Power line route optimisation in a finite spatial grid

Nieke Aerts (TU Berlin), Emile Broeders (Utrecht University), Erik Bruin (Utrecht University), Ross J. Kang (Radboud University Nijmegen), Pedro Munari (Federal University of Sao Carlos)

1 Introduction

Power lines are integral to one of our most important infrastructure systems: the power supply network. They are run all over the world, across all kinds of landscape. While they are vital to our day-to-day life, they also have some negative influence on the environment, the view, and on other man-made structures. Building cost-effective power lines with limited environmental and social impact is a critical task. This paper considers a problem introduced by the British company NM Group, which specialises in surveying and planning for power line infrastructure. The main question which NM Group posed and which we address in our report is the following.

"How can we find a power line route of minimum cost?"

The cost of building a power line route is determined by many factors, not all of which are purely material. In order to specify these costs, NM Group has identified a few main factors, all of which are later incorporated into the mathematical model we study.

1. Occupation of an area by power line structures.

   (a) SVETIG.
   We assume that for each specific area of land the negative influence (cost) of existing infrastructure and landscape can be categorised. The categories, the so-called SVETIG factors, are Socioeconomic/Political, Vegetation, protected Environment, Terrain, technical Infrastructure, and Geotechnical. The way that this is

quantified is out of scope for this report, and for modeling purposes we may assume these are arbitrary positive weights in the plane, over which we integrate.

(b) Structure type.
Aside from the attributes of the area itself, the type of structure also may be taken into account, depending on the granularity. In particular, the area can be passed over with heavy electric cables suspended in the air or it can be occupied by the base of a power line support tower. Clearly the presence of a support tower is more disruptive than that of a suspended cable.

2. Material construction costs.
The more lengthy the power line is, the more costly the materials used for that line, in linear proportion. For costs associated with the construction and placement of towers, two further (multiplicative) penalty factors come into play.

(a) The distance between two consecutive towers. ("Stretch penalty")
When building a power line, the cables go straight from the top of one tower to the top of the next, i.e. along a straight line segment in the plane. In the third dimension, however, gravity takes its toll in terms of both the weight and sag of the suspended wire between those towers. Therefore, the further apart two consecutive towers are placed, the greater must be the load-bearing capacity and height of the two towers. Extremely long distance between consecutive towers is highly costly or impossible. Rarely is a very long span required, when there are no other options, for instance if the power line must cross a river or canyon.

(b) The angle between the incoming and outgoing lines of a tower. ("Angle penalty")
When the power line segments supported by one tower form a sharper angle (with respect to the plane), the weight of the cables exerts a force on the tower roughly in the direction of the angle's bisector. To counteract this force, some counterbalance or reinforcement of the tower is necessary. For instance when the angle is roughly 180°, only a simple tower is needed, whereas if the angle is off by more than 10°, then a stronger type of tower must be used.

In fact, the penalty factor is more akin to a step function, due to the need for increasingly robust types of tower.

1.1 Discretisation and macro/micro separation

For practical reasons, the problem as stated above must be discretised — that is, the spatial region under consideration is dissected using a regular grid, each cell of which is assigned a weight according to an amortised SVETIG factor. Moreover, NM Group deemed it necessary to make a separation into two levels of discretisation, the second of which may be interpreted to be a refinement of the first. Naturally, we refer to the first as the macro scale problem and the second as the micro scale problem. Later, the reader doubtless will notice that the macro and micro level problems are qualitatively distinct, and as such there could be justifiable objection about whether the combination of solutions to these two sub-problems constitutes an overall solution to the original problem as stated above. However, in our work we have taken this separation heuristic as given, partly because it is justified by the limitations of data-acquisition resources, and partly because we also propose heuristic and/or approximative approaches due to computational difficulties inherent to the global problem. It is worth noting that in our report we have independently chosen the type of tessellation — be it hexagonal, triangular, square — out of convenience, though with suitable routine modifications our methods apply to any regular grid pattern, at either scale. In summary, we split the problem as follows:

1. Macro scale.
   At this level, NM Group has indicated that the overall area of consideration is typically a region of about 100 by 100 kilometres. This area is divided into cells of approximately 2 kilometres in diameter. The desired output of this sub-problem is a "rough" routing consisting of a connected sequence of macro cells, which we refer to as a corridor.

2. Micro scale.
   The small subset of the cells identified at the macro scale are examined more closely for the micro scale problem. The selected macro cells are divided according to a finer grid consisting of micro cells at most 50 metres in diameter. The desired output of this sub-problem is the overall desired minimum cost power line route.

A major reason for the two-layer separation of the problem is based on costs incurred by NM Group for determining the SVETIG weighting factors. The SVETIG factors applied to the macro cells are inferred from cheap or free satellite-based imagery and mapping data. By contrast, it is assumed that, after the specific desired macro cells are identified, special surveillance missions are carried out over those regions, with the aid of lasers mounted on helicopters and similar (in particular, rather costly) data-acquisition methods, in order to determine the more detailed SVETIG factors used at micro scale.

Since the micro level data is gathered and formulated by NM Group itself, there is some flexibility in the defined size of micro scale cells. Naturally, the quality of the solution may improve by using a finer tessellation on input; however, much more computation time might be required to obtain the output.

1.2 Routing costs at different scales

As alluded to above, we permit significant differences in treatment of the problem when considered at different scales.

From the macro viewpoint, one may interpret it that we are unable to "see" precise details of the routing: we need not account for the most accurate length of the route, the specific number of towers placed, nor the corresponding angles. Instead, we only roughly account for the presence and SVETIG impact of power line routing through a given macro cell. As we will see later, this macro problem reduces to a relatively simple minimum cost path problem, solved efficiently using Dijkstra's algorithm.

For the micro viewpoint, we narrow in on the output of the macro scale solution, i.e. we restrict attention to the subset of macro cells identified in the first step and divide them into smaller cells. Here, more intricate aspects of the routing problem come into effect. Specifically, for a placement of the towers at the centres of some micro cells, we calculate the cost of the corresponding power line routing by incorporating all the detailed cost contributions. That means we consider not only the (detailed) SVETIG factors, but also structure type, inter-tower distances, and angles at support towers.

Although essentially any reasonable mathematical combination of the cost factors described above can be handled by our method, we assume for concrete-

ness and simplicity that the following cost functions apply for routing through a given micro cell:

- The cost of placing a tower is
  - the material unit cost of a support tower
  - multiplied by the SVETIG factor for placing a tower in that cell
  - multiplied by the stretch penalty
  - multiplied by the angle penalty.
- The cost of suspending power line is
  - the material cost per unit length of power line
  - multiplied by the SVETIG factor for suspending power line in that cell
  - multiplied by the length of power line intersecting that cell.

We require no special assumptions about the unit costs, SVETIG factors, or penalty functions, except merely that they are fixed or monotone (decreasing with respect to angle or increasing with respect to stretch length) and that they are provided to us beforehand by NM Group.

1.3 Avoidance of "rubberbanding"

As a side remark, it is worth noting that NM Group originally gave a slightly different formulation of the power line routing problem. They had suggested to split the problem into three sub-problems, the macro and micro layers followed by an ad hoc "rubberbanding" protocol. Essentially, this delays all angle penalty considerations until the end. In other words, they suggested first to find a micro solution excluding angle considerations, and then afterwards to run a local perturbation procedure to fix and compensate for any undesirable angle penalties.

In our work, we have circumvented the need for (or at least limited the benefit of) this local adjustment procedure by directly incorporating angle penalties

into the mathematical model. Of course, because of discretisation, there is still the possibility of cost saving by "rubberbanding" upon the output of our suggested algorithms, but typically this saving will be of less interest as it will be much smaller than it was for their original approach. In order to eliminate as much as possible the potential "rubberbanding" savings, the discretisation could be made as fine as possible, but this would at the same time increase computational requirements. We in fact recommend NM Group survey according to micro cells that are the size of a tower's base (rather than 50 metres in diameter).

1.4 Outline and overview

In this report, we are mainly concerned with algorithmic solution strategies for the above discretised problem. That is, subject to pre-determined unit costs, SVETIG factors, and penalty functions, we demonstrate how to effectively compute the desired optimal power line route, by solving first the macro and then the micro level sub-problem. We first show that the macro level problem can be solved both exactly and efficiently by a relatively straightforward application of Dijkstra's algorithm; we describe this solution in Section 2. We then show that the micro level problem can be solved both exactly and efficiently using a more involved modification and application of Dijkstra's algorithm; we describe this solution in Section 3. The efficiency for the exact micro level solution is of a theoretical nature (i.e. the computational problem is solvable in polynomial time), and unfortunately our algorithm in unvarnished form is unlikely to produce timely solutions for the problems typically encountered by NM Group in practice. We therefore found it appropriate to propose various potential practical approaches to solving the micro level problem which we describe in Section 4.

2 Macro step efficient exact solution

In the macro step a corridor is to be found. Within this corridor lies the minimum cost path that connects the start and end points of the power line route. Afterwards, more detailed information about the corridor is gathered. Our algorithm will find the minimum cost corridor.

First an auxiliary weighted graph is built, then a minimum cost path in this graph is found by applying for example Dijkstra's shortest path algorithm [2]. We describe how to construct the weighted graph such that the shortest path output is a minimum cost corridor. Moreover, the output corridor minimises the Euclidean length of a polygonal curve between its end cells, optimised over all such polygonal curves whose corners lie only at the centres of cells.

The previous algorithm used by NM Group was based on finding the minimum cost path in a graph with weight on each of the vertices. In the hexagonal grid this graph is obtained by letting every hexagon be a vertex which is connected to all its neighbouring hexagons (see Figure 1). A problem that arises is that the two paths connecting the top-left and bottom-right hexagons in Figure 2 both go through the same number of hexagons. If every hexagon has the same cost, the algorithm of Dijkstra might well select the top-right path. When the actual power line is built it will not necessarily pass through the centers of the hexagons. Therefore the other path turns out to be much cheaper, as the eventual total length is shorter. Hence, our first goal is to devise an algorithm that selects a path of minimum cost and takes into account some measure of Euclidean length of the path.

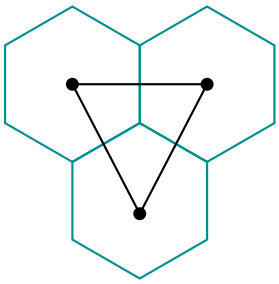

Figure 1: Each hexagon gets one vertex inside and every vertex is connected with the six vertices in the six neighbouring hexagons.

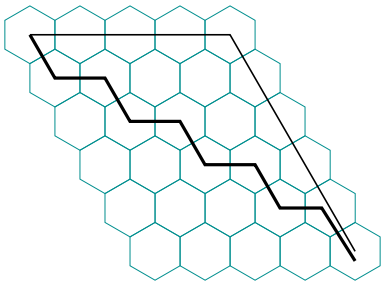

Figure 2: The two paths connecting the top-left and bottom-right hexagons go through the same number of hexagons.

2.1 Hexagonal grid

Instead of changing the algorithm, a different graph is constructed. The new graph also encodes the Euclidean distance between the two neighbours in a path as a cost. We will first describe how the graph is obtained and then show why a minimum cost path algorithm will output the desired path.

Each hexagon gets six vertices inside which are cyclically connected forming a new hexagon. Let $v(h, e)$ be the vertex in the hexagon h closest to border-edge e. Let h^* be the neighbouring hexagon on the edge e, see Figure **??** where e is the dashed edge between the hexagons h and h^*.

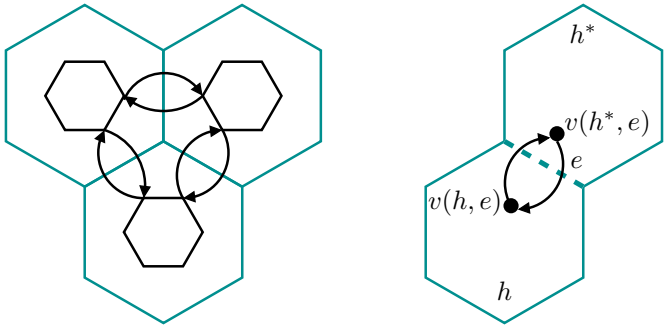

Figure 3: Left: Each hexagon (in cyan) gets a hexagon inside, where each corner is a vertex. Each vertex is connected with an undirected edge to its neighbours in the hexagon and with a directed edge to and from the closest vertex in the neighbouring hexagon. Right: the vertices inside a hexagon are labeled $v(h, e)$ for begin in hexagon h closest to border e of the hexagon.

There is a directed edge from $v(h, e) \rightarrow v(h^*, e)$ having weight equal to the cost of the hexagon h^*, i.e., the weight of the hexagon it enters. There is also an edge from $v(h^*, e) \rightarrow v(h, e)$ having weight equal to the cost of the hexagon h, again the weight of the hexagon it enters. Each interior edge in the hexagon has very small weight. Let c_{min} be the cost of the hexagon with minimal cost, and n the number of hexagons. Then every edge interior to a hexagon is assigned a weight $c_{min}/(3n)$.

A minimum cost path in this graph is a minimum cost path on the map. Moreover, for each inner polygon in this path, the distance between the centers of

its two neighbours is taken into account as a measure of the length of the path. In Figure 4 two examples are given that illustrate how the distance measure works. Let x_h be the distance between the center points of the neighbours of a hexagon h, that is interior to a path. The sum of x_h over all inner hexagons h of the path is less for the red paths than for the black paths. The red paths are also of lower cost than the black paths since there are less interior edges used.

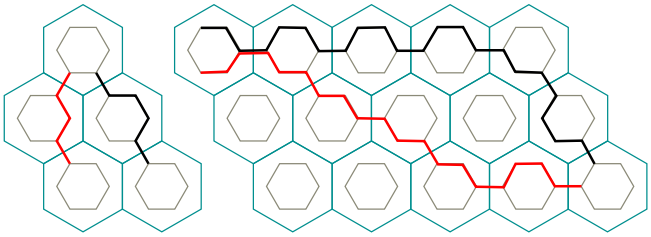

Figure 4: The path on the left (in red) is of lower cost, as it uses fewer interior edges.

2.2 Square grid

In a similar way as for the hexagonal grid, an auxiliary graph can be constructed for the square grid. A minimum cost path in this graph relates to a minimum cost routing in the square grid. Each square gets four vertices inside, cyclically connected. Each vertex has an outgoing edge to the vertex in the next square. This edge gets the weight of entering the neighbouring square. Let c_{min} be the cost of the square with minimal cost, and n the number of squares. Then every edge interior to a hexagon is assigned a weight $c_{min}/(2n)$.

Finding a minimum cost path in this graph gives a minimum cost path on the map such that expensive squares are avoided and the path takes into account some measure of Euclidean length of the path.

3 Micro step exact solution

The corridor that is obtained in the macro-step is now filled in with more detail. We assume that the grid on top of the map is fine, that is, the area covered by one polygon (triangle, square or hexagon) is close to the area needed to build a post. This ensures that we do not have to take into account where in a polygon the post is placed, each post will be placed in the centre of a polygon. For each polygon we now consider the more specific costs as defined in Section 1.2.

In Section 3.1 we will describe how to build an auxiliary graph, with costs on the edges, such that a minimum cost path in this graph gives a minimum cost solution to the micro problem. Dijkstra's algorithm can be used to find such a minimum cost path in the auxiliary graph. In Section 3.2 we will show that there is a bijection between the paths in the auxiliary graph and the routings in the micro corridor. Moreover we show that the minimum cost paths in the auxiliary graph are in bijection to the minimum cost routings in the micro corridor. In Section 3.3 we describe how to calculate the costs on the edges of the auxiliary graph in the case of a square grid. We conclude by discussing the complexity of Dijkstra's algorithm on the auxiliary graph in Section 3.4.

3.1 Preprocessing for Dijkstra's algorithm

We assume that the maximal distance between two towers is bounded by some value K. This is a valid assumption, as there is a value for which the cost that comes from the distance between two consecutive posts becomes too high to be feasible or reasonable. Recall that this cost is denoted by stretch penalty. We will construct a directed graph such that a minimum cost path in this graph describes a minimum cost route in the original grid.

Suppose that from a polygon we can reach k other polygons with one stretch, i.e., the two polygons are consecutive posts in a path. Note that this includes steps of all lengths up to the maximum length K (see Figure 5). Recall that the posts are placed in the centre of a polygon and the maximum distance between two posts is thus K. Therefore, the set of reachable polygons from a polygon p is the set of polygons with centre point at distance at most K from the centre point of p.

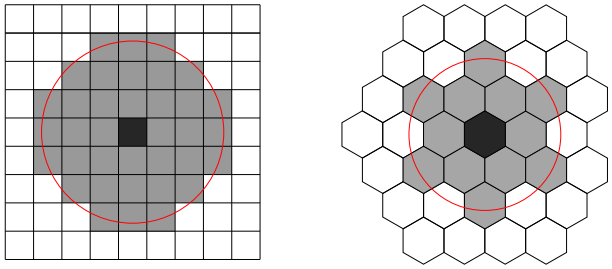

Figure 5: The dark square (hexagon) represents the current location, the lighter areas represent the possible landing spots coming from the dark area.

Let A be the starting point, B be the target point and let p be any polygon. For each polygon p let $n_1, \ldots, n_k$ be the reachable polygons. We introduce k dummy vertices for p. Each dummy vertex is labeled (p, n_l) for $l = 1, \ldots, k$. A dummy vertex (p, n_l) represents that *the current location is* p and this location is reached by *coming from* n_l. For A there is only one vertex introduced, denoted by $(a, \emptyset)$, which is the starting point of the path. For B there is a vertex (b, j) for all polygons j from which B can be reached and a vertex $(\emptyset, b)$ which is the end point of the path.

The vertex $(a, \emptyset)$ has only outgoing edges: $(a, \emptyset) \to (m, a)$ for every reachable polygon $m = 1, \ldots, k$ of A. The vertex $(\emptyset, b)$ has only incoming edges: $(m, b) \to (b, \emptyset)$ for every reachable polygon $m = 1, \ldots, k$ of B. For every other vertex (p, n_l) there is a directed edge to (n_m, p) for each reachable polygon $m = 1, \ldots, k$. A small example is given in Figure 6. The weight on such an edge between (p, i) and (j, p) represents the costs made for placing a post in p.

The costs involved in placing a post in p between i and j are:

- the material unit cost of a support tower;
- multiplied by the cost of the placement at p (the SVETIG factor);
- multiplied by the cost of the angle structure needed at p;
- multiplied by the stretch penalty, which depends on the maximum of the two distances between p and i and between p and j.

There are also costs of suspending the power line. On an edge $(p, i) \to (j, p)$

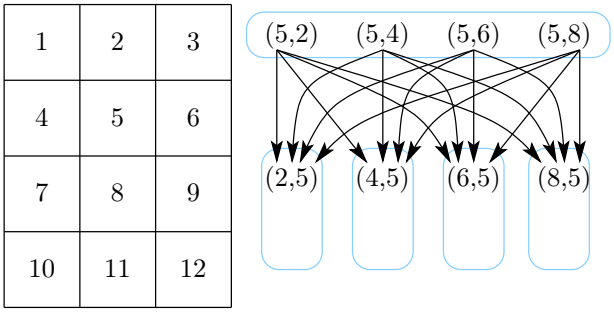

Figure 6: Part of the graph that is built: the square labeled 5 can stretch to any of its direct neighbours 2, 4, 6 and 8, therefore the vertices $(5,2), (5,4), (5,6)$ and $(5,8)$ are introduced. From each of these there is a directed edge to all the reachable neighbours: $(2,5), (4,5), (6,5)$ and $(8,5)$.

we incorporate the suspending costs between p and j. The suspending costs between i and p will be taken into account in the edge that ends in (p,i).

Recall that the suspending costs per polygon are:

- the material cost per unit length of power line;
- multiplied by the SVETIG factor for suspending the power line in this polygon;
- multiplied by the length of the power line intersecting this polygon.

All the relevant information for the placing costs in p and the suspending costs between p and j is known at the time of selecting the edge $(p,i) \to (j,p)$. Therefore, all the costs can be incorporated on this edge.

The outgoing edges of $(a,\emptyset)$ are charged in the same way, the placing of the post in A and the suspending costs between A and the target. The incoming edges of $(\emptyset,b)$ are only charged with the placing cost of the tower in B. In Section 3.3 we will describe in detail how to calculate the costs on an edge in the case of a square grid.

The directed graph has at most k dummy vertices for each polygon in the grid. The polygons on the boundary will give rise to strictly less than k dummy vertices. Suppose the grid has m polygons. Then the graph will have approx-

imately $k \cdot m$ vertices. Each vertex has at most k outgoing edges. Therefore the resulting graph will have approximately $k^2 \cdot m$ edges.

On the resulting directed graph, Dijkstra's algorithm or any other shortest path solver can be applied to obtain the minimum cost route in the original grid. We will discuss the running time of this algorithm later in Section 3.4. First we will show that there is a bijection between a minimum cost path in the directed graph and a minimum cost routing in the grid.

3.2 Validation

Let A and B be the points that need to be connected by a power line. Let $\mathcal{P}$ be the collection of paths that connect A and B, such that the distance between two neighbouring posts is at most K. Let k be the maximum number of polygons at distance at most K. Let D be the directed graph obtained as described in the previous section, where $(a, \emptyset)$ and $(\emptyset, b)$ represent A and B and D is such that at most k polygons are at distance at most K. Let $\mathcal{Q}$ the set of directed paths from A to B in D.

Claim. There is a bijection ϕ between $\mathcal{P}$ and $\mathcal{Q}$ such that for every pair $p \in \mathcal{P}$ and $\phi(p) = q \in \mathcal{Q}$ the costs of p and q are equal.

Proof. Let $p \in \mathcal{P}$ be a path from A to B using the polygons $A = i_0, i_1, \ldots, i_m, i_{m+1} = B$. We show how to obtain $q = \phi(p)$. Start with $q = ((a, \emptyset), (i_1, a))$. For every step between two polygons i_j and i_{j+1} in p, the distance is at most K. Therefore, the directed edge $(i_j, i_{j-1}) \to (i_{j+1}, i_j)$ must exist in D and we can add (i_{j+1}, i_j) to q. Last, we add $(\emptyset, b)$ to q and we have $q \in \mathcal{Q}$.

On the other hand, every path $q \in \mathcal{Q}$ is mapped to a path $\phi^{-1}(q) = p \in \mathcal{P}$. Let q be the path:

$$(a, \emptyset) \to (i_1, a) \to \ldots \to (i_j, i_{j-1}) \to (i_{j+1}, i_j) \to \ldots \to (b, i_m) \to (\emptyset, b)\,.$$

The path p can be obtained from q by selecting the path in P that consecutively visits the polygons $A, i_1, \ldots, i_m, B$. This path must exist in P since all steps in q imply that the consecutive posts are at most at distance K from each other.

Let $p \in \mathcal{P}$ and $\phi(p) = q \in \mathcal{Q}$. The costs of the polygons used (with placing a post or by suspending over it) in p are the same as in q. The distances between

two subsequent posts and the angles at the posts are also the same in p as in q. Therefore, the cost of p must be equal to the cost of q. □

It follows that if p is a minimum cost path in $\mathcal{P}$ then $\phi(p) = q$ must be a minimum cost path in $\mathcal{Q}$. Therefore, the solution that a minimum cost path algorithm will give when applied to the directed graph, is a minimum cost path within the macro corridor for the original problem.

3.3 Calculation of costs

In this section we will describe how to calculate the cost of an edge in the directed graph. We will consider the case of a square grid, the calculations can be done for a hexagonal grid as well, if necessary. In the square grid we let the topleft corner be the origin. We start by introducing some notations.

- $p, q, r, \ldots$ generally denote points in the Euclidean plane, which are the center point of the squares $s_p, s_q, s_r, \ldots$ respectively.
- i, j generally denote the horizontal respectively vertical distance from the origin, and therefore also indicate a column (i) and a row (j).
- $c_{\mathcal{S}}(p, q)$ is the cost function of the wires going over the area between p and q, say the suspension costs.
- $c_{\mathcal{P}}(p, q, r)$ is the cost of placing a post in q coming from p and going to r, which is the landing cost of the tower for the given angle.

We consider the edge $(s_q, s_p) \to (s_r, s_q)$, this denotes that we have subsequent post in p then q and then r. The weight of this edge is:

$$c_{\mathcal{S}}(q, r) + c_{\mathcal{P}}(p, q, r) \ .$$

When $(s_q, s_p) \to (s_r, s_q)$ is chosen, the costs of everything that has to do with the post in s_q is charged as well as the cost of the wiring on top of the squares between q and r.

For the incoming edges to $(\emptyset, s_B)$, there is only the costs of placing a post in B:

$$(s_B, s_q) \to (\emptyset, s_B) \quad \text{has cost} \quad c_{\mathcal{P}}(q, B, \emptyset) \, .$$

3.3.1 Calculation of suspension costs

The function $c_{\mathcal{S}}$ depends on the material cost, the SVETIG factors of the polygons over which the suspending takes place and the length of the power line over such a polygon. To capture this in the definition of $c_{\mathcal{S}}$ we introduce some notation.

- l_{pq} is the line on which p and q lie.
- $\text{cros}_{\mathcal{V}}(pq, x)$ is the crossing of the line l_{pq} with the vertical at coordinate x, $\text{cros}_{\mathcal{H}}(pq, y)$ is the crossing of the line l_{pq} with the horizontal at coordinate y.
- S_{ij}, S_p represent the SVETIG factor of the squares s_{ij} and s_p respectively.
- $\text{len}(pq)$ is the length of the line-segment between p and q and $\text{len}(s_{ij}, pq)$ is the length of the line l_{pq} in the square s_{ij}.
- W represents the cost of the wire per length factor.

The value $c_{\mathcal{S}}(p, q)$ is calculated by summing over all columns in which the line segment appears (i), then over the rows in which the line segment appears in the i-th column (j). The sum then consists of the length of the line segment within square s_{ij} ($\text{len}(s_{ij}, qr)$) times the weight of the square s_{ij}. If necessary, the additional costs of the wires of length $\text{len}(p, q)$ can be added. Formally, we have the following expression:

$$c_{\mathcal{S}}(p, q) = \text{len}(pq) \cdot W + \sum_{i=\lfloor p_x \rfloor}^{\lceil q_x \rceil} \sum_{j=\text{cros}_{\mathcal{V}}(pq,i)}^{\text{cros}_{\mathcal{V}}(pq,i+1)} S_{ij} \cdot \text{len}(s_{ij}, pq)\,.$$

We will now explain how to calculate the crossing points and the length of the line-segment in a particular square. Any point on the line that goes through p and q is described with:

$$l_{pq}(t) = (q - p)t + p\,.$$

The line segment between p and q is given by varying t from 0 to 1. To obtain a value for $\text{cros}_{\mathcal{V}}(pq, i)$, we use the time t^* such that the x-coordinate of $l_{pq}(t)$

is i. Here $\hat{\mathbf{e_y}}$ denotes the vertical unit vector.

$$\text{cros}_{\mathcal{V}}(pq, i) = \begin{cases} p_y & \text{if } t^* \leq 0 \\ q_y & \text{if } t^* \geq 0 \\ l_{pq}(t^*) \cdot \hat{\mathbf{e_y}} & \text{otherwise} \end{cases}$$

For every square s_{ij} the point of entry (e_n) and the point of exit (e_x) of the line l_{pq} are computed. The length of the line-segment in the square is then equal to the Euclidean distance between e_n and e_x, that is,

$$\text{len}(s_{ij}, pq) = ||e_n - e_x||_2 \,.$$

3.3.2 Calculation of tower costs

The function $c_{\mathcal{P}}$ depends on the material cost of a support tower, the SVETIG factors of the polygon in which the tower is built, the angle structure that is needed for this tower and the stretch penalty. To capture this in the definition of $c_{\mathcal{P}}$ we introduce some notation.

- $P(q)$ represents the cost of placing a post in s_q.
- $L(p, q, r)$ represents the stretch penalty, that is, the extra costs at q induced by the maximum distance between q and its two neighbours.
- $A(p, q, r)$ represents the angle penalty that depends on the angle at q.

Using this notation, we calculate the cost of placing a post in q coming from p and going to r as:

$$c_{\mathcal{P}}(p, q, r) = P(q) \cdot L(p, q, r) \cdot A(p, q, r) \,.$$

Both L and A are step functions.

The function L depends on the maximum of $\text{len}(pq)$ and $\text{len}(qr)$. There is a table that contains the value of $L(p, q, r)$ given $\max(\text{len}(pq), \text{len}(qr)) < \ell$ for different values of ℓ.

The angle at q can be computed using the coordinates of p, q and r. Consider the two vectors $\vec{a} = (p_x - q_x, p_y - q_y)$ and $\vec{b} = (r_x - q_x, r_y - q_y)$. The angle

at q is the same as the angle between the two vectors. The angle between two vectors, denoted by θ, can be computed using the default dot product

$$\vec{a} \cdot \vec{b} = ||\vec{a}||_2 ||\vec{b}||_2 \cos(\theta)$$

where $\vec{a} \cdot \vec{b}$ represents the dot product between two vectors. Let p_x, p_y (respectively q_x, q_y and r_x, r_y) represent the coordinates of p (respectively q and r) then the angle at q is given by:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}||_2 ||\vec{b}||_2} = \frac{(p_x - q_x)(r_x - q_x) + (p_y - q_y)(r_y - q_y)}{\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}\sqrt{(r_x - q_x)^2 + (r_y - q_y)^2}} .$$

The function A depends on the angle at q. There is a table that contains the value of $A(p, q, r)$ given that the angle at q is at most α for different values of α.

3.4 Complexity of solving shortest path problems

In the previous section we have described how the problem can be formulated as a minimum cost path problem in a directed graph. This is equivalent to a weighted shortest path problem in a directed graph. There are several algorithms known that solve this problem in polynomial time. An example is the famous algorithm of Dijkstra [2]. Dijkstra's original algorithm has a running time of $\mathcal{O}(|V|^2)$ where $|V|$ denotes the number of vertices in the graph.

The implementation due to Fredman and Tarjan [3] is asymptotically the fastest known shortest-path algorithm for directed graphs with non-negative weights. This algorithm has time complexity $\mathcal{O}(|E| + |V| \log(|V|))$ and space complexity $\mathcal{O}(|V|^2)$, where $|E|$ is the number of edges. Consider the graph we have built in the previous sections. The graph has k^2m edges and km vertices, where k is the number of reachable neighbours and m is the number of polygons in the micro corridor. This relates to a running time of $\mathcal{O}(k^2m + km \log(km))$ and space requirement of $O(k^2m^2)$.

Let us briefly consider the rough boundaries of a problem that could arise in practice (as we were informed by NM Group). Suppose that the grid consists of squares and each square covers 20×20 square metres. The distance between A and B is approximately 50 kilometres and the micro corridor is approximately

50 kilometres long and 4 kilometres wide. This yields $m \approx 2500 \times 200 = 500,000$. Suppose the distance between two posts is at most 2 kilometres. Recall that k is the number of squares whose centre point lies within distance $K = 2$ kilometres, from the centre point of some chosen square. For a square grid k can be found using the solution to the so-called Gauss circle problem: "How many lattice points are there in a circle with radius r that is centred at the origin". For bounded radius this number can be found using the following formula [7]:

$$N(r) = 1 + \sum_{i=0}^{\infty} \left(\left\lfloor \frac{r^2}{4i+1} \right\rfloor - \left\lfloor \frac{r^2}{4i+3} \right\rfloor \right).$$

In our case the lattice points are the centres of the squares, therefore one unit relates to 20 metres. The radius in units is $2000/20 = 100$ and $N(100) = 31,417$. In this example the directed graph will have approximately $500,000 \times 31,417 \approx 15.8$ billion vertices and even more edges.

Although we have shown that the micro level problem can be solved efficiently in theory, the above rough calculation suggests that in practice it may be prohibitively expensive, both in terms of computational time and memory storage requirements, to obtain an exact solution. In the next section, we will describe some heuristic algorithmic strategies which could be efficient in practice.

4 Heuristics and metaheuristics

In this section, we propose a set of simple heuristics that have the purpose of quickly providing solutions to the micro level problem. First, we present the main ideas and the algorithmic description of constructive heuristics that try to build initial paths. Then, we propose several improvement heuristics based on quick perturbations of paths. Finally, we describe ways of combining the proposed heuristics by using a metaheuristic, namely the Tabu Search strategy.

To describe the heuristics, we use the following nomenclature and notation. Let N be the number of polygons used in the micro step, so that $H = \{h_1, h_2, \ldots, h_N\}$ is the set of polygons. We assign each polygon to a unique pair (i, j) that corresponds to its grid position. This way we may conveniently refer

to a polygon $h_k \in H$ as $h_{(i,j)}$ by using this map. We assume the grid is given by n_R rows and n_C columns, such that $n_R \times n_C = N$. We denote by h_S and h_E the starting and ending polygons of the problem, respectively. In addition, (i_S, j_S) and (i_E, j_E) are the grid positions of h_S and h_E, respectively.

4.1 Constructive heuristics

The first heuristic we propose is a greedy algorithm that starts from h_S and iteratively selects other polygons as landing points until h_E is reached. This constructive heuristic, called CH1, is detailed in Algorithm 1. Note that we use the function $movecost_{ij}(p, q)$ to compute the total cost of jumping from any polygon $h_{(i,j)}$ to another reachable polygon $h_{(p,q)}$. Such cost can be implemented as described in Section 3.3. This heuristic goes from one polygon to another only if there are no obstacles between them. There is no guarantee of finding a feasible path at the end of the heuristic (we provide some ways of getting rid of this disadvantage ahead in this section).

A tentative way of improving heuristic CH1 would be to allow infeasible jumps, *i.e.* going from one polygon to another even though there are obstacles between them. To be consistent with the optimisation objective, such infeasible jumps should incur penalisation costs. With this in mind, we propose a second constructive heuristic, called CH2, which is similar to CH1 except for allowing infeasible paths. CH2 is presented in Algorithm 2. Note that we use the function $movecostinfeas_{ij}(p, q)$ to compute the cost of jumping from a polygon $h_{(i,j)}$ to a reachable polygon $h_{(p,q)}$. This function is similar to that used in Algorithm 1, but has to incorporate additional costs to penalise infeasible jumps.

The third constructive heuristic, CH3, works in a different way in relation to the two previous ones. It starts with the path $(i_S, j_S) \rightarrow (i_E, j_E)$ even though this may be infeasible (due to obstacles or a large number of jumps). Then, new polygons are inserted in the path iteratively, in order to improve the total cost of the path. The heuristic is defined in Algorithm 3.

In all the algorithms presented above, we can introduce randomness with the aim of improving their performance. This can be done in many different ways and we mention a few in the following, without going into details. One first idea is to insert random perturbation costs in the computation of the moving

PATH = $\{(i_S, j_S)\}$;
$(i, j) = (i_S, j_S)$;
end_reached = 0;
while $i \leq n_R$ *and* $j \leq n_C$ *and end_ reached* $= 0$ **do**
  set $h_{(i,j)}$ as the current landing point;
  $w = 3$;
  candidates $= 0$;
  while *candidates = 0 and* $w \leq n_R$ **do**
    for *r = 1 to w* **do**
      for *s = -w to w* **do**
        if $i + r \leq n_R$ *and* $1 \leq j + s \leq n_C$ *and move is feasible* **then**
          if $i + r = i_E$ *and* $j + s = j_E$ **then**
            $a_{rs} = -1$;
            end_reached = 1;
          else
            $a_{rs} = movecost_{ij}(i + r, j + s)$;
          end
          candidates = 1;
        end
      end
    end
    w = w + 1;
  end
  if *(candidates = 0)* **then**
    STOP: no solution found;
  end
  $(\hat{r}, \hat{s}) = argmin_{r,s}\{a_{rs}\}$;
  update total costs;
  $i = i + \hat{r}$;
  $j = j + \hat{s}$;
  add (i, j) to PATH;
end
return $PATH$ as solution;

Algorithm 1: Constructive heuristic 1 (CH1).

```
PATH = {(i_S, j_S)};
(i, j) = (i_S, j_S);
end_reached = 0;
w = 3;
while i ≤ n_R and j ≤ n_C and end_reached = 0 do
    set h_(i,j) as the current landing point;
    for r = 1 to w do
        for s = -w to w do
            if i + r ≤ n_R and 1 ≤ j + s ≤ n_C then
                if i + r = i_E and j + s = j_E then
                    a_rs = −1;
                else
                    a_rs = movecostinfeas_ij(i + r, j + s);
                end
                candidates = 1;
            end
        end
    end
    (r̂, ŝ) = argmin_{r,s}{a_rs};
    update total costs;
    i = i + r̂;
    j = j + ŝ;
    add (i, j) to PATH;
end
return PATH as solution;
```

Algorithm 2: Constructive heuristic 2 (CH2).

$PATH = \{(i_S, j_S), (i_E, j_E)\}$;
set total cost as $movecostinfeas_{i_S j_S}(i_E, j_E)$;
candidates = 1;
$w = 3$;
while $i < n_R$ *and* $j < n_C$ *and candidates* $= 1$ **do**
  candidates = 0;
  best_cost = ∞;
  foreach *two neighbour pairs* (i_1, j_1) *and* (i_2, j_2) *in* $PATH$ **do**
    $i = (i_1 + i_2)/2$;
    $j = (j_1 + j_2)/2$;
    for r = -w *to* w **do**
      for s = -w *to* w **do**
        if $1 \leq i + r \leq n_R$ *and* $1 \leq j + s \leq n_C$ **then**
          if $i + r \neq i_1$ *or* $i + r \neq i_2$ *or* $j + s \neq j_1$ *or* $j + s \neq j_2$ **then**
            $a_{rs} = insertioncostinfeas(i + r, j + s)$;
            candidates = 1;
          end
        end
      end
    end
    $(\hat{r}, \hat{s}) = argmin_{r,s}\{a_{rs}\}$;
    if *(best_cost* $> a_{\hat{r}\hat{s}}$*)* **then**
      $\hat{i} = i + \hat{r}$;
      $\hat{j} = j + \hat{s}$;
    end
  end
  if *(candidates = 0)* **then**
    STOP: no solution found;
  end
  update total costs;
  add $(\hat{i}, \hat{j})$ to $PATH$;
end
return $PATH$ as solution;

Algorithm 3: Constructive heuristic 3 (CH3).

and insertion costs. This would force the algorithm to choose jumps that are not the best at the moment, but which may contribute to a better global solution (as the algorithm becomes less greedy). Another idea would be to restrict the jumps only to randomly chosen polygons in a small neighbourhood of the current one, instead of considering all the reachable polygons. When included in the previous algorithms, these ideas (and many others) have the potential to construct better routes, especially when used in a metaheuristic context.

4.2 Path perturbation heuristics

The heuristics proposed so far have the purpose of quickly providing an initial solution of the problem. On the other hand, the quality of the solution may be poor and in the worst case no path is obtained. Therefore, we propose some perturbation heuristics, also known as improvement heuristics, with the purpose of improving the paths obtained by the constructive heuristics. These are also simple and quick heuristics and they work by changing an existing path by means of adding, removing or replacing one or more polygons. Hence, we assume that an initial path of K polygons is available, which we denote by the ordered set $PATH = \{(i_1, j_1), (i_2, j_2), \ldots, (i_K, j_K)\}$. The heuristics are described as follows:

1. *Add polygons to path.* For each $k = 1, \ldots, K-1$, select the pair of polygons $h_{(i_k,j_k)}$ and $h_{(i_{k+1},j_{k+1})}$, with (i_k, j_k) and (i_{k+1}, j_{k+1}) in $PATH$. Then, compute the cost of inserting a reachable polygon $h_{(i,j)}$ between those two such that $(i, j) \notin PATH$. After computing all these insertion costs, carry out only one insertion, namely that one with the minimum insertion cost. This will increase the path by one polygon, which may reduce the total cost (e.g. if a penalisation for crossing an obstacle in the original path is removed after the insertion). Even when the total cost of the resulting path becomes worse, this path can be stored as an alternative solution with the purpose of being useful e.g. in a metaheuristic environment, as we describe later in this section.

2. *Remove polygon from path.* Compute the new total cost of removing one polygon at a time from the current path. Then, the resulting path is obtained by removing the polygon that corresponds to the best total cost among all the others. Again, even if the total cost of the resulting

path is worse than that of the original path, it can be useful to store the resulting path to be later used in a metaheuristic environment.

3. *Replace polygons in path.* For each $(i, j) \in PATH$, compute the change in the total cost given by replacing the corresponding polygon $h_{(i,j)}$ by one of its neighbours in the grid, namely $h_{(i+r,j+s)}$ for $-w \leq r \leq w$, $-w \leq s \leq w$ and $w > 0$, such that $1 \leq i + r \leq n_R$ and $1 \leq j + s \leq n_C$. Then, after computing all the changing costs, replace the polygons that lead to a new path with the best total cost among all the others.

We can introduce randomness to these heuristics in a similar way as previously proposed for the constructive heuristics. This can be useful to provide better solutions, especially if we are able to call the heuristics several times. Indeed, to get the best of these heuristics we can combine all them together and use a metaheuristic algorithm to coordinate their interaction, as described in the next section.

4.3 Metaheuristics

There are many metaheuristics proposed in the literature, such as Tabu Search, Genetic Algorithms, Ant Colony Optimisation and Simulated Annealing [4]. These methods have been widely used in the last decades to find feasible solutions for many hard combinatorial optimisation problems [1], especially when modeling practical situations. Although having different motivations, metaheuristics have a common basic idea: they start with one or more initial solutions and then use a set of simpler heuristics to iteratively perturb these solutions, with the aim of obtaining better quality solutions at the end. These perturbations are important not only to improve the quality of the initial solutions, but also to get rid of local optima. This way, a metaheuristic is able to exploit several neighbourhoods of the feasible set, which increases its chances of finding an optimal solution of the problem (or getting closer to it) in relation to a standard heuristic.

All the constructive and perturbation heuristics described earlier in this section can be used to build a metaheuristic method. The constructive heuristics proposed in Section 4.1 can provide initial solutions to the method, especially when randomness is incorporated, so that many different solutions may be provided. Then, at each iteration of the metaheuristic, the path perturba-

tion heuristics proposed in Section 4.2 may be used to try to get paths that are better than those found so far as well as to diversify the current search space.

To exemplify the use of a metaheuristic in this context, we describe a Tabu Search (TS) method [5] that relies on the previously defined heuristics. There are different variants of TS and furthermore the algorithm described below may be implemented in different ways. The most appropriate variant/implementation depends on the type of the problem, heuristics used to construct and perturb solutions, data structures, computer environment, and many other characteristics. This way, to obtain a TS implementation that works well in practice, it is important to test with different strategies and parameter choices first and then stick to the best setting.

The main idea of TS is to keep a *tabu list* to indicate which changes are prohibited when the perturbation heuristics are called. This tabu list is dynamically updated through iterations, so that some prohibited changes remain in the list for a given number of iterations. After a change is removed from the tabu list, it becomes allowed again. This is the way that TS algorithms try to get rid of local optima.

Let $\mathcal{S} = \{PATH_1, PATH_2, \ldots, PATH_P\}$ be an ordered set of solutions, using the total cost of each path as its ranking value. We denote by TL the tabu list that contains the perturbations (moves) that must be prohibited in the current calls to the path perturbation heuristics. This can be implemented as a simple circular list so that old entries are replaced by new ones through the iterations (which makes this tabu list dynamic). Finaly, we define an ordered set $\mathcal{C}$ that contains the paths that result from the path perturbation heuristics. Using this notation, the TS method is presented in Algorithm 4. The stopping criterion of the algorithm is not particularly defined, as it depends on the implementation. The usual criteria are given by reaching a maximum running time or a maximum number of iterations of the algorithm.

There are several open steps in Algorithm 4. One of them is how to choose index i in line 5. For instance, by choosing $i = 0$ we take the path with the currently best total cost. This may be a good choice for getting better solutions in the improvement heuristics, but other choices may be useful to generate solutions that allow to exploit other neighbourhoods of the feasible set. Hence, an implementation of the algorithm should consider different ways

Initialise set $\mathcal{S}$ by calling the constructive heuristics;
$TL = \emptyset$;
while *stopping criterion is not reached* **do**
  $\mathcal{C} = \emptyset$;
  Choose an index i such that $PATH_i \in \mathcal{S}$;
  Apply the improvement heuristics to $PATH_i$ allowing only the perturbations that are not in TL;
  Add to $\mathcal{C}$ all the paths resulting from the perturbation heuristics;
  Select K paths from $\mathcal{C}$, add them to $\mathcal{S}$ and add the perturbation moves that originated them to TL;
end
if *($\mathcal{S} = \emptyset$)* **then**
  STOP: no solution found;
end
Return $PATH_1$ as the best solution found for the problem;

Algorithm 4: Tabu search method (TS).

of carrying out this step. These comments apply to other parts of the algorithm as well, such as the size of K and how to select the paths from $\mathcal{C}$.

5 A probabilistic viewpoint for further study

A classic topic in probability theory is *first passage percolation*, cf. Hammersley and Welsh [6]. This topic is generally concerned with the following basic question. Given some growing sequence of graphs — think of regular structures such as the two-dimensional grid or the complete graph — whose vertices or edges are given a random distribution of weights, what is a good asymptotic description of the lowest weight path (geodesic) between two specified vertices?

As an offshoot of the SWI problem, it could be natural and novel to consider a variant of first passage percolation that incorporates some aspects of power line routing. Although interesting, this is beyond the scope of this project and report. We only sketch a simplistic model problem which might be considered along these lines. The following is a variant of first passage bond percolation. (Note that more complex variants of first passage site percolation might model

the original problem more faithfully.)

The underlying graph is an $n \times n$ grid, embedded in the plane, whose nodes are given some random weights, according to a given probability distribution. Here we define a *power line route* to be a sequence of mutually distinct points of the grid, every consecutive pair of which is within distance k, according to a given norm. The calculation of the cost of a power line route takes into account two components, weighted separately. First, it incorporates the length of the polygonal curve (which accounts for the cost of the power line). Second, it includes the cumulative cost of all the nodes in the sequence (which accounts for the cost of the tower construction). The cost of each node is calculated as a product of the weight at that node and some given function of the lengths and angle θ of the incident line segments. (A particularly simple example of the last-mentioned function is, say, the product of the incident line lengths and $|\pi - \theta|$, though other simple choices could be more interesting or realistic.) The problem is to determine the asymptotic behaviour (as $n \to \infty$) of the lowest cost power line routing between $(1, 1)$ and (n, n).

As far as we are aware, this type of model, particularly with the penalty cost for acute angles in the polygonal curve, has not been studied in the literature before.

6 Conclusion

In this report, we have addressed a real-life problem faced by the company NM Group, a problem called power line routing. We considered the problem at two different scales of discretisation, namely macro and micro levels. For macro level routing, we proposed a new algorithm that overcomes an issue faced by a previous algorithm used by the company. We also comprehensively addressed the more detailed micro level routing problem. We proposed several different algorithmic solution strategies, one of which is exact, the others of which are heuristic or approximate. In particular, we have encoded the problem in an auxiliary weighted directed graph, the shortest path of which corresponds exactly to a minimum cost solution for the micro level routing problem. This leads to an exact algorithm for determining the optimum that, while theoretically efficient, is unlikely to be computationally practical. So we also proposed several heuristics including path perturbation and metaheuristics, all of which

have decent hope of producing reasonable candidate solutions in reasonable time/space.

Acknowledgments

This research was done during the Study Group Mathematics with Industry 2015 (SWI) in Utrecht. We would like to thank the organisers of the study group for creating the wonderful atmosphere.

The authors thank NM Group representatives Paul Richardson and Chris Williams for presenting the problem and readily answering all of our questions about power line routing and mapping.

We gratefully acknowledge Aleksandra Stojanova, Dusan Bikov, Giorgi Khimshiashvili, Natasha Stojkovikj and Zlatko Varbanov for pleasant conversations during the study group and for their valuable comments and suggestions.

References

[1] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[2] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi: 10.1007/BF01386390.

[3] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi: 10.1145/28869.28874.

[4] F. Glover and G. A. Kochenberger. *Handbook of metaheuristics*. Springer Science & Business Media, 2003.

[5] F. Glover and E. Taillard. A user's guide to tabu search. *Annals of Operations Research*, 41(1):1–28, 1993.

[6] J. M. Hammersley and D. J. A. Welsh. First-passage percolation, subadditive processes, stochastic networks, and generalized renewal theory. In

Proc. Internat. Res. Semin., Statist. Lab., Univ. California, Berkeley, CA, pages 61–110. Springer-Verlag, New York, 1965.

[7] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* AMS Chelsea Publishing, 1999.