Preprocessing Steiner Problems from VLSI Layout

Eduardo Uchoa
Marcus Poggi de Aragão
Celso C. Ribeiro

Pontifícia Universidade Católica do Rio de Janeiro
Departamento de Informática
R. Marquês de São Vicente 225
Rio de Janeiro 22453-900, Brasil
e-mail: {uchoa, poggi, celso}@inf.puc-rio.br

December, 1999
Revised June, 2001

Abstract

VLSI layout applications yield instances of the Steiner tree problem over grid graphs with holes, which are considered hard to be solved by current methods. In particular, preprocessing techniques developed for Steiner problems over general graphs are not likely to reduce significantly such VLSI instances. We propose a new preprocessing procedure, extending earlier ideas from the literature and improving their application, so as to make them effective for VLSI problems. We report significant reductions within reasonable computational times, obtained with the application of this procedure to 116 instances of the SteinLib. These reductions allowed a branch-and-cut to solve 28 out of 32 open instances of the SteinLib, some with more than 10,000 vertices and 20,000 edges.

1 Introduction

The *Steiner Problem in Graphs* (SPG) consists in given a connected graph $G = (V, E)$, non-negative edge costs $c(e)$ (also denoted by $c(u, v)$, with $e = (u, v)$), and a set $T \subseteq V$ of *terminal vertices*, find a *Minimum Steiner Tree* (MST), i.e. a subtree (V', E') of G with $T \subseteq V'$ minimizing $\sum_{e \in E'} c(e)$. The SPG is one of the most widely studied NP-hard problems, see Maculan [10], Winter [14], and Hwang et al. [7] for good surveys.

Koch and Martin [8] presented an exact algorithm for SPG, consisting of preprocessing followed by a branch-and-cut procedure. They obtained good practical results and could solve most of the instances then available in the literature in reasonable times, including all classical problems in the OR-Library [4]. They also introduced a new set of instances extracted from real-world problems arising from VLSI layout. The basic characteristic of these instances is their support graph G, formed by rectangular grids with many irregularly placed holes. Figures 5 and 8 show instances diw0559 and taq0903, where line segments are edges, crossings are vertices, and terminals are indicated by filled squares. Edge costs are proportional to their lengths. A total of 116 such VLSI instances, classified in seven series, are available in the so called SteinLib [9] library.

However, Koch and Martin's code was not so successful in solving these new VLSI instances. The traditional reduction tests employed, which perform well for general graphs, are not effective for VLSI instances. This is a serious drawback, since the original instances are quite large, with several thousands of vertices and edges. Moreover, due to the quite regular graph and cost structures, there are many distinct solutions with the same cost. This kind of symmetry damps branch-and-cut performance, leading to highly degenerate linear programs.

We address this issue in this work. We propose a more powerful preprocessing procedure, enhancing some reduction tests from Duin and Volgenant [6] with the idea of expansion introduced by Winter [15] (for the Rectilinear Steiner Tree Problem). The reductions obtained with this procedure were significant, allowing a branch-and-cut code to solve 112 out of 116 VLSI instances in the SteinLib, 28 of them for the first time. Only four really large instances remained open after the experiments reported in this paper, with more than 54,000 edges[1]. In the next section, we review the classical reduction tests for the Steiner problem in graphs. New reduction tests are proposed in Section 3. Section 4 describes the preprocessing strategy based on the above tests and some implementation issues. Computational results illustrating the effectiveness of both the new reduction tests and the preprocessing strategy are reported in Section 5, together with some concluding remarks.

2 Traditional reduction tests

Reduction tests are procedures devised to reduce the original problem to a smaller, but equivalent problem. An edge $e \in E$ is said to be *choosable* if there is at least one MST containing e and *redundant* if there is at least one MST not containing e. Some reduction tests try to identify choosable and redundant edges. Once a choosable edge $e = (u, v)$ is identified, it can be forced into the solution and its endpoints u and v may be contracted. A redundant edge is simply deleted from the graph. In either case, the size of the problem is reduced. Other reduction tests lead to more complex graph transformations. Reduction tests may be successively applied to each reduced graphs until no further reduction is possible. Some simple tests are:

Test 1 Non-terminal with degree 1 (NTD1) - *A non-terminal vertex with degree 1 and its adjacent edge may be deleted.*

Test 2 Non-terminal with degree 2 (NTD2) - *A non-terminal vertex u with degree 2 and its adjacent edges (u, v) and (u, w) may be replaced by a single edge (v, w) with cost $c(u, v) + c(u, w)$.*

Test 3 Terminal with degree 1 (TD1) - *If $|T| \geq 2$, the edge adjacent to a terminal vertex with degree 1 is choosable.*

Some definitions are needed before we present more sophisticated tests . Let u and v be two distinct vertices in V. Let $\mathcal{P}(u, v)$ denote the set of all paths joining u to v. The standard *distance* between vertices u and v is defined as

$$d(u, v) = \min\{c(P) \mid P \in \mathcal{P}(u, v)\}, \tag{1}$$

where $c(P)$ denotes the sum of the costs of the edges in path P. For $P \in \mathcal{P}(u,v)$, let $T(P)$ be $\{u,v\} \cup (T \cap P)$, i.e. the vertex-set formed by u, v and the terminals in P. Two vertices x and y in $T(P)$ are said to be *consecutive* if the subpath from x to y in P contains no other vertices in $T(P)$. The *Steiner distance* $SD(P)$ is the length of the longest subpath in P joining two consecutive vertices in $T(P)$. The *bottleneck Steiner distance* between vertices u and v is defined as

$$B(u,v) = \min\{SD(P) \mid P \in \mathcal{P}(u,v)\}. \tag{2}$$

The bottleneck Steiner distance without passing through an edge e is defined as

$$B(u,v)^{-e} = \min\{SD(P) \mid P \in \mathcal{P}(u,v);\ e \notin P\}. \tag{3}$$

If e disconnects u and v, $B(u,v)^{-e} = \infty$.

The following three tests were introduced by Duin and Volgenant [6], generalizing some tests which appeared earlier in the literature [2, 3]. We give short proofs of their validity in order to introduce the ideas used in the proofs of new tests.

Test 4 Special distance (SD) - *Let (u,v) be an edge in E. If $B(u,v) < c(u,v)$, then (u,v) is redundant.*

Proof: Suppose that $B(u,v) < c(u,v)$, $P \in \mathcal{P}(u,v)$ is a path such that $SD(P) = B(u,v)$, and $T(P) = \{u,v\} \cup (T \cap P)$. Let R be a MST tree using edge (u,v). Removing (u,v) from R creates two subtrees. Let R_u be the one containing vertex u and R_v be the other, containing v. Pick two consecutive vertices x and y from $T(P)$ such that $x \in R_u$ and $y \in R_v$. Let $P(x,y)$ be the subpath in P from x to y. Since $c(P(x,y)) \leq B(u,v) < c(u,v)$, then $R' = R_u \cup R_v \cup P(x,y)$ is a Steiner tree and $c(R') < c(R)$. This contradiction shows that (u,v) is redundant. ∎

Test 5 Bottleneck degree 3 (BD3) - *Let u be a non-terminal vertex with degree 3, adjacent to vertices v, w, and z. If*

$$\min\{B(v,w) + B(v,z), B(w,v) + B(w,z), B(z,v) + B(z,w)\} \leq c(u,v) + c(u,w) + c(u,z),$$

then there is a MST where the degree of u is at most 2. Therefore, u and its three adjacent edges can be replaced by the following three edges: (v,w) with cost $c(u,v) + c(u,w)$, (v,z) with cost $c(u,v) + c(u,z)$, and (w,z) with cost $c(u,w) + c(u,z)$.

Proof: Without loss of generality, suppose that $B(v,w) + B(v,z) \leq c(u,v) + c(u,w) + c(u,z)$, since the other two cases are similar. Let $P_{vw} \in \mathcal{P}(v,w)$ be a path such that $SD(P_{vw}) = B(v,w)$, $T(P_{vw}) = \{v,w\} \cup (T \cap P_{vw})$, $P_{vz} \in \mathcal{P}(v,z)$ be a path such that $SD(P_{vz}) = B(v,z)$, and $T(P_{vz}) = \{v,z\} \cup (T \cap P_{vz})$. Let R be a MST using vertex u with degree 3. Removing u and its adjacent edges from R, we obtain the subtrees R_v, R_w, and R_z. Pick two consecutive vertices x_1 and y_1 from $T(P_{vw})$, such that $x_1 \in R_w$ and $y_1 \notin R_w$. Now pick two consecutive vertices x_2 and y_2 from $T(P_{vz})$ such that (i) if $y_1 \in R_v$ then $x_2 \in R_z$ and $y_2 \notin R_z$, (ii) if $y_1 \in R_z$ then $x_2 \in R_v$ and $y_2 \notin R_v$. Since $c(P_{vw}(x_1,y_1)) \leq B(v,w)$ and $c(P_{vz}(x_2,y_2)) \leq B(v,z)$, then $c(P_{vw}(x_1,y_1)) + c(P_{vz}(x_2,y_2)) \leq c(u,v) + c(u,w) + c(u,z)$ and $R' = R_v \cup R_w \cup R_z \cup P_{vw}(x_1,y_1) \cup P_{vz}(x_2,y_2)$ is a Steiner tree such that $c(R') \leq c(R)$. Vertex u has degree at most 2 in R'. ∎

Although the BD3 reduction test does not seem to be so advantageous, since the number of edges remains the same, some of the new edges are now quite likely to be eliminated by the SD test. Duin and Volgenant actually introduced a more general BDK test, for non-terminal vertices with degree K. However, the complexity of performing such test grows exponentially with K.

Test 6 Terminal distance (TDist) - *Let $[W, \bar{W}]$ be a partition of the vertices in V such that the subgraphs induced by W and $\bar{W}$ are both connected, with $W \cap T \neq \emptyset$ and $\bar{W} \cap T \neq \emptyset$. Let $\delta(W)$ be the cut induced by this partition. If $\delta(W) = \{e\}$, then e is choosable. If $|\delta(W)| \geq 2$, let $e = \mathrm{argmin}_{e' \in \delta(W)} c(e')$ and $f = \mathrm{argmin}_{f' \in \delta(W) \setminus \{e\}} c(f')$ be respectively a shortest and a second shortest edge in the cut. Suppose $e = (u, v)$ with $u \in W$ and $v \in \bar{W}$. If*

$$\min\{d(u, t_1) \,|\, t_1 \in T \cap W\} + c(e) + \min\{d(v, t_2) \,|\, t_2 \in T \cap \bar{W}\} \;\leq\; c(f),$$

then e is choosable.

Proof: If $\delta(W) = \{e\}$, then the test is trivial. Suppose $|\delta(W)| \geq 2$ and let R be a MST not using a shortest edge e in the cut. There is exactly one path P in R between t_1 and t_2. Pick one edge $g \in (P \cap \delta(W))$, $c(g) \geq c(f)$. Removing g from R creates two subtrees, one containing t_1 and the other containing t_2. Connecting this subtrees by a shortest path from t_1 to t_2 passing through e, we obtain a new Steiner tree R' using e such that $c(R') \leq c(R)$. ■

The above tests are exactly those used in [8]. They are quite effective for many general Steiner instances. For example, they allow the reduction of e18, the hardest instance in the OR-Library, from 62,500 to 5,996 edges. However, they are almost useless for VLSI instances. For example, instance diw0559 is only reduced from 7,013 to 6,883 edges. Some improvement is obtained by using a slight generalization of the SD test suggested in Duin [5]:

Test 7 Special distance with equality (SDE) - *Let (u, v) be an edge in E. If $B(u, v)^{-(u,v)} \leq c(u, v)$, then (u, v) is redundant.*

As an example, we notice that tests 1–3 and 5–7 by themselves are able to reduce instance diw0559 from 7,013 to 2,694 edges, as shown in Figures 5 and 6. In fact, the SDE test combined with the simple tests is always able to remove the portion of the graph outside of a minimal rectilinear convex hull of T not crossing any hole. This reduction may be significant for some instances with no terminals near corners of the grid and few holes. It is the case of diw0559, but not that of taq0903, which is only reduced from 10,490 to 8,485 edges. These reductions are not enough to help substantially in solving those VLSI instances.

There are other known reduction tests, such as the R and CR tests described in Hwang et al. [7], that are based on bounds on the value of an optimal solution. These tests are not likely to be effective on large VLSI instances.

3 New reduction tests

We propose two new reduction tests by generalizing tests SDE and BD3 with the idea of expansion, used by Winter [15] in a test for the rectilinear Steiner tree problem. Both tests are first introduced by examples.

The first example is depicted in Figure 1. Edge (v_1, u_1) can not be removed by the SDE test, as far as $c(v_1, u_1) = 3 < B(v_1, u_1)^{-(v_1,u_1)} = 4$. However, suppose that edge (v_1, u_1) belongs to every MST (Supposition 1). Then, let R be one of these MSTs. Vertex u_1 is a non-terminal and must have degree at least two in R. Now, suppose that (u_1, p) belongs to R (Supposition 2). By removing (v_1, u_1) from R, we obtain two subtrees R_{v_1} and R_{u_1}, such that vertex p belongs to R_{u_1}. Since $B(p, v_1) = 2 < c(u_1, v_1) = 3$, therefore R_{v_1} and R_{u_1} can be reconnected by edge (v_1, t) (if t belongs to R_{u_1}) or edge (t, p) (if t belongs to R_{v_1}), obtaining a new Steiner tree R' such that $c(R') < c(R)$. This contradiction shows that Supposition 2 is false. Therefore, (u_1, u_2) must belong to R. We say that path $\{(v_1, u_1)\}$ was *expanded* to path $\{(v_1, u_1), (u_1, u_2)\}$. However, $B(u_2, v_1)^{-(u_1,v_1)} = 5 \leq c(v_1, u_1) + c(u_1, u_2) = 5$. Therefore, it is possible to obtain a Steiner tree R' not using (v_1, u_1) and such that $c(R') \leq c(R)$. This contradiction shows that Supposition 1 is false. Therefore, (v_1, u_1) is redundant.

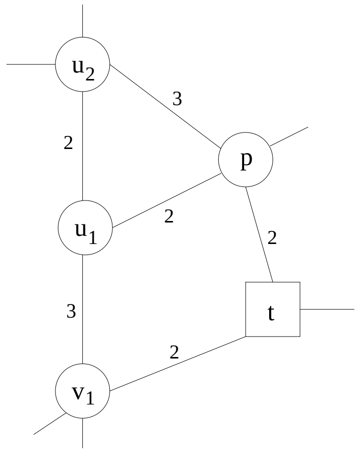

Figure 1: Example of the application of the SDE with expansion test

The sketch in Figure 2 summarizes the pseudo-code of the SDE with expansion test, inspired in this kind of reasoning:

Test 8 SDE with Expansion - *If procedure* `SDE_with_expansion` *applied to edge* $e = (v_1, u_1)$ *returns* success, *then* e *is redundant.*

Proof: Suppose that every MST contains path $P = \{(u_m, u_{m-1}), \ldots, (u_1, v_1), \ldots, (v_{n-1}, v_n)\}$, initially $P = \{(u_1, v_1)\}$. Let R be one of those MSTs. Let v_n be a non-terminal endpoint of P and let $p \notin P$ be a vertex adjacent to v_n. If $B(p, v_n)^{-(u_1,v_1)} \leq c(P)$, then (v_n, p) can not belong to R. Otherwise, by a reasoning similar to the proof of test 4, it would be possible to construct a Steiner tree R' not containing P and such that $c(R') \leq c(R)$. Testing this condition for every possible vertex p, we have one of the following situations, depending on the variable i:

```
Procedure SDE_with_expansion ( Input: edge e = (u_1, v_1) )
    P ⟵ {(u_1, v_1)};
    repeat {
        expanded ⟵ false;
        /* Let P = {(u_m, u_{m-1}), ..., (u_1, v_1), ..., (v_{n-1}, v_n)} be the currrent path */
        if (v_n is non-terminal) {
            i ⟵ 0;
            for each vertex p ∉ P adjacent to v_n
                if (B(p, u_m)^{-(u_1,v_1)} > c(P)) {
                    i ⟵ i + 1;
                    v_{n+1} ⟵ p;
                }
            if (i = 0 ) return success;
            if (i = 1 ) {
                P ⟵ {(u_m, u_{m-1}), ..., (u_1, v_1), ..., (v_{n-1}, v_n), (v_n, v_{n+1})};
                expanded ⟵ true;
            }
        }
        if (u_m is non-terminal) {
            i ⟵ 0;
            for each vertex p ∉ P adjacent to u_m
                if (B(p, v_n)^{-(u_1,v_1)} > c(P)) {
                    i ⟵ i + 1;
                    u_{m+1} ⟵ p;
                }
            if (i = 0 ) return success;
            if (i = 1 ) {
                P ⟵ {(u_{m+1}, u_m), (u_m, u_{m-1}), ..., (u_1, v_1), ..., (v_{n-1}, v_n)};
                expanded ⟵ true;
            }
        }
    } until (expanded = false)
    if (B(u_m, v_n)^{-(u_1,v_1)} ≤ c(P)) return success;
    else return failure;
end SDE_with_expansion
```

Figure 2: Pseudo-code of the SDE with expansion test.

(a) $i = 0$: then vertex v_n has degree 1 in R. This contradiction proves that there is an MST not containing P. The procedure is stopped and the overall conclusion is that edge (u_1, v_1) is redundant.

(b) $i = 1$: then vertex v_n has degree 2 in R, which must also contain edge (v_n, v_{n+1}). We say that P is *expanded* to $\{(u_m, u_{m-1}), \ldots, (u_1, v_1), \ldots, (v_{n-1}, v_n), (v_n, v_{n+1})\}$. The supposition that every MST contains the original path P can only be true if every MST also contains the new expanded path.

(c) $i \geq 2$: no expansion is possible.

The procedure continues, until both endpoints of P are terminals or no more expansions are possible. Finally, a test similar to SDE is applied, considering the whole path P as a single edge. If $B(u_m, v_n)^{-(u_1,v_1)} \leq c(P)$, then it is proved that there is an MST not containing P. The overall conclusion is that edge (u_1, v_1) is redundant. ■

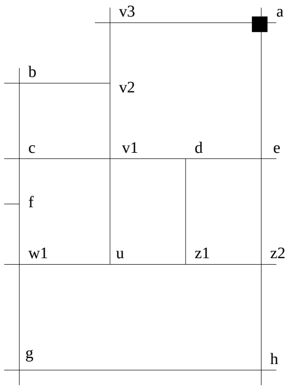

Figure 3: Example of BD3 with Expansion Test

We now consider the example illustrated in Figure 3, showing part of a VLSI-like instance. Edge costs are proportional to the segment lengths. The condition of the BD3 test is not satisfied for vertex u. Suppose that u is used with degree 3 by every MST and let R be such an MST. Since v_1 is a non-terminal vertex, its degree in R is at least 2. Since $d(c, w_1) \leq c(u, v_1)$, edge (v_1, c) does not belong to R. Otherwise, edge (u, v_1) could be replaced by the path between c and w_1 producing a new Steiner tree R', $c(R) = c(R')$, in which u would have degree equal to 2. Since $d(d, z_1) \leq c(u, v_1)$, edge (v_1, d) does not appear in R. Therefore, (v_1, v_2) must belong to R. We say that path $\{(u, v_1)\}$ is *expanded* to path $\{(u, v_1), (v_1, v_2)\}$. Since $d(b, w_1) \leq c(u, v_1) + c(v_1, v_2)$, this path can be further expanded to $\{(u, v_1), (v_1, v_2), (v_2, v_3)\}$. Applying the same reasoning to path $\{(u, z_1)\}$, we may expand it to $\{(u, z_1), (z_1, z_2)\}$, since $d(d, v_1) \leq c(u, z_1)$. We can now apply an improved BD3 test by considering every expanded path as a single edge. Since $B(v_3, z_2) + B(z_2, w_1) \leq c(\{(u, v_1), (v_1, v_2), (v_2, v_3)\}) + c(\{u, w_1\}) + c(\{(u, z_1), (z_1, z_2)\})$, the overall conclusion is that there is an MST in which the degree of u is less than or equal to 2 and the graph transformation originated by the application of BD3 is valid. Vertex u and its adjacent edges could be replaced by the new edges (w_1, z_1), (v_1, z_1) and (v_1, w_1). (Those last two edges would be immediately deleted by the SDE test.)

Before describing the procedure `BD3_with_expansion` which implements this test, we define $B(u, v)^{+(x,y)}$ as the bottleneck distance between vertices u and v in a modified graph, where edge (x, y) has zero cost and both x and y belong to the set of terminal nodes. Then:

$$B(u,v)^{+(x,y)} = \min \left\{ \begin{array}{c} B(u,v), \\ \max\{B(u,x), B(y,v)\}, \max\{B(u,y), B(x,v)\}, \\ \max\{B(u,x), B(x,v)\}, \max\{B(u,y), B(y,v)\} \end{array} \right\} \quad (4)$$

If P is a simple path containing vertices u and v, $P(u, v)$ denotes the subpath in P from u to v.

Test 9 BD3 with Expansion - *If procedure* `BD3_with_expansion` *applied to a vertex u with degree 3 returns* success, *then there exists a MST where u has degree at most 2. Therefore, the transformations of the traditional BD3 test can be applied to u.*

Proof: Suppose that every MST contain paths $P_v = \{(u, v_1), \ldots, (v_{n-1}, v_n)\}$, P_w, and P_z (initially, all these paths contain a single edge). Let R be one of those MSTs. Let v_n be a non-terminal endpoint of P_v and let $p \notin P_v$ be a vertex adjacent to v_n. The key point of this proof is to show that if the expansion condition in Figure 4 holds, then (v_n, p) does not belong to R.

We start by supposing otherwise. If $S = \langle G, T, c\rangle$ is the SPG instance, let $S' = \langle G, T \cup \{v_n, p\}, c'\rangle$ be a modified instance where $c'(v_n, p) = 0$. Tree R is still a MST for S', with cost $c'(R) = c(R) - c(v_n, p)$. For each pair of vertices $x, y \in V$, $B'(x, y) = B(x, y)^{+(v_n,p)}$. Considering paths P_v, $P_w(u, w_j)$, and $P_z(u, z_k)$ as edges and applying a reasoning similar to the proof of test 5, we conclude that there exists a Steiner tree R' with cost $c'(R') \leq c(R) - c(v_n, p)$ in which u has degree at most 2. Then, $R' \cup \{(v_n, p)\}$ is a Steiner tree for S, in which u has degree at most 2 and with cost smaller than or equal to $c(R)$, characterizing a contradiction.

Testing this condition for every possible vertex p, we have one of the following situations, depending to variable i:

(a) $i = 0$: then vertex v_n has degree 1 in R. This contradiction proves that there is an MST where u has degree at most 2. The procedure is stopped.

(b) $i = 1$: then vertex v_n has degree 2 in R, which must also contain edge (v_n, v_{n+1}). We say that path P_v was *expanded* to $\{(u, v_1), \ldots, (v_{n-1}, v_n), (v_n, v_{n+1})\}$. The supposition that every MST contains the paths P_v, P_w, and P_z can only be true if every MST also contains the new expanded path.

(c) $i \geq 2$: no expansion is possible.

This procedure is applied to paths P_v, P_w, and P_z, until all endpoints are terminals or no more expansions are possible. Finally, a test similar to BD3 is applied, considering the whole paths P_v, P_w, and P_z as edges. ■

The BD3 with expansion test has strong similarities with respect to the Star test proposed by Winter [15] for the rectilinear Steiner tree problem. This test actually resorts to geometrical properties of the latter. Adapting it to a pure SPG context, it would be equivalent to a BD3 with expansion procedure, where the if statement labeled as *expansion conditions* is replaced by a less general condition:

if (there exists a vertex w_j in P_w such that $d(p, w_j) \leq \max\{c(P_v), c(P_w(u, w_j))\}$ or there exists a vertex z_j in P_z such that $d(p, z_j) \leq \max\{c(P_v), c(P_z(u, z_j))\}$).

```
Procedure BD3_with_expansion ( Input: vertex u adjacent to v_1, w_1, and z_1 )
    P_v ⟵ {(u, v_1)};
    P_w ⟵ {(u, w_1)};
    P_z ⟵ {(u, z_1)};
    repeat {
        expanded ⟵ false;
        /* Try to expand the current P_v = {(u, v_1), ..., (v_{n-1}, v_n)} path */
        if (v_n is non-terminal) {
            i ⟵ 0;
            for each vertex p ∉ P_v adjacent to v_n
                /* expansion conditions */
                if (there exists vertices w_j ∈ P_w and z_k ∈ P_z such that
```

$$\min \left\{ \begin{array}{l} B(v_n, w_j)^{+(v_n,p)} + B(v_n, z_k)^{+(v_n,p)}, \\ B(w_j, v_n)^{+(v_n,p)} + B(w_j, z_k)^{+(v_n,p)}, \\ B(z_k, v_n)^{+(v_n,p)} + B(z_k, w_j)^{+(v_n,p)} \end{array} \right\} \leq c(P_v) + c(P_w(u, w_j)) + c(P_z(u, z_k));$$

```
                ) {
                    i ⟵ i + 1;
                    v_{n+1} ⟵ p;
                }
            if (i = 0 ) return success;
            if (i = 1 ) {
                P_v ⟵ {(u, v_1), ..., (v_{n-1}, v_n), (v_n, v_{n+1})};
                expanded ⟵ true;
            }
        }

        Try to expand the current P_w path;

        Try to expand the current P_z path;

    } until (expanded = false)
    if (BD3 Test conditions considering P_v, P_w, and P_z as edges) return success;
    else return failure;
end BD3_with_expansion
```

Figure 4: Pseudo-code of the BD3 with expansion test.

4 Preprocessing and implementation strategies

At a given stage of the preprocessing, many possible choices of valid reductions are usually available. The size of the final reduced instance obtained, when no further reductions are possible, strongly depends on those choices. After some preliminary computational experiences, we decided to divide the preprocessing into two phases. Both phases use tests 1-3, 6 and 8 in any order and the first valid reduction found is always performed.

In the first phase, the transformation derived from the application of the BD3 with expansion test (test 9) to a certain vertex is only performed if at least two out of the three newly created edges can be immediately deleted by the SDE with expansion test. In this case, the overall transformation amounts to deleting an edge incident to u and applying a NTD2 test. In the second phase, there are no restrictions to the application of the BD3 with expansion test.

The point is that phase I preserves the structure of the grid graph with holes (except for a few edge contractions due to the TDist test). The degree of almost all vertices remain at most 4. Low vertex degrees seems to be helpful for the success of the expansion and the important BD3 test is only applied to vertices with degree 3. On the other hand, if phase II is applied directly to the original instance, the degree of many vertices may quickly increase, making further reductions impossible.

The simple tests NTD1, NTD2, and TD1 are computationally cheap to perform. Test TDist is also reasonably cheap. Let $T = (V, E')$ be any minimum cost spanning tree of the current graph G. Each edge $e \in E'$ induces a partition of V corresponding to the two connected components of $(V, E' \setminus \{e\})$. Duin [5] showed that only these $|V| - 1$ partitions need to to be considered in the TDist test. This leads to an $O(|V|^2)$ algorithm to identify all edges fulfilling the TDist condition.

The cost of performing tests SD and BD3 lies in the computation of bottleneck distances. This is still true even when expansion is applied. The standard approach suggested by Duin and Volgenant is to construct a table with all pairs bottleneck distances on the original graph, which can be done in $O(|V|^3)$ time. Any sequence of reductions from NTD1, NTD2, SD, or BD3 does not affect the bottleneck distances on the remaining vertices. On the other hand, edge contractions derived from the application of tests TD1 or TDist may reduce bottleneck distances. In this case, that table will then give upper bounds for the current bottleneck distances and lead to valid reductions. When no more reductions can be found with the current table, it may be advantageous to update the latter as another attempt to identify reductions. If the $O(|V|^3)$ time for the computation of all exact bottleneck distances too costly, then one may resort to one of the many cheaper approximations giving upper bounds of B. In this case some valid reductions may be missed.

Keeping a full table of exact or approximate bottleneck distances is not convenient for the VLSI instances. First, because grid graphs with holes are very sparse, with $|V| > |E|/2$. Therefore, the processing time and even the memory requirements would be huge for the bigger instances. For example, in the case of instance `alue7080` with 34,479 vertices, this distance table would have 594 million entries. In contrast, instance `e18` has about the same number of edges as `alue7080`, but only 2,500 vertices. Second, the quite regular cost structure make it important to use the SDE test, which requires the B^{-e} distances. It is not only hard to construct this table, but also the reductions resulting from SDE tests (when equality holds) may increase the bottleneck distances for the remaining vertices [5].

The tables would then have to be updated after each application of an SDE test, which would be far too expensive.

To reduce the computational cost, our approach consists in computing approximate bottleneck distances on-the-fly, as they are needed by the tests. We approximate $B(u,v)$ (and, similarly, B^{-e} and B^{+e}) by considering at most two terminals of T in the bottleneck path:

$$\tilde{B}(u,v) = \min\{d(u,v), \max_{x\in T}\{d(u,x), d(x,v)\}, \max_{x,y\in T}\{d(u,x), d(x,y), d(y,v)\}\}. \qquad (5)$$

These repeated calculations are somehow expensive, but not prohibitive. This, because the SDE and BD3 tests are quite local, even with expansions. In most of the time we compute distances between close vertices and only a small subgraph of G has to be considered. Our computational experiments indicated that the gain in reduction rates observed for the VLSI instances with the use of the actual $B(u,v)$ distances is marginal, except perhaps for terminal-dense instances like alue7080. The corresponding increase in the computational cost does not seem to pay off.

5 Computational results and concluding remarks

The above preprocessing strategy, with all reduction tests, was implemented in C++ and applied to all VLSI instances in the SteinLib. To further evaluate the effectiveness of the proposed preprocessing procedure, we also implemented a branch-and-cut algorithm to exactly solve the reduced instances. A detailed description of our implementation is out of the scope of this paper. We basically followed the scheme described in Koch and Martin [8]. The branch-and-cut procedure was also coded in C++, using the ABACUS framework [13] and the CPLEX 3.0 linear programming solver.

Computational results on a Sun ULTRA 167 MHz workstation are reported on Tables 1 to 4. Each row in these tables summarizes the results for the instance named in its first column. The next three columns give the original dimensions of this instance (number of nodes, edges, and terminals), followed by three columns with the reduced dimensions after the two preprocessing phases (once again, the number of nodes, edges, and terminals). The next two columns report the rounded up computation times in seconds consumed by each of the preprocessing (PTime) and branch-and-cut (BTime) procedures. The next two columns give the value of the best solution found by our branch-and-cut procedure (UPR) and the previous best known solution value (KM), found by [8]. Numbers in boldface indicate proven optimal values. Finally, the following branch-and-cut statistics are presented: the number of branch-and-bound nodes (nodes), the number of linear problems solved (LPs), and the number of cuts introduced (cuts). A "—" appears in the appropriate entries for the instances solved to optimality by preprocessing itself, without the application of branch-and-cut.

The observed reduction rates are significant. The preprocessing procedure by itself was able to solve 43 out of the 116 instances. Among them, we find nine instances with more than 2,000 edges and, in particular, the formerly open instance dmxa1010. Another 48 instances were reduced to less than 10% of their original size (measured in terms of the number of edges). All the remaining instances were reduced to less than 32% of their original size, except alue7065 which was reduced to 38%. These reductions were enough to bring almost all instances to the reach of exact solvability by the branch-and-cut procedure, except those four instances with more than 54,000 edges. Some of these results are graphically illustrated in Figures 7 and 9 for two different instances. The reduced instance obtained from diw0559 after the first phase is shown in Figure 7. This instance is further reduced after the second phase to only 164 edges. Figure 9 illustrates the case of instance taq0903, which shows a more modest reduction after the first phase. This instance is further reduced to 3,034 edges at the end of the second phase.

The time spent in the reductions was somehow expensive for the larger instances. For example, preprocessing instance alue7080 took more than three hours. However, for the larger instances, these preprocessing times were one order of magnitude smaller than those needed to solve the reduced instances by branch-and-cut. This is a clear indication that preprocessing pays off, at least if exact optimal solutions are desired.

Although created for VLSI instances, the new tests with expansion can also be effective for other kinds of *sparse* SPG instances. For example, instance e20 from the OR-Library has 2,500 vertices and 62,500 edges. Tests 1–7 are able to reduce it to 1,257 vertices and 2,330 edges. Applying the new tests from this point leads to exactly solving this instance.

The tests with expansion have a distinct philosophy, with respect to the traditional ones. Instead of procedures to check if a certain predefined condition holds, they may be seen as "on-line theorem proving procedures". The idea of expansion can be viewed as a relatively cheap way to construct a chain of logical implications, trying to prove that an initial hypothesis leads to a contradiction. The procedures given in this paper do not exhaust this idea and could be strengthened by using other kinds of logical implications.

Notes:

[1] On October 11, 2000 we received from T. Polzin and S. Vahdati an yet unpublished article [11], describing a branch-and-cut algorithm that obtained good results on VLSI instances and claiming to having solved in 1998 those same 28 formerly open instances. We also note that, by that time, an improved branch-and-cut described in [12] had already solved (and, to the best of our knowledge, for the first time) instances alue7065 (optimal value 23881), alue7080 (optimal value 62449), alut2610 (optimal value 12239), and alut2625 (optimal value 35459). The preprocessing procedure described in this paper was essential to this achievement. Other recent results on VLSI instances appear in Bahiense et al. [1].

References

[1] L. Bahiense, F. Barahona and O. Porto, "Solving Steiner tree problems in graphs with Lagrangean relaxation", IBM Report RC21846, Yorktown Heights, 2000.

[2] A. Balakrishnan and N. Patel, "Problem reduction methods and a tree generation algorithm for the Steiner network problem", *Networks* 17 (1987), 65–85.

[3] J. Beasley, "An algorithm for the Steiner problem in graphs", *Networks* 14 (1984), 147–159.

[4] J. Beasley, "OR-Library: Distributing test problems by electronic mail", *Journal of the Operational Research Society* 41 (1990), 1069–1072 (on line document at `http://mscmga.ms.ic.ac.uk/info.html`).

[5] C. Duin, *Steiner's problem in graphs*, PhD thesis, University of Amsterdam, 1993.

[6] C. Duin and A. Volgenant, “Reduction tests for the Steiner problem in graphs”, *Networks* 19 (1989), 549–567.

[7] F. Hwang, D. Richards, and P. Winter, *The Steiner tree problem*, North-Holland, 1992.

[8] T. Koch and A. Martin, “Solving Steiner tree problems in graphs to optimality”, *Networks* 32 (1998), 207–232.

[9] T. Koch, A. Martin, and S. Voß, “SteinLib: An updated library on Steiner tree problems in graphs”, Konrad-Zuse-Zentrum für Informationstechnik Berlin, ZIB-Report 00-37, 2000 (online document at `http://elib.zib.de/steinlib`).

[10] N. Maculan, “The Steiner problem in graphs”, *Annals of Discrete Mathematics* 31 (1987), 185–212.

[11] T. Polzin and S. Vahdati, “Improved algorithms for the Steiner problem in networks”, *Discrete Applied Mathematics* 112 (2001), 263–300.

[12] M. Poggi de Aragão, E. Uchoa, and R.F. Werneck, “Dual heuristics on the exact solution of large Steiner problems”, *Electronic Notes in Discrete Mathematics* 7 (2001), Elsevier (online document at `http://www.elsevier.nl/locate/endm`).

[13] S. Thienel, *ABACUS - A Branch-And-Cut System*, PhD thesis, Universität zu Köln, 1995.

[14] P. Winter, “Steiner problems in networks: A survey”, *Networks* 17 (1987), 129–167.

[15] P. Winter, “Reductions for the rectlinear Steiner tree problem”, *Networks* 26 (1995), 187–198.

Table 1: Runs for series alue, alut and diw.

	Original size			Preprocessed size			Times		Solutions		B&C statistics		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	UPR	KM	nodes	LPs	cuts
alue2087	1244	1971	34	40	65	13	2	1	**1049**	**1049**	1	5	65
alue2105	1220	1858	34	1	0	1	1	—	**1032**	**1032**	—	—	—
alue3146	3626	5869	64	65	104	26	15	5	**2240**	2240	3	20	253
alue5067	3524	5560	68	300	504	38	77	54	**2586**	**2586**	1	20	1057
alue5345	5179	8165	68	1100	1860	61	109	2930	**3507**	3560	13	103	16094
alue5623	4472	6938	68	768	1307	57	52	1703	**3413**	3463	7	102	12941
alue5901	11543	18429	68	1066	1821	59	81	1286	**3912**	3994	1	51	9910
alue6179	3372	5213	67	441	752	49	49	112	**2452**	**2452**	5	28	1062
alue6457	3932	6137	68	793	1356	53	93	552	**3057**	3062	1	37	3231
alue6735	4119	6696	68	340	556	50	46	83	**2696**	**2696**	1	23	1416
alue6951	2818	4419	67	257	413	46	35	58	**2386**	**2386**	1	24	1130
alue7065	34046	54841	544	11911	20956	426	7581	not solved	not solved	24827			
alue7066	6405	10454	16	1768	3109	9	300	9571	**2256**	2275	1	95	6001
alue7080	34479	55494	2344	8355	14387	1328	11222	not solved	not solved	not solved			
alue7229	940	1474	34	1	0	1	1	—	**824**	**824**	—	—	—
alut0787	1160	2089	34	9	13	5	1	1	**982**	**982**	1	6	15
alut0805	966	1666	34	34	54	16	2	1	**958**	**958**	1	9	78
alut1181	3041	5693	64	207	369	39	26	56	**2353**	2390	1	31	1102
alut2010	6104	11011	68	374	641	45	74	69	**3307**	3322	1	20	1131
alut2288	9070	16595	68	1195	2136	60	269	1190	**3843**	3889	3	33	3733
alut2566	5021	9055	68	701	1223	62	63	813	**3073**	3127	3	59	4415
alut2610	33901	62816	204	10576	19829	182	6990	not solved	not solved	12760			
alut2625	36711	68117	879	11759	21628	763	8832	not solved	not solved	36763			
alut2764	387	626	34	1	0	1	1	—	**640**	**640**	—	—	—
diw0234	5349	10086	25	760	1413	21	221	69	**1996**	**1996**	1	15	641
diw0250	353	608	11	1	0	1	1	—	**350**	**350**	—	—	—
diw0260	539	985	12	1	0	1	1	—	**468**	**468**	—	—	—
diw0313	468	822	14	1	0	1	1	—	**397**	**397**	—	—	—
diw0393	212	381	11	1	0	1	1	—	**302**	**302**	—	—	—

Table 2: Runs for series diw (continued) and dmxa.

	Original size			Preprocessed size			Times		Solutions		B&C statistics		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	UPR	KM	nodes	LPs	cuts
diw0445	1804	3311	33	27	42	10	5	1	**1363**	**1363**	1	12	78
diw0459	3636	6789	25	1	0	1	2	—	**1362**	**1362**	—	—	—
diw0460	339	579	13	1	0	1	1	—	**345**	**345**	—	—	—
diw0473	2213	4135	25	14	22	6	2	1	**1098**	**1098**	1	9	33
diw0487	2414	4386	25	4	5	3	4	1	**1424**	**1424**	1	2	3
diw0495	938	1655	10	1	0	1	1	—	**616**	**616**	—	—	—
diw0513	918	1684	10	1	0	1	1	—	**604**	**604**	—	—	—
diw0523	1080	2015	10	1	0	1	1	—	**561**	**561**	—	—	—
diw0540	286	465	10	1	0	1	1	—	**374**	**374**	—	—	—
diw0559	3738	7013	18	83	141	11	17	6	**1570**	1570	1	30	394
diw0778	7231	13727	24	187	337	15	40	19	**2173**	2173	1	23	601
diw0779	11821	22516	50	2377	4490	39	1390	34077	**4440**	4566	3	105	11022
diw0795	3221	5938	10	225	400	10	104	19	**1550**	1553	1	24	624
diw0801	3023	5575	10	334	605	10	155	79	**1587**	1587	1	35	1303
diw0819	10553	20066	32	1394	2604	26	803	2956	**3399**	3430	1	51	4451
diw0820	11749	22384	37	1816	3407	32	1756	34190	**4167**	4259	1	191	15773
dmxa0296	233	386	12	1	0	1	1	—	**344**	**344**	—	—	—
dmxa0368	2050	3676	18	47	76	9	2	1	**1017**	**1017**	1	20	182
dmxa0454	1848	3286	16	1	0	1	1	—	**914**	**914**	—	—	—
dmxa0628	169	280	10	1	0	1	1	—	**275**	**275**	—	—	—
dmxa0734	663	1154	11	1	0	1	1	—	**506**	**506**	—	—	—
dmxa0848	499	861	16	34	54	11	1	1	**594**	**594**	1	8	78
dmxa0903	632	1087	10	53	90	7	3	1	**580**	**580**	1	21	208
dmxa1010	3983	7108	23	1	0	1	3	—	**1488**	1488	—	—	—
dmxa1109	343	559	17	9	13	5	1	1	**454**	**454**	1	3	11
dmxa1200	770	1383	21	29	42	13	1	1	**750**	**750**	1	7	55
dmxa1304	298	503	10	1	0	1	1	—	**311**	**311**	—	—	—
dmxa1516	720	1269	11	1	0	1	1	—	**508**	**508**	—	—	—
dmxa1721	1005	1731	18	4	5	3	1	1	**780**	**780**	1	2	2
dmxa1801	2333	4137	17	310	553	17	40	80	**1365**	**1365**	1	42	1328

Table 3: Runs for series gap and msm.

Instance	Original size			Preprocessed size			Times		Solutions		B&C statistics		
	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	UPR	KM	nodes	LPs	cuts
gap1307	342	552	17	1	0	1	1	—	**549**	**549**	—	—	—
gap1413	541	906	10	1	0	1	1	—	**457**	**457**	—	—	—
gap1500	220	374	17	1	0	1	1	—	**254**	**254**	—	—	—
gap1810	429	702	17	1	0	1	1	—	**482**	**482**	—	—	—
gap1904	735	1256	21	7	9	4	4	1	**763**	**763**	1	2	6
gap2007	2039	3548	17	41	70	9	2	1	**1104**	**1104**	3	19	153
gap2119	1724	2975	29	1	0	1	3	—	**1244**	**1244**	—	—	—
gap2740	1196	2084	14	13	19	5	1	1	**745**	**745**	1	10	36
gap2800	386	653	12	1	0	1	1	—	**386**	**386**	—	—	—
gap2975	179	293	10	1	0	1	1	—	**245**	**245**	—	—	—
gap3036	346	583	13	28	42	9	1	1	**457**	**457**	1	12	101
gap3100	921	1558	11	16	25	8	1	1	**640**	**640**	1	5	31
gap3128	10393	18043	104	2146	3903	60	769	1366	**4292**	4315	1	26	2632
msm0580	338	541	11	7	9	4	1	1	**467**	**467**	1	3	7
msm0654	1290	2270	10	1	0	1	1	—	**823**	**823**	—	—	—
msm0709	1442	2403	16	1	0	1	2	—	**884**	**884**	—	—	—
msm0920	752	1264	26	7	9	4	3	1	**806**	**806**	1	4	9
msm1008	402	695	11	13	18	6	1	1	**494**	**494**	1	7	35
msm1234	933	1632	13	7	9	4	1	1	**550**	**550**	1	3	8
msm1477	1199	2078	31	12	16	6	2	1	**1068**	**1068**	1	7	25
msm1707	278	478	11	1	0	1	1	—	**564**	**564**	—	—	—
msm1844	90	135	10	6	8	4	1	1	**188**	**188**	1	4	9
msm1931	875	1522	10	4	5	3	1	1	**604**	**604**	1	2	3
msm2000	898	1562	10	1	0	1	1	—	**594**	**594**	—	—	—
msm2152	2132	3702	37	191	333	24	13	21	**1590**	**1590**	1 v	20	607
msm2326	418	723	14	9	12	5	1	1	**399**	**399**	1	2	7
msm2492	4045	7094	12	39	61	9	6	1	**1459**	1459	3	13	108
msm2525	3031	5239	12	11	15	6	3	1	**1290**	**1290**	1	5	16
msm2601	2961	5100	16	178	305	12	27	10	**1440**	1440	1	18	533

Table 4: Runs for series msm (continued) and taq.

	Original size			Preprocessed size			Times		Solutions		B&C statistics		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	UPR	KM	nodes	LPs	cuts
msm2705	1359	2458	13	4	5	3	1	1	**714**	**714**	1	2	3
msm2802	1709	2963	18	7	11	4	2	1	**926**	**926**	1	5	12
msm2846	3263	5783	89	347	595	58	49	226	**3135**	3141	3	48	2492
msm3277	1704	2991	12	1	0	1	1	—	**869**	**869**	—	—	—
msm3676	957	1554	10	1	0	1	1	—	**607**	**607**	—	—	—
msm3727	4640	8255	21	1	0	1	11	—	**1376**	**1376**	—	—	—
msm3829	4221	7255	12	338	594	10	52	35	**1571**	1571	1	32	599
msm4038	237	390	11	1	0	1	1	—	**353**	**353**	—	—	—
msm4114	402	690	16	1	0	1	1	—	**393**	**393**	—	—	—
msm4190	391	666	16	1	0	1	1	—	**381**	**381**	—	—	—
msm4224	191	302	11	1	0	1	1	—	**311**	**311**	—	—	—
msm4312	5181	8893	10	1299	2355	10	199	1299	**2016**	2049	1	47	2756
msm4414	317	476	11	1	0	1	1	—	**408**	**408**	—	—	—
msm4515	777	1358	13	35	55	8	1	1	**630**	**630**	1	15	109
taq0014	6466	11046	128	1717	3064	82	290	10507	**5326**	5442	1	77	11423
taq0023	572	963	11	37	63	7	1	1	**621**	**621**	1	17	131
taq0365	4186	7074	22	984	1771	21	77	696	**1914**	1914	1	36	2199
taq0377	6836	11715	136	2020	3577	115	319	27365	**6393**	6565	3	141	25856
taq0431	1128	1905	13	108	188	10	14	5	**897**	**897**	1	24	253
taq0631	609	932	10	8	11	4	1	1	**581**	**581**	1	6	17
taq0739	837	1438	16	64	105	12	6	3	**848**	**848**	3	23	236
taq0741	712	1217	16	84	140	14	7	5	**847**	**847**	1	21	374
taq0751	1051	1791	16	104	178	14	7	4	**939**	**939**	1	15	259
taq0891	331	560	10	1	0	1	1	—	**319**	**319**	—	—	—
taq0903	6163	10490	130	1704	3034	90	244	14124	**5099**	5162	3	102	13833
taq0910	310	514	17	1	0	1	1	—	**370**	**370**	—	—	—
taq0920	122	194	17	1	0	1	1	—	**210**	**210**	—	—	—
taq0978	777	1239	10	1	0	1	1	—	**566**	**566**	—	—	—

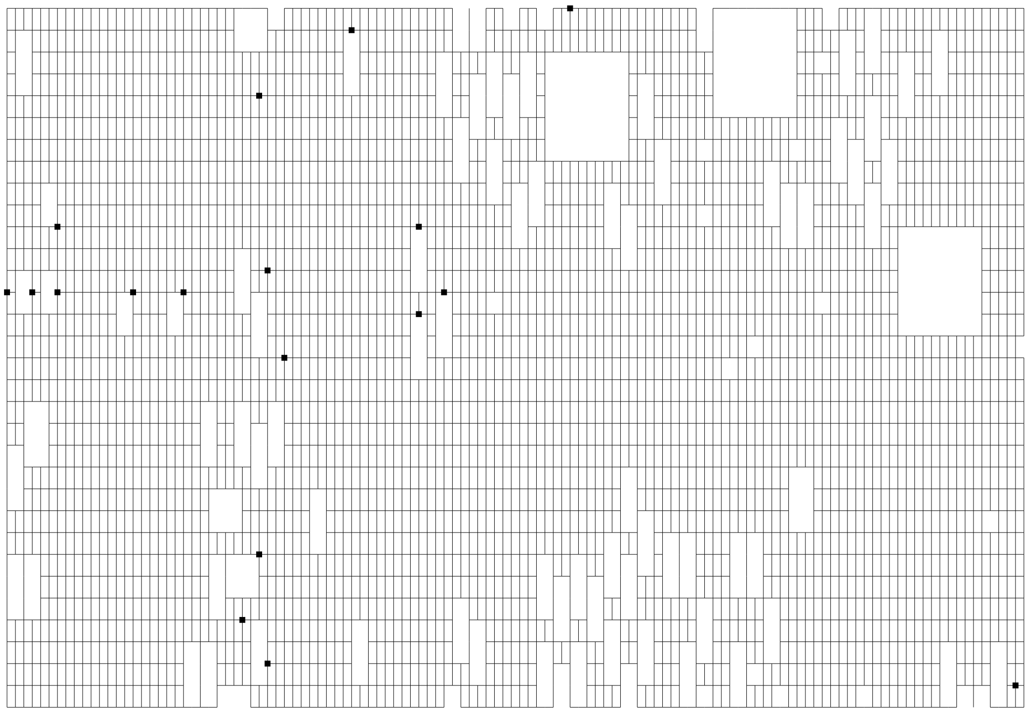

Figure 5: Original instance diw0559: $|V| = 3738$, $|E| = 7013$, and $|T| = 18$.

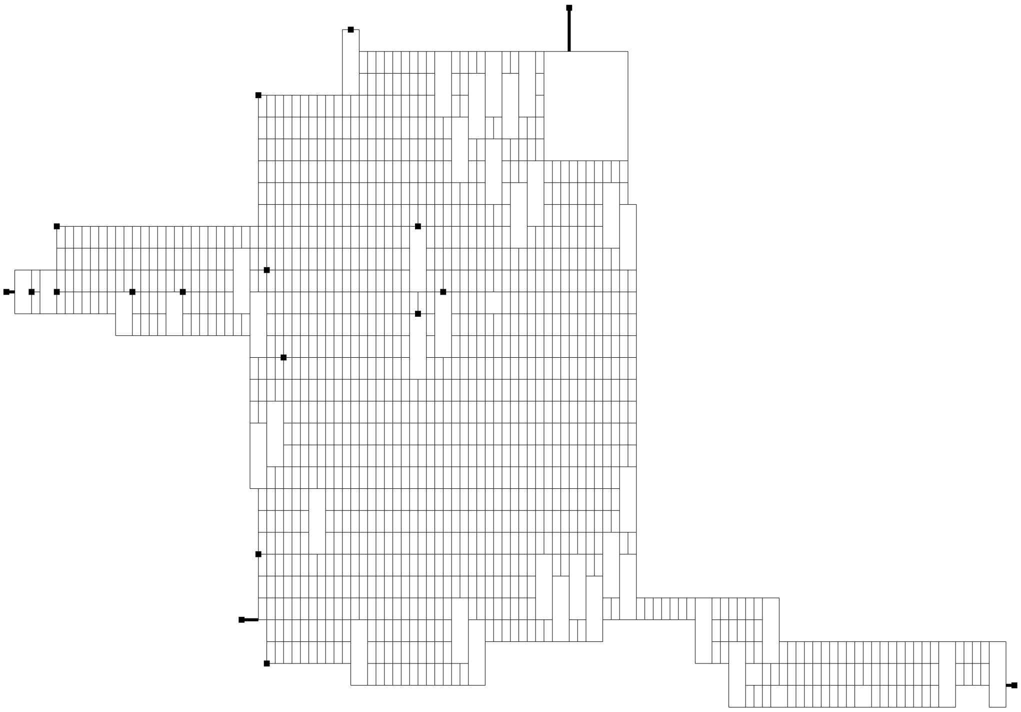

Figure 6: Reduced instance diw0559 after tests 1-3 and 5-7: $|V| = 1452$, $|E| = 2694$, and $|T| = 18$.

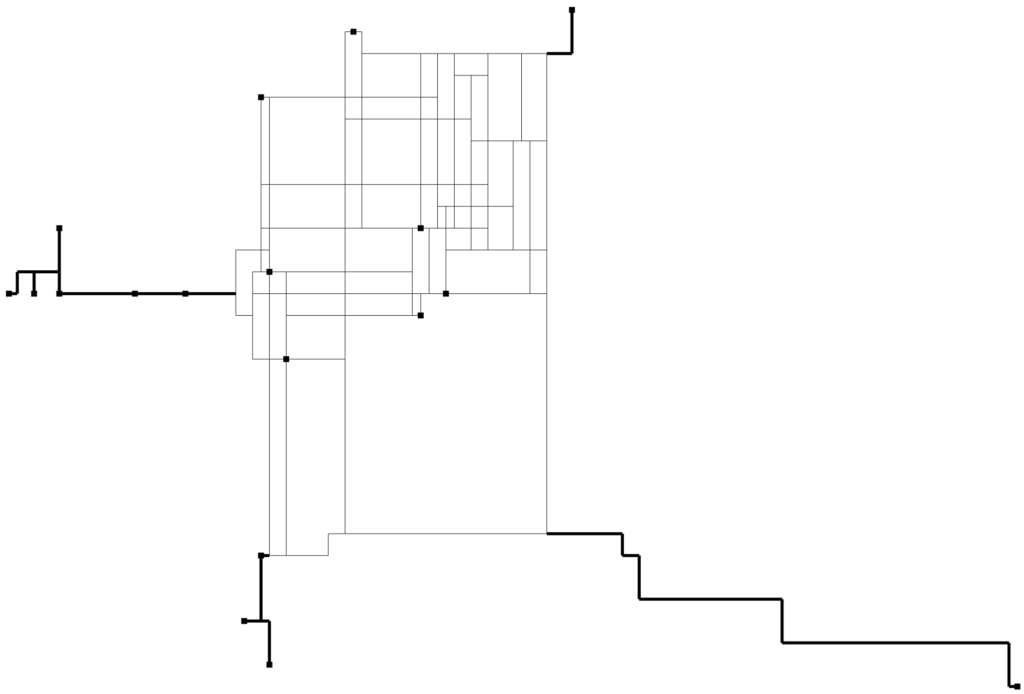

Figure 7: Instance diw0559 after the first preprocessing phase: $|V| = 92$, $|E| = 151$, and $|T| = 11$.

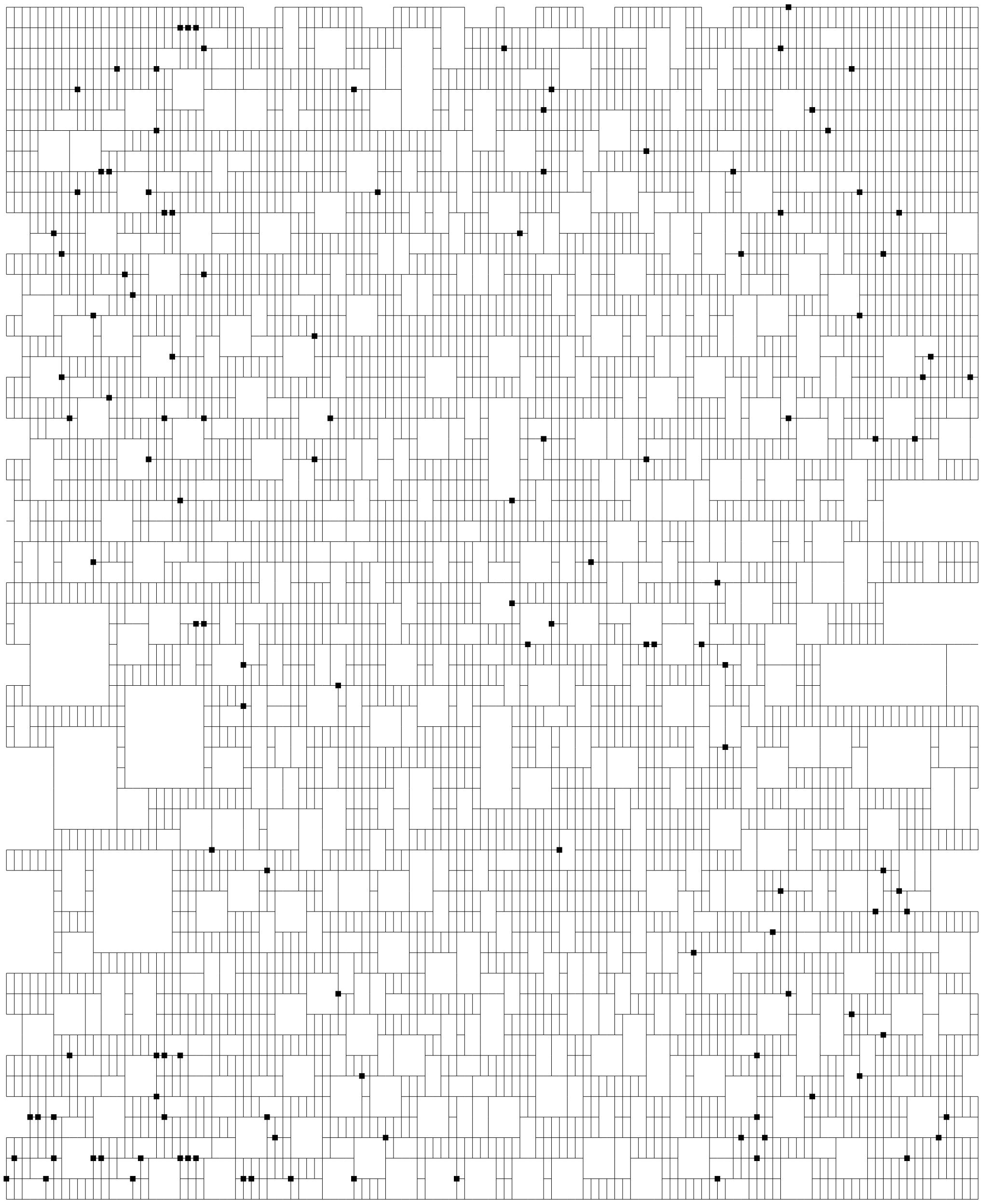

Figure 8: Original instance taq0903: $|V| = 6163$, $|E| = 10,490$, and $|T| = 130$.

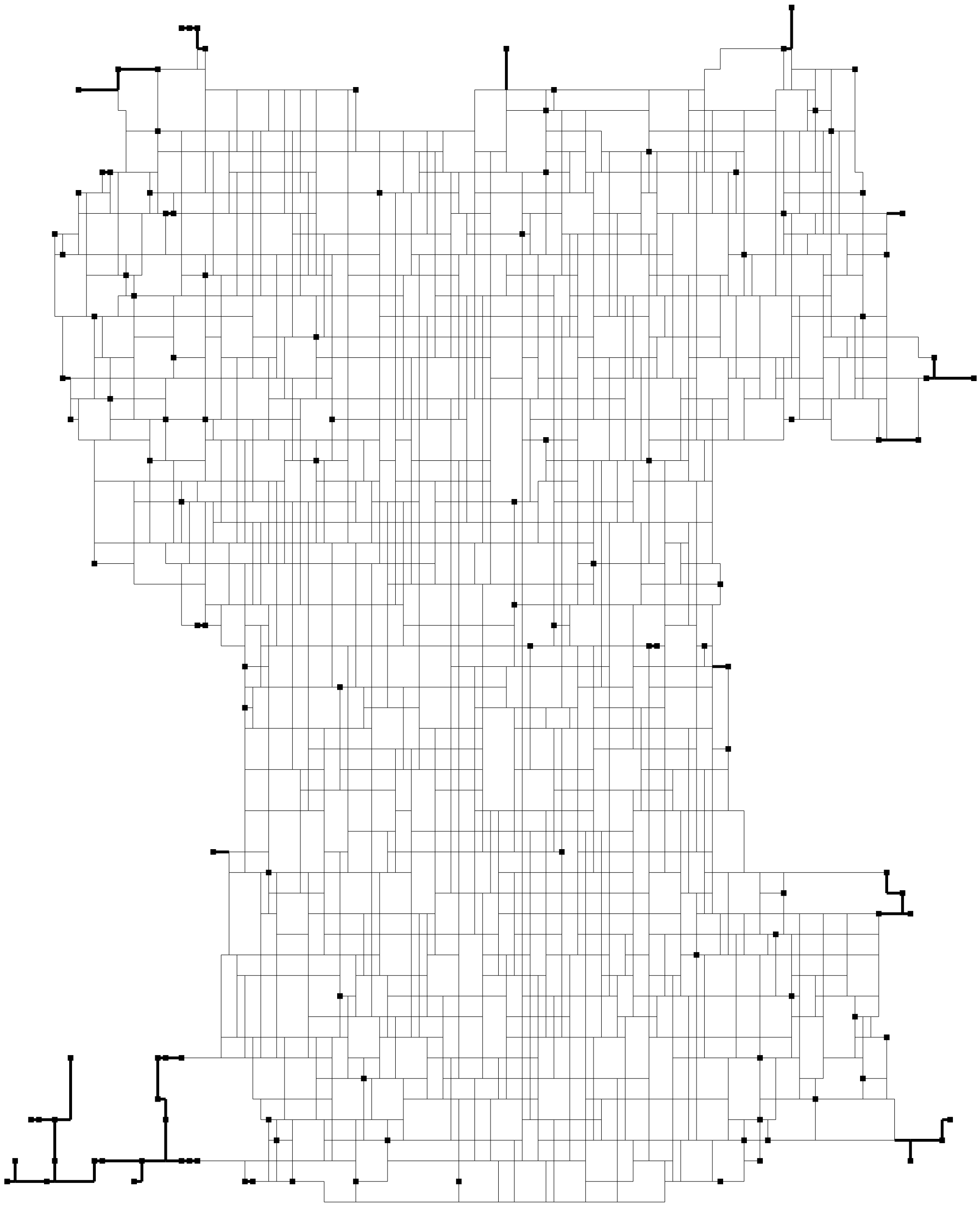

Figure 9: Instance taq0903 after the first preprocessing phase: $|V| = 1995$, $|E| = 3387$, and $|T| = 92$.