a new node k. Finally, we create a new edge $\{i, k\}$ connecting node i of subtree T_i, and node k of subtree T_j.



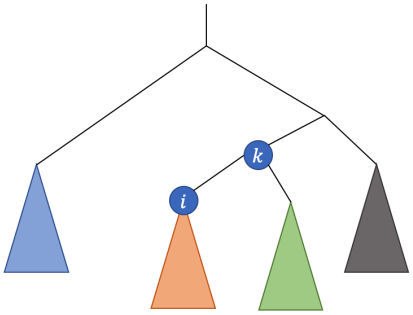
(a) Tree T.



(b) Subtrees T_i and T_j, after removing edge $\{i, j\}$.



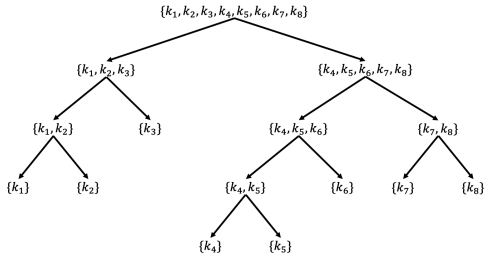
(c) Original subtree T_i, and new subtree T_j.



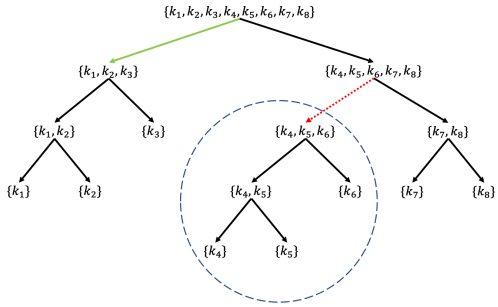
(d) Neighbor of tree T using SPR procedure.

Figure 4: A tree T and a neighbor under the SPR procedure.

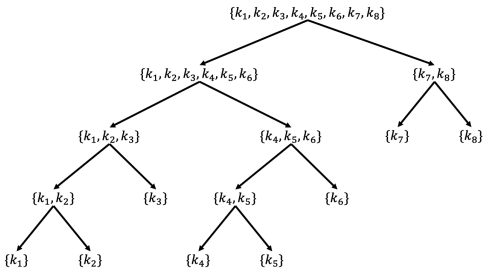
Figure 5 shows an example of the SPR process applied to our setting.



(a) Original tree T.



(b) The subtree within dashed circle is pasted in green edge. The red dashed edge is removed.



(c) SPR neighbor of tree T.

Figure 5: SPR neighbor of a laminar family.

In Figure 5a, we can see the original tree representation of a given laminar family. In Figure 5b, we can see the subtree which is going to be removed and then regrafted, which is highlighted with the blue dashed circle. The red dashed edge is going to be removed, and the green edge is where the subtree is going to be regrafted. Figure 5c shows the SPR neighbor, note that one set was removed $\{k_4, k_5, k_6, k_7, k_8\}$, and one was added $\{k_1, k_2, k_3, k_4, k_5, k_6\}$. This has to be done since the neighbor is the tree representation of a laminar family.

Note that we only need to make a few extra computations to have the optimal solution to the laminar family shown in Figure 5c. Since we are using the algorithm presented in Section 3, we only need to compute $z(i, \{k_1, \ldots, k_6\})$ for all $i \in V$, and recompute $z(r, K)$, since the child sets of $K = \{k_1, \ldots, k_8\}$ are different in the new laminar family. Everything else can be reused from the computations done to get an optimal solution of laminar family shown in Figure 5a.

5 Simulated annealing

Simulated annealing is a metaheuristic, which is widely used in local search frameworks [20, 32, 1, 4]. The main drawback when using local search algorithms, is that we often get stuck in local optima.

To address this in a minimization problem, simulated annealing also allows movements to neighbors with higher cost with a certain probability, which depends on how much worse the new the solution is, and the iteration of the algorithm. The high-level idea of simulated annealing is the following. We set an initial temperature T_0, and a starting initial solution. Then, at each iteration j, we decrease the temperature of the system having $T_j = f(T_0, j)$, where f is a function depending on T_0 and j. Also, at each iteration we randomly choose a solution in the neighborhood of the current solution, and we compute the cost difference between the current solution and the neighbor, denoted by Δ_j. If $\Delta_j < 0$ then we move to the neighbor solution. If $\Delta_j \geq 0$ we move to the neighbor solution with probability $p(\Delta_j, T_j)$. Note that the probability is a function of the difference in cost of the solutions and the current temperature of the system.

There are three main decisions to make when using simulated annealing. How we set the initial temperature T_0, which function we use to reduce the temperature of the system, and what probability function we use to accept increasing cost neighbors. We use the probability function given by

$$p(\Delta_j, T_j) = \left[1 + exp\left(\frac{\Delta_j}{T_j}\right)\right]^{-1} \in \left(0, \frac{1}{2}\right)$$

Note that higher Δ_j leads to lower $p(\Delta_j, T_j)$, and smaller T_j leads to lower acceptance probability.

Since we want Δ_j and T_j to be of the same order of magnitude, we use the cost of the first solution found as T_0. Finally, we use the temperature cooling function given by

$$T_j = f(T_0, j) = T_0(0.95)^j$$

The simulated annealing framework is given in Algorithm 3.

Algorithm 3 Simulated Annealing framework to solve directed Steiner tree problem

1: Set N_{iter}, $j = 1$.
2: $l_{current} \leftarrow initial\_laminar()$, $x_{current} \leftarrow solve\_DP(l_{current})$, $c_{current} \leftarrow c(x_{current})$, $T_0 \leftarrow c_{current}$
3: $l_{best} \leftarrow l_{current}$, $x_{best} \leftarrow x_{current}$, $c_{best} \leftarrow c_{current}$
4: **while** $j \leq N_{iter}$ **do**
5: Set $T_j = f(T_0, j)$
6: $l_{new} \leftarrow SPR(l_{current})$
7: $x_{new} \leftarrow solve\_DP(l_{new})$
8: $c_{new} \leftarrow c(x_{new})$
9: **if** $c_{new} < c_{current}$ **then**
10: $l_{current} \leftarrow l_{new}$
11: $x_{current} \leftarrow x_{new}$
12: $c_{current} \leftarrow c_{new}$
13: **if** $c_{new} < c_{best}$ **then**
14: $l_{best} \leftarrow l_{new}$
15: $x_{best} \leftarrow x_{new}$
16: $c_{best} \leftarrow c_{new}$
17: **else**
18: $\Delta_j = c_{new} - c_{current}$
19: Sample $u \sim U(0, 1)$
20: **if** $u \leq p(\Delta_j, T_j)$ **then**
21: $l_{current} \leftarrow l_{new}$
22: $x_{current} \leftarrow x_{new}$
23: $c_{current} \leftarrow c_{new}$
24: $j \leftarrow j + 1$
return x_{best}

The first 3 steps correspond to initialization of the algorithm. In the first step, we define the number of iterations N_{iter}, and set the iterations counter j to 1. In the second step, we create an initial laminar family using Algorithm 4, then we compute the best solution to the given laminar family, denoted by $x_{current}$, and the cost of the solution. We also define the initial temperature value. In the third step, we save the current solution as the best solution found so far.

Then, for each iteration j we, first, compute the temperature given by function $f(T_0, j)$, and after that, we create a random SPR neighbor of the current laminar family, and we compute the best solution of the neighbor and its cost (steps 6 to 8). We check whether the solution of the neighbor is better than the current solution, and if it is, we update the current solution. Moreover, if the neighbor solution is better than the best solution found so far, we update the best solution (steps 9 to 16). If the solution of the neighbor is worse than the current solution, we compute the cost difference between the solutions and we sample a uniform $(0, 1)$ random variable u. If the value of u is less than or equal to the acceptance probability given by $p(\Delta_j, T_j)$, then we update the current solution (steps 17 to 23). Finally, we increase the iterations counter by 1.

The process to construct the initial structure to solve is the following. First, we create a complete graph G' whose nodes are the terminal nodes. For every edge $e' = (t', t'')$, its cost is the length of the shortest path between t' and t'' in the original graph G. Second, we construct a minimum spanning tree T' in G'. Finally, we run Algorithm 4 to construct an initial laminar family using T' as input.

Algorithm 4 Algorithm *initial_laminar*(), which creates an initial laminar family

Require: Tree T', we assume T' is a set of edges ordered in increasing order of their lengths
1: Set $S(l) \leftarrow \{\{k_1\}, \{k_2\}, \ldots, \{k_b\}\}$
2: **for** $e' \in T'$ **do**
3: Let $e' = (t', t'')$, and let k' and k'' be the commodities of terminals t' and t'', respectively.
4: Let s' be largest set in $S(l)$ containing k'
5: Let s'' be largest set in $S(l)$ containing k''
6: $\hat{s} \leftarrow s' \cup s''$
7: $S(l) \leftarrow S(l) \cup \hat{s}$
return $S(l)$.

Algorithm 4 constructs a laminar family l based on the single linkage clustering algorithm, which is a widely used aglomerative clustering method [35]. It starts with the collection $S(l)$ of all the singletons. Then, it goes over the set of all edges of T', which are assumed to be ordered in increasing order of their lengths. For each edge $e' = (t', t'')$, we take the commodities k' and k'' of terminals t' and t'', respectively. Then, we take sets s' and s'', which are the largest sets in $S(l)$ that contain k' and k'', respectively. Finally, we create a set $\hat{s}$ to be the union of s' and s'', and we add it to $S(l)$. After we visit all edges, we have that $S(l)$ is a collection of sets that forms an admissible laminar family.

5.1 Simulated annealing with solution improvement

As previously pointed out, sometimes an optimal solution to a laminar family subproblem may not be a directed Steiner tree, which is why it may be beneficial to improve the solution obtained at every iteration of the simulated annealing algorithm. We use the solution improvement routine introduced in Section 3.3 in the simulated annealing framework, which is presented below.

Algorithm 5 Simulated Annealing for directed Steiner tree problem with solution improvement

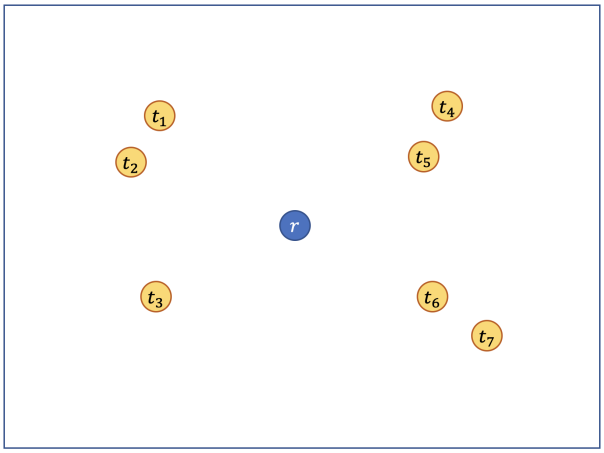
1: Set N_{iter}, $j = 1$.
2: $l_{current} \leftarrow random\_laminar()$, $x_{current} \leftarrow solve\_DP(l_{current})$, $c_{current} \leftarrow c(x_{current})$, $T_0 \leftarrow c_{current}$
3: $l_{best} \leftarrow l_{current}$, $x_{best} \leftarrow x_{current}$, $c_{best} \leftarrow c_{current}$
4: **while** $j \leq N_{iter}$ **do**
5: Set $T_j = f(T_0, j)$
6: $l_{new} \leftarrow SPR(l_{current})$
7: $x_{new} \leftarrow solve\_DP(l_{new})$
8: $c_{new} \leftarrow c(x_{new})$
9: **if** $A(l_{new})$ is not an r-arborescence **then**
10: $T \leftarrow MST(A(l_{new}))$
11: $l_{improved} \leftarrow LaminarFamily(T)$
12: **if** $l_{improved}$ is not full-binary tree **then**
13: $l_{improved} \leftarrow SampleFullBinaryTree(l_{improved})$
14: $l_{new} \leftarrow l_{improved}$
15: $x_{new} \leftarrow solve\_DP(l_{new})$
16: $c_{new} \leftarrow c(x_{new})$
17: **if** $c_{new} < c_{current}$ **then**
18: $l_{current} \leftarrow l_{new}$
19: $x_{current} \leftarrow x_{new}$
20: $c_{current} \leftarrow c_{new}$
21: **if** $c_{new} < c_{best}$ **then**
22: $l_{best} \leftarrow l_{new}$
23: $x_{best} \leftarrow x_{new}$
24: $c_{best} \leftarrow c_{new}$
25: **else**
26: $\Delta_j = c_{new} - c_{current}$
27: Sample $u \sim U(0, 1)$
28: **if** $u \leq p(\Delta_j, T_j)$ **then**
29: $l_{current} \leftarrow l_{new}$
30: $x_{current} \leftarrow x_{new}$
31: $c_{current} \leftarrow c_{new}$
32: $j \leftarrow j + 1$
return x_{best}

Algorithm 5 shows the pseudocode of simulated annealing with the solution improvement. The solution improvement, or tester, corresponds to lines 9 to 16. First, we check whether the solution of l_{new} is an r-arborescence or not, in case it is not, then it can be improved. In step 10, we define T to be the minimum cost r-arborescence whose leaves correspond to terminal nodes, as described above. In line 11, we define $l_{improved}$ to be the laminar family extracted from T. In line 12, we check whether $l_{improved}$ has a full binary tree representation or not. In case it is not full-binary, then we sample a full binary tree that contains $l_{improved}$. Finally, we update l_{new}, x_{new} and c_{new}.

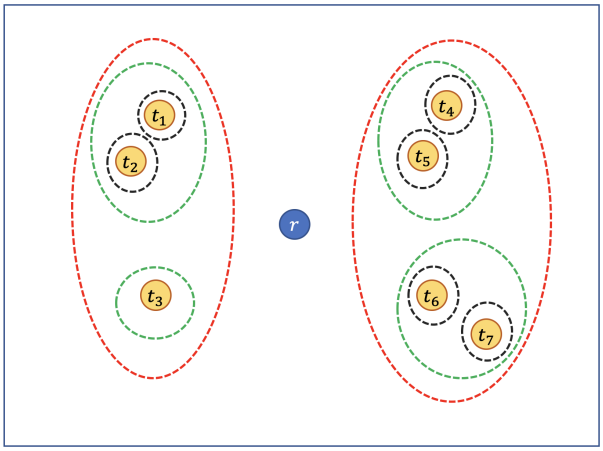
5.2 Rectilinear graphs

The nodes of rectilinear graphs are placed in the $\mathbb{R}^2$ plane and the distance between nodes is given by the $\|\cdot\|_1$ distance. We can take advantage of this property since we can divide the plane into regions, each one containing a set of terminals, which will be used to create an initial laminar family. For instance, in Figure 6a, we can see that terminals t_1 and t_7 may not share arcs in an optimal solution, but it is very likely that t_1 shares arcs with t_2, and t_7 with t_6, since they are in similar regions of the plane.

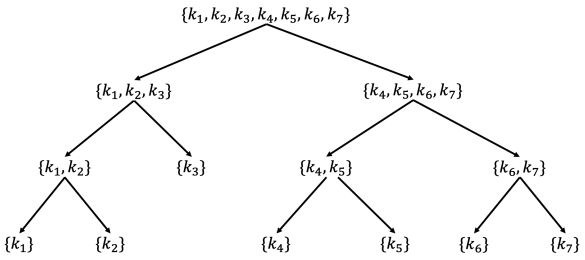
The idea is to first, divide the set of terminal nodes in two sets, which will be interpreted as the first bipartition for the laminar family we want to construct. Then, each set of commodities is divided into another two sets, and so on, until all sets are singletons. In particular, the first partition corresponds to the partition given by clusterizing all the terminals into two clusters. Then each cluster is clusterized again into another two clusters, and so on, until each cluster has size 1. To illustrate this idea see Figure 6. In Figure 6a, we only show the root node and the terminal nodes of a rectilinear graph. Figure 6b shows a potential partition based on the euclidean distance between terminals. In red dashed ovals, we find the first bipartition, inside each of the red zones, we find the second bipartition which is delimited by the green dashed oval, finally, inside each green zone with at least 2 terminals, we find a new partition denoted by the black dashed circles. Figure 6c shows the tree representation of the laminar family associated with the partition shown in Figure 6b.



(a) Rectilinear graph, only showing root and terminal nodes



(b) Example of a clusterization of terminals in rectilinear graph



(c) Tree representation of laminar family of clusterization

Figure 6: Example of terminals clusterization

There are many algorithms to cluster elements in the plane [18]. We decided to use a k-means algorithm [22] which is widely used in practice, because it is easy to implement, it runs very quickly, and it performs well in these types of clustering problems. Since the k-means output depends on the initial centroids, the quality of our constructed laminar family also depends on the initial centroids of each clusterization. This is why, we run the clustering-based algorithm several times to create laminar families, and then we select the laminar family with the best solution among all the candidates.

Algorithm 6 Algorithm $Part(s)$, which creates a bipartition of input set s, and all the created subsets

Require: Set s.
1: Define $\hat{S} = \{s\}$.
2: **if** $|s| = 1$ **then return** $\hat{S}$
3: **else if** $|s| = 2$ **then**
4: $\quad \hat{S} \leftarrow \hat{S} \cup \{\{e_1\}, \{e_2\}\}$, where $s = \{e_1, e_2\}$
5: **else**
6: $\quad (s_1, s_2) = kMeans(s, 2)$
7: $\quad \hat{S}_1 = Part(s_1)$, $\hat{S}_2 = Part(s_2)$
8: $\quad \hat{S} \leftarrow \hat{S} \cup \hat{S}_1 \cup \hat{S}_2$
return $\hat{S}$

Algorithm 6, named $Part(s)$, determines the way a new laminar family is created based on the clustering process previously described. This algorithm takes as input a given set s. If the set has only one element, then a collection of sets $\hat{S}$, containing the singleton set s, is returned as shown in line 4. If set s has exactly 2 elements, then the algorithm returns a collection of sets $\hat{S}$, containing set s and the two singletons, as shown in line 6. If set s has 3 or more elements, then s is split into two subsets s_1 and s_2, which is the result of running the k-means algorithm for $k = 2$, i.e., two clusters. Then, the algorithm recurses getting two collections of sets $\hat{S}_1$ and $\hat{S}_2$, which is the result of applying algorithm $Part(\cdot)$ to s_1 and s_2 respectively. The returned collection of sets $\hat{S}$, contains all sets in $\hat{S}_1$, all sets in $\hat{S}_2$, and the set s. Consequently, to get the desired laminar family, we have to run algorithm $Part(s)$, for $s = K$.

When the instance we want to solve is undirected, then we need to choose the root node among the terminal nodes. For all $i \in R$, let $L(i)$ and $R(i)$ be the number of terminal nodes to the left of i, and to the right of i, respectively. And, let $U(i)$ and $B(i)$ be the number of terminals that are above i, and below i in the plane, respectively. Finally, let $\Delta_x(i) = |R(i) - L(i)|$ and $\Delta_y(i) = |U(i) - B(i)|$, then we choose r as follows.

$$r = \underset{i \in R}{\text{Argmin}}\{\Delta_x(i) + \Delta_y(i)\}$$

We are basically choosing r to be the most centrical terminal node. We pick r in this fashion, since we can have a better guess of the structure of an optimal directed Steiner tree. For instance, in the example shown in Figure 6a, it is very likely that the paths from r of the terminals that are above and to the left of r, i.e., terminals t_1 and t_2, are not going to share many arcs with the paths from r to terminals that are below and to the right of r, i.e., terminals t_6 and t_7.

For rectilinear graphs, we use algorithm 6 to create the initial laminar family in the simulated annealing framework. As described in Algorithms 3 and 5, the initial laminar family is created at random. In rectilinear graphs, we run Algorithm 6 several times (the number of clusterizations is a parameter of the algorithm), and then we select a laminar family from the set of laminar families with the lowest optimal cost. The rest of the algorithm is the same as algorithm 3, or algorithm 5 if we use the solution improvement. Moreover, if the instance is undirected, we choose r as the most centrical terminal, as previously described.

5.3 Worst case analysis

Let OPT be the value of an optimal solution for the directed Steiner tree problem instance, let OPT_{SA} be the value of the solution returned by the simulated annealing framework, and for $l \in \mathcal{L}_b$, let $OPT(l)$ be the value of an optimal solution to $\mathcal{Z}_l$.

Lemma 1. *For any $l \in \mathcal{L}_b$ we have $OPT \leq |R| \times OPT(l)$*

Proof. Let l be an arbitrary laminar family of $\mathcal{L}_b$. Note that we can always construct the following feasible solution. For all $s \in S(l)$ with $|s| \geq 2$, we fix $f_a^s = 0$ for all $a \in A$, and for all $k \in K$, which correspond to

$s \in S(l)$ with $|s| = 1$, we take a shortest path from r to t_k. It is known that such solution is at most $|R|$ times the value of an optimal solution of the problem [31]. Consequently, any optimal solution to $\mathcal{Z}_l$ is at most $|R|$ times the value of an optimal solution to the problem. □

Proposition 1. *For every instance of the directed Steiner tree problem, we have* $OPT \leq |R| \times OPT_{SA}$

Proof. Since at every iteration we solve to optimality $\mathcal{Z}_l$ for some $l \in \mathcal{L}_b$, then using Lemma 1, we have that at every iteration, the best solution found cannot be more than $|R|$ times the value of an optimal solution to the problem, and the statement holds. □

Let $T(l)$ be the tree representation of laminar family $l \in \mathcal{L}_b$, and let $l^* \in \mathcal{L}_b$ be the laminar family of an optimal solution to the problem . We define $d_{SPR}(T(l_1), T(l_2))$ as the minimum number of SPR moves to transition from tree $T(l_1)$ to tree $T(l_2)$. It has been proven that computing $d_{SPR}(T(l_1), T(l_2))$ for any pair of l_1, l_2 is NP-Hard [5]. Nevertheless, it was proven in [30], that for any two $l_1, l_2 \in \mathcal{L}_b$, we have that $d_{SPR}(T(l_1), T(l_2)) \leq |R| - 2$. Consequently, if we use at least $|R| - 2$ iterations in the simulated annealing framework, no matter the initial laminar family chosen, the probability of solving $\mathcal{Z}_{l^*}$ in a given iteration, is strictly positive.

6 Computation experiments

In this section we present our computational results. We compare our proposed simulated annealing framework with all the algorithms studied in [33].

In [33], 6 algorithms are compared. The first algorithm, denoted by ShP_1, takes a shortest path from r to each terminal, and then returns the union of such shortest paths as a solution. The second algorithm, denoted by ShP_2, takes a shortest path from r to its closest terminal, sets the cost of all used arcs to 0, and then proceeds in the same fashion with the rest of the terminals, until all terminals are reached. The third algorithm, denoted by DuAs, corresponds to using the dual ascent algorithm presented in [34]. If the solution is fractional, then it takes a minimum cost spanning tree within the support of the fractional solution. If some of the leave nodes are non-terminal nodes, then the tree is pruned until all leaves are terminal nodes. The fourth algorithm, denoted by Roos, corresponds to the algorithm presented in [8], which has an approximation ratio $\mathcal{O}\left(|R|^{\frac{1}{t}}\right)$ using $t = 2$, and is implemented using Roos modified algorithm which is described in [23]. The fifth algorithm, denoted by FLAC, is one of the algorithms introduced by the authors in [33]. This algorithm takes each arc as a pipe with capacity, in liters, equal to its cost. Then the algorithm tries to send water to the terminals, at a rate of 1 liter of water per second. Initially, only the arcs incoming to terminals are going to be considered. When an arc is up to its capacity, it is said to be saturated. Once an arc is saturated, then we start looking at the arcs incoming to the tail node of the saturated arcs too. This process is done until we reach the root node. Then, within the support of the saturated arcs, we have a directed Steiner tree. The sixth algorithm, denoted by $\text{FLAC}^{\triangleright}$, is the FLAC algorithm applied to the shortest path instance of the problem, i.e., to a complete directed graph where the cost from u to v is given by the shortest path, in the original graph, between u and v. Finally, we define the Best Benchmark (BB) to be the algorithm that, for each instance, takes the best result among the previous 6 algorithms. The solutions provided by our algorithms will be compared with the solutions provided by BB.

On the other hand, we present 3 algorithms, SA, SA-Test and SA-Rect algorithm. The SA algorithm corresponds to Algorithm 3, SA-Test corresponds to Algorithm 5, and SA-Rect corresponds to Algorithm 5 but using Algorithm 6 to construct the initial laminar family. Since all of the proposed algorithms have a random component, we run 10 replications of each one, and the result of each algorithm corresponds to the solution with the lowest cost among the 10 replications. Furthermore, we run each algorithm with 1,000 and with 5,000 iterations, since the number of iterations is a parameter of the proposed algorithms. In the

SA-Rect case, we run 50 replications of Algorithm 6, and we select the solution with lowest cost as the initial laminar family of the simulated annealing algorithm.

We run experiments using the same instances studied in [33], which correspond to directed graphs constructed based on undirected instances from the SteinLib library [21]. We only consider instances with at most 160 terminal nodes, and less than 3,500 nodes, since the proposed approach requires the computation of the shortest path between every pair of nodes. In total, we studied just over 800 instances, whose details can be found in Table 5 of Appendix A.

The proposed algorithms were implemented in Java 8. All the experiments were run in an AWS *c5d.2xlarge* machine, with am *Intel Xeon Platinum 8000-series* processor (3.0 GHz) of 8 cores and 16 GB of memory.

6.1 Non-rectilinear graphs

In this section, we compare the solution quality of the regular simulated annealing algorithm (see Algorithm 3), the simulated annealing with solution improvement (see Algorithm 5), and the best benchmark algorithm. We only consider instances which are not rectilinear graphs, since there is a specific algorithm for such cases, which are analyzed in section 6.2. We use the performance profile approach to compare the solution quality delivered by the studied algorithms [11]. Figure 7 shows the cumulative distribution of instances versus the error of the obtained solution, this graph should be read as follows. If, for instance, we have a point $(1.5, 0.85)$ it means that 85% of the instances solved have a gap smaller or equal to 1.5%.

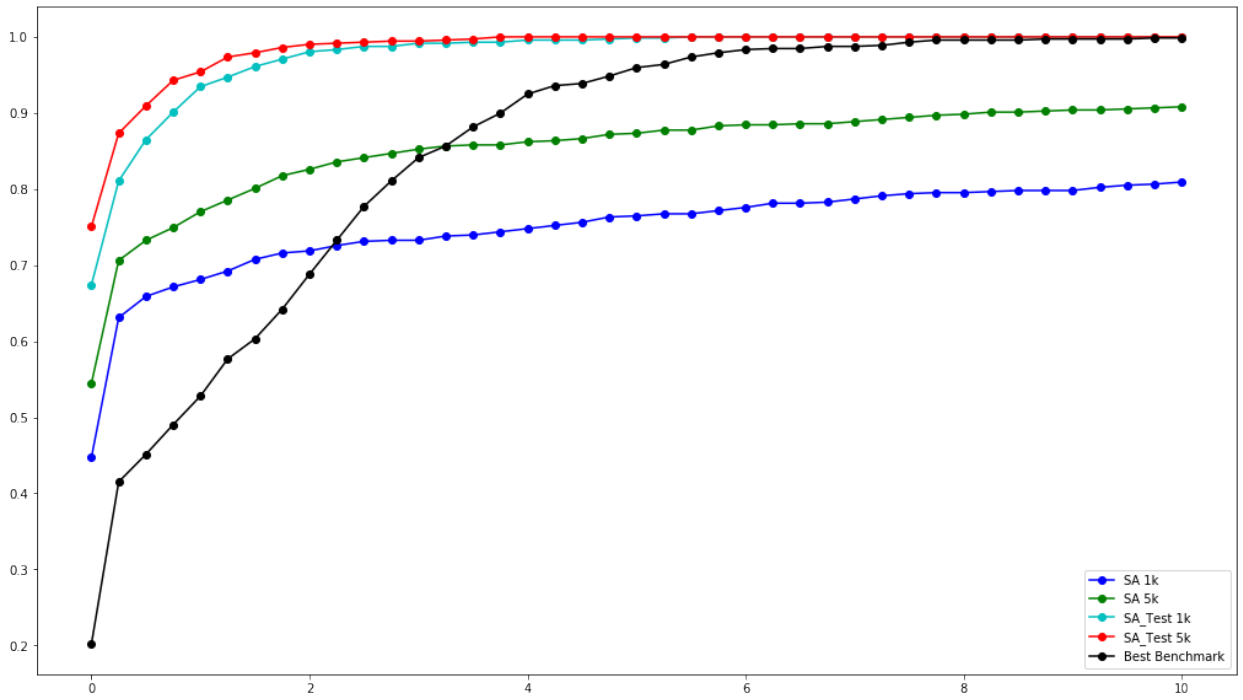


Figure 7: Performance profile for SA and SA-Test, for 1,000 and 5,000 iterations

It is clear from Figure 7 that SA-Test outperforms SA, and BB. First of all, we expected that, for the same algorithm, using more iterations will lead to better results. This happens with both, the SA and the SA-Test algorithm. The difference is more pronounced in the SA case; while in the SA-Test, the difference still exists, specially for smaller gaps, but after 2%, the two curves are almost identical. Also, note that

SA-Test algorithm with just 1,000 iterations, outperforms by far the BB algorithm, and the SA algorithm with 5,000 iterations. Another interesting observation is the difference in the proportion of instances solved to optimality. The BB algorithm solves around 20% of the instances to optimality, being the worst of all the algorithms studied in this topic. Furthermore, while the SA algorithm cannot solve more than 55% of the instances to optimality, we have that SA-Test can solve over 67% of the instances to optimality with 1,000 iterations, and 75% of the instances with 5,000.

Algorithm	Worse than SA 5k	Equal to SA 5k	Better than SA 5k
SA-Test 1k	4.7%	55.4%	39.8%
SA-Test 5k	1.8%	55.8%	42.3%

Table 1: Proportion of instances where SA-Test is strictly worse, equal, or strictly better than SA with 5,000 iterations

Table 1 shows the proportion of instances where SA-Test algorithm does strictly worse, equal, or strictly better than SA with 5,000 iterations. Even when we use 1,000 iterations, SA-Test only delivers worse results than SA in fewer than 5% of the cases studied; this number reduces to 1.8% when we use 5,000 iterations in SA-Test.

Algorithm	Worse than BB	Equal to BB	Better than BB
SA-Test 1k	3.6%	21.0%	75.3%
SA-Test 5k	2.2%	20.6%	77.2%

Table 2: Proportion of instances where SA-Test is strictly worse, equal, or strictly better than best benchmark

Table 2 show the proportion of instances where SA-Test algorithm does strictly worse, equal, or strictly better than BB. When we use 1,000 iterations, SA-Test only deliver worse results than BB in 3.6% of the cases studied, this number reduces to 2.2% when we use 5,000 iterations in SA-Test.

We conclude that SA-Test outperforms, in solution quality, the BB and SA algorithms. Although SA-Test with 5,000 iterations gives better performance than SA-Test with 1,000 iterations, the results are not considerably better.

6.2 Rectilinear graphs

In this section we consider rectilinear graphs. We compare the solution quality of the regular simulated annealing algorithm (see Algorithm 3), the simulated annealing for rectilinear graphs (see Algorithms 5 and 6), and the best benchmark algorithm. We compare the three algorithms in the same way we compared SA, SA-Test, and BB in section 6.1.

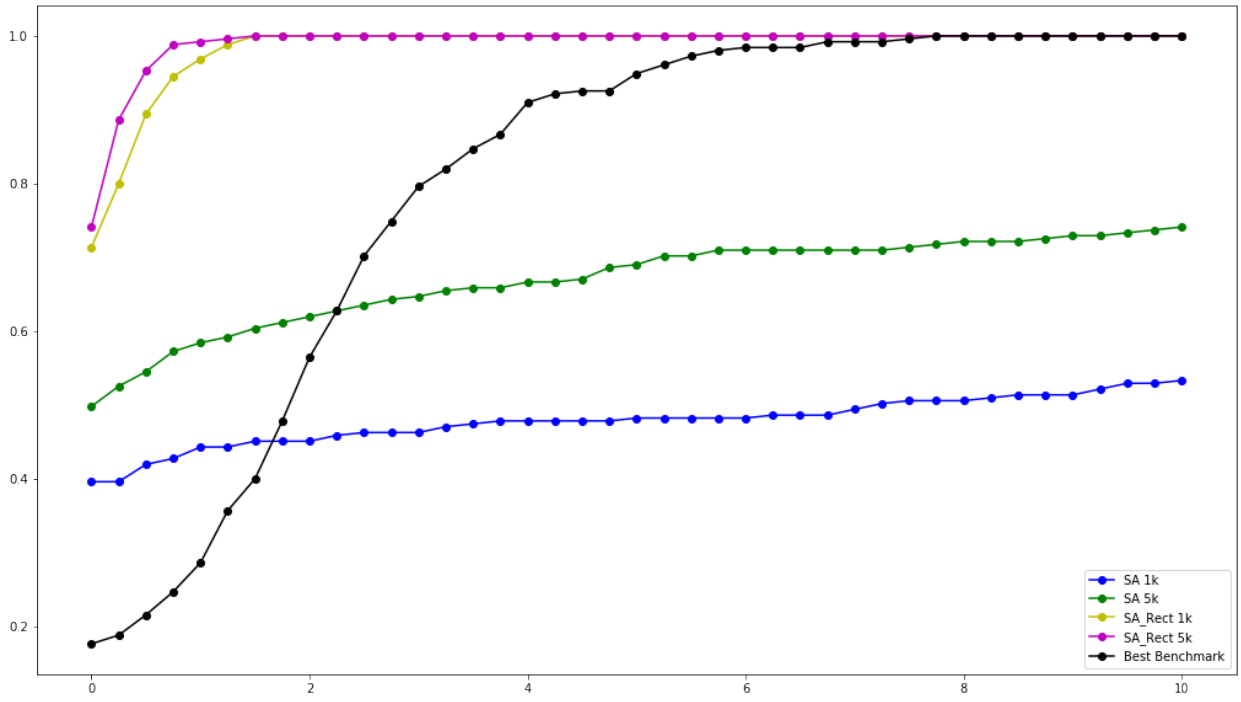


Figure 8: Performance profile for SA and SA-Rect, for 1,000 and 5,000 iterations

It is clear from Figure 8 that SA-Rect outperforms SA, and BB. The results are even more pronounced than the non-rectilinear graphs, since SA-Rect always delivers solutions with at most 1.75% gap. With respect to the SA and SA-Rect algorithms, again we see that by solving the problem with more iterations, the quality of the solutions improve. And again, the difference is more pronounced in the SA case. Indeed, for the SA-Rect the performance profiles are very similar, having a slightly better performance with 5,000 iterations.

In this case, we also have that SA-Rect algorithm with just 1,000 iterations, outperforms by far the BB algorithm, and the SA algorithm with 5,000 iterations. BB solves less than 18% of the instances to optimality, while SA solves almost 40% of instances to optimality with 1,000 iterations, and almost 50% with 5,000 iterations. In contrast, SA-Rect solves over 71% of instances to optimality with 1,000 iterations, and almost 75% with 5,000 iterations.

Algorithm	Worse than SA 5k	Equal to SA 5k	Better than SA 5k
SA-Rect 1k	1.2%	49.8%	49.0%
SA-Rect 5k	1.2%	50.2%	48.6%

Table 3: Proportion of instances where SA-Rect is strictly worse, equal, or strictly better than SA with 5,000 iterations

Table 3 shows the proportion of instances where SA-Rect algorithm does strictly worse, equal, or strictly better than SA with 5,000 iterations. We can see that when the number of iterations is 1,000, SA-Rect only delivers worse results than SA in 1.2% of the cases studied, the same amount when we use 5,000 iterations in SA-Rect.

Algorithm	Worse than BB	Equal to BB	Better than BB
SA-Rect 1k	0.4%	17.3%	82.3%
SA-Rect 5k	0.0%	17.6%	82.4%

Table 4: Proportion of instances where SA-Rect is strictly worse, equal, or strictly better than best benchmark

Table 4 shows the proportion of instances where SA-Rect algorithm does strictly worse, equal, or strictly better than BB. We can see that when we use 1,000 iterations, SA-Rect only delivers worse results than BB in 0.4% of the cases studied, which in this case corresponds to only one instance. When we use 5,000 iterations, SA-Rect performance is at least as well as BB in all the studied instances.

We conclude, that SA-Rect outperforms, in solution quality, the BB and SA algorithms. Although the conclusions in the rectilinear case are similar to the non-rectilinear case, the solution quality obtained by the improved version of SA in rectilinear graphs is better than the non-rectilinear case. All the instances in the rectilinear case present a gap smaller or equal to 1.75%, while in the non-rectilinear case it is 3.5% when we run SA-Test with 5,000 iterations, and 5% with 1,000 iterations. Moreover, the proportion of instances where SA-Rect performs worse than, either SA or BB, is lower than SA-Test.

6.3 Execution times

At each iteration of simulated annealing, we compute the new solution based on the solution of the previous iteration. Therefore, simulated annealing, by nature, is a sequential algorithm. There are some researchers that study parallel versions of the algorithm [16, 26, 10], but we just focused on the original version. Consequently, the running time of the algorithm will depend on the number of iterations.

Figure 9 shows the histogram of the average execution time per iteration for simulated annealing with, and without the solution improvement routine.

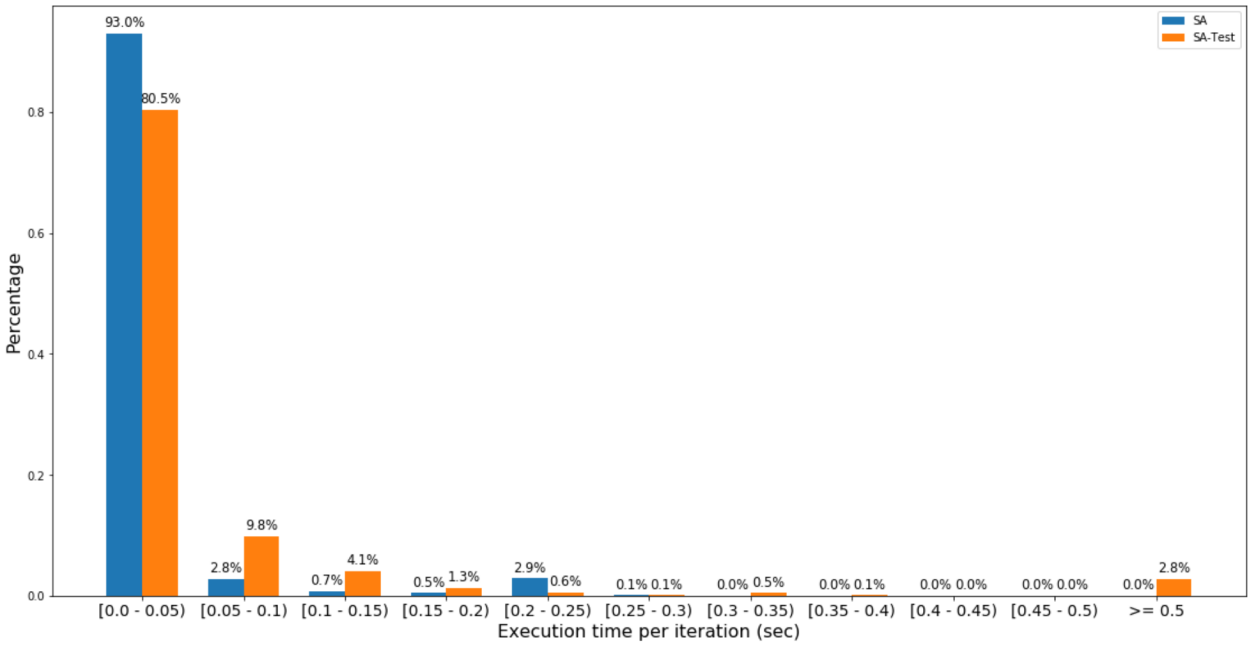


Figure 9: Execution time performance profile for SA with no solution improvement

As expected, the execution times when not using the solution improvement routine are lower, since there are fewer steps to complete at each iteration, but the distribution of the execution time when using solution improvement is not much worse. In both cases, the majority of the instances have an average execution time below 50 milliseconds.

The execution time per iteration, in both cases of the simulated annealing framework, are in the same order of magnitude as the execution times of all the studied algorithms in [33]. The main difference is that the total execution time of the simulated annealing algorithm is larger given the number of iterations to perform. In any case, the vast majority of the instances solve within seconds, or a few minutes, which makes this approach appealing to use given the better results in solution quality. Moreover, we can always use the solution given by the best benchmark algorithm, which takes a few milliseconds to solve, as the initial laminar family and start the simulated annealing framework from there.

7 Conclusions and future work

We developed a simulated annealing framework to solve the directed Steiner tree problem based on the approach proposed in [29]. We gave an efficient algorithm to solve the subproblem of each tree structure and then we used simulated annealing to find better solutions. We compared the proposed framework with the algorithms studied in [33], and we concluded that our approach outperforms these algorithms in solution quality.

Future research might be directed in applying the insights obtained in this paper to other solution techniques for the problem. For instance, we can use our algorithm to solve each laminar family subproblem to find better upper bounds in a Branch and Bound setting. In particular, we wonder if we can obtain a significant reduction in execution times when we use the Branch and Ascent approach proposed in [2]. At any node of the search tree, we can get candidate laminar families from the support of the fractional solution, and then solve the problem for that set of tree structures to compute better primal bounds.

Acknowledgement

This research was partially supported by Office on Naval Research grants N00014-15-1-2078 and N00014-18-1-2075 to the Georgia Institute of Technology, and by the CONICYT (Chilean National Commission for Scientific and Technological Research) through the Doctoral Fellowship program "Becas Chile", Grant No. 72160393

References

[1] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines," 1988.

[2] M. P. de Aragão, E. Uchoa, and R. F. Werneck, "Dual heuristics on the exact solution of large steiner problems," *Electronic Notes in Discrete Mathematics*, vol. 7, pp. 150–153, 2001.

[3] M. Bern and P. Plassmann, "The steiner problem with edge lengths 1 and 2," *Information Processing Letters*, vol. 32, no. 4, pp. 171–176, 1989.

[4] D. Bertsimas, J. Tsitsiklis, *et al.*, "Simulated annealing," *Statistical science*, vol. 8, no. 1, pp. 10–15, 1993.

[5] M. Bordewich and C. Semple, "On the computational complexity of the rooted subtree prune and regraft distance," *Annals of combinatorics*, vol. 8, no. 4, pp. 409–423, 2005.

[6] D. Bryant, "The splits in the neighborhood of a tree," *Annals of Combinatorics*, vol. 8, no. 1, pp. 1–11, 2004.

[7] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanità, "Steiner tree approximation via iterative randomized rounding," *Journal of the ACM (JACM)*, vol. 60, no. 1, p. 6, 2013.

[8] M. Charikar, C. Chekuri, T.-y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed steiner problems," *Journal of Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

[9] M. Chlebík and J. Chlebíková, "The steiner tree problem on graphs: Inapproximability results," *Theoretical Computer Science*, vol. 406, no. 3, pp. 207–214, 2008.

[10] Z. J. Czech and P. Czarnas, "Parallel simulated annealing for the vehicle routing problem with time windows," in *Proceedings 10th Euromicro workshop on parallel, distributed and network-based processing*, IEEE, 2002, pp. 376–383.

[11] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

[12] S. E. Dreyfus and R. A. Wagner, "The steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971.

[13] C. Duin, "Preprocessing the steiner problem in graphs," in *Advances in Steiner Trees*, Springer, 2000, pp. 175–233.

[14] C. Duin and S. Voβ, "Efficient path and vertex exchange in steiner tree algorithms," *Networks: An International Journal*, vol. 29, no. 2, pp. 89–105, 1997.

[15] J. Felenstein, *Inferring phylogenies*. Sinauer associates Sunderland, MA, 2004, vol. 2.

[16] D. R. Greening, "Parallel simulated annealing techniques," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 293–306, 1990.

[17] E. Halperin and R. Krauthgamer, "Polylogarithmic inapproximability," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, ACM, 2003, pp. 585–594.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[19] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, Springer, 1972, pp. 85–103.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[21] T. Koch, A. Martin, and S. Voß, "Steinlib: An updated library on steiner tree problems in graphs," in *Steiner trees in industry*, Springer, 2001, pp. 285–325.

[22] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[23] E. MING-IHsieh and M Tsai, "Fasterdsp: A faster approximation algorithm for directed steiner tree problem," *Journal of information science and Engineering*, vol. 22, pp. 1409–1425, 2006.

[24] T. Polzin and S. V. Daneshmand, "Extending reduction techniques for the steiner tree problem," in *European Symposium on Algorithms*, Springer, 2002, pp. 795–807.

[25] ——, "Improved algorithms for the steiner problem in networks," *Discrete Applied Mathematics*, vol. 112, no. 1-3, pp. 263–300, 2001.

[26] D. J. Ram, T. Sreenivas, and K. G. Subramaniam, "Parallel simulated annealing algorithms," *Journal of parallel and distributed computing*, vol. 37, no. 2, pp. 207–212, 1996.

[27] D. Rehfeldt, "A generic approach to solving the steiner tree problem and variants," 2015.

[28] G. Robins and A. Zelikovsky, "Tighter bounds for graph steiner tree approximation," *SIAM Journal on Discrete Mathematics*, vol. 19, no. 1, pp. 122–134, 2005.

[29] M. Siebert, S. Ahmed, and G. Nemhauser, "A linear programming based approach to the steiner tree problem with a fixed number of terminals," *Networks*, 2019. DOI: 10.1002/net.21913. [Online]. Available: https://doi.org/10.1002/net.21913.

[30] Y. S. Song, "On the combinatorics of rooted binary phylogenetic trees," *Annals of Combinatorics*, vol. 7, no. 3, pp. 365–379, 2003.

[31] H. Takahashi, "An approximate solution for the steiner problem in graphs," *Math. Japonica.*, vol. 6, pp. 573–577, 1990.

[32] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*, Springer, 1987, pp. 7–15.

[33] D. Watel and M.-A. Weisser, "A practical greedy approximation for the directed steiner tree problem," *Journal of Combinatorial Optimization*, vol. 32, no. 4, pp. 1327–1370, 2016.

[34] R. T. Wong, "A dual ascent approach for steiner tree problems on a directed graph," *Mathematical programming*, vol. 28, no. 3, pp. 271–287, 1984.

[35] R. Xu and D. C. Wunsch, "Survey of clustering algorithms," 2005.

[36] A. Z. Zelikovsky, "An 11/6-approximation algorithm for the network steiner problem," *Algorithmica*, vol. 9, no. 5, pp. 463–470, 1993.

Appendix A Simulated annealing instances

Table 5 contains the information about all the instances studied in Section 6. Table 5 contains the name of the instances, as well as the number of nodes, arcs and number of terminal nodes (excluding the root node). Column 'OPT' contains the value of an optimal solution to the instance. For instances whose optimal value is not known, we use the symbol '-'. Column 'BB' corresponds to the value of the best benchmark solution. Columns 'SA 1k' , 'SA-Test 1k' , and 'SA-Rect 1k' correspond to the values of the simulated annealing (SA), SA with solution improvement, and SA for rectilinear graphs using 1,000 iterations. Columns 'SA 5k' , 'SA-Test 5k' , and 'SA-Rect 5k' correspond to the values of the SA, SA with solution improvement, and SA for rectilinear graphs using 5,000 iterations. For not rectilinear instances, we use the symbol '-' in columns 'SA-Rect 1k' and 'SA-Rect 5k'.

Table 5: Studied instances

Instance	$\|V\|$	$\|A\|$	$\|R\|$	OPT	BB	SA 1k	SA-Test 1k	SA-Rect 1k	SA 5k	SA-Test 5k	SA-Rect 5k
wrp3-11	128	454	10	1,100,361	1,100,374	1,100,361	1,100,361	-	1,100,361	1,100,361	-
wrp3-12	84	298	11	1,200,237	1,200,237	1,200,237	1,200,237	-	1,200,237	1,200,237	-
wrp3-13	311	1226	12	1,300,497	1,300,524	1,300,497	1,300,497	-	1,300,497	1,300,497	-
wrp3-14	128	494	13	1,400,250	1,400,251	1,400,250	1,400,250	-	1,400,250	1,400,250	-
wrp3-15	138	514	14	1,500,422	1,500,422	1,500,422	1,500,422	-	1,500,422	1,500,422	-
wrp3-16	204	748	15	1,600,208	1,600,224	1,600,208	1,600,208	-	1,600,208	1,600,208	-
wrp3-17	177	708	16	1,700,442	1,700,443	1,700,442	1,700,442	-	1,700,442	1,700,442	-
wrp3-19	189	706	18	1,900,439	1,900,448	1,900,441	1,900,439	-	1,900,439	1,900,439	-
wrp3-20	245	908	19	2,000,271	2,000,302	2,000,271	2,000,271	-	2,000,271	2,000,271	-
wrp3-21	237	888	20	2,100,522	2,100,529	2,100,525	2,100,522	-	2,100,522	2,100,522	-
wrp3-22	233	862	21	2,200,557	2,200,575	2,200,567	2,200,557	-	2,200,557	2,200,557	-
wrp3-23	132	460	22	2,300,245	2,300,245	2,300,252	2,300,245	-	2,300,245	2,300,245	-
wrp3-24	262	974	23	2,400,623	2,400,630	2,400,637	2,400,623	-	2,400,623	2,400,623	-
wrp3-25	246	936	24	2,500,540	2,500,563	2,500,557	2,500,540	-	2,500,540	2,500,540	-
wrp3-26	402	1560	25	2,600,484	2,600,499	2,600,494	2,600,484	-	2,600,484	2,600,484	-
wrp3-27	370	1442	26	2,700,502	2,700,514	2,700,528	2,700,502	-	2,700,502	2,700,502	-
wrp3-28	307	1118	27	2,800,379	2,800,396	2,800,433	2,800,379	-	2,800,379	2,800,379	-
wrp3-29	245	872	28	2,900,479	2,900,510	2,900,502	2,900,479	-	2,900,479	2,900,479	-
wrp3-30	467	1792	29	3,000,569	3,000,594	3,000,642	3,000,569	-	3,000,572	3,000,569	-
wrp3-31	323	1184	30	3,100,635	3,100,641	3,100,643	3,100,635	-	3,100,635	3,100,635	-
wrp3-33	437	1676	32	3,300,513	3,300,518	3,300,560	3,300,513	-	3,300,513	3,300,513	-
wrp3-34	1244	4948	33	3,400,646	3,400,696	3,400,749	3,400,646	-	3,400,648	3,400,646	-
wrp3-36	435	1636	35	3,600,610	3,600,642	3,600,722	3,600,610	-	3,600,612	3,600,610	-
wrp3-37	1011	4020	36	3,700,485	3,700,517	3,700,571	3,700,485	-	3,700,492	3,700,485	-
wrp3-38	603	2414	37	3,800,656	3,800,676	3,800,804	3,800,656	-	3,800,661	3,800,656	-
wrp3-39	703	3232	38	3,900,450	3,900,469	3,900,512	3,900,450	-	3,900,451	3,900,450	-
wrp3-41	178	614	40	4,100,466	4,100,467	4,100,566	4,100,466	-	4,100,471	4,100,466	-
wrp3-42	705	2746	41	4,200,598	4,200,634	4,200,752	4,200,598	-	4,200,607	4,200,598	-
wrp3-43	173	596	42	4,300,457	4,300,459	4,300,523	4,300,457	-	4,300,457	4,300,457	-
wrp3-45	1414	5626	44	4,500,860	4,500,915	4,501,149	4,500,861	-	4,500,892	4,500,860	-
wrp3-48	925	3476	47	4,800,552	4,800,584	4,800,790	4,800,553	-	4,800,580	4,800,552	-
wrp3-49	886	3600	48	4,900,882	4,900,914	4,901,181	4,900,882	-	4,900,928	4,900,882	-
wrp3-50	1119	4502	49	5,000,673	5,000,719	5,000,961	5,000,673	-	5,000,718	5,000,673	-
wrp3-52	701	2704	51	5,200,825	5,200,873	5,201,116	5,200,830	-	5,200,865	5,200,825	-
wrp3-53	775	2942	52	5,300,847	5,300,899	5,301,338	5,300,850	-	5,300,888	5,300,854	-
wrp3-55	1645	6372	54	5,500,888	5,500,950	5,501,238	5,500,888	-	5,500,926	5,500,888	-
wrp3-56	853	3180	55	5,600,872	5,600,930	5,601,250	5,600,877	-	5,600,919	5,600,872	-
wrp3-60	838	3526	59	6,001,164	6,001,173	6,001,541	6,001,164	-	6,001,201	6,001,164	-
wrp3-62	670	2632	61	6,201,016	6,201,057	6,201,359	6,201,016	-	6,201,101	6,201,016	-
wrp3-64	1822	7220	63	6,400,931	6,400,985	6,401,558	6,400,948	-	6,401,017	6,400,931	-
wrp3-66	2521	9716	65	6,600,922	6,600,997	6,601,439	6,600,941	-	6,601,017	6,600,922	-
wrp3-67	987	3846	66	6,700,776	6,700,820	6,701,134	6,700,782	-	6,700,831	6,700,777	-
wrp3-69	856	3242	68	6,900,841	6,900,879	6,901,393	6,900,841	-	6,900,971	6,900,841	-
wrp3-70	1468	5862	69	7,000,890	7,000,956	7,001,397	7,000,893	-	7,001,052	7,000,890	-
wrp3-71	1221	4828	70	7,101,028	7,101,118	7,101,764	7,101,041	-	7,101,240	7,101,028	-
wrp3-73	1890	7226	72	7,301,207	7,301,269	7,301,780	7,301,212	-	7,301,405	7,301,207	-
wrp3-74	1019	3882	73	7,400,759	7,400,789	7,401,302	7,400,769	-	7,400,850	7,400,763	-
wrp3-75	729	2790	74	7,501,020	7,501,030	7,501,678	7,501,020	-	7,501,150	7,501,020	-
wrp3-76	1761	6740	75	7,601,028	7,601,067	7,602,077	7,601,032	-	7,601,253	7,601,028	-
wrp3-78	2346	9312	77	7,801,094	7,801,190	7,802,045	7,801,115	-	7,801,411	7,801,098	-
wrp3-79	833	3190	78	7,900,444	7,900,486	7,900,831	7,900,448	-	7,900,525	7,900,444	-
wrp3-80	1491	5662	79	8,000,849	8,000,918	8,001,550	8,000,852	-	8,001,046	8,000,849	-
wrp3-83	3168	12440	82	8,300,906	8,300,966	8,301,849	8,300,926	-	8,301,150	8,300,918	-
wrp3-84	2356	9094	83	8,401,094	8,401,165	8,401,790	8,401,096	-	8,401,385	8,401,094	-
wrp3-85	528	2034	84	8,500,739	8,500,793	8,501,556	8,500,750	-	8,500,880	8,500,749	-
wrp3-86	1360	5214	85	86,000,746	86,000,818	86,001,855	86,000,749	-	86,000,999	86,000,746	-
wrp3-88	743	2818	87	88,001,175	88,001,186	88,002,079	88,001,175	-	88,001,552	88,001,175	-
wrp3-91	1343	5188	90	91,000,866	91,000,940	91,001,448	91,000,886	-	91,001,149	91,000,869	-
wrp3-92	1765	7226	91	92,000,764	92,000,825	92,001,246	92,000,768	-	92,001,042	92,000,766	-
wrp3-94	1976	7672	93	94,001,181	94,001,220	94,002,005	94,001,197	-	94,001,541	94,001,183	-
wrp3-96	2518	9970	95	96,001,172	96,001,311	96,001,949	96,001,195	-	96,001,645	96,001,174	-
wrp3-98	2265	9090	97	98,001,224	98,001,311	98,001,888	98,001,233	-	98,001,754	98,001,230	-
wrp3-99	2076	8144	98	99,001,097	99,001,201	99,001,889	99,001,117	-	99,001,624	99,001,099	-
wrp4-11	123	466	10	1,100,179	1,100,192	1,100,179	1,100,179	-	1,100,179	1,100,179	-
wrp4-13	110	376	12	1,300,798	1,300,806	1,300,798	1,300,798	-	1,300,798	1,300,798	-
wrp4-14	145	566	13	1,400,290	1,400,291	1,400,290	1,400,290	-	1,400,290	1,400,290	-
wrp4-15	193	738	14	1,500,405	1,500,408	1,500,405	1,500,405	-	1,500,405	1,500,405	-
wrp4-16	311	1158	15	1,601,190	1,601,237	1,601,190	1,601,190	-	1,601,190	1,601,190	-
wrp4-17	223	808	16	1,700,525	1,700,541	1,700,525	1,700,525	-	1,700,525	1,700,525	-
wrp4-18	211	760	17	1,801,464	1,801,499	1,801,464	1,801,464	-	1,801,464	1,801,464	-
wrp4-19	119	412	18	1,901,446	1,901,461	1,901,446	1,901,446	-	1,901,446	1,901,446	-
wrp4-21	529	2064	20	2,103,283	2,103,345	2,103,284	2,103,283	-	2,103,283	2,103,283	-
wrp4-22	294	1136	21	2,200,394	2,200,403	2,200,411	2,200,394	-	2,200,394	2,200,394	-
wrp4-23	257	1030	22	2,300,376	2,300,416	2,300,410	2,300,376	-	2,300,376	2,300,376	-
wrp4-24	493	1926	23	2,403,332	2,403,428	2,403,473	2,403,332	-	2,403,332	2,403,332	-
wrp4-25	422	1616	24	2,500,828	2,500,838	2,500,868	2,500,828	-	2,500,828	2,500,828	-
wrp4-26	396	1562	25	2,600,443	2,600,459	2,600,469	2,600,443	-	2,600,443	2,600,443	-
wrp4-27	243	994	26	2,700,441	2,700,464	2,700,473	2,700,441	-	2,700,441	2,700,441	-

wrp4-28	272	1090	27	2,800,466	2,800,488	2,800,470	2,800,466	-	2,800,466	2,800,466	-
wrp4-29	247	1010	28	2,900,484	2,900,518	2,900,513	2,900,484	-	2,900,486	2,900,484	-
wrp4-30	361	1448	29	3,000,526	3,000,563	3,000,586	3,000,527	-	3,000,530	3,000,526	-
wrp4-31	390	1572	30	3,100,526	3,100,553	3,100,576	3,100,526	-	3,100,526	3,100,526	-
wrp4-32	311	1264	31	3,200,554	3,200,626	3,200,640	3,200,554	-	3,200,554	3,200,554	-
wrp4-33	304	1142	32	3,300,655	3,300,677	3,300,743	3,300,655	-	3,300,656	3,300,655	-
wrp4-34	314	1300	33	3,400,525	3,400,572	3,400,634	3,400,532	-	3,400,539	3,400,525	-
wrp4-35	471	1908	34	3,500,601	3,500,672	3,500,698	3,500,612	-	3,500,616	3,500,601	-
wrp4-36	363	1500	35	3,600,596	3,600,638	3,600,672	3,600,596	-	3,600,612	3,600,596	-
wrp4-37	522	2108	36	3,700,647	3,700,709	3,700,811	3,700,651	-	3,700,651	3,700,647	-
wrp4-38	294	1236	37	3,800,606	3,800,632	3,800,740	3,800,606	-	3,800,620	3,800,606	-
wrp4-39	802	3106	38	3,903,734	3,903,854	3,904,607	3,903,734	-	3,903,806	3,903,734	-
wrp4-40	538	2176	39	4,000,758	4,000,793	4,000,931	4,000,767	-	4,000,764	4,000,758	-
wrp4-41	465	1910	40	4,100,695	4,100,752	4,100,860	4,100,695	-	4,100,706	4,100,695	-
wrp4-42	552	2262	41	4,200,701	4,200,715	4,200,931	4,200,721	-	4,200,724	4,200,701	-
wrp4-43	596	2296	42	4,301,508	4,301,605	4,301,895	4,301,508	-	4,301,536	4,301,508	-
wrp4-44	398	1576	43	4,401,504	4,401,577	4,402,007	4,401,504	-	4,401,524	4,401,504	-
wrp4-45	388	1630	44	4,500,728	4,500,817	4,500,998	4,500,745	-	4,500,781	4,500,742	-
wrp4-46	632	2574	45	4,600,756	4,600,812	4,601,058	4,600,772	-	4,600,795	4,600,756	-
wrp4-47	555	2196	46	4,701,318	4,701,434	4,701,884	4,701,318	-	4,701,350	4,701,318	-
wrp4-48	451	1650	47	4,802,220	4,802,305	4,803,377	4,802,220	-	4,802,334	4,802,220	-
wrp4-49	557	2160	48	4,901,968	4,902,073	4,902,856	4,901,968	-	4,902,009	4,901,968	-
wrp4-50	564	2224	49	5,001,625	5,001,697	5,002,201	5,001,625	-	5,001,687	5,001,625	-
wrp4-51	668	2612	50	5,101,616	5,101,738	5,102,081	5,101,616	-	5,101,712	5,101,616	-
wrp4-52	547	2230	51	5,201,081	5,201,255	5,201,494	5,201,081	-	5,201,139	5,201,081	-
wrp4-53	615	2464	52	5,301,351	5,301,395	5,302,217	5,301,351	-	5,301,406	5,301,351	-
wrp4-54	688	2776	53	5,401,534	5,401,644	5,402,504	5,401,550	-	5,401,600	5,401,534	-
wrp4-55	610	2402	54	5,501,952	5,502,057	5,502,658	5,501,953	-	5,502,007	5,501,952	-
wrp4-56	839	3234	55	5,602,299	5,602,412	5,603,612	5,602,311	-	5,602,375	5,602,299	-
wrp4-58	757	2986	57	5,801,466	5,801,588	5,802,021	5,801,480	-	5,801,609	5,801,466	-
wrp4-59	904	3612	58	5,901,592	5,901,667	5,902,401	5,901,597	-	5,901,773	5,901,592	-
wrp4-60	693	2740	59	6,001,782	6,001,889	6,002,447	6,001,792	-	6,001,999	6,001,782	-
wrp4-61	775	3076	60	6,102,210	6,102,361	6,103,257	6,102,210	-	6,102,443	6,102,210	-
wrp4-62	1283	4986	61	6,202,100	6,202,231	6,203,213	6,202,101	-	6,202,375	6,202,100	-
wrp4-63	1121	4454	62	6,301,479	6,301,646	6,302,331	6,301,482	-	6,301,649	6,301,480	-
wrp4-64	632	2562	63	6,401,996	6,402,099	6,403,005	6,402,012	-	6,402,199	6,402,002	-
wrp4-66	844	3382	65	6,602,931	6,603,034	6,604,878	6,602,938	-	6,603,146	6,602,931	-
wrp4-67	1518	6120	66	6,702,800	6,702,917	6,704,241	6,702,813	-	6,703,156	6,702,803	-
wrp4-68	917	3700	67	6,801,753	6,801,846	6,802,614	6,801,758	-	6,802,011	6,801,758	-
wrp4-69	574	2330	68	6,902,328	6,902,452	6,903,483	6,902,381	-	6,902,735	6,902,342	-
wrp4-70	637	2538	69	7,003,022	7,003,140	7,004,996	7,003,034	-	7,003,615	7,003,034	-
wrp4-71	802	3218	70	7,102,320	7,102,452	7,104,656	7,102,327	-	7,102,613	7,102,320	-
wrp4-72	1151	4548	71	7,202,807	7,202,974	7,204,473	7,202,819	-	7,203,096	7,202,810	-
wrp4-73	1898	7232	72	7,302,643	7,302,838	7,303,630	7,302,671	-	7,302,926	7,302,656	-
wrp4-74	802	3240	73	7,402,046	7,402,147	7,403,189	7,402,050	-	7,402,483	7,402,046	-
wrp4-75	938	3738	74	7,501,712	7,501,871	7,503,025	7,501,720	-	7,502,056	7,501,713	-
wrp4-76	766	3070	75	7,602,040	7,602,179	7,602,701	7,602,090	-	7,602,639	7,602,042	-
es10fst01	18	40	9	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745	22,920,745
es10fst02	14	26	9	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104	19,134,104
es10fst03	17	40	9	26,003,678	26,496,603	26,003,678	26,003,678	26,003,678	26,003,678	26,003,678	26,003,678
es10fst04	18	40	9	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116	20,461,116
es10fst05	12	22	9	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916	18,818,916
es10fst06	17	40	9	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768	26,540,768
es10fst07	14	26	9	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072	26,025,072
es10fst08	21	56	9	25,056,214	25,488,037	25,056,214	25,056,214	25,056,214	25,056,214	25,056,214	25,056,214
es10fst09	21	58	9	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355	22,062,355
es10fst10	18	42	9	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095	23,936,095
es10fst11	14	26	9	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535	22,239,535
es10fst12	13	24	9	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318	19,626,318
es10fst13	18	42	9	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914	19,483,914
es10fst14	24	64	9	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128	21,856,128
es10fst15	16	36	9	18,641,924	19,045,863	18,641,924	18,641,924	18,641,924	18,641,924	18,641,924	18,641,924
es20fst01	29	56	19	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886	33,703,886
es20fst02	29	56	19	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486	32,639,486
es20fst03	27	52	19	27,847,417	27,847,417	27,961,830	27,847,417	27,847,417	27,847,417	27,847,417	27,847,417
es20fst04	57	166	19	27,624,394	28,089,782	27,624,394	27,624,394	27,624,394	27,625,414	27,624,394	27,624,394
es20fst05	54	154	19	34,033,163	34,134,855	34,134,855	34,033,163	34,033,163	34,033,163	34,033,163	34,033,163
es20fst06	29	56	19	36,014,241	36,014,241	36,271,878	36,014,241	36,014,241	36,014,241	36,014,241	36,014,241
es20fst07	45	118	19	34,934,874	35,468,385	34,934,874	34,934,874	34,934,874	34,934,874	34,934,874	34,934,874
es20fst08	52	148	19	38,016,346	38,607,292	38,824,337	38,016,346	38,016,346	38,016,346	38,016,346	38,016,346
es20fst09	36	84	19	36,739,939	37,254,007	36,739,939	36,739,939	36,739,939	36,739,939	36,739,939	36,739,939
es20fst10	49	134	19	34,024,740	34,693,096	34,484,566	34,509,767	34,389,405	34,261,290	34,389,405	34,389,405
es20fst11	33	72	19	27,123,908	27,123,908	27,344,510	27,123,908	27,123,908	27,123,908	27,123,908	27,123,908
es20fst12	33	72	19	30,451,397	30,451,397	31,564,941	30,451,397	30,451,397	30,451,397	30,451,397	30,451,397
es20fst13	35	80	19	34,438,673	34,596,753	35,517,810	34,438,673	34,438,673	34,438,673	34,438,673	34,438,673
es20fst14	36	88	19	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374	34,062,374
es20fst15	37	86	19	32,303,746	32,428,792	32,576,338	32,303,746	32,303,746	32,303,746	32,303,746	32,303,746
es30fst01	79	230	29	40,692,993	41,508,595	46,635,339	40,750,608	40,750,608	41,038,444	40,750,608	40,750,608
es30fst02	71	194	29	40,900,061	41,408,520	44,777,630	41,004,120	40,993,674	40,940,009	41,004,120	41,004,120
es30fst03	83	240	29	43,120,444	43,302,772	46,224,298	43,120,444	43,120,444	43,121,549	43,120,444	43,120,444
es30fst04	80	230	29	42,150,958	42,663,072	48,402,684	42,150,958	42,150,958	42,460,642	42,150,958	42,150,958
es30fst05	58	142	29	41,739,748	42,752,354	46,636,621	41,739,748	41,739,748	42,117,575	41,739,748	41,739,748
es30fst06	83	238	29	39,955,139	41,126,430	43,293,460	39,955,139	39,955,139	40,030,642	39,955,139	39,955,139
es30fst07	53	128	29	43,761,391	44,377,168	49,358,523	43,761,391	43,761,391	43,761,391	43,761,391	43,761,391
es30fst08	69	186	29	41,691,217	42,806,267	45,472,224	42,247,524	42,247,524	42,212,169	42,247,524	42,247,524
es30fst09	43	88	29	37,133,658	37,133,658	41,502,232	37,133,658	37,133,658	37,133,658	37,133,658	37,133,658
es30fst10	48	104	29	42,686,610	42,844,829	46,635,392	42,686,610	42,686,610	42,686,610	42,686,610	42,686,610
es30fst11	79	224	29	41,647,993	42,510,264	46,604,387	41,647,993	41,647,993	41,833,021	41,647,993	41,647,993
es30fst12	46	96	29	38,416,720	38,416,720	41,553,416	38,416,720	38,416,720	38,416,720	38,416,720	38,416,720
es30fst13	65	168	29	37,406,646	37,715,130	42,161,698	37,624,311	37,624,311	38,185,028	37,624,311	37,624,311
es30fst14	53	116	29	42,897,025	42,922,387	47,111,518	42,897,025	42,897,025	42,897,025	42,897,025	42,897,025
es30fst15	118	376	29	43,035,576	44,071,776	46,046,662	43,371,181	43,371,181	43,609,541	43,524,349	43,035,576
es40fst01	93	254	39	44,841,522	44,841,522	55,672,219	44,876,594	44,876,594	45,882,448	44,876,594	44,841,522
es40fst02	82	210	39	46,811,310	47,733,795	60,279,350	46,823,632	46,823,632	47,878,200	46,823,632	46,823,632
es40fst03	87	232	39	49,974,157	51,372,751	58,550,475	50,246,476	49,974,157	50,058,838	49,974,157	49,974,157
es40fst04	55	110	39	45,289,864	45,289,864	58,702,492	45,289,864	45,289,864	45,609,206	45,289,864	45,289,864
es40fst05	121	360	39	51,940,413	53,715,084	62,954,731	52,843,545	52,296,671	53,502,935	52,296,671	52,061,953
es40fst06	92	246	39	49,753,385	50,850,792	57,830,227	49,765,043	49,765,043	50,353,499	49,765,043	49,765,043
es40fst07	77	190	39	45,639,009	46,204,077	56,813,767	45,639,009	45,639,009	46,260,208	45,639,009	45,639,009
es40fst08	98	274	39	48,745,996	50,139,513	60,425,216	48,745,996	48,745,996	50,322,448	48,866,198	48,745,996
es40fst09	107	306	39	51,761,789	53,119,640	61,897,881	52,226,536	52,226,536	52,740,289	52,226,536	52,226,536
es40fst10	107	304	39	57,136,852	57,362,614	73,145,243	57,136,852	57,136,852	58,670,009	57,136,852	57,136,852
es40fst11	97	270	39	46,734,214	47,342,154	54,532,390	46,734,214	46,734,214	47,459,975	46,734,214	46,734,214
es40fst12	67	150	39	43,843,378	44,130,716	52,822,175	43,843,378	43,843,378	44,091,409	43,843,378	43,843,378
es40fst13	78	190	39	51,884,545	52,450,232	64,426,257	51,884,545	51,884,545	52,115,588	51,884,545	51,884,545
es40fst14	98	268	39	49,166,952	50,997,712	62,137,970	49,166,952	49,166,952	50,597,720	49,166,952	49,166,952
es40fst15	93	258	39	50,828,067	51,720,259	64,042,047	50,828,067	50,828,067	51,611,361	50,828,067	50,828,067

es50fst01	118	320	49	54,948,660	55,965,169	74,011,207	54,948,660	54,948,660	57,115,538	55,085,443	55,085,443
es50fst02	125	354	49	55,484,245	57,184,363	77,041,142	55,484,245	55,484,245	57,671,027	55,484,245	55,484,245
es50fst03	128	364	49	54,691,035	56,855,907	73,453,032	54,938,564	54,921,607	57,794,358	54,921,607	54,921,607
es50fst04	106	276	49	51,535,766	51,850,247	67,855,697	51,535,766	51,535,766	54,147,360	51,535,766	51,535,766
es50fst05	104	270	49	55,186,015	55,846,131	73,184,480	55,186,015	55,186,015	57,713,227	55,186,015	55,186,015
es50fst06	126	364	49	55,804,287	56,660,049	76,598,562	56,224,277	55,959,024	57,654,655	56,202,663	55,804,287
es50fst07	143	422	49	49,961,178	51,444,886	66,399,782	50,003,494	50,003,494	51,253,194	49,961,178	49,961,178
es50fst08	83	192	49	53,754,708	54,249,829	77,477,037	53,754,708	53,754,708	56,215,144	53,754,708	53,754,708
es50fst09	139	404	49	53,456,773	54,657,716	72,987,917	53,754,484	53,711,442	56,503,192	53,565,420	53,639,007
es50fst10	139	414	49	54,037,963	55,264,396	72,784,990	54,159,145	54,159,145	56,510,901	54,159,145	54,159,145
es50fst11	100	262	49	52,532,923	52,532,923	73,327,233	52,532,923	52,532,923	53,574,460	52,532,923	52,532,923
es50fst12	110	298	49	53,409,291	54,357,004	73,398,085	53,474,688	53,474,345	55,762,154	53,474,688	53,474,345
es50fst13	92	232	49	53,891,019	54,485,465	76,579,667	54,032,241	53,891,019	58,018,036	53,891,019	53,891,019
es50fst14	120	334	49	53,551,419	55,364,841	71,018,201	53,624,641	53,602,223	56,239,136	53,602,223	53,602,223
es50fst15	112	294	49	52,180,862	52,599,988	74,663,147	52,180,862	52,180,862	54,568,774	52,451,296	52,180,862
es60fst01	123	318	59	53,761,423	54,755,587	81,863,765	53,761,423	53,761,423	56,416,838	53,761,423	53,761,423
es60fst02	186	560	59	55,367,804	57,498,851	85,483,381	55,782,575	55,511,788	60,693,584	55,652,489	55,434,945
es60fst03	113	284	59	56,566,797	57,540,673	86,790,698	56,672,299	56,566,797	62,917,304	56,672,299	56,672,299
es60fst04	162	476	59	55,371,042	56,741,428	82,273,812	55,512,897	55,421,710	61,521,694	55,421,710	55,421,710
es60fst05	119	296	59	54,704,991	55,944,103	84,177,043	54,704,991	54,704,991	59,524,029	54,704,991	54,704,991
es60fst06	130	348	59	60,421,961	61,506,301	92,979,217	60,495,383	60,421,961	68,196,539	60,495,383	60,495,383
es60fst07	188	560	59	58,978,041	60,653,649	87,830,332	58,978,041	59,071,187	66,954,247	59,071,187	59,071,187
es60fst08	109	266	59	58,138,178	59,024,671	79,628,946	58,159,255	58,138,178	65,783,731	58,138,178	58,138,178
es60fst09	151	432	59	55,877,112	57,365,065	83,691,100	56,054,006	56,194,566	62,668,746	55,877,112	55,877,112
es60fst10	133	354	59	57,624,488	58,794,018	88,464,280	57,624,488	57,624,488	63,588,122	57,697,771	57,697,771
es60fst11	121	308	59	56,141,666	57,006,563	82,628,025	56,724,002	56,478,202	60,986,525	56,686,467	56,141,803
es60fst12	176	514	59	59,791,362	61,381,577	91,591,210	59,791,362	59,791,362	64,465,095	59,791,362	59,791,362
es60fst13	157	452	59	61,213,533	63,635,933	85,257,041	61,715,012	61,561,409	69,717,005	61,568,573	61,561,409
es60fst14	118	298	59	56,035,528	56,555,699	84,708,470	56,243,459	56,238,070	61,833,829	56,035,528	56,035,528
es60fst15	117	302	59	56,622,581	57,679,180	95,323,644	56,622,581	56,622,581	62,247,204	56,622,581	56,622,581
es70fst01	154	418	69	62,058,863	63,014,190	101,837,680	62,058,863	62,058,863	72,377,860	62,058,863	62,058,863
es70fst02	147	394	69	60,928,488	62,363,798	96,797,636	60,928,488	60,928,488	69,527,624	60,928,488	60,928,488
es70fst03	181	528	69	61,934,664	63,426,170	92,429,399	62,003,547	62,003,547	70,603,398	62,003,547	62,003,547
es70fst04	167	462	69	62,938,583	64,980,889	112,090,729	62,938,583	62,938,583	72,812,948	62,938,583	62,938,583
es70fst05	169	462	69	62,256,993	63,734,181	96,258,098	62,565,241	62,565,241	70,889,824	62,476,165	62,476,165
es70fst06	187	536	69	62,124,528	63,098,455	101,661,617	62,124,528	62,124,528	75,675,756	62,124,528	62,124,528
es70fst07	167	460	69	62,223,666	63,468,895	100,242,277	62,687,106	62,486,945	71,904,773	62,538,263	62,223,666
es70fst08	209	628	69	61,872,849	63,960,318	94,965,770	62,298,465	62,349,496	70,595,710	61,983,289	61,983,289
es70fst09	161	440	69	62,986,133	63,703,624	92,605,950	63,293,977	62,986,133	72,579,562	63,251,040	63,251,040
es70fst10	165	450	69	62,511,830	64,034,461	106,167,511	62,677,619	62,512,257	72,762,985	62,511,830	62,511,830
es70fst11	177	508	69	66,455,760	67,194,990	101,807,921	66,786,476	66,818,973	75,215,186	66,455,760	66,455,760
es70fst12	142	362	69	63,047,132	63,636,016	102,637,913	63,047,132	63,047,132	72,411,546	63,047,132	63,047,132
es70fst13	160	438	69	62,912,258	64,737,500	102,759,829	63,140,852	63,140,852	74,075,031	62,912,258	62,912,258
es70fst14	143	368	69	60,411,124	61,562,929	100,069,731	60,584,213	60,488,022	68,957,369	60,488,022	60,444,492
es70fst15	178	502	69	62,318,458	63,452,839	103,263,238	62,573,437	62,717,697	71,622,372	62,462,718	62,573,437
es80fst01	187	510	79	70,927,442	72,132,147	116,409,767	71,083,285	71,162,601	84,636,408	71,111,210	71,050,741
es80fst02	183	498	79	65,273,810	67,090,295	111,842,465	65,324,609	65,427,854	79,937,663	65,273,810	65,273,810
es80fst03	189	522	79	65,332,546	66,721,458	119,314,620	65,342,854	65,332,546	80,973,001	65,421,298	65,332,546
es80fst04	198	560	79	64,193,446	65,634,556	103,922,381	64,494,783	64,494,783	81,067,119	64,228,815	64,228,815
es80fst05	172	456	79	66,350,529	67,182,043	113,922,379	66,418,239	66,361,552	82,130,710	66,361,552	66,361,552
es80fst06	172	448	79	71,007,444	71,976,669	120,513,807	71,384,971	71,247,118	88,484,489	71,206,127	71,206,127
es80fst07	193	542	79	68,228,475	70,631,658	113,247,391	69,243,794	69,141,321	79,031,047	68,700,773	68,618,343
es80fst08	217	612	79	67,452,377	69,093,982	105,770,374	67,603,229	67,603,229	80,483,235	67,552,401	67,495,599
es80fst09	236	686	79	69,825,651	73,799,332	115,098,751	70,209,428	70,417,175	83,921,337	69,968,010	69,968,010
es80fst10	156	394	79	65,497,988	66,048,297	112,801,043	65,593,502	65,558,842	83,456,794	65,558,842	65,558,842
es80fst11	209	590	79	66,283,099	68,336,640	117,623,500	66,625,179	66,596,846	81,419,928	66,472,435	66,472,435
es80fst12	147	360	79	65,070,089	65,811,851	120,873,101	65,070,089	65,070,089	79,206,221	65,192,803	65,192,803
es80fst13	164	422	79	68,022,647	68,870,098	122,958,169	68,022,647	68,022,647	86,542,708	68,022,647	68,022,647
es80fst14	209	594	79	70,077,902	71,710,883	118,723,015	70,416,450	70,416,450	86,779,194	70,197,701	70,416,450
es80fst15	197	564	79	69,939,071	72,673,765	118,785,339	70,090,765	69,939,071	83,092,180	70,090,765	69,939,071
es90fst01	181	462	89	68,350,357	69,340,636	150,931,602	68,350,357	68,350,357	91,635,890	68,350,357	68,350,357
es90fst02	221	626	89	71,294,845	72,831,603	126,432,017	71,425,203	71,694,120	93,178,255	71,694,120	71,294,845
es90fst03	284	860	89	74,817,473	77,047,659	126,077,645	75,105,945	75,063,090	90,528,230	74,852,873	75,063,090
es90fst04	217	598	89	70,910,063	72,157,062	122,983,174	70,917,414	71,024,704	89,896,156	70,910,063	70,910,063
es90fst05	190	508	89	71,831,224	73,172,031	139,882,707	71,831,224	71,831,224	93,017,653	71,831,224	71,831,224
es90fst06	215	580	89	68,640,346	69,723,973	135,734,285	68,640,346	68,640,346	89,951,662	68,640,346	68,739,214
es90fst07	175	442	89	72,036,885	72,605,732	131,262,102	72,250,976	72,327,131	89,595,537	72,110,874	72,110,874
es90fst08	234	664	89	72,341,668	73,986,750	127,623,103	72,572,503	72,483,361	87,034,353	72,572,503	72,572,503
es90fst09	234	662	89	67,856,007	71,215,888	126,590,077	68,347,576	68,255,241	88,576,349	67,901,571	68,248,494
es90fst10	246	712	89	72,310,409	74,583,122	123,328,360	72,718,116	72,627,121	91,590,148	72,781,214	72,603,106
es90fst11	225	646	89	72,310,039	73,459,010	131,228,732	72,514,834	72,539,712	87,630,890	72,390,983	72,390,983
es90fst12	207	568	89	69,367,257	71,471,301	113,416,866	69,629,971	69,664,913	85,874,569	69,539,644	69,539,644
es90fst13	240	698	89	72,810,663	74,166,435	118,232,381	73,168,302	73,061,207	92,251,455	73,008,303	72,959,602
es90fst14	185	486	89	69,188,992	70,262,339	138,627,523	69,496,894	69,465,880	93,278,634	69,663,555	69,645,014
es90fst15	207	572	89	71,778,294	73,826,179	125,045,782	72,211,375	71,964,639	93,853,615	71,802,943	71,802,943
es100fst01	250	708	99	72,522,165	74,996,386	113,646,598	73,162,848	73,121,878	96,737,605	72,833,811	72,857,813
es100fst02	339	1044	99	75,176,630	76,681,625	131,323,543	75,815,432	75,569,313	99,417,305	75,340,116	75,567,309
es100fst03	189	466	99	72,746,006	73,849,547	123,680,439	72,902,717	73,187,062	100,744,772	72,902,717	72,902,717
es100fst04	188	470	99	74,342,392	75,693,600	161,542,475	74,693,569	74,579,736	101,101,314	74,579,736	74,579,736
es100fst05	188	476	99	75,670,198	76,897,439	152,269,080	75,938,104	75,846,518	105,331,568	75,932,463	75,932,463
es100fst06	301	904	99	74,414,990	76,297,034	131,514,016	74,873,914	74,774,210	98,133,810	74,488,218	74,788,218
es100fst07	276	802	99	77,740,576	80,265,546	137,610,940	78,404,419	78,617,102	101,561,113	78,521,419	78,318,231
es100fst08	210	552	99	73,033,178	74,839,270	121,805,114	73,525,091	73,519,753	92,789,504	73,519,753	73,519,753
es100fst09	248	684	99	77,952,027	80,059,348	137,043,274	79,275,897	79,014,040	101,586,959	78,836,698	78,865,289
es100fst10	229	624	99	75,952,202	77,683,619	129,303,378	76,702,908	76,634,779	101,901,491	76,486,096	76,375,025
es100fst11	253	724	99	78,674,859	80,283,998	136,353,831	78,968,098	78,848,637	102,233,108	78,848,637	78,848,637
es100fst12	266	770	99	76,131,099	78,387,376	153,396,885	76,564,015	76,689,893	103,218,548	76,301,181	76,282,932
es100fst13	254	722	99	74,604,990	76,822,679	152,766,695	75,196,104	74,844,199	99,738,847	74,988,766	74,858,345
es100fst14	198	506	99	78,632,795	79,774,666	149,842,856	78,632,795	78,632,795	108,976,665	78,632,795	78,632,795
es100fst15	231	638	99	70,446,493	72,805,295	127,537,710	70,798,205	70,699,648	93,858,665	70,652,858	70,573,355
diw0250	353	1216	10	350	355	350	350	350	350	350	350
diw0260	539	1970	11	468	468	468	468	468	468	468	468
diw0313	468	1644	13	397	397	397	397	397	397	397	397
diw0393	212	762	10	302	307	302	302	302	302	302	302
diw0445	1804	6622	32	1,363	1,388	1,445	1,363	1,363	1,370	1,363	1,363
diw0459	3636	13578	24	1,362	1,377	1,382	1,362	1,362	1,366	1,362	1,362
diw0460	339	1158	12	345	345	345	345	345	345	345	345
diw0473	2213	8270	24	1,098	1,110	1,105	1,098	1,098	1,098	1,098	1,098
diw0487	2414	8772	24	1,424	1,466	1,455	1,424	1,424	1,425	1,426	1,424
diw0495	938	3310	9	616	626	616	616	616	616	616	616
diw0513	918	3368	9	604	618	604	604	604	604	604	604
diw0523	1080	4030	9	561	561	561	561	561	561	561	561
diw0540	286	930	9	374	374	374	374	374	374	374	374
diw0559	3738	14026	17	1,570	1,613	1,570	1,570	1,570	1,570	1,570	1,570
diw0795	3221	11876	9	1,550	1,630	1,550	1,550	1,550	1,550	1,550	1,550
diw0801	3023	11150	9	1,587	1,609	1,587	1,587	1,587	1,587	1,587	1,587
dmxa0296	233	772	11	344	351	344	344	344	344	344	344

dmxa0296	233	772	11	344	351	344	344	344	344	344	344
dmxa0368	2050	7352	17	1,017	1,036	1,017	1,017	1,017	1,017	1,017	1,017
dmxa0454	1848	6572	15	914	958	914	914	914	914	914	914
dmxa0628	169	560	9	275	295	275	275	275	275	275	275
dmxa0734	663	2308	10	506	506	506	506	506	506	506	506
dmxa0848	499	1722	15	594	602	599	594	594	594	594	594
dmxa0903	632	2174	9	580	603	580	580	580	580	580	580
dmxa1109	343	1118	16	454	465	454	454	454	454	454	454
dmxa1200	770	2766	20	750	766	752	750	750	750	750	750
dmxa1304	298	1006	9	311	327	311	311	311	311	311	311
dmxa1516	720	2538	10	508	533	508	508	508	508	508	508
dmxa1721	1005	3462	17	780	789	780	780	780	780	780	780
dmxa1801	2333	8274	16	1,365	1,420	1,365	1,365	1,365	1,365	1,365	1,365
gap1307	342	1104	16	549	559	549	549	549	549	549	549
gap1413	541	1812	9	457	470	457	457	457	457	457	457
gap1500	220	748	16	254	254	254	254	254	254	254	254
gap1810	429	1404	16	482	487	482	482	482	482	482	482
gap1904	735	2512	20	763	778	765	763	763	763	763	763
gap2007	2039	7096	16	1,104	1,140	1,107	1,104	1,104	1,104	1,104	1,104
gap2119	1724	5950	28	1,244	1,307	1,336	1,244	1,244	1,244	1,244	1,244
gap2740	1196	4168	13	745	745	745	745	745	745	745	745
gap2800	386	1306	11	386	386	386	386	386	386	386	386
gap2975	179	586	9	245	245	245	245	245	245	245	245
gap3036	346	1166	12	457	473	457	457	457	457	457	457
gap3100	921	3116	10	640	682	640	640	640	640	640	640
msm0580	338	1082	10	467	490	467	467	467	467	467	467
msm0654	1290	4540	9	823	838	823	823	823	823	823	823
msm0709	1442	4806	15	884	884	884	884	884	884	884	884
msm0920	752	2528	25	806	826	834	806	806	806	806	806
msm1008	402	1390	10	494	518	494	494	494	494	494	494
msm1234	933	3264	12	550	563	550	550	550	550	550	550
msm1477	1199	4156	30	1,068	1,109	1,197	1,068	1,068	1,073	1,068	1,068
msm1707	278	956	10	564	574	564	564	564	564	564	564
msm1844	90	270	9	188	193	188	188	188	188	188	188
msm1931	875	3044	9	604	604	604	604	604	604	604	604
msm2000	898	3124	9	594	599	594	594	594	594	594	594
msm2152	2132	7404	36	1,590	1,649	1,782	1,590	1,590	1,604	1,590	1,590
msm2326	418	1446	13	399	409	399	399	399	399	399	399
msm2525	3031	10478	11	1,290	1,298	1,290	1,290	1,290	1,290	1,290	1,290
msm2601	2961	10200	15	1,440	1,458	1,440	1,440	1,440	1,440	1,440	1,440
msm2705	1359	4916	12	714	735	714	714	714	714	714	714
msm2802	1709	5926	17	926	951	926	926	926	926	926	926
msm2846	3263	11566	88	3,135	3,252	5,303	3,166	3,181	3,839	3,141	3,143
msm3277	1704	5982	11	869	869	869	869	869	869	869	869
msm3676	957	3108	9	607	607	607	607	607	607	607	607
msm4038	237	780	10	353	367	353	353	353	353	353	353
msm4114	402	1380	15	393	403	393	393	393	393	393	393
msm4190	391	1332	15	381	381	381	381	381	381	381	381
msm4224	191	604	10	311	315	311	311	311	311	311	311
msm4414	317	952	10	408	408	408	408	408	408	408	408
msm4515	777	2716	12	630	660	630	630	630	630	630	630
taq0023	572	1926	10	621	635	621	621	621	621	621	621
taq0431	1128	3810	12	897	907	897	897	897	897	897	897
taq0631	609	1864	9	581	591	581	581	581	581	581	581
taq0739	837	2876	15	848	877	848	848	848	848	848	848
taq0741	712	2434	15	847	897	847	847	847	847	847	847
taq0751	1051	3582	15	939	980	939	939	939	939	939	939
taq0891	331	1120	9	319	321	319	319	319	319	319	319
taq0910	310	1028	16	370	370	370	370	370	370	370	370
taq0920	122	388	16	210	210	210	210	210	210	210	210
taq0978	777	2478	9	566	571	566	566	566	566	566	566
b01	50	126	8	82	82	82	82	-	82	82	-
b02	50	126	12	83	83	83	83	-	83	83	-
b03	50	126	24	138	138	140	138	-	138	138	-
b04	50	200	8	59	59	59	59	-	59	59	-
b05	50	200	12	61	62	61	61	-	61	61	-
b06	50	200	24	122	123	125	122	-	122	122	-
b07	75	188	12	111	111	111	111	-	111	111	-
b08	75	188	18	104	104	104	104	-	104	104	-
b09	75	188	37	220	220	241	220	-	221	220	-
b10	75	300	12	86	90	86	86	-	86	86	-
b11	75	300	18	88	90	88	88	-	88	88	-
b12	75	300	37	174	174	194	174	-	174	174	-
b13	100	250	16	165	177	165	165	-	165	165	-
b14	100	250	24	235	236	235	235	-	235	235	-
b15	100	250	49	318	318	393	318	-	327	318	-
b16	100	400	16	127	133	127	127	-	127	127	-
b17	100	400	24	131	132	133	131	-	131	131	-
b18	100	400	49	218	222	263	218	-	222	218	-
i080-001	80	240	5	1,787	1,787	1,787	1,787	-	1,787	1,787	-
i080-002	80	240	5	1,607	1,607	1,607	1,607	-	1,607	1,607	-
i080-003	80	240	5	1,713	1,713	1,713	1,713	-	1,713	1,713	-
i080-004	80	240	5	1,866	1,866	1,866	1,866	-	1,866	1,866	-
i080-005	80	240	5	1,790	1,885	1,790	1,790	-	1,790	1,790	-
i080-011	80	700	5	1,479	1,488	1,479	1,479	-	1,479	1,479	-
i080-012	80	700	5	1,484	1,494	1,484	1,484	-	1,484	1,484	-
i080-013	80	700	5	1,381	1,383	1,381	1,381	-	1,381	1,381	-
i080-014	80	700	5	1,397	1,397	1,397	1,397	-	1,397	1,397	-
i080-015	80	700	5	1,495	1,496	1,495	1,495	-	1,495	1,495	-
i080-021	80	6320	5	1,175	1,175	1,175	1,175	-	1,175	1,175	-
i080-022	80	6320	5	1,178	1,178	1,178	1,178	-	1,178	1,178	-
i080-023	80	6320	5	1,174	1,174	1,174	1,174	-	1,174	1,174	-
i080-024	80	6320	5	1,161	1,161	1,161	1,161	-	1,161	1,161	-
i080-025	80	6320	5	1,162	1,162	1,162	1,162	-	1,162	1,162	-
i080-031	80	320	5	1,570	1,570	1,570	1,570	-	1,570	1,570	-
i080-032	80	320	5	2,088	2,089	2,088	2,088	-	2,088	2,088	-
i080-033	80	320	5	1,794	1,815	1,794	1,794	-	1,794	1,794	-
i080-034	80	320	5	1,688	1,771	1,688	1,688	-	1,688	1,688	-
i080-035	80	320	5	1,862	1,862	1,862	1,862	-	1,862	1,862	-
i080-041	80	1264	5	1,276	1,279	1,276	1,276	-	1,276	1,276	-
i080-042	80	1264	5	1,287	1,287	1,287	1,287	-	1,287	1,287	-
i080-043	80	1264	5	1,295	1,297	1,295	1,295	-	1,295	1,295	-
i080-044	80	1264	5	1,366	1,366	1,366	1,366	-	1,366	1,366	-
i080-045	80	1264	5	1,310	1,362	1,310	1,310	-	1,310	1,310	-
i080-101	80	240	7	2,608	2,704	2,608	2,608	-	2,608	2,608	-
i080-102	80	240	7	2,403	2,405	2,403	2,403	-	2,403	2,403	-
i080-103	80	240	7	2,603	2,672	2,603	2,603	-	2,603	2,603	-

i080-104	80	240	7	2,486	2,580	2,486	2,486	-	2,486	2,486	-
i080-105	80	240	7	2,203	2,203	2,203	2,203	-	2,203	2,203	-
i080-111	80	700	7	2,051	2,054	2,051	2,051	-	2,051	2,051	-
i080-112	80	700	7	1,885	1,893	1,885	1,885	-	1,885	1,885	-
i080-113	80	700	7	1,884	1,984	1,884	1,884	-	1,884	1,884	-
i080-114	80	700	7	1,895	1,895	1,895	1,895	-	1,895	1,895	-
i080-115	80	700	7	1,868	1,870	1,868	1,868	-	1,868	1,868	-
i080-121	80	6320	7	1,561	1,561	1,561	1,561	-	1,561	1,561	-
i080-122	80	6320	7	1,561	1,561	1,561	1,561	-	1,561	1,561	-
i080-123	80	6320	7	1,569	1,569	1,569	1,569	-	1,569	1,569	-
i080-124	80	6320	7	1,555	1,555	1,555	1,555	-	1,555	1,555	-
i080-125	80	6320	7	1,572	1,572	1,572	1,572	-	1,572	1,572	-
i080-131	80	320	7	2,284	2,371	2,284	2,284	-	2,284	2,284	-
i080-132	80	320	7	2,180	2,197	2,180	2,180	-	2,180	2,180	-
i080-133	80	320	7	2,261	2,261	2,261	2,261	-	2,261	2,261	-
i080-134	80	320	7	2,070	2,090	2,070	2,070	-	2,070	2,070	-
i080-135	80	320	7	2,102	2,102	2,105	2,102	-	2,102	2,102	-
i080-141	80	1264	7	1,788	1,823	1,788	1,788	-	1,788	1,788	-
i080-142	80	1264	7	1,708	1,708	1,708	1,708	-	1,708	1,708	-
i080-143	80	1264	7	1,767	1,777	1,767	1,767	-	1,767	1,767	-
i080-144	80	1264	7	1,772	1,772	1,772	1,772	-	1,772	1,772	-
i080-145	80	1264	7	1,762	1,762	1,762	1,762	-	1,762	1,762	-
i080-201	80	240	15	4,760	4,764	4,760	4,760	-	4,760	4,760	-
i080-202	80	240	15	4,650	4,680	4,650	4,650	-	4,652	4,650	-
i080-203	80	240	15	4,599	4,708	4,692	4,599	-	4,599	4,599	-
i080-204	80	240	15	4,492	4,583	4,499	4,499	-	4,499	4,499	-
i080-205	80	240	15	4,564	4,660	4,576	4,564	-	4,564	4,564	-
i080-211	80	700	15	3,631	3,746	3,631	3,631	-	3,631	3,631	-
i080-212	80	700	15	3,677	3,794	3,679	3,679	-	3,677	3,677	-
i080-213	80	700	15	3,678	3,751	3,730	3,693	-	3,730	3,678	-
i080-214	80	700	15	3,734	3,759	3,734	3,734	-	3,734	3,734	-
i080-215	80	700	15	3,681	3,767	3,681	3,681	-	3,681	3,681	-
i080-221	80	6320	15	3,158	3,158	3,158	3,158	-	3,158	3,158	-
i080-222	80	6320	15	3,141	3,141	3,141	3,141	-	3,141	3,141	-
i080-223	80	6320	15	3,156	3,156	3,156	3,156	-	3,156	3,156	-
i080-224	80	6320	15	3,159	3,159	3,159	3,159	-	3,159	3,159	-
i080-225	80	6320	15	3,150	3,150	3,150	3,150	-	3,150	3,150	-
i080-231	80	320	15	4,354	4,466	4,363	4,354	-	4,360	4,354	-
i080-232	80	320	15	4,199	4,392	4,290	4,199	-	4,199	4,199	-
i080-233	80	320	15	4,118	4,265	4,136	4,118	-	4,136	4,118	-
i080-234	80	320	15	4,274	4,274	4,302	4,274	-	4,274	4,274	-
i080-235	80	320	15	4,487	4,490	4,487	4,487	-	4,534	4,487	-
i080-241	80	1264	15	3,538	3,589	3,538	3,538	-	3,538	3,538	-
i080-242	80	1264	15	3,458	3,503	3,459	3,458	-	3,458	3,458	-
i080-243	80	1264	15	3,474	3,494	3,474	3,474	-	3,474	3,474	-
i080-244	80	1264	15	3,466	3,583	3,472	3,473	-	3,472	3,471	-
i080-245	80	1264	15	3,467	3,559	3,467	3,467	-	3,467	3,467	-
i080-301	80	240	19	5,519	5,628	5,519	5,519	-	5,519	5,519	-
i080-302	80	240	19	5,944	6,139	5,944	5,944	-	5,944	5,944	-
i080-303	80	240	19	5,777	5,978	5,788	5,852	-	5,777	5,780	-
i080-304	80	240	19	5,586	5,586	5,586	5,586	-	5,586	5,586	-
i080-305	80	240	19	5,932	5,966	5,936	5,932	-	5,932	5,932	-
i080-311	80	700	19	4,554	4,762	4,574	4,554	-	4,573	4,554	-
i080-312	80	700	19	4,534	4,659	4,563	4,534	-	4,534	4,534	-
i080-313	80	700	19	4,509	4,638	4,532	4,509	-	4,509	4,509	-
i080-314	80	700	19	4,515	4,554	4,515	4,515	-	4,515	4,515	-
i080-315	80	700	19	4,459	4,474	4,459	4,459	-	4,459	4,459	-
i080-321	80	6320	19	3,932	3,932	3,932	3,932	-	3,932	3,932	-
i080-322	80	6320	19	3,937	3,937	3,937	3,937	-	3,937	3,937	-
i080-323	80	6320	19	3,946	3,946	3,946	3,946	-	3,946	3,946	-
i080-324	80	6320	19	3,932	3,932	3,932	3,932	-	3,932	3,932	-
i080-325	80	6320	19	3,924	3,924	3,924	3,924	-	3,924	3,924	-
i080-331	80	320	19	5,226	5,334	5,301	5,230	-	5,226	5,226	-
i080-332	80	320	19	5,362	5,388	5,378	5,362	-	5,362	5,362	-
i080-333	80	320	19	5,381	5,467	5,441	5,441	-	5,441	5,441	-
i080-334	80	320	19	5,264	5,269	5,270	5,265	-	5,267	5,267	-
i080-335	80	320	19	4,953	5,059	5,038	4,953	-	4,953	4,953	-
i080-341	80	1264	19	4,236	4,268	4,236	4,265	-	4,236	4,236	-
i080-342	80	1264	19	4,337	4,375	4,345	4,342	-	4,342	4,337	-
i080-343	80	1264	19	4,246	4,371	4,246	4,246	-	4,246	4,246	-
i080-344	80	1264	19	4,310	4,405	4,310	4,310	-	4,310	4,310	-
i080-345	80	1264	19	4,341	4,451	4,391	4,351	-	4,403	4,341	-
i160-001	160	480	6	2,490	2,513	2,490	2,490	-	2,490	2,490	-
i160-002	160	480	6	2,158	2,160	2,158	2,158	-	2,158	2,158	-
i160-003	160	480	6	2,297	2,297	2,297	2,297	-	2,297	2,297	-
i160-004	160	480	6	2,370	2,495	2,370	2,370	-	2,370	2,370	-
i160-005	160	480	6	2,495	2,594	2,495	2,495	-	2,495	2,495	-
i160-011	160	1624	6	1,677	1,677	1,677	1,677	-	1,677	1,677	-
i160-012	160	1624	6	1,750	1,774	1,750	1,750	-	1,750	1,750	-
i160-013	160	1624	6	1,661	1,759	1,661	1,661	-	1,661	1,661	-
i160-014	160	1624	6	1,778	1,796	1,778	1,778	-	1,778	1,778	-
i160-015	160	1624	6	1,768	1,768	1,768	1,768	-	1,768	1,768	-
i160-021	160	25440	6	1,352	1,352	1,352	1,352	-	1,352	1,352	-
i160-022	160	25440	6	1,365	1,365	1,365	1,365	-	1,365	1,365	-
i160-023	160	25440	6	1,351	1,351	1,351	1,351	-	1,351	1,351	-
i160-024	160	25440	6	1,371	1,371	1,371	1,371	-	1,371	1,371	-
i160-025	160	25440	6	1,366	1,366	1,366	1,366	-	1,366	1,366	-
i160-031	160	640	6	2,170	2,170	2,170	2,170	-	2,170	2,170	-
i160-032	160	640	6	2,330	2,425	2,330	2,330	-	2,330	2,330	-
i160-033	160	640	6	2,101	2,107	2,101	2,101	-	2,101	2,101	-
i160-034	160	640	6	2,083	2,083	2,083	2,083	-	2,083	2,083	-
i160-035	160	640	6	2,103	2,103	2,103	2,103	-	2,103	2,103	-
i160-041	160	5088	6	1,494	1,543	1,494	1,494	-	1,494	1,494	-
i160-042	160	5088	6	1,486	1,486	1,486	1,486	-	1,486	1,486	-
i160-043	160	5088	6	1,549	1,561	1,549	1,549	-	1,549	1,549	-
i160-044	160	5088	6	1,478	1,478	1,478	1,478	-	1,478	1,478	-
i160-045	160	5088	6	1,554	1,560	1,554	1,554	-	1,554	1,554	-
i160-101	160	480	11	3,859	3,859	3,859	3,859	-	3,859	3,859	-
i160-102	160	480	11	3,747	3,824	3,747	3,747	-	3,747	3,747	-
i160-103	160	480	11	3,837	3,837	3,837	3,837	-	3,837	3,837	-
i160-104	160	480	11	4,063	4,065	4,063	4,063	-	4,063	4,063	-
i160-105	160	480	11	3,563	3,655	3,563	3,563	-	3,563	3,563	-
i160-111	160	1624	11	2,869	2,869	2,869	2,869	-	2,869	2,869	-
i160-112	160	1624	11	2,924	2,934	2,924	2,924	-	2,924	2,924	-
i160-113	160	1624	11	2,866	2,898	2,866	2,866	-	2,866	2,866	-
i160-114	160	1624	11	2,989	3,047	3,010	3,010	-	3,024	3,024	-
i160-115	160	1624	11	2,937	2,944	2,937	2,937	-	2,937	2,937	-

i160-121	160	25440	11	2,363	2,363	2,363	2,363	-	2,363	2,363	-
i160-122	160	25440	11	2,348	2,348	2,348	2,348	-	2,348	2,348	-
i160-123	160	25440	11	2,355	2,355	2,355	2,355	-	2,355	2,355	-
i160-124	160	25440	11	2,352	2,352	2,352	2,352	-	2,352	2,352	-
i160-125	160	25440	11	2,351	2,351	2,351	2,351	-	2,351	2,351	-
i160-131	160	640	11	3,356	3,431	3,356	3,356	-	3,356	3,356	-
i160-132	160	640	11	3,450	3,576	3,450	3,450	-	3,450	3,450	-
i160-133	160	640	11	3,585	3,763	3,585	3,585	-	3,585	3,585	-
i160-134	160	640	11	3,470	3,489	3,470	3,470	-	3,470	3,470	-
i160-135	160	640	11	3,716	3,732	3,716	3,716	-	3,716	3,716	-
i160-141	160	5088	11	2,549	2,650	2,549	2,549	-	2,549	2,549	-
i160-142	160	5088	11	2,562	2,634	2,562	2,571	-	2,571	2,562	-
i160-143	160	5088	11	2,557	2,663	2,557	2,557	-	2,557	2,557	-
i160-144	160	5088	11	2,607	2,612	2,610	2,607	-	2,607	2,607	-
i160-145	160	5088	11	2,578	2,578	2,578	2,578	-	2,578	2,578	-
i160-201	160	480	23	6,923	7,200	6,927	6,923	-	6,923	6,923	-
i160-202	160	480	23	6,930	7,038	6,930	6,930	-	6,934	6,930	-
i160-203	160	480	23	7,243	7,330	7,337	7,243	-	7,252	7,243	-
i160-204	160	480	23	7,068	7,273	7,149	7,077	-	7,068	7,068	-
i160-205	160	480	23	7,122	7,323	7,214	7,132	-	7,122	7,122	-
i160-211	160	1624	23	5,583	5,650	5,613	5,640	-	5,670	5,648	-
i160-212	160	1624	23	5,643	5,839	5,739	5,649	-	5,725	5,652	-
i160-213	160	1624	23	5,647	5,743	5,724	5,681	-	5,720	5,686	-
i160-214	160	1624	23	5,720	5,769	5,857	5,808	-	5,808	5,734	-
i160-215	160	1624	23	5,518	5,849	5,633	5,603	-	5,633	5,597	-
i160-221	160	25440	23	4,729	4,729	4,729	4,729	-	4,729	4,729	-
i160-222	160	25440	23	4,697	4,697	4,697	4,697	-	4,697	4,697	-
i160-223	160	25440	23	4,730	4,730	4,730	4,730	-	4,730	4,730	-
i160-224	160	25440	23	4,721	4,721	4,721	4,721	-	4,721	4,721	-
i160-225	160	25440	23	4,728	4,728	4,728	4,728	-	4,728	4,728	-
i160-231	160	640	23	6,662	6,742	6,761	6,725	-	6,726	6,732	-
i160-232	160	640	23	6,558	6,842	6,639	6,558	-	6,566	6,566	-
i160-233	160	640	23	6,339	6,427	6,500	6,339	-	6,339	6,339	-
i160-234	160	640	23	6,594	6,610	6,675	6,594	-	6,594	6,594	-
i160-235	160	640	23	6,764	6,930	6,976	6,767	-	6,846	6,846	-
i160-241	160	5088	23	5,086	5,145	5,107	5,086	-	5,086	5,086	-
i160-242	160	5088	23	5,106	5,251	5,142	5,150	-	5,139	5,106	-
i160-243	160	5088	23	5,050	5,165	5,094	5,095	-	5,095	5,051	-
i160-244	160	5088	23	5,076	5,212	5,141	5,139	-	5,118	5,114	-
i160-245	160	5088	23	5,084	5,167	5,094	5,098	-	5,098	5,084	-
i160-301	160	480	39	11,816	12,025	12,986	11,905	-	11,918	11,904	-
i160-302	160	480	39	11,497	11,640	12,558	11,591	-	11,686	11,497	-
i160-303	160	480	39	11,445	11,553	12,326	11,445	-	11,540	11,445	-
i160-304	160	480	39	11,448	11,542	12,138	11,521	-	11,546	11,521	-
i160-305	160	480	39	11,423	11,520	12,065	11,465	-	11,583	11,456	-
i160-311	160	1624	39	9,135	9,242	9,463	9,256	-	9,355	9,255	-
i160-312	160	1624	39	9,052	9,288	9,454	9,226	-	9,197	9,223	-
i160-313	160	1624	39	9,159	9,209	9,578	9,369	-	9,431	9,309	-
i160-314	160	1624	39	8,941	8,958	9,359	9,065	-	9,045	9,057	-
i160-315	160	1624	39	9,086	9,225	9,362	9,154	-	9,161	9,143	-
i160-321	160	25440	39	7,876	7,903	7,903	7,903	-	7,903	7,876	-
i160-322	160	25440	39	7,859	7,892	7,892	7,892	-	7,892	7,859	-
i160-323	160	25440	39	7,876	7,883	7,883	7,883	-	7,883	7,883	-
i160-324	160	25440	39	7,884	7,912	7,912	7,884	-	7,912	7,884	-
i160-325	160	25440	39	7,862	7,915	7,877	7,877	-	7,877	7,877	-
i160-331	160	640	39	10,414	10,614	11,015	10,505	-	10,583	10,489	-
i160-332	160	640	39	10,806	10,845	11,605	10,871	-	10,836	10,806	-
i160-333	160	640	39	10,561	10,651	11,220	10,637	-	10,668	10,624	-
i160-334	160	640	39	10,327	10,697	10,928	10,489	-	10,590	10,392	-
i160-335	160	640	39	10,589	10,730	11,192	10,772	-	10,672	10,594	-
i160-341	160	5088	39	8,331	8,427	8,433	8,404	-	8,412	8,397	-
i160-342	160	5088	39	8,348	8,518	8,538	8,370	-	8,489	8,384	-
i160-343	160	5088	39	8,275	8,318	8,354	8,342	-	8,358	8,352	-
i160-344	160	5088	39	8,307	8,363	8,389	8,324	-	8,363	8,324	-
i160-345	160	5088	39	8,327	8,441	8,434	8,382	-	8,436	8,380	-
alue2087	1244	3942	33	1,049	1,055	1,189	1,052	1,049	1,052	1,052	1,052
alue2105	1220	3716	33	1,032	1,051	1,105	1,032	1,032	1,045	1,032	1,032
alue3146	3626	11738	63	2,240	2,316	3,130	2,254	2,245	2,408	2,240	2,243
alue5067	3524	11120	67	2,586	2,688	3,902	2,608	2,613	2,934	2,586	2,586
alue6179	3372	10426	66	2,452	2,483	3,674	2,465	2,475	2,679	2,452	2,454
alue6951	2818	8838	66	2,386	2,519	3,604	2,414	2,413	2,674	2,386	2,387
alue7229	940	2948	33	824	826	930	824	824	842	824	824
alut0787	1160	4178	33	982	989	1,050	982	982	988	982	982
alut0805	966	3332	33	958	966	1,094	958	958	958	958	958
alut1181	3041	11386	63	2,353	2,449	3,569	2,381	2,381	2,592	2,358	2,355
alut2764	387	1252	33	640	650	723	640	640	640	640	640
i320-001	320	960	7	2,672	2,672	2,672	2,672	-	2,672	2,672	-
i320-002	320	960	7	2,847	2,847	2,847	2,847	-	2,847	2,847	-
i320-003	320	960	7	2,972	2,984	2,972	2,972	-	2,972	2,972	-
i320-004	320	960	7	2,905	2,989	2,905	2,905	-	2,905	2,905	-
i320-005	320	960	7	2,991	3,093	2,991	2,991	-	2,991	2,991	-
i320-011	320	3690	7	2,053	2,061	2,053	2,053	-	2,053	2,053	-
i320-012	320	3690	7	1,997	2,062	1,997	1,997	-	1,997	1,997	-
i320-013	320	3690	7	2,072	2,072	2,072	2,072	-	2,072	2,072	-
i320-014	320	3690	7	2,061	2,073	2,061	2,061	-	2,061	2,061	-
i320-015	320	3690	7	2,059	2,070	2,059	2,059	-	2,059	2,059	-
i320-021	320	102080	7	1,553	1,553	1,553	1,553	-	1,553	1,553	-
i320-022	320	102080	7	1,565	1,565	1,565	1,565	-	1,565	1,565	-
i320-023	320	102080	7	1,549	1,549	1,549	1,549	-	1,549	1,549	-
i320-024	320	102080	7	1,553	1,553	1,553	1,553	-	1,553	1,553	-
i320-025	320	102080	7	1,550	1,550	1,550	1,550	-	1,550	1,550	-
i320-031	320	1280	7	2,673	2,764	2,673	2,673	-	2,673	2,673	-
i320-032	320	1280	7	2,770	2,982	2,770	2,770	-	2,770	2,770	-
i320-033	320	1280	7	2,769	2,865	2,769	2,769	-	2,769	2,769	-
i320-034	320	1280	7	2,521	2,529	2,521	2,521	-	2,521	2,521	-
i320-035	320	1280	7	2,385	2,656	2,385	2,385	-	2,385	2,385	-
i320-041	320	20416	7	1,707	1,761	1,707	1,707	-	1,707	1,707	-
i320-042	320	20416	7	1,682	1,686	1,682	1,682	-	1,682	1,682	-
i320-043	320	20416	7	1,723	1,760	1,723	1,723	-	1,723	1,723	-
i320-044	320	20416	7	1,681	1,770	1,681	1,681	-	1,681	1,681	-
i320-045	320	20416	7	1,686	1,686	1,686	1,686	-	1,686	1,686	-
i320-101	320	960	16	5,548	5,554	5,548	5,548	-	5,548	5,548	-
i320-102	320	960	16	5,556	5,754	5,556	5,556	-	5,556	5,556	-
i320-103	320	960	16	6,239	6,448	6,239	6,239	-	6,243	6,239	-
i320-104	320	960	16	5,703	6,057	5,797	5,780	-	5,703	5,703	-
i320-105	320	960	16	5,928	6,139	5,928	5,932	-	5,928	5,936	-
i320-111	320	3690	16	4,273	4,370	4,283	4,286	-	4,278	4,276	-

i320-112	320	3690	16	4,213	4,328	4,213	4,213	-	4,213	4,213	-
i320-113	320	3690	16	4,205	4,328	4,205	4,205	-	4,212	4,205	-
i320-114	320	3690	16	4,104	4,215	4,104	4,104	-	4,142	4,104	-
i320-115	320	3690	16	4,238	4,261	4,317	4,238	-	4,314	4,238	-
i320-121	320	102080	16	3,321	3,321	3,321	3,321	-	3,321	3,321	-
i320-122	320	102080	16	3,314	3,314	3,314	3,314	-	3,314	3,314	-
i320-123	320	102080	16	3,332	3,332	3,332	3,332	-	3,332	3,332	-
i320-124	320	102080	16	3,323	3,323	3,323	3,323	-	3,323	3,323	-
i320-125	320	102080	16	3,340	3,345	3,340	3,340	-	3,340	3,340	-
i320-131	320	1280	16	5,255	5,388	5,255	5,268	-	5,255	5,255	-
i320-132	320	1280	16	5,052	5,264	5,058	5,052	-	5,060	5,052	-
i320-133	320	1280	16	5,125	5,237	5,125	5,125	-	5,125	5,125	-
i320-134	320	1280	16	5,272	5,301	5,295	5,272	-	5,335	5,272	-
i320-135	320	1280	16	5,342	5,447	5,367	5,342	-	5,355	5,342	-
i320-141	320	20416	16	3,606	3,638	3,611	3,606	-	3,610	3,606	-
i320-142	320	20416	16	3,567	3,590	3,567	3,567	-	3,567	3,567	-
i320-143	320	20416	16	3,561	3,648	3,561	3,561	-	3,561	3,561	-
i320-144	320	20416	16	3,512	3,648	3,512	3,512	-	3,512	3,512	-
i320-145	320	20416	16	3,601	3,610	3,601	3,601	-	3,601	3,601	-
i320-201	320	960	33	10,044	10,344	10,650	10,044	-	10,189	10,044	-
i320-202	320	960	33	11,223	11,254	11,731	11,231	-	11,235	11,233	-
i320-203	320	960	33	10,148	10,520	10,838	10,227	-	10,161	10,148	-
i320-204	320	960	33	10,275	10,464	11,015	10,367	-	10,429	10,361	-
i320-205	320	960	33	10,573	10,797	10,987	10,725	-	10,651	10,642	-
i320-211	320	3690	33	8,039	8,299	8,488	8,157	-	8,172	8,116	-
i320-212	320	3690	33	8,044	8,231	8,375	8,140	-	8,184	8,143	-
i320-213	320	3690	33	7,984	8,080	8,291	8,222	-	8,228	8,155	-
i320-214	320	3690	33	8,046	8,191	8,399	8,112	-	8,193	8,051	-
i320-215	320	3690	33	8,015	8,202	8,425	8,139	-	8,177	8,041	-
i320-221	320	102080	33	6,679	6,700	6,697	6,697	-	6,697	6,697	-
i320-222	320	102080	33	6,686	6,688	6,688	6,688	-	6,688	6,688	-
i320-223	320	102080	33	6,695	6,709	6,709	6,709	-	6,709	6,709	-
i320-224	320	102080	33	6,694	6,694	6,694	6,694	-	6,694	6,694	-
i320-225	320	102080	33	6,691	6,701	6,701	6,701	-	6,701	6,701	-
i320-231	320	1280	33	9,862	10,242	10,326	10,091	-	10,142	9,863	-
i320-232	320	1280	33	9,933	10,630	10,508	10,116	-	10,183	10,090	-
i320-233	320	1280	33	9,787	10,228	10,289	9,961	-	9,988	9,888	-
i320-234	320	1280	33	9,517	9,851	10,147	9,611	-	9,605	9,520	-
i320-235	320	1280	33	9,945	10,394	10,357	9,945	-	10,118	9,945	-
i320-241	320	20416	33	7,027	7,209	7,162	7,027	-	7,027	7,056	-
i320-242	320	20416	33	7,072	7,159	7,153	7,138	-	7,140	7,135	-
i320-243	320	20416	33	7,044	7,092	7,152	7,113	-	7,132	7,097	-
i320-244	320	20416	33	7,078	7,133	7,191	7,165	-	7,129	7,131	-
i320-245	320	20416	33	7,046	7,133	7,131	7,066	-	7,084	7,081	-
i320-301	320	960	79	23,279	23,621	29,059	23,524	-	24,925	23,367	-
i320-302	320	960	79	23,387	24,152	28,655	23,742	-	24,684	23,652	-
i320-303	320	960	79	22,693	23,140	28,646	22,986	-	24,345	22,787	-
i320-304	320	960	79	23,451	24,182	28,930	23,635	-	24,970	23,554	-
i320-305	320	960	79	22,547	22,923	27,089	22,663	-	23,487	22,550	-
i320-311	320	3690	79	17,945	18,364	20,230	18,909	-	18,642	18,605	-
i320-312	320	3690	79	18,122	18,567	20,488	18,831	-	19,121	18,783	-
i320-313	320	3690	79	17,991	18,161	20,341	18,809	-	18,777	18,440	-
i320-314	320	3690	79	18,088	18,493	20,139	18,770	-	19,141	18,668	-
i320-315	320	3690	79	17,987	18,380	20,293	18,849	-	19,223	18,616	-
i320-321	320	102080	79	15,648	15,725	15,727	15,687	-	15,708	15,687	-
i320-322	320	102080	79	15,646	15,711	15,697	15,695	-	15,693	15,693	-
i320-323	320	102080	79	15,654	15,688	15,693	15,698	-	15,685	15,698	-
i320-324	320	102080	79	15,667	15,756	15,714	15,713	-	15,712	15,709	-
i320-325	320	102080	79	15,649	15,739	15,724	15,711	-	15,712	15,704	-
i320-331	320	1280	79	21,517	22,036	25,752	22,132	-	23,171	22,099	-
i320-332	320	1280	79	21,674	22,109	25,979	22,105	-	23,199	22,027	-
i320-333	320	1280	79	21,339	21,877	25,817	22,083	-	23,075	21,695	-
i320-334	320	1280	79	21,415	21,779	26,403	21,950	-	23,132	21,767	-
i320-335	320	1280	79	21,378	21,839	25,364	21,839	-	22,836	21,622	-
i320-341	320	20416	79	16,296	16,474	17,050	16,592	-	16,638	16,460	-
i320-342	320	20416	79	16,228	16,408	16,865	16,678	-	16,475	16,347	-
i320-343	320	20416	79	16,281	16,482	16,967	16,590	-	16,655	16,582	-
i320-344	320	20416	79	16,295	16,474	16,991	16,643	-	16,660	16,587	-
i320-345	320	20416	79	16,289	16,500	16,897	16,609	-	16,587	16,471	-
i640-001	640	1920	8	4,033	4,033	4,033	4,033	-	4,033	4,033	-
i640-002	640	1920	8	3,588	3,588	3,588	3,588	-	3,588	3,588	-
i640-003	640	1920	8	3,438	3,631	3,438	3,438	-	3,438	3,438	-
i640-004	640	1920	8	4,000	4,169	4,000	4,000	-	4,000	4,000	-
i640-005	640	1920	8	4,006	4,098	4,006	4,006	-	4,006	4,006	-
i640-011	640	8270	8	2,392	2,392	2,392	2,392	-	2,392	2,392	-
i640-012	640	8270	8	2,465	2,466	2,466	2,465	-	2,465	2,465	-
i640-013	640	8270	8	2,399	2,577	2,399	2,399	-	2,399	2,399	-
i640-014	640	8270	8	2,171	2,171	2,171	2,171	-	2,171	2,171	-
i640-015	640	8270	8	2,347	2,351	2,347	2,347	-	2,347	2,347	-
i640-021	640	408960	8	1,749	1,749	1,749	1,749	-	1,749	1,749	-
i640-022	640	408960	8	1,756	1,756	1,756	1,756	-	1,756	1,756	-
i640-023	640	408960	8	1,754	1,754	1,754	1,754	-	1,754	1,754	-
i640-024	640	408960	8	1,751	1,751	1,751	1,751	-	1,751	1,751	-
i640-025	640	408960	8	1,745	1,745	1,745	1,745	-	1,745	1,745	-
i640-031	640	2560	8	3,278	3,591	3,278	3,278	-	3,278	3,278	-
i640-032	640	2560	8	3,187	3,370	3,187	3,187	-	3,187	3,187	-
i640-033	640	2560	8	3,260	3,327	3,260	3,260	-	3,260	3,260	-
i640-034	640	2560	8	2,953	3,091	2,953	2,953	-	2,953	2,953	-
i640-035	640	2560	8	3,292	3,580	3,292	3,292	-	3,292	3,292	-
i640-041	640	81792	8	1,897	1,967	1,897	1,897	-	1,897	1,897	-
i640-042	640	81792	8	1,934	1,976	1,934	1,934	-	1,934	1,934	-
i640-043	640	81792	8	1,931	1,943	1,931	1,931	-	1,931	1,931	-
i640-044	640	81792	8	1,938	1,950	1,938	1,938	-	1,938	1,938	-
i640-045	640	81792	8	1,866	1,866	1,866	1,866	-	1,866	1,866	-
i640-101	640	1920	24	8,764	9,344	9,059	8,773	-	8,778	8,846	-
i640-102	640	1920	24	9,109	9,298	9,478	9,191	-	9,191	9,191	-
i640-103	640	1920	24	8,819	9,188	9,129	8,819	-	8,819	8,848	-
i640-104	640	1920	24	9,040	9,365	9,040	9,084	-	9,040	9,040	-
i640-105	640	1920	24	9,623	9,920	9,823	9,785	-	9,701	9,715	-
i640-111	640	8270	24	6,167	6,232	6,314	6,256	-	6,172	6,189	-
i640-112	640	8270	24	6,304	6,466	6,597	6,400	-	6,424	6,424	-
i640-113	640	8270	24	6,249	6,381	6,486	6,278	-	6,309	6,311	-
i640-114	640	8270	24	6,308	6,323	6,488	6,385	-	6,419	6,308	-
i640-115	640	8270	24	6,217	6,467	6,331	6,228	-	6,233	6,226	-
i640-121	640	408960	24	4,906	4,906	4,906	4,906	-	4,906	4,906	-
i640-122	640	408960	24	4,911	4,911	4,911	4,911	-	4,911	4,911	-
i640-123	640	408960	24	4,913	4,915	4,913	4,913	-	4,913	4,913	-

i640-124	640	408960	24	4,906	4,908	4,906	4,906	-	4,906	4,906	-
i640-125	640	408960	24	4,920	4,920	4,920	4,920	-	4,920	4,920	-
i640-131	640	2560	24	8,097	8,240	8,313	8,201	-	8,216	8,179	-
i640-132	640	2560	24	8,154	8,788	8,500	8,154	-	8,154	8,154	-
i640-133	640	2560	24	8,021	8,279	8,277	8,087	-	8,099	8,084	-
i640-134	640	2560	24	7,754	8,044	7,941	7,754	-	7,825	7,754	-
i640-135	640	2560	24	7,696	8,075	7,927	7,696	-	7,698	7,696	-
i640-141	640	81792	24	5,199	5,309	5,269	5,205	-	5,200	5,200	-
i640-142	640	81792	24	5,193	5,253	5,260	5,210	-	5,232	5,193	-
i640-143	640	81792	24	5,194	5,296	5,252	5,194	-	5,243	5,194	-
i640-144	640	81792	24	5,205	5,307	5,264	5,232	-	5,257	5,234	-
i640-145	640	81792	24	5,218	5,238	5,277	5,277	-	5,273	5,256	-
i640-201	640	1920	49	16,079	16,322	18,124	16,195	-	16,517	16,130	-
i640-202	640	1920	49	16,324	17,236	18,497	16,339	-	16,524	16,409	-
i640-203	640	1920	49	16,124	16,657	18,136	16,389	-	16,721	16,210	-
i640-204	640	1920	49	16,239	16,852	17,735	16,510	-	16,794	16,415	-
i640-205	640	1920	49	16,616	16,787	18,336	16,927	-	17,206	16,643	-
i640-211	640	8270	49	11,984	12,477	12,896	12,222	-	12,542	12,329	-
i640-212	640	8270	49	11,795	12,160	12,746	12,354	-	12,329	12,232	-
i640-213	640	8270	49	11,879	12,128	13,123	12,265	-	12,226	12,077	-
i640-214	640	8270	49	11,898	12,157	12,965	12,352	-	12,391	12,088	-
i640-215	640	8270	49	12,081	12,277	13,142	12,512	-	12,429	12,299	-
i640-221	640	408960	49	9,821	9,864	9,827	9,822	-	9,821	9,821	-
i640-222	640	408960	49	9,798	9,835	9,798	9,807	-	9,798	9,798	-
i640-223	640	408960	49	9,811	9,871	9,858	9,817	-	9,813	9,813	-
i640-224	640	408960	49	9,805	9,851	9,819	9,805	-	9,807	9,817	-
i640-225	640	408960	49	9,807	9,860	9,860	9,815	-	9,815	9,809	-
i640-231	640	2560	49	15,014	15,212	16,687	15,248	-	15,468	15,175	-
i640-232	640	2560	49	14,630	14,945	16,097	15,012	-	15,353	14,877	-
i640-233	640	2560	49	14,797	15,215	16,703	15,087	-	15,297	15,051	-
i640-234	640	2560	49	15,203	15,732	16,942	15,534	-	15,824	15,556	-
i640-235	640	2560	49	14,803	15,514	16,155	15,223	-	15,443	15,024	-
i640-241	640	81792	49	10,230	10,361	10,518	10,431	-	10,355	10,349	-
i640-242	640	81792	49	10,195	10,345	10,437	10,304	-	10,358	10,219	-
i640-243	640	81792	49	10,215	10,348	10,493	10,263	-	10,360	10,355	-
i640-244	640	81792	49	10,246	10,323	10,489	10,369	-	10,397	10,293	-
i640-245	640	81792	49	10,223	10,449	10,410	10,342	-	10,428	10,355	-
i640-301	640	1920	159	45,005	45,656	63,687	46,507	-	53,785	45,506	-
i640-302	640	1920	159	45,736	46,969	65,625	47,042	-	55,674	46,333	-
i640-303	640	1920	159	44,922	45,360	61,364	45,659	-	53,320	45,151	-
i640-304	640	1920	159	46,233	47,152	67,574	47,976	-	56,179	47,221	-
i640-305	640	1920	159	45,902	46,984	64,921	47,555	-	55,740	46,941	-
i640-311	640	8270	159	-	36,527	44,266	38,607	-	41,123	37,760	-
i640-312	640	8270	159	-	36,445	44,709	38,862	-	40,962	37,509	-
i640-321	640	408960	159	31,094	31,228	31,391	31,247	-	31,244	31,205	-
i640-322	640	408960	159	31,068	31,175	31,391	31,233	-	31,307	31,223	-
i640-323	640	408960	159	31,080	31,218	31,420	31,296	-	31,318	31,244	-
i640-324	640	408960	159	31,092	31,219	31,369	31,219	-	31,230	31,227	-
i640-325	640	408960	159	31,081	31,199	31,419	31,273	-	31,247	31,224	-
i640-331	640	2560	159	42,796	43,818	57,927	45,213	-	49,781	44,086	-
i640-332	640	2560	159	42,548	43,830	56,681	44,824	-	51,006	43,918	-
i640-333	640	2560	159	42,345	43,099	58,567	44,781	-	50,785	43,617	-
i640-334	640	2560	159	42,768	43,426	57,959	44,792	-	50,666	43,857	-
i640-335	640	2560	159	43,035	44,118	58,094	44,917	-	51,126	44,168	-
i640-341	640	81792	159	32,042	32,374	34,331	33,033	-	32,868	32,719	-
i640-342	640	81792	159	31,978	32,280	34,185	33,121	-	33,081	32,519	-
i640-343	640	81792	159	32,015	32,328	34,217	33,053	-	33,031	32,738	-
i640-344	640	81792	159	31,991	32,354	34,226	33,172	-	32,952	32,675	-
i640-345	640	81792	159	31,994	32,319	34,206	33,161	-	32,911	32,752	-
lin01	53	160	3	503	503	503	503	503	503	503	503
lin02	55	164	5	557	557	557	557	557	557	557	557
lin03	57	168	7	926	926	926	926	926	926	926	926
lin04	157	532	5	1,239	1,307	1,239	1,239	1,239	1,239	1,239	1,239
lin05	160	538	8	1,703	1,709	1,703	1,703	1,703	1,703	1,703	1,703
lin06	165	548	13	1,348	1,383	1,348	1,348	1,348	1,348	1,348	1,348
lin07	307	1052	5	1,885	1,897	1,885	1,885	1,885	1,885	1,885	1,885
lin08	311	1060	9	2,248	2,252	2,248	2,248	2,248	2,248	2,248	2,248
lin09	313	1064	11	2,752	2,815	2,752	2,752	2,752	2,752	2,752	2,752
lin10	321	1080	19	4,132	4,294	4,149	4,132	4,132	4,135	4,132	4,132
lin11	816	2920	9	4,280	4,323	4,280	4,280	4,280	4,280	4,280	4,280
lin12	818	2924	11	5,250	5,356	5,250	5,250	5,250	5,250	5,250	5,250
lin13	822	2932	15	4,609	4,633	4,609	4,609	4,609	4,609	4,609	4,609
lin14	828	2944	21	5,824	6,143	5,967	5,824	5,824	5,824	5,824	5,824
lin15	840	2968	33	7,145	7,427	7,800	7,145	7,145	7,145	7,145	7,145
lin16	1981	7266	11	6,618	6,696	6,618	6,618	6,618	6,618	6,618	6,618
lin17	1989	7282	19	8,405	8,970	8,405	8,405	8,405	8,405	8,405	8,405
lin18	1994	7292	24	9,714	10,245	10,010	9,724	9,714	9,714	9,720	9,714
lin19	2010	7324	40	13,268	13,632	15,475	13,338	13,338	13,360	13,268	13,268
lin20	3675	13418	10	6,673	7,175	6,673	6,673	6,673	6,673	6,673	6,673
lin21	3683	13434	19	9,143	9,498	9,220	9,143	9,143	9,143	9,143	9,143
lin22	3692	13452	27	10,519	10,740	11,034	10,519	10,519	10,519	10,519	10,519
lin23	3716	13500	51	17,560	18,288	21,763	17,560	17,615	18,480	17,585	17,585