Preprocessing Steiner Problems from VLSI Layout

Eduardo Uchoa
Marcus Poggi de Aragão
Celso Ribeiro

PUC-Rio.Inf.MCC 32/99 December, 1999

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, combining earlier ideas from the literature to make them effective for VLSI problems. Testing this procedure over the 116 instances of the SteinLib, we obtained significant reductions within reasonable computational times. 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.*

Keywords: Steiner Problem, Preprocessing, VLSI Design.

Resumo: O projeto de circuitos VLSI demanda soluções de instâncias do problema de Steiner em grafos. Essas instâncias tem como característica a estrutura de grade bi-dimensional com buracos e são consideradas de difícil resolução pelos métodos conhecidos. Em particular, as técnicas de pré-processamento desenvolvidas pra problemas de Steiner em grafos gerais não funcionam bem. Este trabalho apresenta um novo procedimento de pré-processamento, adaptando testes de redução já conhecidos na literatura tornando-os efetivos para as instâncias de VLSI. O procedimento foi aplicado sobre 116 instâncias de VLSI da SteinLib, disponíveis na rede. Foram obtidas reduções bastante significativas, permitindo que um algoritmo de *branch-and-cut* resolvesse 28 das 32 instâncias de VLSI da SteinLib até então em em aberto.

Palavras-chave: Problema de Steiner, Pré-processamento, Projeto de VLSI.

1 Introduction

The *Steiner Tree Problem in Graphs* (STPG) consists in given a connected graph $G = (V, E)$, non-negative costs $c(e)$ (also denoted $c(u, v)$ when $e = (u, v)$) associated to the edges of E and a set $T \subseteq V$ of *terminal vertices*, find a *Minimum Cost Steiner Tree* (MCST), i.e. a subtree (V', E') of G with $T \subseteq V'$ minimizing $\sum_{e \in E'} c(e)$. The STPG is one of the most widely studied NP-hard problems.

Koch and Martin [3] presented an efficient implementation of an exact algorithm for STPG, consisting of preprocessing followed by a branch-and-cut. It could solve most of the instances then available in the literature in reasonable times, including all the classical problems in the OR-Library (available at ftp://mscmga.ms.ic.ac.uk/pub). They also introduced a new set of STPG instances extracted from real-world problems arising from VLSI layout. The basic characteristic of these instances is their support graph G: rectangular grids with many irregularly placed holes. Figures 2 and 5 show instances *diw*0559 and *taq*0903, where line segments are edges, crossings are vertices, terminals are indicated by filled squares. Costs are proportional to edge lengths. A number of 116 such VLSI instances, classified in seven series, are available in the so called SteinLib (at ftp://ftp.zib.de/pub/mp-testdata/steinlib).

However, Koch and Martin's code was not so successful in solving these new instances. The applied reduction tests, 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 structure, there are many distinct solutions with the same cost. This kind of symmetry damps branch-and-cut performance, because it leads to highly degenerate linear programs.

We addressed the first difficulty. This is done by building a powerful preprocessing procedure, enhancing some reduction tests from Duin and Volgenant [2] with the idea of expansion introduced by Winter [6] (for the Rectlinear Steiner Tree Problem). The reductions obtained 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 4 really large instances, with more than 55,000 edges, remain open.

2 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 MCST containing e and *redundant* if there is at least one MCST 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 vertices u and v may be contracted. A redundant edge is simply deleted. In either case, the size of the problem is reduced. There are also reduction tests leading to more

complex graph transformations. Reduction tests may be applied again and again over the reduced graphs until no more reduction is found. Some very simple tests are:

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

Test 2 Non-Terminal of degree 2 (NTD2) - *A non-terminal vertex u of degree 2 and its 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 of degree 1 (TD1) - *The edge adjacent to a terminal vertex of degree 1 is choosable.*

Before we present more sophisticated tests some definitions are needed. 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 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. The *Steiner distance* $SD(P)$ between u and v along P is the length of the longest subpath in P joining two vertices in $T(P)$, consecutives in P. The *bottleneck Steiner distance* between u and v is defined as

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

A nice interpretation for bottleneck distances is to consider G as a road system, $\{u, v\} \cup T$ as petrol stations and a driver who wants to go from u to v. Then $B(u, v)$ is the minimum distance he must be able to drive without refilling in order to reach his destination.

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 [2], generalizing several tests which appeared earlier in the literature.

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.*

Test 5 Bottleneck Degree 3 (BD3) - *Let u be a non-terminal vertex of 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 MCST where the degree of u is at most 2. Therefore u and its three adjacent edges can be replaced by 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 $c(u,w)+c(u,z)$.

Although this reduction 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.

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 $e = \text{argmin}_{e' \in \delta(W)} c(e')$ and $f = \text{argmin}_{f' \in \delta(W) \setminus \{e\}} c(f')$ be respectively a shortest and 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.

The TD1 Test is a special case of this one. The tests presented so far are exactly those used by [3]. They are quite efficient for many general Steiner instances. For example, in the case of $e18$, the hardest instance in the OR-Library, they allow its reduction 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 [1]:

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.*

Tests 1-3, 5-7 reduces $diw0559$ to 2,694 edges, as shown in Figure 3. (When plotting preprocessed instances, we keep the original positions of vertices and edges. This allows an easy comparison with the original instances. The bold lines indicate edges already chosen.) 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 significative for some instances with no terminals near corners of the grid and few holes. It is the case of $diw0559$, but not of $taq0903$ which is only reduced from 10,490 to 8,485 edges. The reduction rate obtained is not enough to help substantially on solving those VLSI instances. The real breakthrough in preprocessing VLSI instances comes from the idea of expansion proposed by Winter [6]. This is better introduced by an example:

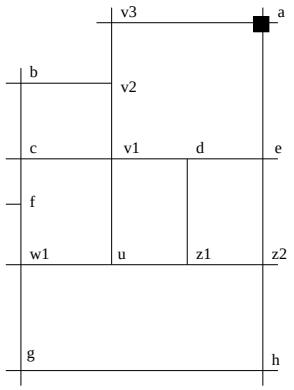

Figure 1: Example of BD3 with Expansion Test

Example: In Figure 1, BD3 condition over vertex u is not satisfied. Suppose that u is used with degree 3 by every MCST. As v_1 is a non-terminal vertex, its degree in every MCST is at least 2. Since $d(c, w_1) \leq c(u, v_1)$, edge (v_1, c) does not belong to any MCST. Otherwise, edge (u, v_1) could be replaced by the path between c and w_1 producing a new MCST in which u would have degree 2. Since $d(d, z_1) \leq c(u, v_1)$, edge (v_1, d) does not appear in any MCST. Therefore, (v_1, v_2) must belong to every MCST. We say that path $\{(u, v_1)\}$ was *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 expand it to $\{(u, z_1), (z_1, z_2)\}$, since $d(d, v_1) \leq c(u, z_1)$. Now we can apply an improved BD3 test by considering every expanded path as a single edge. Since $B(v_3, z_2) + B(z_2, w1) \leq c(\{(u, v_1), (v_1, v_2), (v_2, v_3)\}) + c(\{u, w_1\}) + c(\{(u, z_1), (z_1, z_2)\})$, the overall conclusion is that no MCST uses u with degree 3 and graph transformation of BD3 is valid. (Applying the SDE test over the reduced graph, new edges (v_1, z_1) and (v_1, w_1) would be immediately deleted.)

Sometimes it is possible to conclude that a vertex is not used with degree 3 by some MCST before applying BD3 condition. In Figure 1, this is the case of vertex z_1. This follows because path $\{(z_1, d)\}$ can not be expanded, since $d(v_1, u) \leq c(z_1, d)$ and $d(e, z_2) \leq c(z_1, d)$.

Test 8 General BD3 with Expansion - *Let u be a non-terminal vertex of degree 3 and initialize paths $P_v = \{(u, v_1)\}$, $P_w = \{(u, w_1)\}$ and $P_z = \{(u, z_1)\}$. Choose a path, say P_v, such that its last vertex, say v_n, is a non-terminal. For each edge (v_n, p), where p is adjacent to v_n but not in P_v, apply the* expansion conditions, *i.e. conditions that try to show that, supposing u has degree 3 in every MCST, (v_n, p) can not be in any MCST.*

- *If no edge pass the conditions, the whole procedure ends and it is proved that u has degree at most 2 in some MCST.*

- *If exactly one edge (v_n, p) pass the conditions, path $P_v = \{(u, v_1), \ldots, (v_{n-1}, v_n)\}$ is expanded to $\{(u, v_1), \ldots, (v_{n-1}, v_n), (v_n, p)\}$, $p = v_{n+1}$, and a new round of expansions may be tried.*

- *If more than one edge adjacent to v_n pass the conditions, P_v can not be further expanded, at least while a possible expansion of any of the other paths (P_w and P_z) does not change the conditions.*

When no more expansions are possible, apply the BD3 Test considering every path as a single edge.

Winter's expansion conditions actually resort to geometrical properties of the Rectilinear Steiner Tree Problem (RSTP), but they are directly adapted for a pure STPG context. If P is a simple path and i and j are vertices in P, let $P(i, j)$ be the subpath of P from i to j.

BD3 Expansion Conditions 1 - *Edge (v_n, p) can not be in any MCST if there is 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 is a vertex z_j in P_z such that $d(p, z_j) \leq \max\{c(P_v), c(P_z(u, z_j))\}$.*

We used stronger expansion conditions. Define the bottleneck distance between u and v supposing edge (x, y) has zero cost and that vertices $\{x, y\}$ are part of the terminal-set as:

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

BD3 Expansion Conditions 2 - *Edge (v_n, p) can not be in any MCST if there is a vertex w_j in P_w and z_k in P_z such that*

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

We also propose an enhancement of the SDE test with the idea of expansion:

Test 9 SDE with Expansion - *Let (u_1, v_1) be an edge in E. Initialize path $P = \{(u_1, v_1)\}$. Let the current path be denoted by $P = \{(u_m, u_{m-1}), \ldots, (u_1, v_1), \ldots, (v_{n-1}, v_n)\}$. Choose a non-terminal endpoint of P, if one exists, say v_n. For each vertex p adjacent to v_n and not in P, apply the following expansion condition to edge (v_n, p): If $B(p, u_n)^{-(u_1,v_1)} \leq c(P)$ then (v_n, p) can not be in any MCST (supposing (u_1, v_1) is not redundant).*

- *If no edge (v_n, p) pass the conditions, the procedure ends and it is proved that (u_1, v_1) is redundant.*

- *If exactly one edge (v_n, p) pass the criteria, path P is expanded to $\{(u_m, u_{m-1}), \ldots, (u_1, v_1), \ldots, (v_{n-1}, v_n), (v_n, v_{n+1}\}$, $p = v_{n+1}$, and a new round of expansions may be tried.*

- *If more than one edge adjacent to v_n pass the conditions, this endpoint of P can not be further expanded, at least while a possible expansion of the other endpoint does not change the conditions.*

When no more expansions are possible, test $B(u_m, v_n)^{-(u_1,v_1)} \leq c(P)$. If this is true then (u_1, v_1) is redundant.

The validity of all reduction tests is proved in the appendix.

At a given stage of the preprocessing, many possible choices of valid reductions are usually available. It is important to realize that the size of the final instance obtained, when no more reductions are possible, depends on those choices. After some experiences, we found best to divide the preprocessing into two phases. Both phases use tests 1-3, 6 and 9 in any order and the first valid reduction found is performed. In **phase I**, the transformation of BD3 with expansion (test 8) over a certain vertex is only performed if at least two out of the three new 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. In **phase II**, there are no restrictions to BD3 with expansion.

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

The instance obtained from *diw*0559 after the phase I, is shown in Figure 4. After phase II, this instance is further reduced to 164 edges. For an example of a more modest reduction, Figure 6 shows instance *taq*0903 after phase I. This instance is further reduced to 3,034 edges in phase II. Tables 1-4 show the final reductions obtained for each VLSI instance in the SteinLib. The reduction rates were quite satisfactory, enough to bring them to the reach of exact solvability by our branch-and-cut procedure, except for those four instances with more than 55,000 edges.

3 Implementing the Tests

Simple tests NTD1, NTD2 and TD1 are computationally cheap to perform. TDist Test 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 [1] showed that only these $|V| - 1$ partitions need to to be considered in TDist Test. This leads to an $O(|V|^2)$ algorithm to identify all edges fulfilling the TDist condition.

The cost of performing SD and BD3 tests lies on 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 over the original graph. This can be done in $O(|V|^3)$ time. Any sequence of reductions from NTD1, NTD2, SD orBD3 does not affect bottleneck distances over the remaining vertices. On the other hand, edge contractions from TD1 or TDist tests may reduce bottleneck distances. This is not a problem, since the table will then give upper bounds of current bottleneck distances, leading to valid reductions. When no more reductions are found, it may advantageous to update the table and try once more. If the $O(|V|^3)$ time for the calculation of all exact bottleneck distances is considered too costly, one may resort to one of the many cheaper approximations giving upper bounds of B. In this case some valid reductions may be missed.

This approach of keeping a full table of bottleneck distances, exact or approximate, is not convenient for VLSI instances. (i) grid graphs with holes are very sparse, $|V| > |E|/2$. Therefore, the cpu and even the memory requirements would be huge for the bigger instances. For example, $alue7080$ has 34,479 vertices, such a table would have 594 million entries. By contrast, $e18$ has about the same number of edges as $alue7080$ but only 2500 vertices. (ii) Due to the quite regular cost structure, it is important to use the SDE test, which requires of B^{-e} distances (the B^{+e} distances used in our BD3 expansion conditions can be obtained from a B table). Not only is hard to construct a such a table, but a reduction from a SDE test (when equality holds) may **increase** the bottleneck distances over the remaining vertices [1]. This means that the tables would have to be updated after each such application of an SDE Test. This would be far too expensive.

Our approach is to compute approximate bottleneck distances on-the-fly, as they are needed by the tests. We approximate $B(u, v)$ (and B^{-e}, B^{+e}) by considering at most 2 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 because SDE and BD3 tests, even with expansion, are quite local. It means that most of the time we have to calculate distances between vertices that are very close and only a small subgraph of G have to be considered.

Our experiences indicated that the gain in reduction rates for the VLSI instances obtained by calculating the true $B(u, v)$ would be marginal, except perhaps for terminal-dense instances like $alue7080$. This increase in computational cost does not pay-off.

4 Computational Results

To verify the effectiveness of the proposed preprocessing procedure, we also implemented a branch-and-cut in order to solve the reduced instances. A detailed description of our implementation is outside the scope of this paper. We followed several suggestions described in Koch and Martin [3] and also our own experience with a related problem [4].

Both preprocessing and the branch-and-cut codes were written in C++, the latter also used the ABACUS framework [5] and CPLEX 3.0 LP solver. All cpu times were obtained on a Sun ULTRA workstation.

The columns in tables 1-4 have the following meaning. The first columns give the original size of the instance. Next columns present the size of the preprocessed instance, after phase I and II are applied. Columns **PTime** and **BTime** are the cpu time in seconds (rounded up) for preprocessing and for the branch-and-cut, respectively. Next two columns are the value of the solutions we found and the previous best solutions, found by [3]. Bold numbers indicate solutions proven optimal. Finally, we have branch-and-cut statistics: the number of branch-and-bound nodes (**Nd**), the number of solved **LPs** and the number of **Cuts** introduced. When the instance is solved only by preprocessing, a "-" appear in the corresponding branch-and-cut entries. On the other hand, for those 4 instances considered too big for our branch-and-cut, the corresponding entries are blank.

Some remarks and additional informations are:

- It was possible to solve 43 out of 116 instances only with preprocessing. It includes 9 instances with more than 2,000 edges and the formerly open instance $dmxa1010$. Another 48 instances were reduced to less than 10% of their original size (measured by the number of edges). All the remaining instances were reduced to less than 32%, except for $alue7065$ which was only reduced to 38%.

- The time spent in reductions was somehow expensive for the larger instances. For example, preprocessing $alue7080$ took more than 3 hours of cpu. It is certainly possible to improve our implementation for those big instances. One idea is to select local subgraphs of G with several hundreds of vertices, and then apply reduction tests for many vertices and edges in this subgraph while keeping full subgraph B tables updated. The trouble is to decide the size of the subgraphs: if they are large, the updating cost is too high; if they are small, the tests may miss many valid reductions. Another promising idea is to use geometrical information, the vertex coordinates on the grid, to guide the reductions. Anyway, for the larger instances, our preprocessing times were one order of magnitude smaller than the times to solve the reduced instances by branch-and-cut. This indicates that our preprocessing procedure pays, at least if one desires optimal solutions.

- The performance of our branch-and-cut code is similar to Koch and Martin's. We are able to solve larger instances due only to the preprocessing.

Appendix

We give short proofs of Duin and Volgenant's SDE and BD3 tests in order to introduce the ideas used in the proofs of the versions with expansion. For the sake of completeness, we also give a proof of the TDist Test:

Lemma 1 *The TDist Test is valid.*

Proof: If $\delta(W) = \{e\}$, the test is trivial. Suppose $|\delta(W)| \geq 2$ and let R be a MCST 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, we get two subtrees, one containing t_1 other containing t_2. Connecting this subtrees using a shortest path from t_1 to t_2 passing through e, we get a new Steiner tree R' using e such that $c(R') \leq c(R)$. ■

Lemma 2 *The SD and SDE Tests are valid.*

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 MCST using edge (u,v). Removing (u,v) from R, we get two subtrees, let the one containing vertex u be R_u and the one containing v be R_v. Pick two vertices x and y from $T(P)$, consecutive in 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 $c(P(x,y)) \leq B(u,v) < c(u,v)$, $R' = R_u \cup R_v \cup P(x,y)$ is a Steiner tree and $c(R') < c(R)$. This contradiction shows that no MCST can use (u,v). If $B(u,v)^{-(u,v)} \leq c(u,v)$, a similar reasoning gives a Steiner tree R' not containing (u,v) such that $c(R') \leq c(R)$ ■

Lemma 3 *The BD3 Test is valid.*

Proof: Suppose that $B(v,w) + B(v,z) \leq c(u,v) + c(u,w) + c(u,z)$, 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 MCST using vertex u with degree 3. Removing u and its adjacent edges from R, we get subtrees R_v, R_w and R_z. Pick two vertices x_1 and y_1 from $T(P_{vw})$, consecutive in P_{vw}, such that $x_1 \in R_w$ and $y_1 \notin R_w$. Now pick two vertices x_2 and y_2 from $T(P_{vz})$, consecutive in 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 $c(P_{vz}(x_2,y_2)) \leq B(v,z)$, $c(P_{vw}(x_1,y_1)) + c(P_{vz}(x_2,y_2)) \leq c(u,v) + c(u,w) + c(u,z)$ 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'. ■

The validity of the general BD3 with expansion test comes almost directly from its definition. Winter's expansion conditions are also clear. The only point deserving a formal proof are our expansion conditions:

Lemma 4 *BD3 Expansion Conditions 2 are valid.*

Proof: Suppose that expansion conditions hold. Let $S = (G, T, c)$ be the STPG instance and suppose that exists a MCST R containing (v_n, p). By hypothesis, R must also contain paths P_v, P_w and P_z. Let $S' = (G, T \cup \{v_n, p\}, c')$ be a modified STPG instance where $c'(v_n, p) = 0$. Tree R is still a MCST for S' with cost $c'(R) = c(R) - c(v_n, p)$. For any vertex $x, y \in V$, $B'(x, y) = B(x, y)^{+(v_n, p)}$. Applying the reasoning of lemma 3 over S', considering paths P_v, $P_w(u, w_j)$ and $P_z(u, z_k)$ as edges, it can be concluded that exists a Steiner tree R' with cost $c'(R') \leq c(R) - c(v_n, p)$ where u has degree at most 2. Therefore, $R' \cup \{(v_n, p)\}$ is a Steiner tree for S where u has degree at most 2 and has cost at most $c(R)$. ■

Lemma 5 *The SDE with Expansion Test is valid.*

Proof: The validity of the general SDE test with expansion also comes directly from its definition. We analyze the expansion condition over an edge (v_n, p). Suppose R is MCST containing (v_n, p). By hypothesis, R contains P. If $B(p, u_m)^{-(u_1, v_1)} \leq c(P)$, we can apply a reasoning similar to lemma 2 to replace P for a subpath of P', where P' is a path not containing (u_1, v_1) in $\mathcal{P}(u, v)$ such that $SD(P') = B(p, u_m)^{-(u_1, v_1)}$. The new Steiner tree R' has cost at most $c(R)$ and not uses (u_1, v_1). ■

References

[1] C. Duin, *Steiner's Problem in Graphs*, PhD thesis, University of Amsterdam, 1993.

[2] C. Duin and A. Volgenant, *Reduction Tests for the Steiner Problem in Graphs*, Networks, Vol. 19, 549-567, 1989.

[3] T. Koch and A. Martin, *Solving Steiner Tree Problems in Graphs to Optimality*, Networks, Vol. 32, 207-232, 1998.

[4] E. Uchoa and M. Poggi de Aragão, *Vertex-Disjoint Packing of Two Steiner Trees: Polyhedra and Branch-and-Cut*, Proceedings of the IPCO'99 Conference, Lecture Notes in Computer Science, Vol. 1610, 439-452, 1999.

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

[6] P. Winter, *Reductions for the Rectlinear Steiner Tree Problem*, Networks, Vol. 26, 187-198, 1995.

Table 1: *alue*, *alut* and *diw* series.

	Original Size			Preprocessed Size			Times		Solutions		B&C Stats		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	Ours	Previous	Nds	LPs	Cuts
*alue*2087	1244	1971	34	40	65	13	2	1	**1049**	**1049**	1	5	65
*alue*2105	1220	1858	34	1	0	1	1	-	**1032**	**1032**	-	-	-
*alue*3146	3626	5869	64	65	104	26	15	5	**2240**	2240	3	20	253
*alue*5067	3524	5560	68	300	504	38	77	54	**2586**	**2586**	1	20	1057
*alue*5345	5179	8165	68	1100	1860	61	109	2930	**3507**	3560	13	103	16094
*alue*5623	4472	6938	68	768	1307	57	52	1703	**3413**	3463	7	102	12941
*alue*5901	11543	18429	68	1066	1821	59	81	1286	**3912**	3994	1	51	9910
*alue*6179	3372	5213	67	441	752	49	49	112	**2452**	**2452**	5	28	1062
*alue*6457	3932	6137	68	793	1356	53	93	552	**3057**	3062	1	37	3231
*alue*6735	4119	6696	68	340	556	50	46	83	**2696**	**2696**	1	23	1416
*alue*6951	2818	4419	67	257	413	46	35	58	**2386**	**2386**	1	24	1130
*alue*7065	34046	54841	544	11911	20956	426	7581			24827			
*alue*7066	6405	10454	16	1768	3109	9	300	9571	**2256**	2275	1	95	6001
*alue*7080	34479	55494	2344	8355	14387	1328	11222						
*alue*7229	940	1474	34	1	0	1	1	-	**824**	**824**	-	-	-
*alut*0787	1160	2089	34	9	13	5	1	1	**982**	**982**	1	6	15
*alut*0805	966	1666	34	34	54	16	2	1	**958**	**958**	1	9	78
*alut*1181	3041	5693	64	207	369	39	26	56	**2353**	2390	1	31	1102
*alut*2010	6104	11011	68	374	641	45	74	69	**3307**	3322	1	20	1131
*alut*2288	9070	16595	68	1195	2136	60	269	1190	**3843**	3889	3	33	3733
*alut*2566	5021	9055	68	701	1223	62	63	813	**3073**	3127	3	59	4415
*alut*2610	33901	62816	204	10576	19829	182	6990			12760			
*alut*2625	36711	68117	879	11759	21628	763	8832			36763			
*alut*2764	387	626	34	1	0	1	1	-	**640**	**640**	-	-	-
*diw*0234	5349	10086	25	760	1413	21	221	69	**1996**	**1996**	1	15	641
*diw*0250	353	608	11	1	0	1	1	-	**350**	**350**	-	-	-
*diw*0260	539	985	12	1	0	1	1	-	**468**	**468**	-	-	-
*diw*0313	468	822	14	1	0	1	1	-	**397**	**397**	-	-	-
*diw*0393	212	381	11	1	0	1	1	-	**302**	**302**	-	-	-

Table 2: *diw* (continued) and *dmxa* series.

Instance	Original Size			Preprocessed Size			Times		Solutions		B&C Stats		
	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	Ours	Previous	Nds	LPs	Cuts
*diw*0445	1804	3311	33	27	42	10	5	1	**1363**	**1363**	1	12	78
*diw*0459	3636	6789	25	1	0	1	2	-	**1362**	**1362**	-	-	-
*diw*0460	339	579	13	1	0	1	1	-	**345**	**345**	-	-	-
*diw*0473	2213	4135	25	14	22	6	2	1	**1098**	**1098**	1	9	33
*diw*0487	2414	4386	25	4	5	3	4	1	**1424**	**1424**	1	2	3
*diw*0495	938	1655	10	1	0	1	1	-	**616**	**616**	-	-	-
*diw*0513	918	1684	10	1	0	1	1	-	**604**	**604**	-	-	-
*diw*0523	1080	2015	10	1	0	1	1	-	**561**	**561**	-	-	-
*diw*0540	286	465	10	1	0	1	1	-	**374**	**374**	-	-	-
*diw*0559	3738	7013	18	83	141	11	17	6	**1570**	1570	1	30	394
*diw*0778	7231	13727	24	187	337	15	40	19	**2173**	2173	1	23	601
*diw*0779	11821	22516	50	2377	4490	39	1390	34077	**4440**	4566	3	105	11022
*diw*0795	3221	5938	10	225	400	10	104	19	**1550**	1553	1	24	624
*diw*0801	3023	5575	10	334	605	10	155	79	**1587**	1587	1	35	1303
*diw*0819	10553	20066	32	1394	2604	26	803	2956	**3399**	3430	1	51	4451
*diw*0820	11749	22384	37	1816	3407	32	1756	34190	**4167**	4259	1	191	15773
*dmxa*0296	233	386	12	1	0	1	1	-	**344**	**344**	-	-	-
*dmxa*0368	2050	3676	18	47	76	9	2	1	**1017**	**1017**	1	20	182
*dmxa*0454	1848	3286	16	1	0	1	1	-	**914**	**914**	-	-	-
*dmxa*0628	169	280	10	1	0	1	1	-	**275**	**275**	-	-	-
*dmxa*0734	663	1154	11	1	0	1	1	-	**506**	**506**	-	-	-
*dmxa*0848	499	861	16	34	54	11	1	1	**594**	**594**	1	8	78
*dmxa*0903	632	1087	10	53	90	7	3	1	**580**	**580**	1	21	208
*dmxa*1010	3983	7108	23	1	0	1	3	-	**1488**	1488	-	-	-
*dmxa*1109	343	559	17	9	13	5	1	1	**454**	**454**	1	3	11
*dmxa*1200	770	1383	21	29	42	13	1	1	**750**	**750**	1	7	55
*dmxa*1304	298	503	10	1	0	1	1	-	**311**	**311**	-	-	-
*dmxa*1516	720	1269	11	1	0	1	1	-	**508**	**508**	-	-	-
*dmxa*1721	1005	1731	18	4	5	3	1	1	**780**	**780**	1	2	2
*dmxa*1801	2333	4137	17	310	553	17	40	80	**1365**	**1365**	1	42	1328

Table 3: *gap* and *msm* series.

	Original Size			Preprocessed Size			Times		Solutions		B&C Stats		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	Ours	Previous	Nds	LPs	Cuts
*gap*1307	342	552	17	1	0	1	1	-	**549**	**549**	-	-	-
*gap*1413	541	906	10	1	0	1	1	-	**457**	**457**	-	-	-
*gap*1500	220	374	17	1	0	1	1	-	**254**	**254**	-	-	-
*gap*1810	429	702	17	1	0	1	1	-	**482**	**482**	-	-	-
*gap*1904	735	1256	21	7	9	4	4	1	**763**	**763**	1	2	6
*gap*2007	2039	3548	17	41	70	9	2	1	**1104**	**1104**	3	19	153
*gap*2119	1724	2975	29	1	0	1	3	-	**1244**	**1244**	-	-	-
*gap*2740	1196	2084	14	13	19	5	1	1	**745**	**745**	1	10	36
*gap*2800	386	653	12	1	0	1	1	-	**386**	**386**	-	-	-
*gap*2975	179	293	10	1	0	1	1	-	**245**	**245**	-	-	-
*gap*3036	346	583	13	28	42	9	1	1	**457**	**457**	1	12	101
*gap*3100	921	1558	11	16	25	8	1	1	**640**	**640**	1	5	31
*gap*3128	10393	18043	104	2146	3903	60	769	1366	**4292**	4315	1	26	2632
*msm*0580	338	541	11	7	9	4	1	1	**467**	**467**	1	3	7
*msm*0654	1290	2270	10	1	0	1	1	-	**823**	**823**	-	-	-
*msm*0709	1442	2403	16	1	0	1	2	-	**884**	**884**	-	-	-
*msm*0920	752	1264	26	7	9	4	3	1	**806**	**806**	1	4	9
*msm*1008	402	695	11	13	18	6	1	1	**494**	**494**	1	7	35
*msm*1234	933	1632	13	7	9	4	1	1	**550**	**550**	1	3	8
*msm*1477	1199	2078	31	12	16	6	2	1	**1068**	**1068**	1	7	25
*msm*1707	278	478	11	1	0	1	1	-	**564**	**564**	-	-	-
*msm*1844	90	135	10	6	8	4	1	1	**188**	**188**	1	4	9
*msm*1931	875	1522	10	4	5	3	1	1	**604**	**604**	1	2	3
*msm*2000	898	1562	10	1	0	1	1	-	**594**	**594**	-	-	-
*msm*2152	2132	3702	37	191	333	24	13	21	**1590**	**1590**	1	20	607
*msm*2326	418	723	14	9	12	5	1	1	**399**	**399**	1	2	7
*msm*2492	4045	7094	12	39	61	9	6	1	**1459**	1459	3	13	108
*msm*2525	3031	5239	12	11	15	6	3	1	**1290**	**1290**	1	5	16
*msm*2601	2961	5100	16	178	305	12	27	10	**1440**	1440	1	18	533

Table 4: *msm* (continued) and *taq* series.

	Original Size			Preprocessed Size			Times		Solutions		B&C Stats		
Instance	$\|V\|$	$\|E\|$	$\|T\|$	$\|V\|$	$\|E\|$	$\|T\|$	PTime	BTime	Ours	Previous	Nds	LPs	Cuts
*msm*2705	1359	2458	13	4	5	3	1	1	**714**	**714**	1	2	3
*msm*2802	1709	2963	18	7	11	4	2	1	**926**	**926**	1	5	12
*msm*2846	3263	5783	89	347	595	58	49	226	**3135**	3141	3	48	2492
*msm*3277	1704	2991	12	1	0	1	1	-	**869**	**869**	-	-	-
*msm*3676	957	1554	10	1	0	1	1	-	**607**	**607**	-	-	-
*msm*3727	4640	8255	21	1	0	1	11	-	**1376**	**1376**	-	-	-
*msm*3829	4221	7255	12	338	594	10	52	35	**1571**	1571	1	32	599
*msm*4038	237	390	11	1	0	1	1	-	**353**	**353**	-	-	-
*msm*4114	402	690	16	1	0	1	1	-	**393**	**393**	-	-	-
*msm*4190	391	666	16	1	0	1	1	-	**381**	**381**	-	-	-
*msm*4224	191	302	11	1	0	1	1	-	**311**	**311**	-	-	-
*msm*4312	5181	8893	10	1299	2355	10	199	1299	**2016**	2049	1	47	2756
*msm*4414	317	476	11	1	0	1	1	-	**408**	**408**	-	-	-
*msm*4515	777	1358	13	35	55	8	1	1	**630**	**630**	1	15	109
*taq*0014	6466	11046	128	1717	3064	82	290	10507	**5326**	5442	1	77	11423
*taq*0023	572	963	11	37	63	7	1	1	**621**	**621**	1	17	131
*taq*0365	4186	7074	22	984	1771	21	77	696	**1914**	1914	1	36	2199
*taq*0377	6836	11715	136	2020	3577	115	319	27365	**6393**	6565	3	141	25856
*taq*0431	1128	1905	13	108	188	10	14	5	**897**	**897**	1	24	253
*taq*0631	609	932	10	8	11	4	1	1	**581**	**581**	1	6	17
*taq*0739	837	1438	16	64	105	12	6	3	**848**	**848**	3	23	236
*taq*0741	712	1217	16	84	140	14	7	5	**847**	**847**	1	21	374
*taq*0751	1051	1791	16	104	178	14	7	4	**939**	**939**	1	15	259
*taq*0891	331	560	10	1	0	1	1	-	**319**	**319**	-	-	-
*taq*0903	6163	10490	130	1704	3034	90	244	14124	**5099**	5162	3	102	13833
*taq*0910	310	514	17	1	0	1	1	-	**370**	**370**	-	-	-
*taq*0920	122	194	17	1	0	1	1	-	**210**	**210**	-	-	-
*taq*0978	777	1239	10	1	0	1	1	-	**566**	**566**	-	-	-

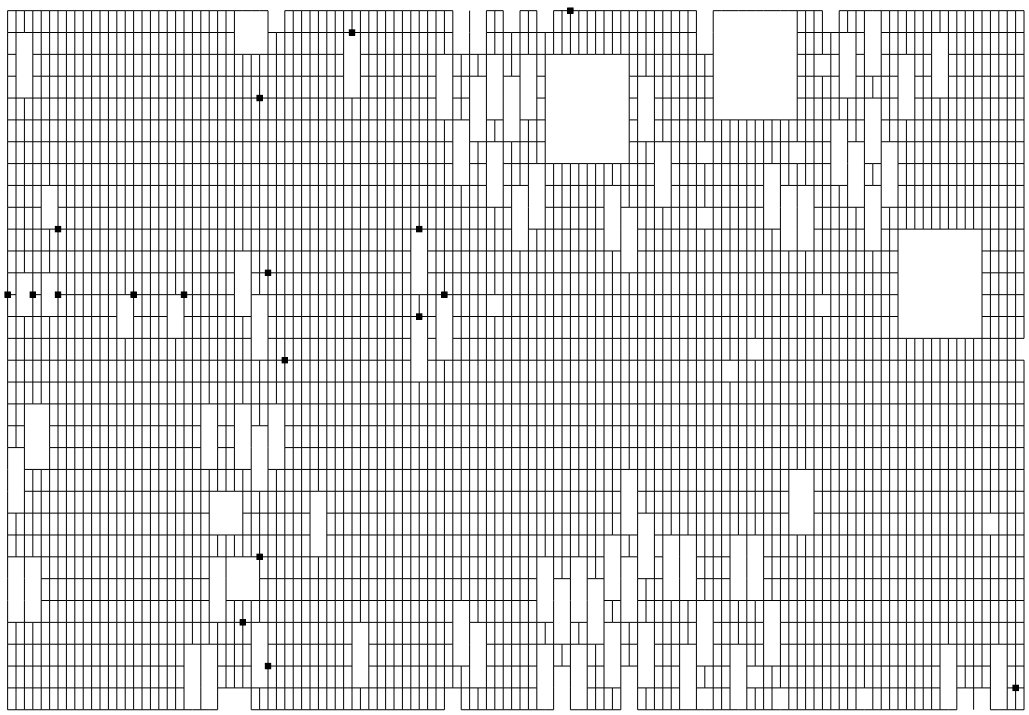

Figure 2: Original *diw*0559, $|V| = 3738$, $|E| = 7013$, $|T| = 18$.

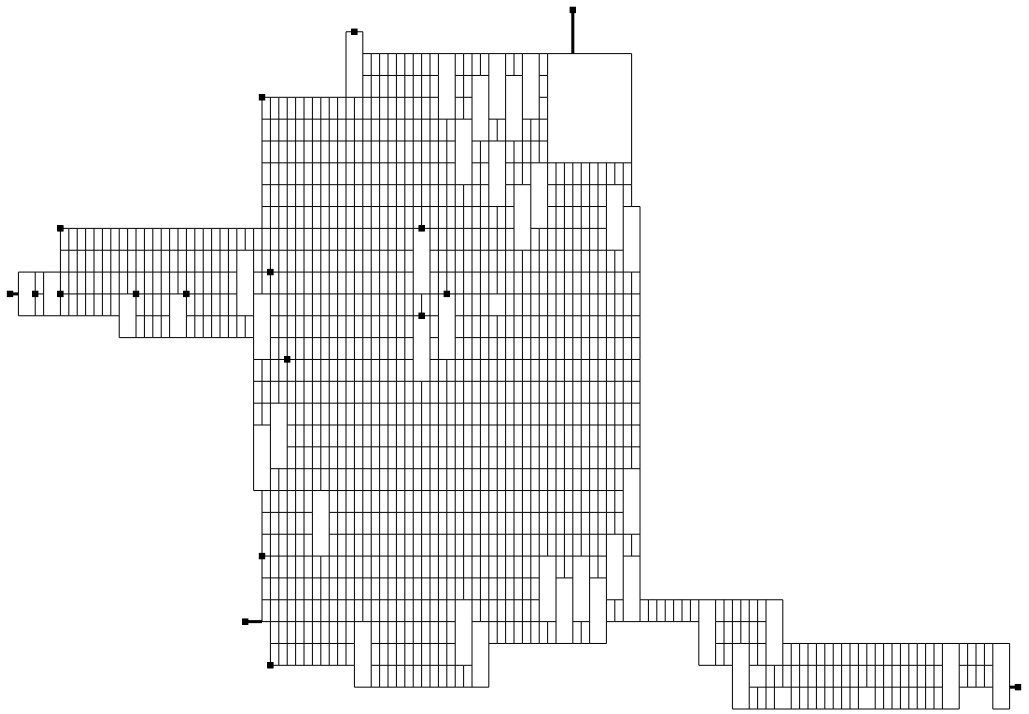

Figure 3: *diw*0559 after tests 1-3, 5-7, $|V| = 1452$, $|E| = 2694$, $|T| = 18$.

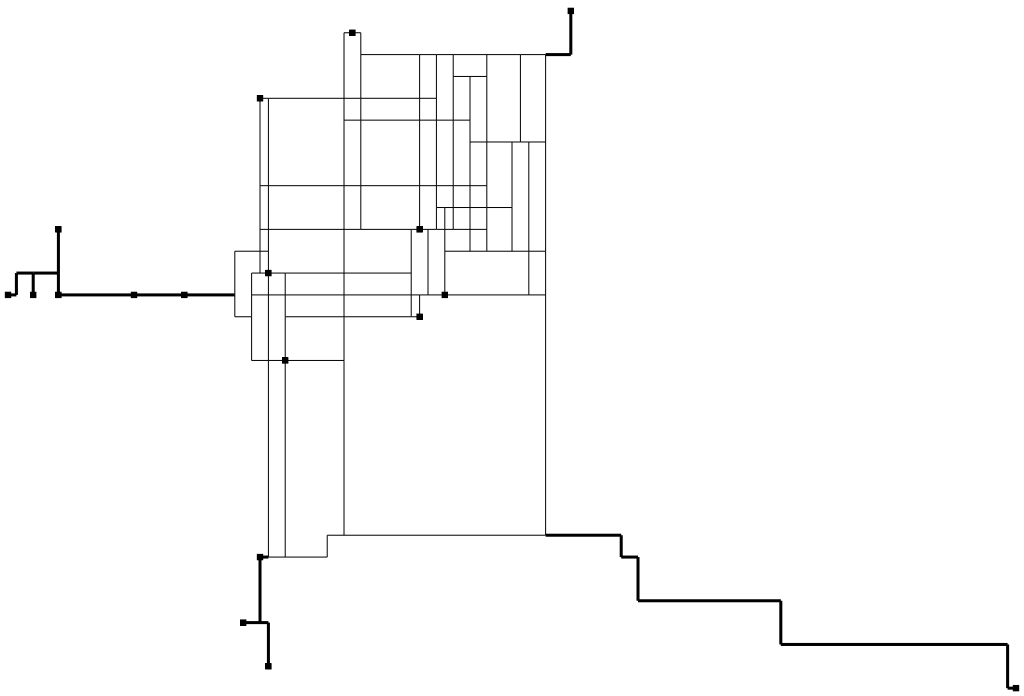

Figure 4: *diw*0559 after phase I preprocessing, $|V| = 92$, $|E| = 151$, $|T| = 11$.

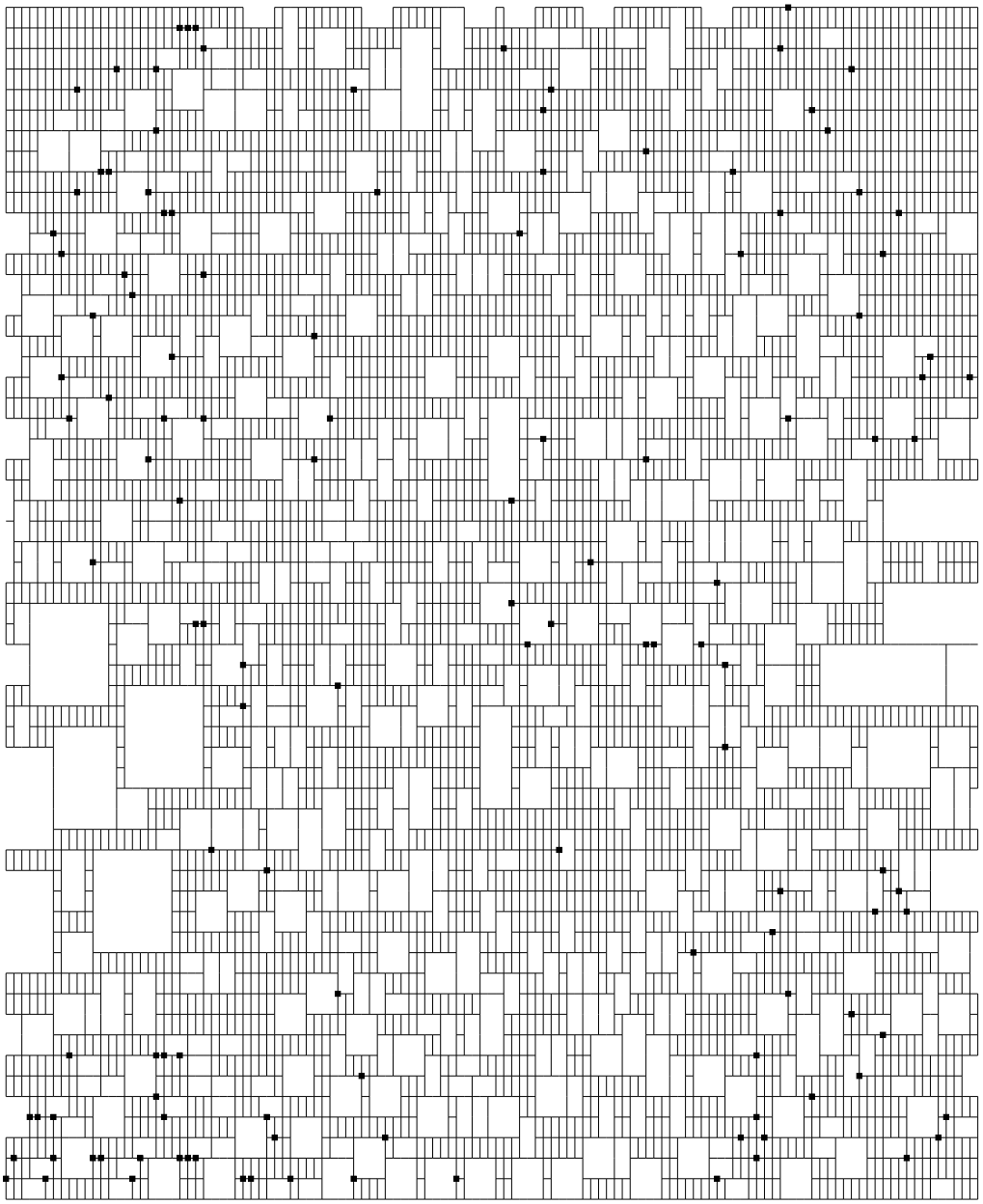

Figure 5: Original *taq*0903, $|V| = 6163$, $|E| = 10,490$, $|T| = 130$.

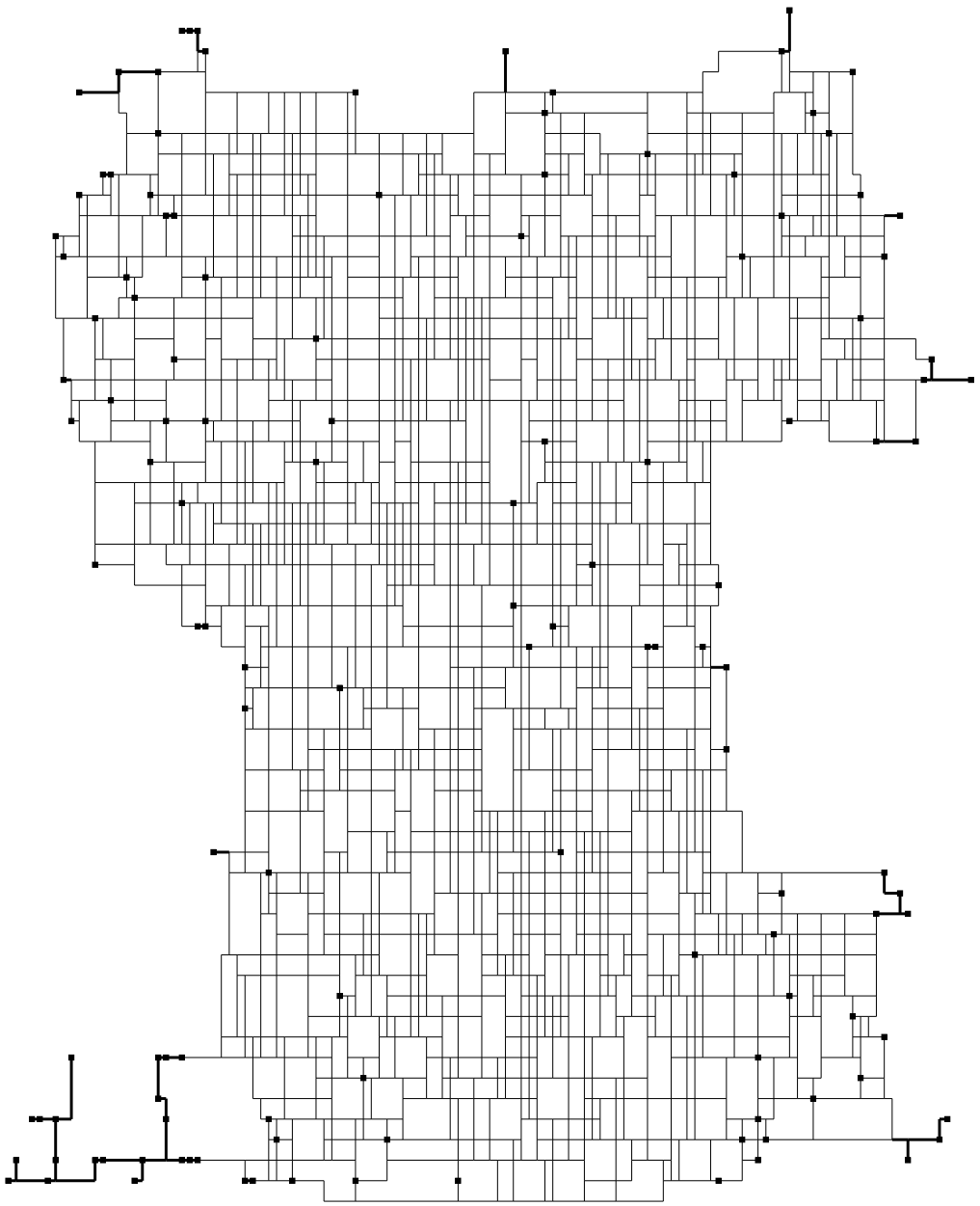

Figure 6: *taq*0903 after phase I preprocessing, $|V| = 1995$, $|E| = 3387$, $|T| = 92$.