Substation Location in Offshore Wind Farms – A Planar Multi-Facility Location-Routing Problem

Thomas Amland
Department of Informatics
University of Bergen

March 14, 2014

Contents

Chapter 1

Introduction

In offshore wind farms, two important parts of the design are to determine locations for substations and a cabling layout that connects every turbine to a substation. These problems are interconnected, as the cable layout depends on the choice of location for the substation. In this thesis we investigate how to set the location of substations such that the total cable cost is minimized.

The *Substation Location Problem* is defined as follows: given the turbine positions, cable capacity and number of substations to install; determine the locations for the substations and the cable layout minimizing the total cable cost such that the number of turbines per cable do not exceed the capacity, and no cables cross each other. The problem of finding the cable layout has been described as the Offshore Wind Farm Array Cable Layout (OWFACL) [1] and amounts to the Open Vehicle Routing Problem (OVRP) with the additional constraint that routes do not cross each other. The Substation Location amounts to a facility location problem taking into account aspects of vehicle routing. In the literature, this is known as the Location-routing problem (LRP). The LRP is NP-hard, as both the routing- and location subproblems are NP-hard. Most of the literature deals with discrete location problems, while little work have been done for continuous space [2]. Work in continuous LRP includes a local search algorithm [3], and for the single facility instance, a method using self-organizing map [4] and hierarchical search method [3]. To the best of our knowledge, exact methods have not been studied for the continuous problem.

In order to relate to existing literature, terminology from location analysis and vehicle routing is used.

A detailed description of the problem is given in Chapter 2 as well as a proposed method known as the The Big Square-Small Square. In Chapter 3 this method is adapted for the LRP and multi-facility problems, and improvements such as using quads instead of squares and the use of local search is introduced. Furthermore, the performance is improved by the use of re-

laxed bounds. Lastly, a branch-and-cut approach for the LRP is introduced, and is the topic of Chapter 4.

Test Instances

Test instances are created from publicly available data for three Offshore Wind Farm (OWF) installations: Barrow [5], Walney 1 [6] and Sheringham Shoal [7], consisting of 30, 51 and 88 turbines, and 1, 1 and 2 installed substations respectively. For each OWF capacity is set between 5 and 10, and number of substations between 1 and 2 yielding 36 test instances. The instances are denoted as *b30-cX*, *b51-cX* and *s88-cX* with X indicating the capacity. Instances with two substations are prefixed *m2*.

Chapter 2

Background

2.1 The Hop-indexed formulation

The formulation used to model the OWFACL is based on [1]. This model is chosen as it is easy to implement and can be solved to optimality for the OWFACL instances within a reasonable time.

Parameters:

V_c : The set of clients
V_d : The set of depots
C : Maximum number of clients per route
p_i : Coordinates for the vertices

Variables:

$$x_{ij}^h = \begin{cases} 1 & \text{if } (i,j) \text{ is used and } i \text{ is the } h\text{th node in the route} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

$$\min \quad \sum_{i \in V} \sum_{j \in V} \sum_{h=0}^{C-1} c_{ij} x_{ij}^h \tag{2.2}$$

$$\text{s.t.} \quad \sum_{i \in V} \sum_{h=0}^{C-1} x_{ij}^h = 1 \qquad \forall j \in V_c \tag{2.3}$$

$$\sum_{i \in V} x_{ij}^{h-1} - \sum_{k \in V} x_{jk}^h \geq 0 \qquad \forall j \in V_c, \quad h = 1, ..., C-1 \tag{2.4}$$

$$x_{ij}^0 = 0 \qquad \forall i \in V_c, \quad j \in V_c \tag{2.5}$$

$$x_{ij}^h = 0 \qquad \forall i \in V_d, \quad j \in V_c, \quad h = 1, ..., C-1 \tag{2.6}$$

$$x_{ij}^h \in \{0, 1\} \forall i, j \in V; h = 0, ..., C-1 \tag{2.7}$$

$$\sum_{h=0}^{C-1} x_{ij}^h + x_{ji}^h + x_{uv}^h + x_{vu}^h \leq 1 \qquad \forall i, j, u, v \in V \mid \overline{p_i p_j} \text{ and } \overline{p_u p_v} \text{ intersect} \tag{2.8}$$

Equation (2.3) ensure every client is visited. (2.4) counts the number of clients in a route ensuring the index is incremented for each hop, (2.5) and (2.6) that routes start from a depot. Equation (2.8) are the planarity constraints. This set is large so we resort to constraint generation.

2.1.1 Branching

In this model, branching on a single variable gives a poor balance as most of the variables will be 0. Alternative branching methods include:

Decide whether an edge should be at the beginning or end (given by index H) in the routes:

$$\sum_{i \in V} \sum_{h=0}^{H} x_{ij}^h = 1 \quad \text{or} \quad \sum_{i \in V} \sum_{h=H+1}^{C-1} x_{ij}^h = 1 \quad j \in V_c, H \in \{0, \ldots, C\} \tag{2.9}$$

whether vertex i should connect to j:

$$\sum_{h=1}^{C-1} x_{ij}^h = 0 \quad \text{or} \quad \sum_{h=1}^{C-1} x_{ij}^h = 1 \quad i, j \in V_c \tag{2.10}$$

whether i and j are connected in either direction:

$$\sum_{h=1}^{C-1} x_{ij}^h = 1 \quad \text{or} \quad \sum_{h=1}^{C-1} x_{ji}^h = 1 \quad \text{or} \quad \sum_{h=1}^{C-1} x_{ij}^h + x_{ji}^h = 0 \quad i, j \in V_c \tag{2.11}$$

(2.10) showed the most promising results and is the branching scheme used. The cut with value closest to 0.5 is chosen.

2.2 LRP Formulation

With basis in the Hop-indexed model we formulate the Location-routing problem (LRP) as a non-linear mixed integer problem.

Additional parameters:

$$p_i, \quad i \in V_c: \text{ Coordinates for the clients}$$

Additional variables:

$$p_i, \quad i \in V_d: \text{ Coordinates of the depots}$$

$$\min_x \sum_{i \in V} \sum_{j \in V} \sum_{h=0}^{C-1} x_{ij}^h \|p_i - p_j\| \tag{2.12}$$

$$\text{s.t.} (2.3) - (2.8) \tag{2.13}$$

Recall from the Hop-indexed model that depots will always be the first vertex of a route. The objective (2.12) can be simplified and split into two parts: variables corresponding to edges connecting clients to other clients and those connected the depots to clients:

$$\min_x \sum_{i \in V_c} \sum_{j \in V_c} \sum_{h=1}^{C-1} \|p_i - p_j\| x_{ij}^h + \sum_{i \in V_d} \sum_{j \in V_c} \|p_i - p_j\| x_{ij}^0 \tag{2.14}$$

2.3 The Big Square–Small Square method

The Big Square-Small Square (BSSS) [8] and The Generalized Big Square Small Square (GBSSS) [9] have been suggested as a general method for solving planar single facility location problems. The method is applicable to problems that can be formulated as a function of the distances $G(D(x))$ where x is the location of the facility and D the distance vector for this location. The generalized minisum location problem is stated as

$$\min_{x \in \mathbb{R}^2} F(x) = G(D(x)) \tag{2.15}$$

The idea behind the BSSS method is to split the region of feasible points into rectangles. In a branch and bound manner, rectangles are either pruned or split into smaller rectangle by evaluating a lower bound for each rectangle. Feasible solutions are found by evaluating F for the center of each new rectangle.

Since F is a function of the distances a *lower bound on the distance vector* D for a rectangle R can be transformed into a *lower bound on the objective.* A lower bound on the distance vector here means a bound on the distances G can take for any possible location in a rectangle R. Surely, if the facility must be located inside of R, the distance from point y to the facility is at least the shortest distance from y to R.

Definition 1. *Let $\underline{d}(R, y)$ be the shortest distance between point y and rectangle R. Formally, $\underline{d}(R, y) = \min_{z \in R} \|y - z\|$*

A lower bound $LF(R)$ can then be stated as

$$LF(R) = G(\underline{\mathrm{D}}(R)) \tag{2.16}$$

where $\underline{\mathrm{D}}$ is the vector of $\underline{\mathrm{d}}$ for demand points y_i.

Chapter 3

The Planar Multi-facility Location-routing Problem

In this chapter, we use the formulation of the generalized minisum and the BSSS as a basis for a method for the planar multi-facility location-routing problem.

A general continuous multi-facility location problem with m facilities can be stated as:

$$\min_{s \in \binom{S}{m}} F(s) \tag{3.1}$$

where S is the set of feasible locations, and $\binom{S}{m}$ the set of subsets of S with cardinality 2.

The LRP is obtained by setting:

$$F(p_i) = \min_x \sum_{i \in V_c} \sum_{j \in V_c} \sum_{h=1}^{C-1} d(p_i, p_j) x_{ij}^h + \sum_{i \in V_d} \sum_{j \in V_c} d(p_i, p_j) x_{ij}^0 \tag{3.2}$$

$$\text{s.t. } (2.3) - (2.8) \tag{3.3}$$

and

$$LF(R_i) = \min_x \sum_{i \in V_c} \sum_{j \in V_c} \sum_{h=1}^{C-1} d(p_i, p_j) x_{ij}^h + \sum_{i \in V_d} \sum_{j \in V_c} \underline{d}(R_i, p_j) x_{ij}^0 \tag{3.4}$$

$$\text{s.t. } (2.3) - (2.8) \tag{3.5}$$

where d_{ij} is the L^2-norm and $\underline{\mathrm{d}}$ the shortest distance. Evaluating F and LF amounts to solving the routing problem to optimality for the distance vector given by the locations of the facilities p_i and rectangles R_i.

Perhaps the biggest challenge of applying the BSSS method as is, is the difficulty of this routing problem. The algorithm relies on evaluating F and LF a large number of times. Therefore, they should be relatively easy to compute. For some of the largest OWFACL instances, the first iteration alone, i.e. solving $F(y)$ for the center point y may take more than an hour. We expect only to be able to solve some of the smaller instances to near optimality in any reasonable time.

3.1 Branching

Branching for the multi-facility problem is best illustrated by assigning to each depot its own rectangle such that they can be considered independently. Then rectangles are split either one at a time (alternating by always choosing the largest, or by some other criteria) or simultaneously the same way as the single-facility problem. Splitting each rectangle in 4 simultaneously gives $4m$ new rectangles and 4^m possible combination of locations for m facilities. While its efficiency can be doubted, even for small m, it seems worth considering for the OWF instance as we restrict ourself to only two substations.

The high branching factor can be reduced by reducing the number of sub-rectangles each rectangle is split into. For instance, rectangles can be split in half along the shortest bimedian (effectively alternating between horizontal and vertical). In general, a splitting operation generating l sub-rectangles gives a branching factor of l^m. For instance, dividing rectangles in two with two facilities will generate 4 subproblems (see Figure 3.1 branch (2)). The same is true for single-facility problems achieving a branching factor of 2 by splitting in half.

For the VRP there is no distinction between the depots so the number of subproblems can be reduced even further by considering the cases when the depots *are* located in the same rectangle. This is illustrated in Figure 3.1 as the root node. Observe that (2) has the same solution as its (1) (the only difference is the rectangle assigned). The new problems that need to be solved are (a) and (b).

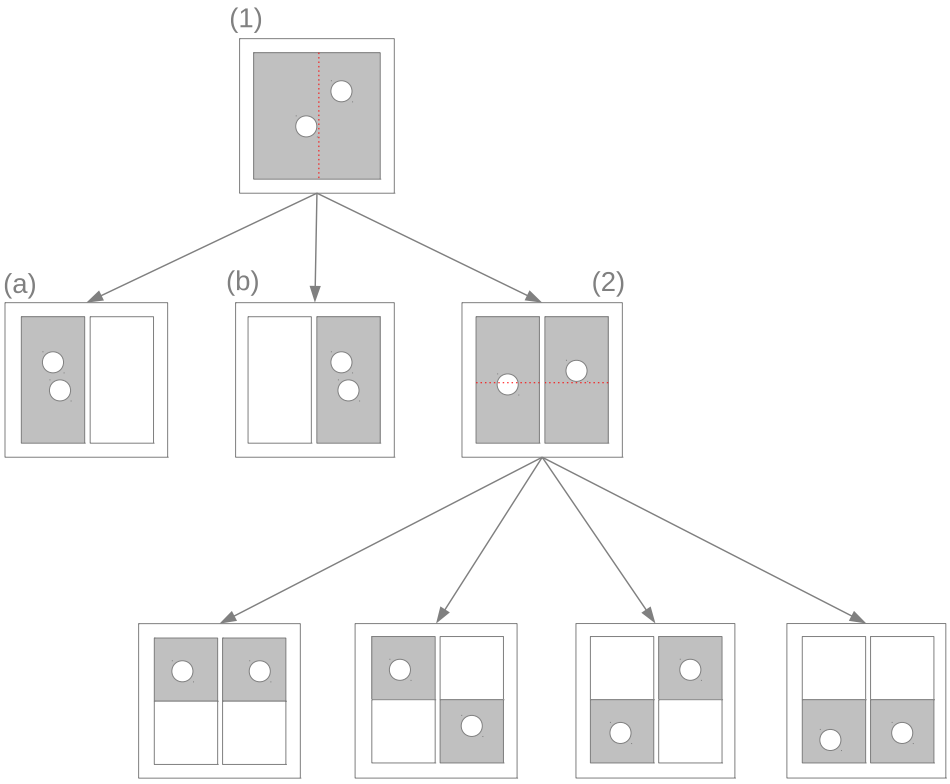

Figure 3.1: Depots are illustrated by dots and colored areas are the associated bounding rectangle. Node (1): branching when depots are located in the same rectangle. Node (2): branching when depots are fixed to different rectangles. $m = l = 2$

The same holds for any l, giving $\binom{l}{m} + l$ new problems when the assigned rectangles are the same. However, at node (2) in Figure 3.1, each depot is already fixed to different parts of the plane for the entire sub-tree. Additionally, since there are no constraints on the number of vehicles or capacity on the depots in the OWFACL, it is not optimal to place multiple depots in the same neighborhood. For simplicity, the case seen in node (2) can instead be extended to the root node by duplicating the rectangle, as initially explained. Then the depots are fixed to different rectangles (of the hyper-plane).

Splitting in half as illustrated in Figure 3.1 proved a much more efficient strategy for the 2-facility instances and is the method of choice. For the single-facility the original method of split in four was kept.

3.2 Planarity Constraints

Formulating the planarity constraint for LF requires some additional considerations. The constraints (2.8) depend on the coordinates p_i, but the depot locations $p_i, i \in V_d$ are not known other than to be in R_i.

The client points $p_i, i \in V_c$ remain fixed and edges between the clients can be treated as normally. Two edges incident to one depot cannot cross each other and need not be considered. Two edges incident to different depots need not be considered either as connecting clients to depots in such a configuration is not optimal.

The remaining are depot-client edges crossing client-client edges. A planarity inequality is valid for rectangle R if a depot edge is crossing *some* client edges for *all* depot locations inside R. Otherwise, there is a depot location that does not violate this particular planarity constraint and the inequality is not valid. An example of this is illustrated in Figure 3.2a where a section of the right hand side would be a feasible location not violating the planarity constraint. In other words, valid inequalities for the planarity constraint are given by: (i) an edge intersecting for all possible depot locations (Figure 3.2b) and (ii) a path of edges together intersecting for all depot locations (Figure 3.2c).

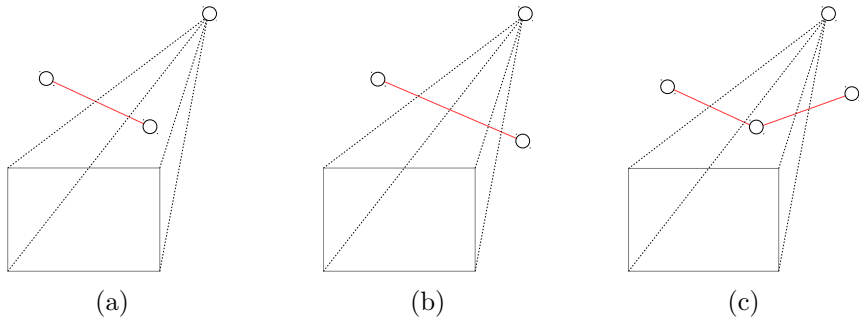

Figure 3.2: Different cases that can arise when the depot location is relaxed. Dashed lines are drawn to 4 possible depot locations. (a): allowed. (b) and (c) not allowed.

Note that (ii) will eventually become redundant when branching on location. As R is reduced to a point the possible edges from R may be treated as a single edge. Because of this we generate constraints only for singe edges (3.2b) in addition to the normal client-client edges. Since R is convex these cuts are easily found as the edges required to be intersected can be restricted to those with end point at each of the 4 corners of R, as illustrated in Figure 3.2.

3.3 Choosing the Starting Polygon

While the BSSS method uses squares or rectangles, this can obviously be generalized to other convex shapes. Some observations for the OWF instances:

- The optimal location is within the convex hull of the client points.
- Solving the routing problem for locations outside the convex hull proved much harder than for locations inside.
- For Walney and Sheringham Shoal the convex hull are exactly quadrilaterals, and for Barrow there is only a small cutoff at two of the corners (see Figure 3.3).

A variation using triangles has previously been suggested [10]. The main advantages noted for using triangles is the elimination of feasibility tests and reduction of the feasible region to exactly the convex hull by triangulating the demand points. The former does not apply to our problem since all points are considered feasible. A triangulation of the demand points will lead to at least $n-2$ triangles. For the Delaunay triangulation the worst case is $2n-5$. This is the number of starting nodes, as every triangle is used as a root. It is easy to see that staring from a single initial region can potentially reduce the overall number of evaluations needed by discarding larger areas given sufficiently bounds. In general, the only other difference between squares, triangles and other shapes is how the distances are calculated. It does not seem important to the problems we are applying the method to as solving the routing problem will dominate any minor difference is such calculations.

From the above observations, we decided to use the minimum bounding quad as the starting region to take advantage of the reduction of the feasible area. The minimum circumscribing k-gon can be found in time $O(n^2)$ [11]. Implementing such a method is out of the scope of this thesis, so we resort to a simple approximation as follows:

(i) Staring from the convex hull: For every 3 consecutive edges extend the first and third edge forming a triangle with the middle edge.

(ii) Take the smallest candidate triangle and replace the 3 old edges with the 2 new extended edges in the convex hull.

(iii) Continue with (i) until 4 edges remain.

Another possibility is to use the minimum oriented bounding box [12] to get around the rotated nature of the convex hull. The results of both are shown in Table 3.1 and Figure 3.3. While the oriented box seems to provide a good approximation for the Barrow and Walney instance, it naturally

performs poorly for the rhomboid shaped Sheringham Shoal OWF. The quad approximation performs well giving the optimal quad for all instances.

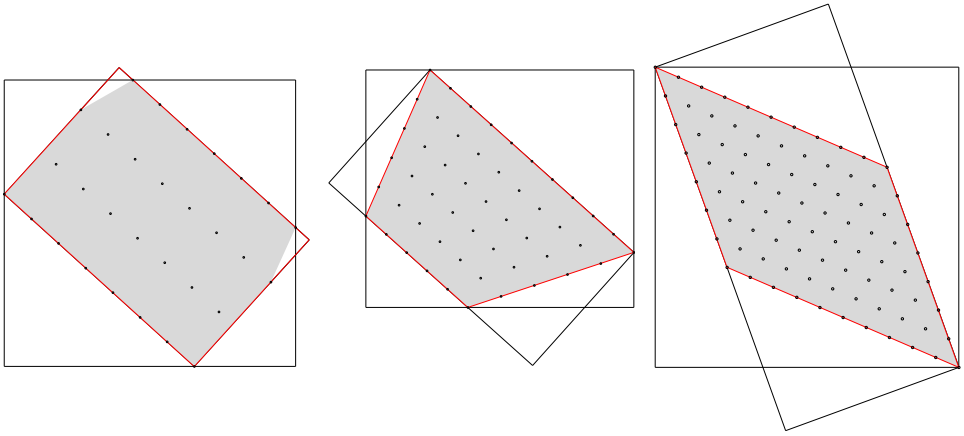

Figure 3.3: Normal- and oriented bounding box (black), quad (red), convex hull (colored area) for Barrow, Walney, and Sheringham Shoal respectively.

Instance	Convex hull	Box	Oriented	Quad
b30	1.0000	1.9495	1.0249	1.0247
w51	1.0000	2.0441	1.3337	1.0000
s88	1.0000	2.3243	1.8153	1.0000

Table 3.1: Size of the different bounding polygons relative to the convex hull.

The shortest distance $\underline{d}$ between a point and any simple polygon can be easily calculated by computing the point-line segment distance for each line segment of the polygon.

Similar to rectangles, the *center* of a quad here refers to the intersection point of the two bimedians.

3.4 Local Improvement in Location

Salhi and Nagy [3] introduced a local search procedure for the continuous multi-depot location-routing problem. The idea behind the method is to, given a fixed routing, simply move the depot to the best position without changing the routing. That is, a pure location problem is constructed from the routing problem by taking the first and last client of each route as the demand point, i.e. every client incident to the depot, and minimize the associated cost. For the *open* routing problem only the first client of each route

is considered. This set is referred to as the *end-points*. Minimize the total distances to these points amounts to finding the geometric median, which is approximated with the well-known Weiszfeld procedure. The algorithm proceeds by solving the routing problem for the new depot locations iteratively alternating between the two until there is no improvement.

One difficulty that arises are the planarity constraints. While the initial depot location will not have crossing edges, since it is feasible, moving the depot may lead to depot edges crossing with client-to-client edge. Because of this, the Weiszfeld procedure may return an infeasible solution. Salhi and Nagy [3] present a similar problem, that is, maximum cost allowed for routes, and propose two solutions: either iteratively assign non-uniform weights to infeasible location, or immediately output the last feasible location when the new becomes infeasible. The problem with using the Weiszfeld procedure is that the last feasible location may not have the desired precision if the previous step size was large. We propose a simple hill-climbing method with fixed step size as an alternative, and instead handle infeasible steps by incrementally reducing the step size.

Algorithm 1 LocationOpt($V_c \cup V_d$, routing R)

1: **for** $i \in V_d$ **do**
2: s $\leftarrow$ the set of clients incident to i in R
3: Apply method for the geometric median to i with demand points s
4: Update i to the new location

The local search is applied once for every routing found. Continuing with a re-routing step seem unnecessary as BSSS is the method of choice for the location problem. Additionally, it is computationally very expensive. The purpose of the search is to quickly obtain locally optimal solutions that may otherwise require a large number of iterations, or in fact may not be found at all given the time constraint. Given that the BSSS method rely on sampling values for F we argue that employing a local search is important to obtaining good solutions fast. In Table 3.2, the improvement obtained with *LocationOpt* during the run of Algorithm 2 for some of the instances, is given.

Instance	Min.	Avg.	Max.
b30-c5	0.04	177.26	1559.61
b30-c6	0.67	104.85	898.73
b30-c7	0.19	101.41	1035.09
b30-c8	0.00	113.93	618.33
b30-c9	1.46	101.51	615.71
b30-c10	0.00	85.42	487.30
w51-c5	3.50	492.17	3437.62
w51-c6	0.56	285.95	3266.48
w51-c7	0.37	239.64	1995.62
w51-c8	0.35	187.07	1705.92

Table 3.2: Improvement obtained with *LocationOpt*. In metre.

3.5 The Algorithm

Algorithm 2 (V_c, m number of depots)

```
1:  OPT ← ∞
2:  Initialize an empty queue Q
3:  Compute a bounding quad H of V_c, and add {H} to Q
4:  loop
5:      S ← select and remove the set with lowest lower bound from Q
6:      if LF(S)(1 + ε) ≥ OPT then
7:          Stop. The optimal value is approximated by sufficient precision
8:      else
9:          S_i ← SPLIT(S)
10:         for all S_i do
11:             s ← the set of center points of S_i
12:             Evaluate F(s) and apply LocationOpt to the routing
13:             if F(s) < OPT then
14:                 OPT ← F(s)
15:             Evaluate the lower bound LF(S_i)
16:             if LF(S_i) > OPT then
17:                 Discard S_i
18:             else
19:                 Add S_i to Q
```

Line 9 is one of several methods of branching on a set of rectangles as discussed in Section 3.1. Termination relies on LF converging as the rectangles are cut into smaller rectangles; which is true as $LF(S) = F(S)$ when the size of all $s \in S$ approach 0.

3.6 Computational Results

The underlying OWFACL problem was modeled and solved with the CPLEX MIP solver. Generation of planarity constraints was implemented with CPLEX library as well, such that each evaluation of F and LF amounts to invoking the solver with a new distance vector. All other code were developed in Scala. Experiments were performed on a personal computer with 2.85GHz processor running CPLEX 12.5 with default setting and OpenJDK 7 with an available heap space of 6GB. The computational results are given in Table 3.3.

Instance	LB	UB	Abs.	Rel. (%)	Tree size	Time
b30-c5	16524.88	16688.99	164.12	0.99	895	275
b30-c6	16236.40	16398.76	162.36	1.00	1609	814
b30-c7	16008.54	16168.58	160.04	1.00	1147	1495
b30-c8	15383.11	15536.55	153.44	1.00	635	420
b30-c9	15159.97	15310.73	150.76	0.99	713	393
b30-c10	15157.81	15309.15	151.34	1.00	1071	702
w51-c5	42273.91	42691.63	417.72	0.99	483	2017
w51-c6	40855.48	41262.22	406.75	1.00	863	6273
w51-c7	39932.54	40331.14	398.60	1.00	1009	14267
w51-c8	39278.76	39789.68	510.92	1.30	944	20000
w51-c9	38614.99	39351.94	736.95	1.91	553	20000
w51-c10	38165.85	39104.51	938.66	2.46	439	20000
s88-c5	49166.29	77832.88	28666.58	58.31	14	20000
s88-c6	46513.39	72133.01	25619.62	55.08	11	20000
s88-c7	44663.95	68491.44	23827.48	53.35	13	20000

Table 3.3: Computational results of Algorithm 2.

The target precision ϵ was set to 1%, and time limit to 20000 seconds. While some of the smallest instances could to some degree be solved withing reasonable time, the limitation of the method becomes apparent when looking at the tree size for larger instance: for s88-c7 the tree size is 13, meaning, a depth of only 3.

3.7 Relaxation

To improve the performance, we propose to (i) relax the integrality constraint (2.7) for LF and (ii) replace F with heuristics. Evaluating $LF(R)$ for some R then amounts to solving the linear program (LP) as opposed to the ILP. This should dramatically improve the number of nodes in the search tree that can be explored. Since we only look for near optimality the

idea is that this relaxation may give good enough bounds, or even improve on the exact method given the time constraint.

F and LF can be viewed as general upper- and lower bounds for the real objective and need not be an LP relaxation.

3.7.1 Termination

A problem with this relaxations is that the optimality gap between LF and F may no longer converge to ϵ. In many cases one cannot use the optimality gap as stopping criteria, as the obtainable upper and lower bound will depend on the strength of LF and F. An additional stopping criteria is needed.

A simple alternative is to use an absolute criteria as originally suggest by Hansen et al. [8], i.e. only branch when the rectangle is larger than some fixed size. As pointed out by Plastria [9], measuring "optimality" in terms of spatial size is not without issues. A rectangle being "small" does not mean one can conclude that a near optimal solution is found, as optimality is always measured in the objective. From a practical standpoint however, it seems one can safely assume that a small increase in the distance vector will consistently give a small increase of the objective. Since we limit ourself to the OWF instances, we can choose a limit that seems adequate. From experiments we have found that the neighborhood of a depot locations in which a routing remains optimal tends to be quite large. An area where the optimal routing does not change is in terms of the LRP objective function a neighborhood that is locally convex. This is an important observation as it means finding the routing and locally optimal location is sufficient to find the optimal solution in that area, and there is no point continuing.

3.7.2 Heuristics

Bauer and Lysgaard [1] have suggested an adaption of the Clarke and Wright heuristic for the OWFACL showing good results for the same test instances. Two different saving heuristics are used: *POS1* and *POS2*, and an exchange heuristics *RouteOpt*, taking the best of the two. While *RouteOpt* improves the routing when the depot is fixed, exchanging depot connected edges after it has been relocated may yield a different result. To avoid creating (globally) worse solution, *LocationOpt* is applied to routes with and without *RouteOpt* returning the best of the four.

Algorithm 3 POS'($V_c \cup V_d$, E)

1: R1 $\leftarrow$ LocationOpt(POS1($V_c \cup V_d$, E))
2: R2 $\leftarrow$ LocationOpt(POS2($V_c \cup V_d$, E))
3: R3 $\leftarrow$ LocationOpt(RouteOpt(POS1($V_c \cup V_d$, E)))
4: R4 $\leftarrow$ LocationOpt(RouteOpt(POS2($V_c \cup V_d$, E)))
5: **return** min(c(R1), c(R2), c(R3), c(R4))

3.7.3 Strengthening the Lower Bound

For large rectangles, the shortest distance used in calculation of the lower bound LF may be a significant underestimate of what the actual minimum distance will be. For instance: If two clients are located at the opposite sides of a rectangle, the minimum distance needed for using these two edges is underestimated by at least the length of the side. To improve the bound on minimum distance we introduce a new continuous variable y_{ij} to the hop-indexed model representing the length of the edge going from depot i to client j. The objective can then be written as:

$$\min \quad \sum_{i \in V_c} \sum_{j \in V_c} \sum_{h=1}^{C-1} d_{ij} x_{ij}^h + \sum_{i \in V_d} \sum_{j \in V_c} y_{ij} \tag{3.6}$$

$$\text{s.t.} \quad y_{ij} \geq \underline{d}(R_i, p_j) x_{ij}^0 \qquad \forall i \in V_d, j \in V_c \tag{3.7}$$

$$(2.3) - (2.8) \tag{3.8}$$

By the triangle inequality we get the following constraints:

$$y_{ij} + y_{ik} \geq (x_{ij} + x_{ik} - 1) d(p_j, p_k) \qquad \forall i \in V_d, \qquad j, k \in V_c \tag{3.9}$$

The constraints (3.9) are very weak for fractional x and will have less effect as the size of R is reduced. In practice, we found that these constraints made both the ILP and the LP for this model significantly more difficult while providing little improvement. It proved more efficient to branch 1-2 additional times to reach a similar bound.

3.7.4 Computational Results

Instance	LB	UB	Abs.	Rel. (%)	Tree size	Time
b30-c5	16343.18	16690.25	347.07	2.12	613	7
b30-c6	15927.92	16396.98	469.07	2.94	941	13
b30-c7	15657.48	16216.34	558.86	3.57	1229	23
b30-c8	15282.98	15542.29	259.31	1.70	917	22
b30-c9	15087.25	15309.27	222.03	1.47	805	26
b30-c10	15087.25	15309.27	222.03	1.47	1057	42
w51-c5	41991.44	43527.48	1536.04	3.66	1305	53
w51-c6	40441.87	41593.00	1151.14	2.85	1661	99
w51-c7	39611.16	40425.25	814.09	2.06	2105	183
w51-c8	39063.28	40249.41	1186.13	3.04	3445	404
w51-c9	38750.62	39679.41	928.79	2.40	3873	578
w51-c10	38592.78	39206.98	614.19	1.59	3949	717
s88-c5	75940.15	83779.16	7839.01	10.32	2557	539
s88-c6	70349.73	77414.76	7065.03	10.04	3005	1098
s88-c7	66444.11	73280.90	6836.79	10.29	3389	1857
s88-c8	63961.43	70494.46	6533.03	10.21	3741	2702
s88-c9	62301.88	67561.41	5259.52	8.44	3805	3409
s88-c10	61180.35	65735.22	4554.87	7.44	3957	4379
b30-c5-m2	14139.06	14581.94	442.88	3.13	44582	587
b30-c6-m2	14138.66	14580.98	442.33	3.13	99362	1536
b30-c7-m2	13890.02	14215.73	325.72	2.34	48410	935
b30-c8-m2	13890.02	14215.73	325.72	2.34	55092	1327
b30-c9-m2	13890.02	14215.73	325.72	2.34	59042	1699
b30-c10-m2	13890.02	14215.73	325.72	2.34	61838	2241
w51-c5-m2	37589.87	38341.77	751.90	2.00	89850	5405
w51-c6-m2	37131.37	37688.98	557.60	1.50	242119	20000

Table 3.4: Computational results with the added relaxations.

The minimum size required for an area to be further divided was set to $100m^2$. Recall that the purpose of the relaxation was to see if the method can still be used to find a good solution, not necessarily to prove optimality. This limit was set to a small value to ensure we find the solution that could be found using the heuristics. The results are given in Table 3.4. While the lower bound found obviously can be improve by splitting rectangles further, doing so comes at a very high cost in running time. For most of the instances the target precision of 1% could not be reached regardless. For instance, running b30-c6 down to a size of $0.01m^2$ took 4450 second and the relative gap was still 1.06% while giving no improvement in UB.

For several of the instances, the size limit could be set higher while retaining the upper bound found, but was kept the same for comparison.

As can be seen from the table, the obtained relative optimality gap tend to decrease as the capacity increase. The minimum number of routes required to satisfy the capacity constraint is $\lceil \frac{n}{C} \rceil$. As the number of depot edges used increase, the total underestimate of distances increases and is affecting the strength of the lower bound. For these low capacities, it is usually optimal to use what is available, resulting in a number of routes close the minimum required.

Chapter 4

A Branch-and-cut Algorithm for the LRP

One of the problems with the BSSS method in general is that no knowledge of the underlying problem it is applied to is used. When applied to the LRP the result is a location-first routing-second type of procedure where the only information used in the location phase is the lower bound on the routing problem to guide where to try next. That is; after a region has been bounded all information about the routing that was found is discarded. This is very inefficient. A better approach is to solve location and routing *simultaneously* as a single problem.

Starting with the linear program (LP) that amount to $LF(H)$ as defined in Section 3.7 with H being the initial bounding region, one can view *routing* and *location* as each a possible branch in a branch and bound tree. That is, at each subproblem $LP(R_i)$ given by the rectangle R_i, either branch on location by dividing the bounding region to solve the location problem; or on integrality by branching a variable to solve the routing. The idea behind this method is two part: (i) by branching on integer infeasibility before the location is determined, one can benefit from a stronger lower bound. Similarly, if the LP gives a sufficient bound, it can quickly be pruned. (ii) LP solutions can be reused to quickly solve subproblems.

4.1 Obtaining Feasible Solutions

An important difference from the BSSS method is how solutions are obtained. As we have seen, the two-part procedure of evaluating lower bounds and upper bounds independently is not very efficient. Instead, we use a traditional branch and cut approach of incrementally creating "less fractional" problems until a feasible solutions is found. It will of course not be possible

to obtain solutions *directly* through branching since it still takes infinitely many steps to reduce a rectangle to a point.

Ignoring the planarity constraint for a moment, let $x(R)$ be an integer feasible solution for $LP(R)$. Observe that $x(R)$ is also feasible for all $p \in R$. Thus, a feasible solution to the LRP can be obtained when an integer feasible lower bound is found. It of course only make sense to do so when it yields a *good* solution, e.g. when x is also optimal for $F(p)$. In other words: a good solution is found when the increase in distance between p and R is sufficiently small such that the optimal routing does not change.

The purpose of trying to identify when the routing does not change is to finally use this to take a diving step in the search tree and obtain the optimal solution within the rectangle as early as possible, while keeping the number of iterations required for re-optimization as low as possible.

Another possibility is to create an additional branch for the center point p of R when splitting R. p can be considered a subproblem as it is not defined for any finite depth sub-rectangles of R. The main difficulty with this approach is that many unnecessary and hard nodes a created with no obvious good node selection strategy.

4.2 Maximum Increase in LP

When reducing the size of the rectangles, the distances vector $\underline{d}$ necessarily increases. We devise a bound on the increase in the objective function from such increase in distance, that will be used for branching and diving strategies.

Let $x^*(R)$ be the optimal solution for $LP(R)$ and $z(R)$ the objective value for this solution. Observe that when splitting R into the set of sub-rectangles R_i, $x^*(R)$ is a feasible solution for $LP(R_i)$. The increase in the distance vector cannot be more than the longest diagonal of R, here denoted as L. Then maximum increase is:

$$maxIncrease\left(LP\left(R_i\right)\right) = \sum_{i \in V_d} \sum_{j \in V_c} L x_{ij}^{*0} \tag{4.1}$$

Reformulating (4.1) as the maximum objective value:

$$z\left(R_i\right) \leq z_{max}(R_i) = \sum_{i \in V_c} \sum_{j \in V_c} \sum_{h=1}^{C-1} d_{ij} x_{ij}^{*h} + \sum_{i \in V_d} \sum_{j \in V_c} L x_{ij}^{*0} \tag{4.2}$$

The bound holds for all $R_i \subseteq R$. By the triangle inequality the increase in distance can be strengthen to:

$$z_{max}(R_i) = \sum_{i \in V_c} \sum_{j \in V_c} \sum_{h=1}^{C-1} d_{ij} x_{ij}^{*h} + \sum_{i \in V_d} \left(\max_{y \in R_i} \sum_{j \in V_c} d(y,j) x_{ij}^{*0} \right) \tag{4.3}$$

The maximization problem of (4.3) amounts to the *maxisum* problem on the plane bounded by the convex polygon R and the optimal values is at one of the extreme points [13].

Regarding branching strategies we have made the following observations:

1. Except for the very small rectangles, branching on locations generally gives a much better increase in lower bound.

2. There is no obvious benefit from fixing variables long before the location is decided.

The solution we proposed is to require the increase in z_{max} to be low, that is $\left(z_{max}(R_i) - z(R_i)\right) / z(R_i) \leq L_1$ for some parameter L_1. It can be viewed as an alternative to the size criteria used in Section 3.7.1, but taking into account changes in objective value. There a still two important cases where one would want to branch on variables, namely:

3. If there is a good chance of finding a better solution than current best.

4. If branching on location will not give a sufficient bound.

We propose to branch on variable when $z_{max}(R) \leq OPT(1 + L_2)$. A parameter L_2 is added as z_{max} is still likely an overestimate of the increase that can be expected.

4.3 The Algorithm

Algorithm 4 (V_c, m number of depots)

1: OPT $\leftarrow \infty$
2: Initialize an empty queue Q
3: Compute a bounding quad H of V_c, and add $\{H\}$ to Q
4: **loop**
5: $S \leftarrow$ select and remove the set with lowest lower bound from Q
6: $\delta \leftarrow (z_{max}(S) - z(S))/z(S)$
7: **if** $x^*(S)$ is infeasible **then**
8: Discard S
9: **else if** $z(S)(1+\epsilon) \geq OPT$ **then**
10: Stop. OPT is approximated with sufficient precision
11: **else if** $x^*(S)$ contains crossings **then**
12: Add violated cuts
13: **else if** x^* not integer **and** $\delta \leq L_1$ **and** $z_{max} \leq OPT(1+L_2)$ **then**
14: Branch on variable
15: **else**
16: Branch on location
17: **if** $x(S)$ is integer **and** $\delta \leq L_1$ **then**
18: Let S' be the set of center points of S
19: Solve $LP(S')$ to integrality
20: Apply *LocationOpt* to S' and update OPT if necessary
21: Prune all $S \in Q$ where $z(S) \geq OPT$

4.4 Memory Constraints

Taking advantage of fast re-optimization when cuts are added to the problem is essential to achieve good performance. Memory quickly becomes a constraint when using the hop-indexed model as there are $O((n+m)^2C)$ variables that needs to be stored for every node. In an attempt to reduce the need for recompute LPs, a basis pool is implemented as a priority queue that discards the basis with the highest lower bound first.

A garbage collection step is performed by immediately discarding nodes that can be pruned by bound every time a new solution is found (Line 21 in Algorithm 4). In addition, since the algorithm terminates when a sufficient precision is reached, not only when Q is empty, all nodes with a sufficient bound can be discarded as long as the lowest lower bound is recorded. I.e all $S \in Q$ where $z(S)(1+\epsilon) \geq OPT$.

Despite these improvements the available memory proved insufficient for the computational results in Section 4.5. For instances s88-c6 to s88-c10, b30-c5-m2, b30-c6-m2 and w51-c5-m2 to s88-c10-m2, one or more LPs had

to be recomputed because of insufficient memory.

4.5 Computational Results

Instance	LB	UB	Abs.	Rel. (%)	Tree size	Time
b30-c5	16520.81	16685.48	164.66	1.00	1663	8
b30-c6	16234.69	16396.98	162.29	1.00	7520	43
b30-c7	16003.18	16163.10	159.92	1.00	13116	95
b30-c8	15383.23	15535.99	152.76	0.99	952	13
b30-c9	15161.68	15309.27	147.59	0.97	792	16
b30-c10	15158.21	15309.15	150.94	1.00	1116	29
w51-c5	42251.09	42668.48	417.39	0.99	1038	21
w51-c6	40853.47	41261.66	408.20	1.00	20748	544
w51-c7	39926.81	40326.03	399.23	1.00	26310	1061
w51-c8	39392.02	39785.93	393.91	1.00	40326	2365
w51-c9	38941.12	39330.45	389.33	1.00	9904	821
w51-c10	38654.78	39041.31	386.53	1.00	5388	717
s88-c5	76996.28	77766.24	769.96	1.00	104866	5479
s88-c6	70724.69	-	-	-	112478	20001
s88-c7	66847.24	-	-	-	80974	20001
s88-c8	64269.22	-	-	-	67036	20000
s88-c9	62562.80	-	-	-	56811	20013
s88-c10	61434.15	-	-	-	49539	20002
b30-c5-m2	14437.57	14581.94	144.37	1.00	787133	18193
b30-c6-m2	14361.64	14581.41	219.77	1.53	747494	20000
b30-c7-m2	14074.98	14215.73	140.75	1.00	225764	1561
b30-c8-m2	14074.98	14215.73	140.75	1.00	276550	3128
b30-c9-m2	14074.98	14215.73	140.75	1.00	311666	4309
b30-c10-m2	14074.98	14215.73	140.75	1.00	336456	5202
w51-c5-m2	37962.15	38341.77	379.62	1.00	384537	9252
w51-c6-m2	37297.02	37671.90	374.88	1.01	511222	20000
w51-c7-m2	36889.45	37258.34	368.89	1.00	383747	17709
w51-c8-m2	36681.06	37258.69	577.63	1.57	345150	20000
w51-c9-m2	36484.52	37175.13	690.61	1.89	277864	20000
w51-c10-m2	36357.15	37009.28	652.13	1.79	242376	20000
s88-c5-m2	62133.35	63597.66	1464.30	2.36	184497	20000
s88-c6-m2	59568.73	61248.04	1679.31	2.82	101405	20001
s88-c7-m2	57872.13	60069.52	2197.39	3.80	98820	20001
s88-c8-m2	57156.85	-	-	-	43488	20001
s88-c9-m2	56313.00	-	-	-	30382	20001
s88-c10-m2	55846.33	-	-	-	24174	20001

Table 4.1: Computational results for Algorithm 4.

The results for Algorithm 4 are given in Table 4.1. L_1 was set to 0.001 and L_2 to 0.05. The values were chosen through experimentation. For s88, L_2 was reduced to 0.025 as too many iteration were used for solving the fixed-point problem (line 19 in Algorithm 4).

4.6 Observations

The average number of diving steps taken across all instances (where UB is provided) was 2.5 with an average number of iteration of 2.1 suggesting the parameters L_1 is sufficiently low such that feasible solutions can be obtained almost immediately.

The new algorithm provide a significant improvement to the previous results. For all b30 and w51 the LB and UB from the initial results are retained, while keeping a comparable running time to the method using heuristic. In some cases even improving. Additionally, w51 with capacity 8 through 10, and s88 with capacity 5, which could not previously be solved within the time limit, have been solved to the target precision of 1% with low running time *and* a better solution given. For s88-c5-m2, s88-c6-m2 and s88-c7-m2, while not solved to the desired precision, near-optimal solutions have been found. However, some of the largest instances remain unsolved. As typical to best-bound searches, the algorithm may fail to find any solution regardless of target precision unless sufficient time and memory is given. This is something that limits the range of near-optimality the method can be used for. To improve the method's capability of determining upper bound, taking diving steps in both location and variable integrality, as opposed to *only* location, should be explored. Alternatively, integrating the routing heuristics in the algorithm to at least provide the same upper bound as in Table 3.4.

Following the observations made in Chapter 3; the underestimate in the lower bound is still high. E.g. for b30 the smallest rectangle encountered range from 67m (c5) to 135m (c10), and they are not solved to integrality, suggesting the optimality gap can be reduced significantly simply by the increase given by the integrality constraint and the distances.

Instance	Old	New	Saving (m)	Saving (%)
b30-c5	20739	16685.48	4053.52	19.55
b30-c6	18375	16396.98	1978.02	10.76
b30-c7	17781	16163.10	1617.9	9.10
b30-c8	16566	15535.99	1030.01	6.22
b30-c9	16553	15309.27	1243.73	7.51
b30-c10	16317	15309.15	1007.85	6.18
w51-c5	43539	42668.48	870.52	2.00
w51-c6	41587	41261.66	325.34	0.78
w51-c7	40789	40326.03	462.97	1.14
w51-c8	40242	39785.93	456.07	1.13
w51-c9	39752	39330.45	419.55	1.06
w51-c10	39541	39041.31	499.69	1.26
s88-c5-m2	64828	63597.66	1230.34	1.90
s88-c6-m2	62031	61248.04	782.96	1.26
s88-c7-m2	60667	60069.52	597.48	0.98

Table 4.2: Saving for best known solutions

Saving for the new substation location and cable layout compared to the optimal cable layout for the installed substations (not the installed cable layout), is given in Table 4.2, showing improvement in all instances where a solution was found. For exact location of the substations and cable layout, see Table A.1, Table A.2 and Figures in Appendix A.

Chapter 5

Concluding Remarks and Future Work

Classical problems in branch-and-cut, such as determining branching rules and node selection strategy, remains to be studied. For the underlying OVRP we have used a straight-forward formulation and branching strategy that is easy to implement. It remains open whether other formulations and cutting planes, such as [14], can be used and improve the LRP branch-and-cut.

Plastria [9] introduced a second phase to the BSSS method for determining a region of near-optimality, i.e. a region where no locations objective value exceeds a certain value. It is argued that "the determination of a region of near-optimality is crucial in location problems [...] any solution method for continuous location problems should be able to generate such near-optimality information in order to be useful in practice to a decision maker." Identifying the optimal value as has been the topic of this thesis is a first step. How the planarity constraint can be formulated for such upper-bound region remain unresolved.

Based on the proposed BSSS method we provided the extension to multi-facility problems and important improvements such as elimination of non-feasible area and the use local search. The method was tried but was very slow even for the small instances. An ad-hoc approach for finding solutions to the Substation location problems by the use of heuristics and relaxations was tried. Finally, we proposed a different approach for the LRP by the way of a branch-and-cut algorithm. We have provided near-optimal solution for both single- and multi-facility instances showing branch-and-cut as a feasible technique for small to medium sized problems.

Bibliography

[1] J. Bauer and J. Lysgaard. Offshore wind farm array cable layout problem. *J Oper Res Soc*, pages –. ISSN 0160-5682. URL http://dx.doi.org/10.1057/jors.2013.188.

[2] Gabor Nagy and Said Salhi. Location-routing: Issues, models and methods, March 2007. URL http://ideas.repec.org/a/eee/ejores/v177y2007i2p649-672.html.

[3] Saïd Salhi and Gábor Nagy. Local improvement in planar facility location using vehicle routing. *Annals of Operations Research*, 167(1): 287–296, 2009. ISSN 0254-5330. doi: 10.1007/s10479-007-0223-z. URL http://dx.doi.org/10.1007/s10479-007-0223-z.

[4] Martin Schwardt and Jan Dethloff. Solving a continuous location-routing problem by use of a self-organizing map. *International Journal of Physical Distribution & Logistics Management*, 35(6): 390–408, 2005. URL http://www.emeraldinsight.com/10.1108/09600030510611639.

[5] Barrow offshore wind farm. URL http://www.bowind.co.uk/pdf/Barrow_coordinates.pdf.

[6] Walney 1 offshore wind farm, 2010. URL http://www.dongenergy.com/Walney/News/data/Documents/WOW_I_grid.pdf.

[7] Sheringham shoal offshore wind farm, 2012. URL http://www.scira.co.uk/construction/foundationsmap.php.

[8] Pierre Hansen, Dominique Peeters, Denis Richard, and Jacques-Francois Thisse. The minisum and minimax location problems revisited. *Operations Research*, 33(6):pp. 1251–1265, 1985. ISSN 0030364X. URL http://www.jstor.org/stable/170636.

[9] Frank Plastria. Gbsss: The generalized big square small square method for planar single-facility location. *European Journal of Operational Research*, 62(2):163 – 174, 1992. ISSN 0377-2217. doi: 10.

1016/0377-2217(92)90244-4. URL http://www.sciencedirect.com/science/article/pii/0377221792902444.

[10] Zvi Drezner and Atsuo Suzuki. The big triangle small triangle method for the solution of nonconvex facility location problems. *Operations Research*, 52(1):128–135, 2004. doi: 10.1287/opre.1030.0077. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.1030.0077.

[11] Alok Aggarwal, MariaM. Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1-4):195–208, 1987. ISSN 0178-4617. doi: 10.1007/BF01840359. URL http://dx.doi.org/10.1007/BF01840359.

[12] Godfried Toussaint. Solving geometric problems with the rotating calipers. *In Proc. IEEE MELECON '83*, pages 10—02, 1983. doi: 10.1.1.40.2140. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.2140.

[13] E. Melachrinoudis and T.P. Cullinane. Locating an obnoxious facility within a polygonal region. *Annals of Operations Research*, 6(5):137–145, 1986. ISSN 0254-5330. doi: 10.1007/BF02026821. URL http://dx.doi.org/10.1007/BF02026821.

[14] Jens Lysgaard, Adam N. Letchford, and Richard W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Math. Program.*, 100(2):423–445, June 2004. ISSN 0025-5610. doi: 10.1007/s10107-003-0481-8. URL http://dx.doi.org/10.1007/s10107-003-0481-8.

Appendix A

Solution Data

Location data and layout for best known solutions.

Instance	E	N
b30-c5	480001.297228	5983127.078478
b30-c6	480193.475510	5982983.992996
b30-c7	479803.995895	5983779.199905
b30-c8	479180.864944	5983461.082077
b30-c9	479463.008331	5983646.217041
b30-c10	479828.982452	5983315.064850
w51-c5	466645.000305	5988342.997528
w51-c6	466512.261246	5988653.808975
w51-c7	466923.986847	5988093.003632
w51-c8	466668.507614	5988585.895157
w51-c9	465528.141861	5989344.325302
w51-c10	465528.141861	5989344.325302
s88-c5	376177.286865	5888853.260498

Table A.1: Substation location

Instance	E1	N1	E2	N2
b30-c5-m2	479496.000170	5982581.499250	481074.015398	5983221.534807
b30-c6-m2	479496.000170	5982581.500164	481621.491270	5982725.283962
b30-c7-m2	478695.278387	5982983.164217	479726.016863	5984118.495260
b30-c8-m2	478695.278387	5982983.164217	479726.016863	5984118.495260
b30-c9-m2	478695.278387	5982983.164217	479726.016863	5984118.495260
b30-c10-m2	478695.278387	5982983.164217	479726.016863	5984118.495260
w51-c5-m2	465028.745361	5987917.491211	467578.507050	5988886.894440
w51-c6-m2	466738.124359	5986199.745544	467200.528976	5989321.129532
w51-c7-m2	464162.575653	5988892.525452	468009.600632	5988794.368332
w51-c8-m2	464162.575653	5988892.525452	466892.524750	5989795.343414
w51-c9-m2	464162.575653	5988892.525452	466892.524750	5989795.343414
w51-c10-m2	463801.116074	5988373.660934	465216.601410	5991296.367889
s88-c5-m2	374237.325730	5890585.359449	377612.693847	5887034.437072
s88-c5-m2	374454.405807	5890854.002335	377366.866211	5887715.846008
s88-c5-m2	374798.149216	5890368.296799	377143.338409	5886776.949890

Table A.2: Substation locations

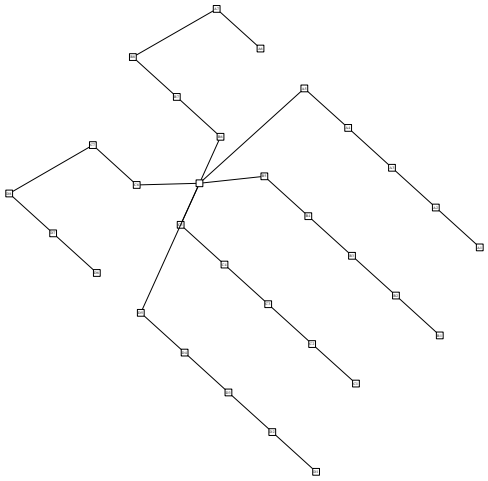

Figure A.1: b30-c5

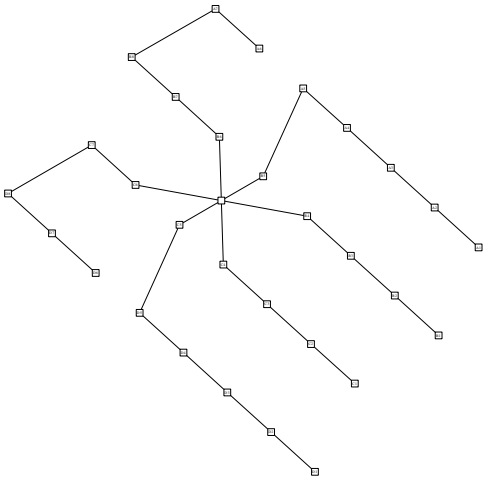

Figure A.2: b30-c6

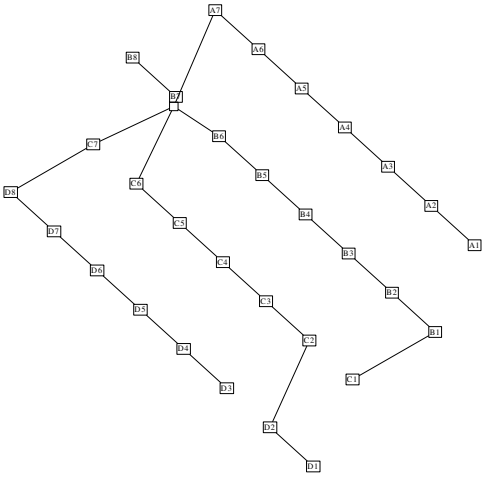

Figure A.3: b30-c7

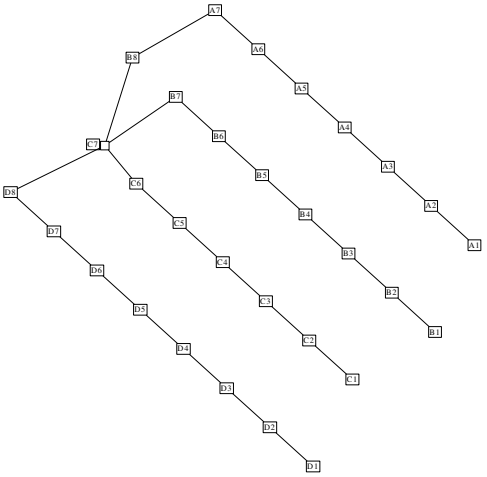

Figure A.4: b30-c8

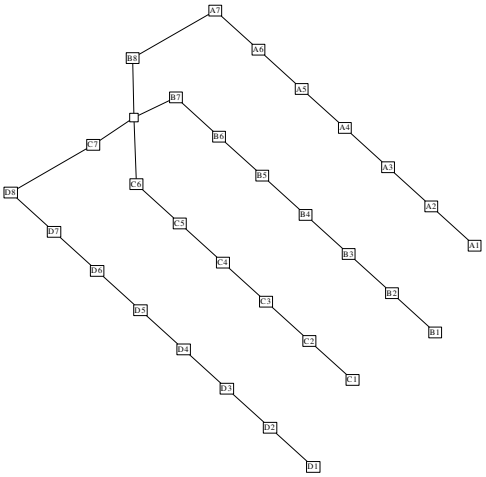

Figure A.5: b30-c9

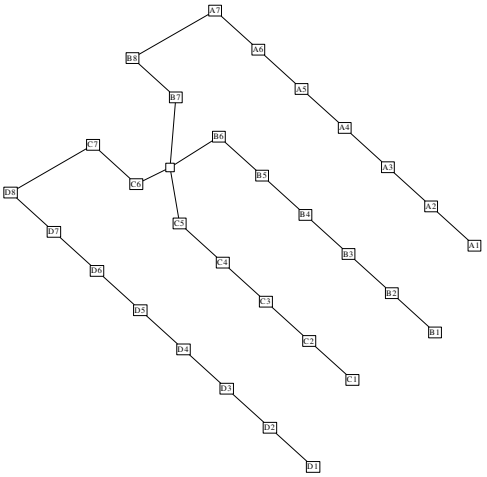

Figure A.6: b30-c10

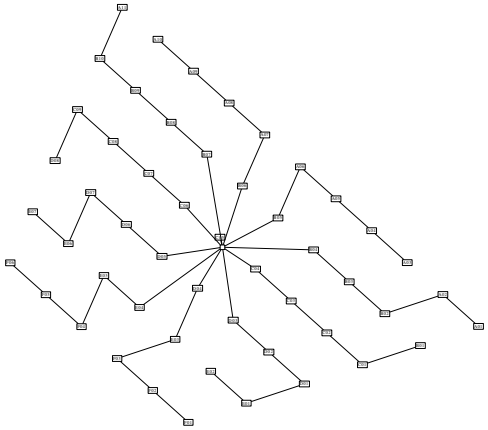

Figure A.7: w51-c5

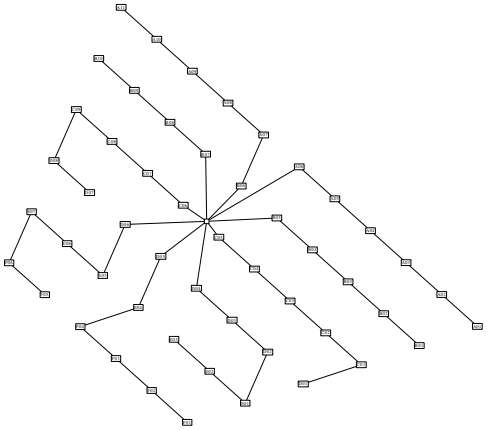

Figure A.8: w51-c6

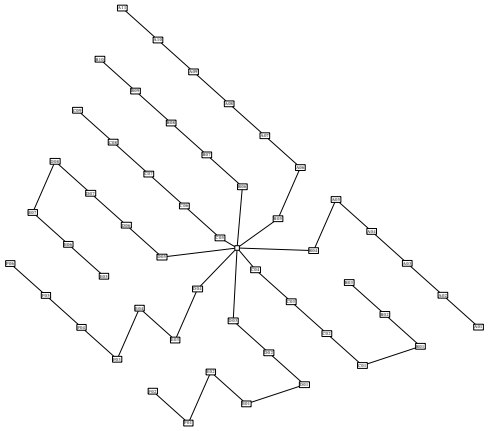

Figure A.9: w51-c7

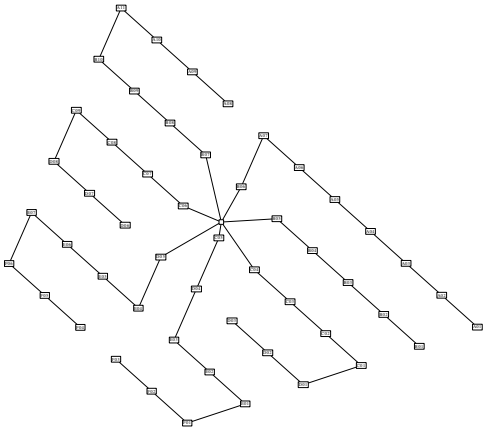

Figure A.10: w51-c8

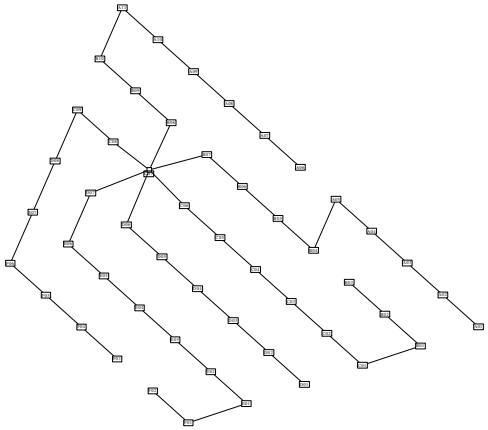

Figure A.11: w51-c9

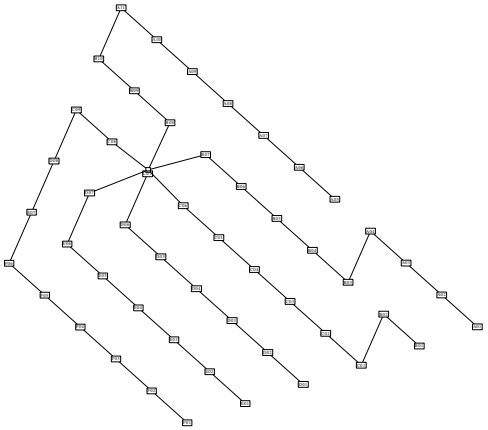

Figure A.12: w51-c10

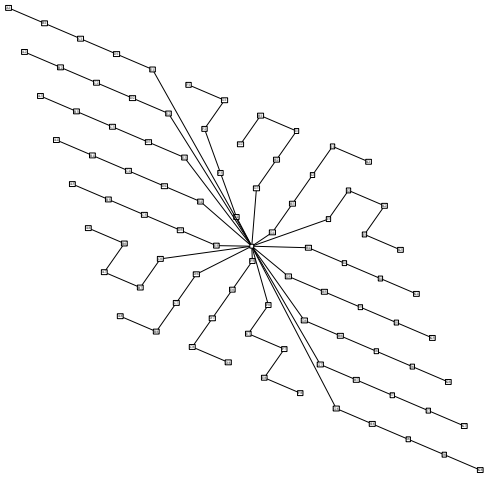

Figure A.13: w88-c5

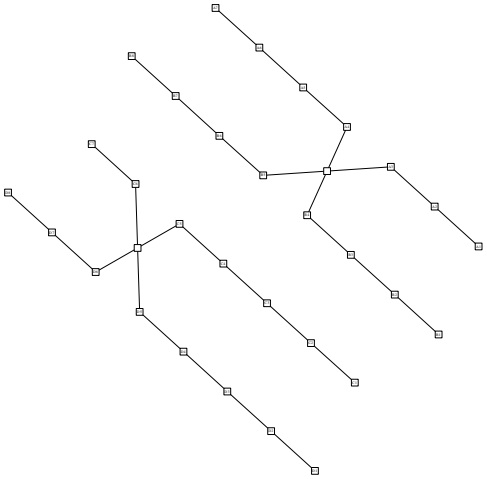

Figure A.14: b30-c5-m2

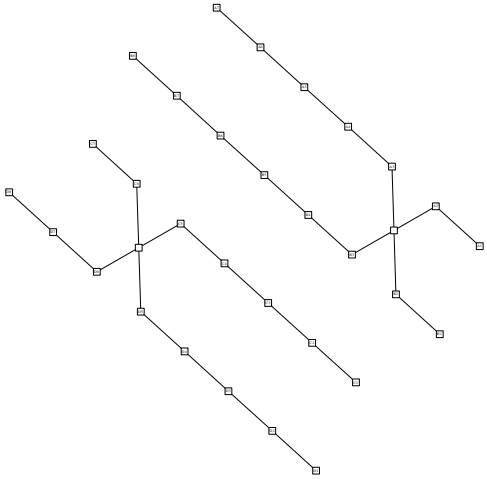

Figure A.15: b30-c6-m2

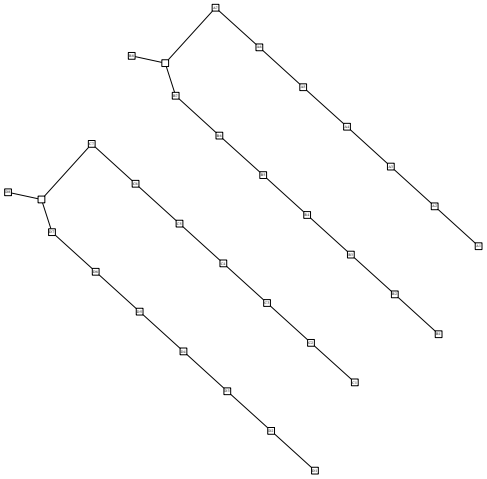

Figure A.16: b30-c7-m2

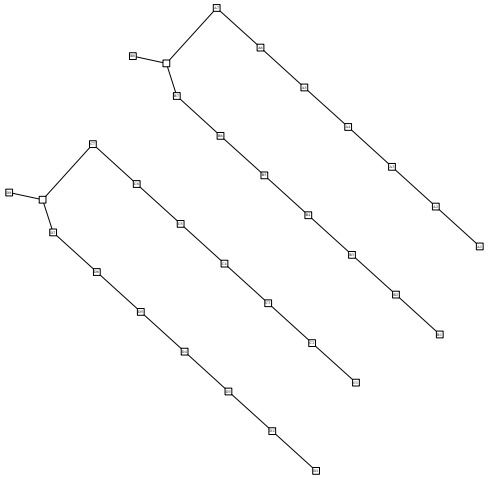

Figure A.17: b30-c8-m2

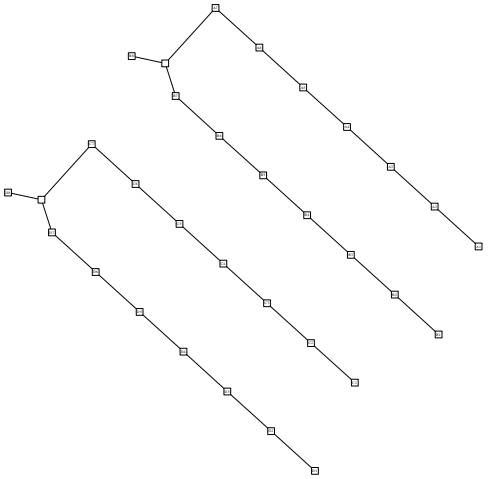

Figure A.18: b30-c9-m2

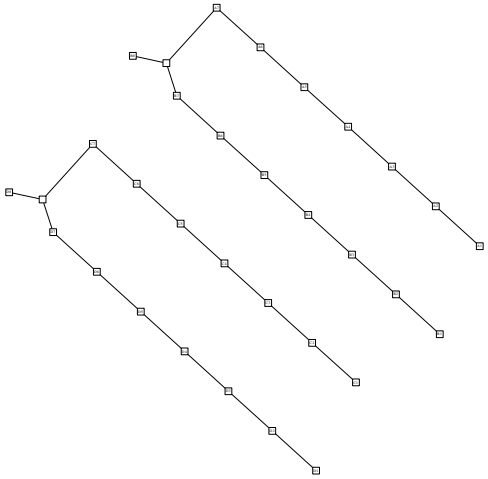

Figure A.19: b30-c10-m2

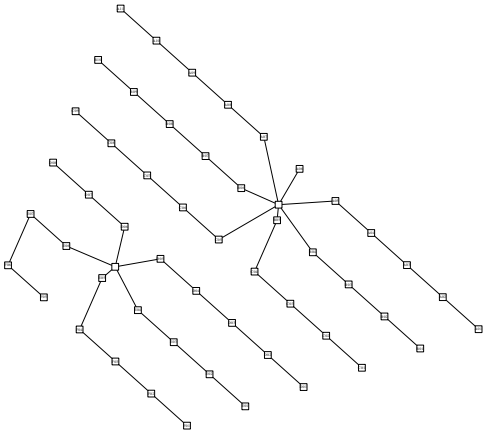

Figure A.20: w51-c5-m2

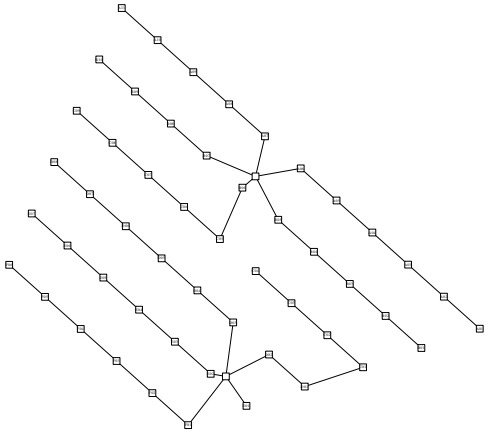

Figure A.21: w51-c6-m2

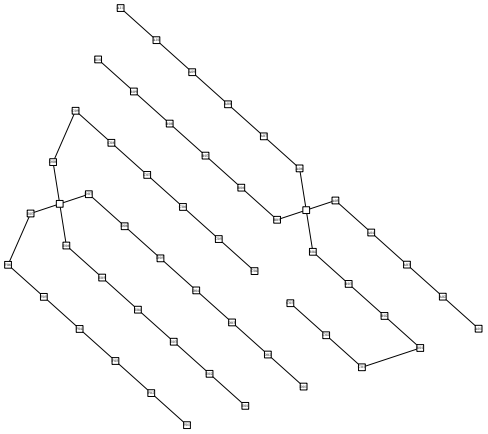

Figure A.22: w51-c7-m2

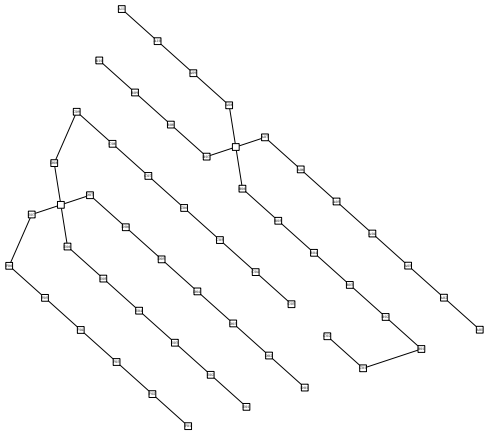

Figure A.23: w51-c8-m2

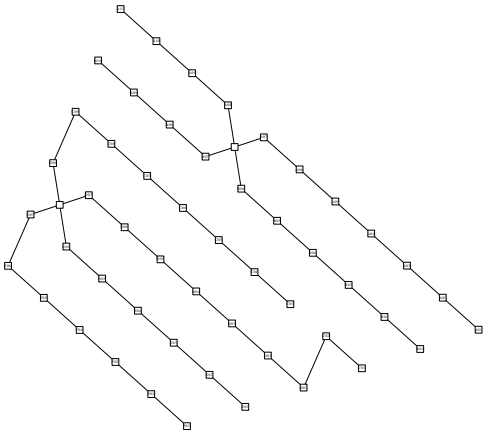

Figure A.24: w51-c9-m2

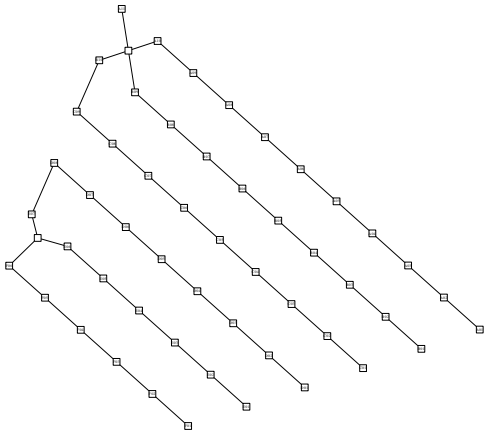

Figure A.25: w51-c10-m2

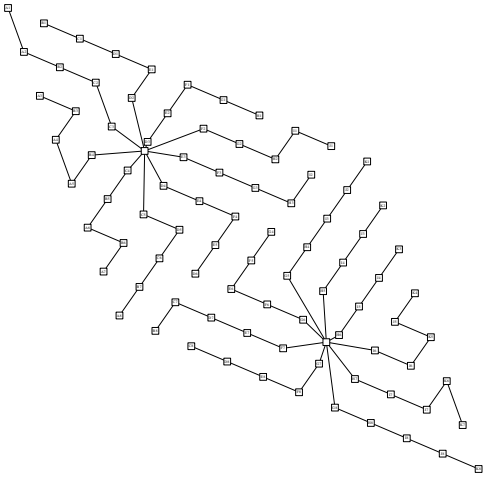

Figure A.26: s88-c5-m2

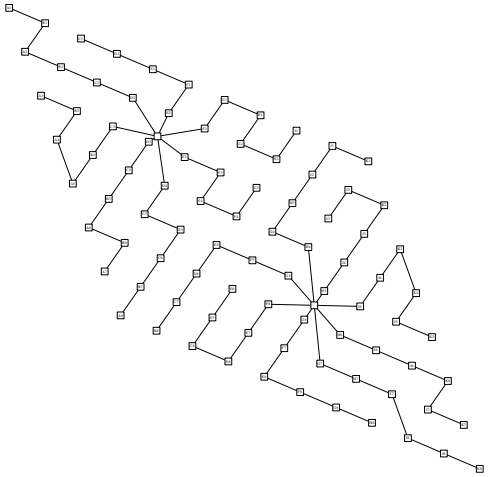

Figure A.27: s88-c6-m2

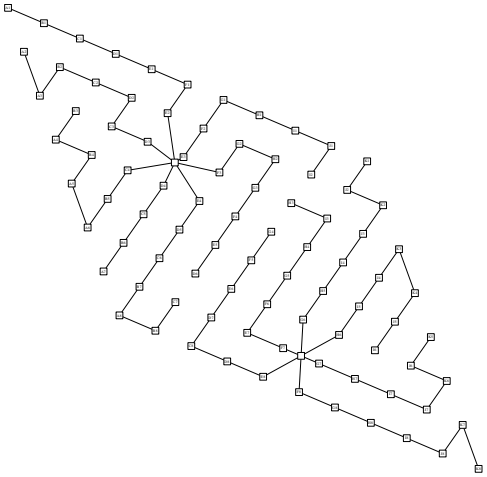

Figure A.28: s88-c7-m2