Ole Kåre Endresen · Dag Haugland

Optimal adjustment of surfaces to point sets

Abstract. We consider the problem of determining a two-dimensional surface in a three-dimensional space. A proposed surface, represented by a discrete point set is already given. The proposal may however not correspond to the true surface. Associated with each proposed point, a set of admissible points is provided. The problem is to select one admissible point for each proposed point such that it intersects each set of admissible point rather than the proposed points. This adjustment must be carried out in such a way that the resulting surface either minimizes the global distance from the proposed surface or maximizes the shape similarity with the proposed surface.

In this work we propose a mathematical model for the problem under study, and we point out that this model, contrary to its equivalent in a two-dimensional space, is $\mathcal{NP}$-hard. Since polynomial exact algorithms are unlikely to be found, we rather develop heuristic methods based on combinatorial relaxations, which are solvable by dynamic programming, in combination with local search. Computational properties of the solution methods are discussed, and results from preliminary experiments are presented.

1. Introduction

Interpretation of seismic maps frequently involve operations like identification of ridges or valleys in layers of various rock qualities, recognition of the border surfaces between different layers, separation of porous rock containing hydro carbons from dry rock, and so forth. Whatever the exact task is, interpretation problems like these imply that some surface or line in space must be identified. Hence an important challenge for a seismologist may be to apply his or her knowledge to the actual problem in order to incorporate new surfaces into digitally available maps.

Surfaces will typically be represented by finite sets of points that they intersect. These points constitute the user input in the actual setting, and are henceforth referred to as *proposed points*. The proposal may however be based on rather subjective judgment, and there is a risk that the proposed surface positions are incompatible with information inherent in the seismic map. Therefore, there is a *set of admissible points* defined for each proposed one, all of which respect any compatibility rule that applies. Typically, the admissible points are located in the vicinity of the proposed point, which it may include in some cases. The problem addressed in the current paper is how the proposed surface should be snapped to the admissible sets, that is,

Dept. of Informatics, University of Bergen, Norway

Correspondence to: {oleke,dag}@ii.uib.no

with which of their corresponding admissible point should the proposed ones be replaced, in order for the resulting surface to stay as close as possible to the original one.

Closeness between surfaces can be defined in terms of either geometrical closeness or shape similarity. The first criterion is modeled by the absolute value of the surface difference, integrated over the whole domain. Shape similarity is represented by summing up absolute values of differences in slopes of corresponding linear segments. In the present paper we consider both of these closeness measures. The mathematical model to be presented turns out to be general enough to capture both, and the choice of minimization criterion is reflected only by parameter values. Alternative interesting criteria, like minimizing curvature or other measures independent of the proposed surface, are beyond the scope of this paper.

To the knowledge of the authors, the problem under study has not received much attention in the mathematical programming literature. However, a model to be presented in later sections, appears to be similar to those suggested by Jünger et al. [4] and Kleinhans et al. [3] for a problem in electronic circuit design.

In order to give a soft introduction to the exact problem formulation, the next section deals with the related, but lower-dimensional, problem of adjusting a line in the plane. In Section 3 we give the three-dimensional model for which dynamic programming heuristics are given in Section 4. Local search methods are studied in Section 5, and computational results from all our methods presented in the paper are given and discussed in Section 6.

2. Exact methods in 2 dimensions

Let $A = \{v^1, \ldots, v^n\} \subset \Re^2$ be the set of proposed points, where $v^i = (x^i, y^i)$. We assume that $x^1 < x^2 < \cdots < x^n$. For each $v^i \in A$, a set $I^i = \{v^i_1, \ldots, v^i_{h^i}\}$ of corresponding admissible points is defined. We let $v^i_p = (x^i_p, y^i_p)$.

For each pair (v^{i-1}_p, v^i_q) where $v^{i-1}_p \in I^{i-1}$ and $v^q_i \in I^i$ a cost coefficient c^i_{pq} is given. The exact definition of the cost coefficient depends on what closeness criterion we use. In this paper we consider the criteria *minimum global distance from the proposed line* and *maximum shape similarity with the proposed line*. The problem is to pick one admissible point from each set, such that the overall cost for the piecewise linear line interpolating selected points is minimized.

Our problems' similarity to the well known shortest path problem suggests a dynamic programming approach to solving the problem to optimality. Let $f^i(q)$ be the total cost from the $n-1$ first segments given that v^i_q is chosen from I^i, and an optimal choice is made from each of $I^1, \ldots, I^{i-1}$. The cost function $f^i(q)$ is then given by the dynamic programming model:

$$f^i(q) = \min_{p=1,\ldots,k^{i-1}} \left\{ f^{i-1}(p) + c^i_{pq} \right\}, i = \{1, \ldots, n\}, q = 1, \ldots, h^{i-1} \quad (1)$$

Let $h = \max_{i \in A}\{|I^i|\}$.

Proposition 1. *An optimal solution to problem (1) is found in $\mathcal{O}(nh^2)$ time*

Proof. Follows immediately by considering an implementation with forward enumeration of the point sets ($\mathcal{O}(n)$), where each $v^i_q \in I^i$ is combined with all $v^{i-1}_p \in I^{i-1}$ and the best is picked ($\mathcal{O}(h^2)$). □

2.1. Calculating the cost coefficient

2.1.1. Maximum shape similarity For the criterion maximum shape similarity with the proposed line we compare the slope of the line between two admissible points in subsequent point sets with the slope made up by the corresponding proposed points. Let ∇^i ($i \in A$) be the gradient of the straight line interpolating v^{i-1} and v^i, and for $p = 1, \ldots, h^{i-1}$ and $q = 1, \ldots, h^i$, let ∇^i_{pq} be defined correspondingly for v^{i-1}_p and v^i_q. We get the cost coefficient c^i_{pq} from:

$$c^i_{pq} = \left| \nabla^i - \nabla^i_{pq} \right| \quad (2)$$

2.1.2. Minimum global distance For $x \in \Re$, let $\text{sign}(x) = 1$ if $x \geq 0$, and 0 otherwise. Let $u^i = (u^i_1, u^i_2) \in \Re^2$, $i = 1, \ldots, 4$. We denote the line segment (vector) starting in u^1 and ending in u^2 by $[u^1, u^2]$, and define the determinant

$$\Delta(\mathbf{u^1}, \mathbf{u^2}, \mathbf{u^3}) = \begin{vmatrix} u^1_1 & u^1_2 & 1 \\ u^2_1 & u^2_2 & 1 \\ u^3_1 & u^3_2 & 1 \end{vmatrix}$$

In the calculations below we shall exploit the facts that $[u^1, u^2]$ intersects $[u^3, u^4]$ if $\text{sign}\left(\Delta(u^1, u^2, u^3)\right) \neq \text{sign}\left(\Delta(u^1, u^2, u^4)\right)$ and $\text{sign}\left(\Delta(u^3, u^4, u^1)\right) \neq \text{sign}\left(\Delta(u^3, u^4, u^2)\right)$, and that the area of the triangle with vertices u^1, u^2 and u^3 equals $\left|\Delta(u^1, u^2, u^3)\right|/2$.

To compute c^i_{pq}, we first need to determine whether or not the segments $[v^{i-1}, v^i]$ and $[v^{i-1}_p, v^i_q]$ intersect. Hence, if

$$\text{sign}\left(\Delta(v^{i-1}, v^i, v^{i-1}_p)\right) = \text{sign}\left(\Delta(v^{i-1}, v^i, v^i_q)\right) \quad (3)$$

or

$$\text{sign}\left(\Delta(v^{i-1}_p, v^i_q, v^{i-1})\right) = \text{sign}\left(\Delta(v^{i-1}_p, v^i_q, v^i)\right) \quad (4)$$

c^i_{pq} is given by the quadrangle composed by the four points in question, and

$$c^i_{pq} = \frac{\left|\Delta(v^{i-1}, v^i, v^{i-1}_p)\right| + \left|\Delta(v^i, v^{i-1}_p, v^i_q)\right|}{2} \tag{5}$$

In the case of intersecting segments, i.e. neither (3) nor (4) holds, let $\hat{v}$ denote the intersection point. Hence the cost is found by

$$c^i_{pq} = \frac{\left|\Delta(v^{i-1}, \hat{v}, v^{i-1}_p)\right| + \left|\Delta(v^i, \hat{v}, v^i_q)\right|}{2} \tag{6}$$

3. The 3-dimensional model

In the three-dimensional case, we let $A = \{v_1, \ldots, v_n\} \subset \Re^3$ denote the proposed points, and we let $I^1, \ldots, I^n \subset \Re^3$ be the corresponding sets of admissible points. In the two-dimensional case, the piecewise linear function interpolating all proposed points is a rather obvious choice of surface (line) to associate with the points. A similar association is made for selected admissible points, and distance measures are defined straightforwardly. Both measures considered are composed of costs involving two consecutive points.

When the proposed points are in $\Re^3$, a piecewise linear interpolating function is not that obvious. That is, it has to be settled what pairs of proposed points that may define costs. The following model defines all such pairs.

3.1. Graph representation

To illustrate the problem we define a graph $G = (V, E)$, where the vertices $V = \{1, \ldots, n\}$ correspond to the set A of proposed points (and belonging set of admissible points), and where edges are defined only between proposed points (or sets of admissible points) that are somewhat geometrically related (G could for instance be a grid graph, an assumption we make in Section 4). Weights are assigned the edges depending on what admissible points are chosen for the end vertices. The goal is to choose one admissible point from vertex such that the sum of edge weights is minimized. Figure 1 is an illustration of the graph.

If we let one vertex denote a cluster of nodes, being the admissible points, we get that G is a special case of the *Generalized Subgraph Problem*, introduced by Labbé et al. [2]. The graph we have explained is in fact a what in [2] is referred to as the *shrunk* graph. A *full* graph representation is obtained by defining the admissible points as the vertices. A set of admissible points I^i represents a cluster of nodes, and for every adjacent point set I^i and I^j, where $(i, j) \in E$, we have a complete bipartite subgraph. The problem is now to choose one edge between adjacent clusters, under the condition that

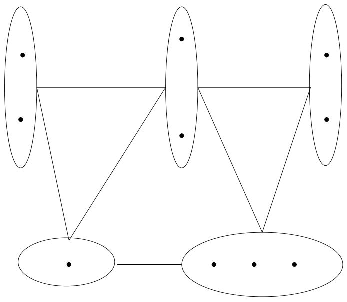

Fig. 1. A shrunk representation of the graph

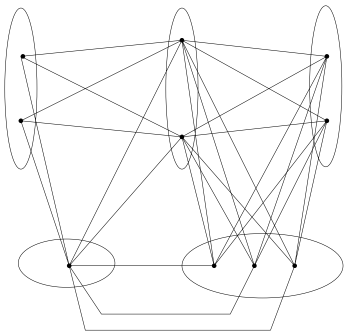

Fig. 2. A complete representation of the graph

only one vertex is chosen in each cluster, that minimizes the overall sum of weights. The full graph corresponding to the shrunk graph in Figure 1 is illustrated in Figure 2.

3.2. A Constrained Quadratic Binary Program

For $(i, j) \in E$ let c_{pq}^{ij} be the cost of choosing admissible point $p \in I^i$ and $q \in I^j$. A decision variable y_p^i is defined:

$$y_p^i = \begin{cases} 1 \text{ , choose } p \in I^i \\ 0 \text{ , otherwise} \end{cases}$$

The Constrained Quadratic Binary Program is then given by:

Constrained Quadratic Binary Program

$$z = min \sum_{(i,j)\in E} \sum_{p\in I^i} \sum_{q\in I^j} c_{pq}^{ij} y_p^i y_q^j \tag{7}$$

subject to:

$$\sum_{p\in I^i} y_p^i = 1, \quad \forall i \in V \tag{8}$$

$$y_p^i \in \{0,1\}, \quad \forall i \in V, p \in I^i \tag{9}$$

Proposition 2. *(7) - (9) is $\mathcal{NP}$-hard.*

Proof. Follows directly from Theorem 3.2 in Jünger et al. [4]. □

Due to the large datasets involved in seismology we need to focus on solving the problem fast. Proposition 2 indicates that it is unlikely to find a polynomial algorithm that solves the problem to optimality. In Section 2 we learned that the 2 dimensional case of the problem is polynomial solvable. Even though the algorithm does not apply when we move to three dimension we can use the idea behind the algorithm in our search for good heuristics for the problem.

4. Dynamic programming heuristics

4.1. Grid assumption

In Section 3 we defined a graph were the edges represent line segments along which we want the admissible surface to be close to the proposed one. We now assume that the projection of A into the plane spanned by $(1,0,0)$ and $(0,1,0)$ is distributed on a rectangular grid. Furthermore, we compare the surface only along horizontal and vertical lines between neighboring points on the grid. This leaves us with G as a grid graph (see Figure 3).

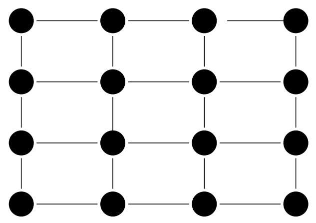

Fig. 3. A grid graph

4.2. A tree relaxation

We have already seen that a dynamic programming model solves the 2-dimensional case of our problem. The 2-dimensional problem corresponds to a path according to the sparse graph definition we gave in Section 3. That means that if we transform our grid graph to a path by eliminating edges, we face a polynomially solvable problem.

This idea can be generalized by transforming G to a tree. For some spanning tree of G let $B^i, i \in V$, be the set of child vertices to i, $f^i(q)$ the best value for choosing $q \in I^i$ and let c^{ij}_{pq} be the cost coefficient of choosing $p \in I^i$ and $q \in I^j$. We then arrive at the *tree relaxation*of the grid graph problem, given by:

$$f^i(q) = \begin{cases} 0 & , B^i = \emptyset \\ \sum_{j \in B^i} \min_{p \in I^j} \left\{ c^{ij}_{pq} + f^j(p) \right\} & , \text{otherwise} \end{cases} \tag{10}$$

Proposition 3. *The tree relaxation can be solved in* $\mathcal{O}(nh^2)$

Proof. Similar to Proposition 1 □

The question is now how to construct such spanning trees in the grid graph. In order to make decisions on what edges to be picked or not, we need to assign weights to them. One way of doing this is to calculate the variance of all possible weights for one edge. For an edge $(i, j) \in E$ we find the new edge weight c^{ij} by:

$$c^{ij} = \frac{\sum_{p=1}^{h^i} \sum_{q=1}^{h^j} \left(c^{ij}_{pq} \right)^2}{h^i h^j} - \left(\frac{\sum_{p=1}^{h^i} \sum_{q=1}^{h^j} c^{ij}_{pq}}{h^i h^j} \right)^2 \tag{11}$$

In this grid graph of new edge weights we find the maximum spanning tree and apply Equation 10 on this tree. The idea behind this approach is that edges with the largest spread, where a bad choice of admissible points for the end vertices will have a considerable impact on the goal value, should be calculated in the DP recursion. If the variance of an edge is small the choice of admissible point is not critical since the contribution to the goal value does not differ much for the choices.

If we let $m = |E|$, calculating the weights c^{ij} $\forall (i, j) \in E$ runs in $\mathcal{O}(mh^2)$ time. We find the maximum spanning tree by applying the famous algorithms of Prim or Kruskal. These have a running time of $\mathcal{O}(m \log m)$ and $\mathcal{O}(m \log n)$ respectively.

In this work we have implemented a maximum spanning tree relaxation of the variance of edge weights using Kruskal. We have also implemented two simple path relaxation algorithms, one that runs along the y-lines and one running along the x-lines. Figure 4 illustrates the two paths. The three relaxations are compared experimentally and discussed in Section 6.

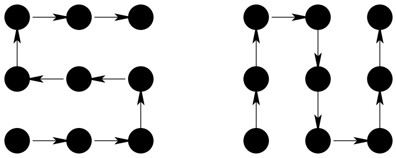

Fig. 4. Two possible path relaxations

5. Local search heuristics

We consider two different neighborhood structures for our local search heuristics, a vertex based and an edge based structure. Both heuristics assume an initial solution, given e.g. by solving any of the relaxations proposed in Section 3. In Section 6 we compare the two algorithms.

5.1. A vertex based search algorithm

In the vertex based local search heuristic we consider a change of admissible point in one vertex at the time. We find the best admissible point for a vertex relative to the admissible points chosen for the adjacent vertices, and if this is different from the admissible point already chosen we make a change.

The situation is illustrated in Figure 5. Here a change in the black vertex is considered depending on the admissible points chosen for the adjacent shaded vertices.

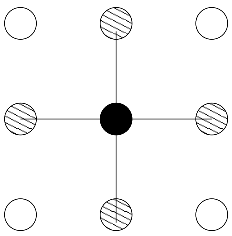

Fig. 5. Vertex based local search

Let $\mathbf{S} = \{s^1, \ldots, s^n\}$, where admissible points $s^1 \in I^1, s^2 \in I^2, \ldots, s^n \in I^n$ constitute the actual solution. For each vertex $i \in V$ we define D^i to be the set of adjacent vertices to i and $\mathcal{N}^i(\mathbf{S}) = \{\mathbf{S}' : (s')^k = s^k \quad \forall k \in V \backslash \{i\}\}$ the neighborhood of solutions. Then, if the admissible point for vertex $i \in V$ becomes p, the contribution to z from incident edges is given by:

$$N_p^i = \sum_{j \in D^i} c_{ps^j}^{ij} \quad i = 1, \ldots, n, \quad p = 1, \ldots, h^i \tag{12}$$

Changes in $\mathbf{S}$ are performed in accordance with:

$$s^i \in \arg\min_{p \in I^i} \{N_p^i\} \quad i = 1, \ldots, n \tag{13}$$

One iteration of the algorithm runs through all the vertices in a consistent order, and performs the above update. Hence it has time complexity $\mathcal{O}(nh)$.

5.2. An edge based search algorithm

For the edge based search algorithm we look at two vertices, the end vertices of an edge, at the same time when a change of admissible points is considered. In the same manner as the vertex based algorithm we run through all possible combinations of admissible points for the two vertices, and update our solution with the best combination if this is different from the actual solution.

In Figure 6 we consider a change in the two black vertices. The change is carried out dependent on what admissible points are chosen for the adjacent shaded vertices.

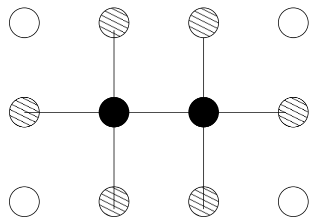

Fig. 6. Edge based local search

We let $\mathbf{S} = \{s^1, \ldots, s^n\}$ be the actual solution and D^i the set of adjacent vertices to i as for the vertex based algorithm. The neighborhood of solutions for an edge $(i,j) \in E$ is given by $\mathcal{N}^{ij}(\mathbf{S}) = \{\mathbf{S}' : (s')^k = s^k \quad \forall k \in V \backslash \{i,j\}\}$. The contribution to z for an edge $(i,j) \in E$ is, if admissible points $p \in I^i$ and $q \in I^j$ are chosen, given by:

$$N_{pq}^{ij} = c_{pq}^{ij} + \textstyle\sum_{k \in D^i \backslash \{j\}} c_{ps^k}^{ik} + \sum_{l \in D^j \backslash \{i\}} c_{qs^l}^{jl} \qquad \begin{array}{l} , i,j = 1, \ldots, m \\ , p = 1, \ldots, h^p \\ , q = 1, \ldots, h^q \end{array} \tag{14}$$

We run through all edges $(i,j) \in E$, and assign new values to (s^i, s^j) such that:

$$(s^i, s^j) \in \arg\min_{p \in I^i, q \in I^j} \{N_{pq}^{ij}\} \tag{15}$$

This is repeated until no change is observed. One such iteration, where we run through the edges in a consistent order runs in $\mathcal{O}(mh^2)$ time.

6. Computational results

As explained in Section 4.2 three different tree relaxation algorithms have been implemented, a path that runs the y-lines (`path 1`), a path that runs the x-lines (`path 2`) and a maximum spanning tree of the edge weights variance (`tree`). In this section we compare the result from the three algorithms on test data given by Schlumberger Stavanger Technology Center. A comparison to an implementation using CPLEX, which solves the problem to optimality, is also provided. We have also implemented the two different local search heuristics discussed in Section 5, and we will see that applying these to the tree relaxation solutions will lead to a considerable improvement. All the test runs have been carried out on a Sun Ultra 10, Solaris 8 machine with 128 MB RAM and a 350MHz Ultra SPARC 3 processor.

We start by looking at test sets that are solvable by CPLEX. Table 1 and Table 2 sum up runs on the criterion maximum shape similarity with the proposed surface. If we look at the three tree relaxations alone we can see that while `path 1` gives the best result for the first test, `tree` is superior in the second. These results, together with numerous other tests, shows no clear indications on what is the preferable relaxation. As we were expecting the more sophisticated `tree` relaxation to be better than the two simple paths, this is rather disappointing. The reason for this might be that the variance measurement we have used for the edges in our maximum spanning tree algorithm is not a good measurement, at least not for the test data we have used.

Comparing the relaxations to the optimal CPLEX solution in Table 1, we can see that while our approximations find a solution in less than a second, optimum is computed in about 2 minutes. In addition to being fast, the heuristics also give satisfying goal values. The best relaxation, `path 1`, is 1.58 times the optimum. We see that, although the differences are small, the edge based heuristic is better than the vertex based. Running the edge based local search using `tree` as the initial solution gives a goal value that is about 1.16 times the optimal value.

	No local search		Vertex based			Edge based		
	Weight	Time (sec)	Weight	Its	Time	Weight	Its	Time
CPLEX	1485	121	-	-	-	-	-	-
Path1	2339	< 1	1919	3	< 1	1871	2	< 1
Path2	2408	< 1	2082	3	< 1	2007	3	< 1
Tree	2556	< 1	1737	6	< 1	1728	2	< 1

Table 1. Results from a shape similarity case with h = 10 and $|V| = 13 \times 11$

For the bigger test set in Table 2, we can see that the result after the tree relaxations differs more than in the previous table. Here `tree` gives the best result, 1.56 times optimum, while `path 1` gives over 2.29 times the optimal value. Due to the calculations done prior to the DP algorithm, using `tree` makes this run one second slower than the paths. CPLEX spends nearly 10 minutes solving this problem.

As we can read from the table, the local search heuristics gain large improvements in this case. We can also see that although the vertex based local search runs more iterations, the edge based structure gives the best results. After the edge based local search the best goal value is down to 1.14 of optimum when starting from the `tree` relaxation solution, whereas it is 1.33 times the optimal value when starting from `path 2`. The vertex based heuristics give results that are between 1.19 and 1.57 times optimum.

	No local search		Vertex based			Edge based		
	Weight	Time (sec)	Weight	Its	Time	Weight	Its	Time
CPLEX	6687	590	-	-	-	-	-	-
Path1	15316	1	10414	12	<1	8762	5	<1
Path2	14111	1	10503	8	<1	8925	4	<1
Tree	10446	2	7961	4	<1	7621	3	<1

Table 2. Results from a shape similarity case with h = 8 and $|V| = 21 \times 21$

For the criterion minimum distance from the proposed surface (Table 3), all three relaxations provide results closer to optimum. The local search heuristics run fewer iterations and gain less than they do for the shape similarity criterion. It seems that this criterion is easier to calculate, at least for the test cases we have. The reason for this may be that in our cases, most admissible points are located far from the proposed points, and hence the few admissible points that are close are easily recognized.

	No local search		Vertex based			Edge based		
	Weight	Time (sec)	Weight	Its	Time	Weight	Its	Time
CPLEX	5757	426	-	-	-	-	-	-
Path1	6034	1	5821	2	<1	5974	1	<1
Path2	5978	1	5821	3	<1	5766	2	<1
Tree	6059	2	5771	3	<1	5770	2	<1

Table 3. Results from a minimum global distance case with h = 8 and $|V| = 21 \times 21$

We now move to test sets that are too big for CPLEX to solve to optimality. Tables 4 and 5 show results from a 74×37 big grid graph with 24 admissible points in each vertex.

For shape similarity (Table 4) we can see that the tree relaxation gives results that are superior to both path relaxations. We can also see that it spends more time solving the problem. The calculation of the variance edges and the maximum spanning tree is the main reason for this.

A lot is gained by running the local search heuristics with one of the relaxation solutions as initial solution in this case. Again, the edge based structure gives the best results, although it runs fewer iterations. It is, however, more expensive to run than the fast node based algorithm.

	Tree Relaxation		Vertex search			Edge search		
	Weight	Time (sec)	Weight	Time	Its	Weight	Time	Its
Path 1	32692	19	19094	7	14	17497	55	5
Path 2	31543	19	23797	4	16	18804	86	8
Tree	24673	64	17626	19	7	15549	76	4

Table 4. Results from a shape similarity case with h = 24 and $|V| = 74 \times 37$

In Table 5 we find the same tendencies as explained earlier. The minimum distance criterion is easier to compute than the shape similarity criterion. We do not have the optimum to compare with, but the fact that the different local search algorithms only run for one or two iterations indicate good initial solutions. From the tables we can also see that the minimum distance weight is harder to compute than the shape similarity one.

	Tree Relaxation		Vertex search			Edge search		
	Weight	Time (sec)	Weight	Time	Its	Weight	Time	Its
Path 1	4530	36	4508	2	1	4505	28	2
Path 2	4537	36	4515	1	2	4513	19	1
Tree	4553	166	4508	30	1	4506	47	1

Table 5. Results from a minimum global distance case with h = 24 and $|V| = 74 \times 37$

Figures 7 to 10 are plots from the test set used in Table 4. In these plots we try to capture the differences between the relaxations, and show how the edge based local search performs better than the vertex based one.

Figure 7 is a plot of a horizontal line of the grid. The plot shows that `path 1` approximates the proposed line best, while `path 2` does not look like the proposal at all. This is easily explained as `path 1` runs the horizontal edges and takes all the linear segments into account while `path 2` runs the vertical edges and does not consider these segments at all. The `tree` relaxation comes out somewhere in between the two paths, as it will pick and calculate some of the horizontal edges.

For vertical lines the result is, as we can see in Figure 8, the other way around. Here path 2 gives the best plot while `path 1` is more accidental compared to the proposed line.

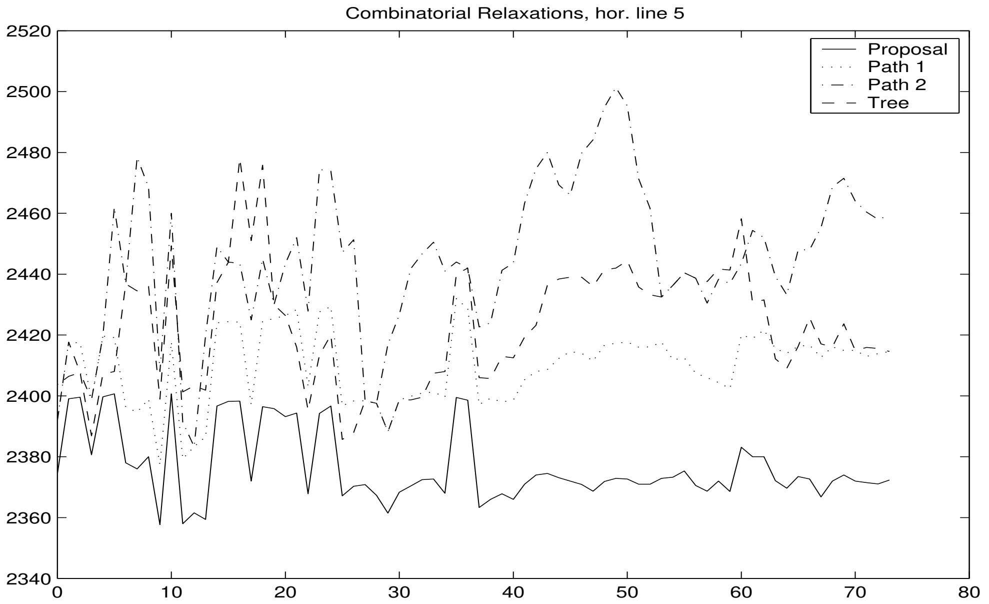

Fig. 7. Plot of relaxations, horizontal line

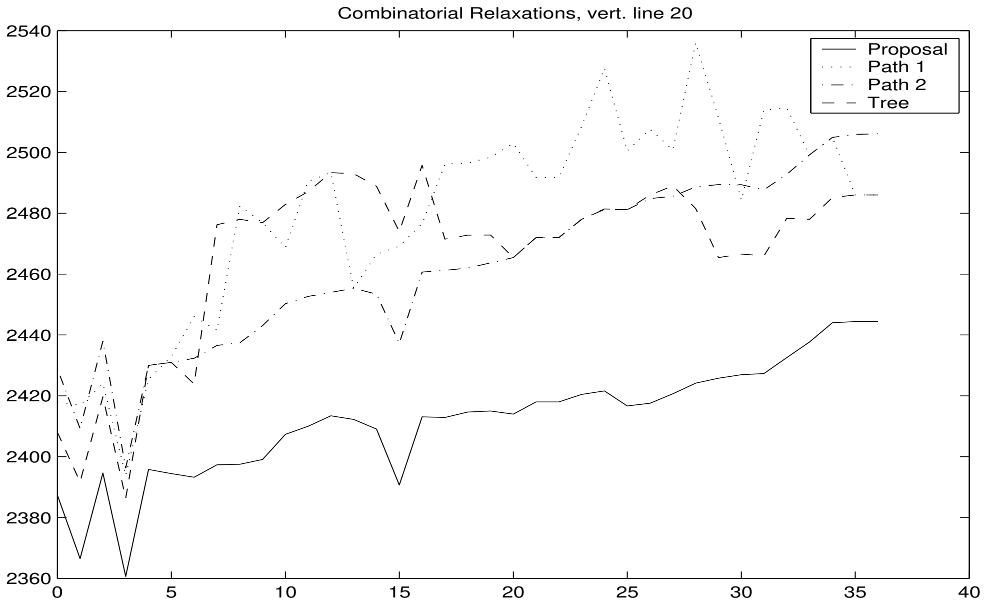

Fig. 8. Plot of relaxations, vertical line

In Figure 9 we can see the result of performing the edge based local search heuristic to the plot from Figure 8. The resulting lines with `path 1` and `tree` as initial solutions have been considerably improved. We observe that the break in the middle of the line has been recognized, and that the lines, especially for `path 1`, are much smoother and thereby in accordance with the proposal. For `path 2` only minor changes have been carried out. It can be argued that these alterations have not made the plot any better compared to the initial solutions. This should come as no surprise as the initial solution is based on calculations made with vertical influences only, whereas local search also takes the horizontal edges into consideration.

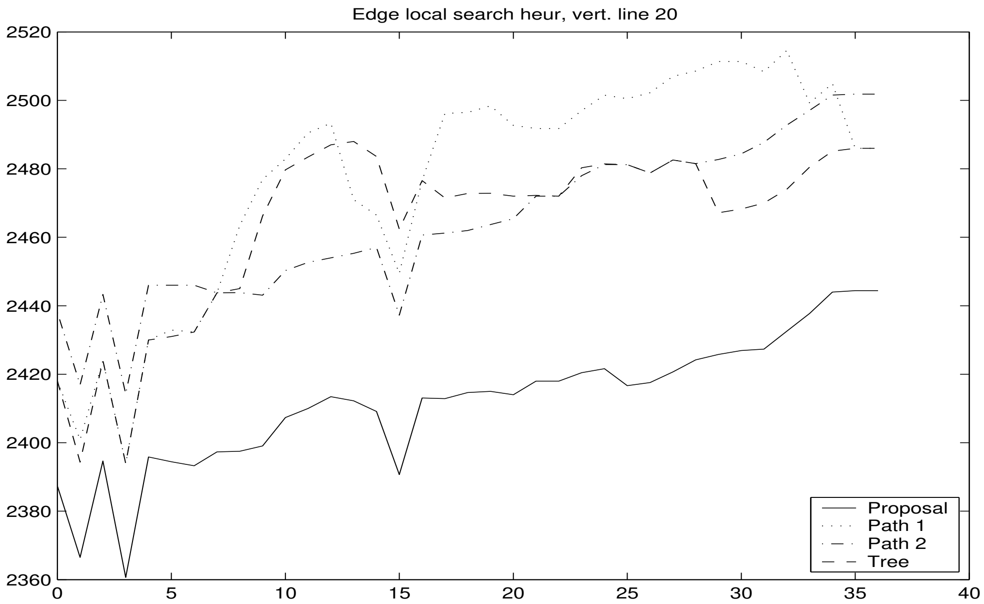

Fig. 9. Result of applying the edge based local search heuristic to the plot in Figure 8

The last plot (Figure 10) shows a comparison of the local search heuristics. This is a plot of a horizontal line with `path 2` as the initial solution. The initial line (dotted) contains several big spikes that cannot be seen in the proposed line. We can see that both of the local search algorithms manage to reduce these spikes, but the edge based structure is far more successful. While we in the vertex based algorithm have used a window of only one vertex the edge based uses two. That means if the solution contains two or more neighboring bad admissible points, the vertex based heuristic is more likely than the edge based to accept the solution.

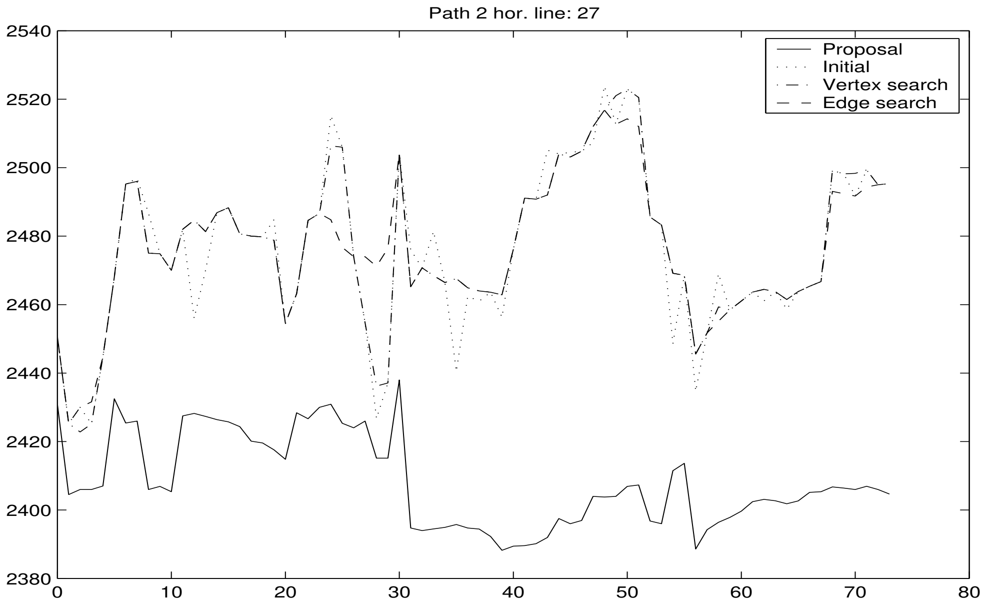

Fig. 10. Comparison of the local search heuristics

7. Conclusion

In this paper we have considered at the problem of determining a two-dimensional surface in a three dimensional space. Given a set of proposed points, where each such proposed point is surrounded by a number of admissible points, we have shown how the closeness criteria, maximum shape similarity with the proposal and minim global distance from the proposal can be computed. We have also given a polynomial dynamic programming algorithm that solves the 2 dimensional case of the problem.

We have pointed out that the three-dimensional problem is $\mathcal{NP}$-hard, and consequently we have focused on heuristic methods for this problem. Based on a simple graph model, we have defined one vertex-oriented and one edge-oriented neighborhood structure.Local search heuristics relying on these structures are developed and compared experimentally. Initial solutions used in the search have been computed through combinatorial relaxations solvable in polynomial time.

The experiments have shown that our heuristics are capable to compute near-optimal solutions (off by 0.2 %) to problems with 441 proposed points and 8 admissible points per proposed point, given that the closeness criterion is minimum distance. In the case of shape similarity, the results are not that good (off by 14 %). It turns out that the edge based neighborhood structure is the more efficient, whereas the choice of initial solution does not seem to have a significant impact on the result. Moreover, the modest computational effort required by our methods indicate a great potential for improvements in the solution quality. This potential will be further explored in an ongoing work involving more advanced search methods (Endresen [1]).

Acknowledgements. The authors express their gratitude to Project Manager Jan Tveiten at Schlumberger Stavanger Technology Center for introducing the problem and providing the test data.

References

1. O.K. Endresen. Optimal adjustment of surfaces to point sets. Master's thesis, University of Bergen, 2003.
2. C. Feremans, M. Labbé, A. Letchford and J.J. Salazar. On the generalized subgraph problem. Paper presented by M. Labbé at the Eighth Meeting of the Nordic Section of the Mathematical Programming Society, Bergen, Norway, 09 2002.
3. J.M. Kleinhans, G. Sigl, F.M. Johannes and K.J. Antreich. Vlsi placement by quadratic programming and slicing optimization. *IEEE Transactions on Computer Aided Design*, 10, 1991.
4. M. Jünger, A. Martin, G. Reinelt and R. Weismantel. Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits. *Mathematical Programming*, 63(3):257–279, 1994.