UNIVERSITY OF OSLO
Department of informatics

Curve approximation and constrained shortest path problems

Geir Dahl and
Bjørnar Realfsen

Preprint 6

August 1996

CURVE APPROXIMATION AND CONSTRAINED SHORTEST PATH PROBLEMS

GEIR DAHL* AND BJØRNAR REALFSEN

Abstract.
We study the following problem *k-constrained shortest path problem*: given an acyclic directed graph $D = (V, E)$ with arc weights $c_{i,j}$, $(i,j) \in E$, two nodes s and t and an integer k, find a shortest st-path containing at most k arcs. An important application of the problem in linear curve approximation is discussed. Vertices and edges of associated polytopes are determined, and integrality of these polytopes for certain graphs are shown. We present a combinatorial algorithm for solving the problem and compare it to other methods based on Lagrangian relaxation and dynamic programming. Numerical results for curve approximation problems are reported.
Keywords: Approximation, cardinality constrained shortest path, polyhedra.

1. Introduction. Piecewise linear functions (of one variable) are often used to approximate complex functions or geometrical objects in areas like computer aided geometric design, cartography, computer graphics, image processing [2] and mathematical programming [4]. Piecewise linear functions are popular because they are easy to obtain and manipulate and may still provide a sufficiently good approximation in the problem studied.

Applications in the mentioned areas often include a huge amount of data (breakpoints in the piecewise linear function), and this may cause difficulties regarding to storage space, transmission rates or the time taken to display the curve on a graphical device. This naturally leads to *data reduction problems* where one wants to determine a new piecewise linear function that approximates the original one, but has fewer breakpoints. It is common to require that the approximating curve should not deviate too much from the old one as measured by some norm defined on the function space.

Different methods have been suggested for this data reduction problem, see [11] and [1] (and the references cited therein) for surveys. The majority of the methods select the new breakpoints, called "critical breakpoints", as a subset of the old ones, while others insert new breakpoints that are close to the old ones in order to improve the data compression rate. The different algorithms may be classified into *local* and *global methods*. A local method makes a single sweep through the data and determines the critical breakpoints based on comparisons of neighbor breakpoints. A global method considers intervals of the curve (possibly the entire curve) and decreases the number of critical breakpoints in each iteration until a "low number" of breakpoints is reached. The local methods may have a linear computational time, i.e., the number of calculations in the algorithm is proportional to the number of breakpoints of the initial curve. For global methods the computational complexity is usually higher, but better solutions are found. A major disadvantage of many of these methods is that they may not provide sufficiently high data compression.

Imai & Iri [10] studied the data reduction problem with the new breakpoints given as a subset of the old ones, and stated that it can be viewed as a longest path problem in an acyclic directed graph. In [13] this idea is developed further and one studies a a shortest path problem in a similar graph.

In this paper we extend this problem and require that the approximating curve

*Institute of Informatics, University of Oslo, P.O.Box 1080, Blindern, 0316 Oslo, Norway (Email:geird@ifi.uio.no) should not contain more than a specified number of breakpoints. This requirement is of interest because it gives direct control of the compression rate. We formulate this problem as a constrained shortest path problem and analyze the structure of this problem and associated polytopes for interesting classes of graphs. For these graphs the compression rate is at most 50 %. Using adjacency results for certain path polytopes we present a fast combinatorial algorithm for solving the approximation problem. Due to integrality properties of the mentioned polytopes the problem can also be solved by linear programming and Lagrangian relaxation. We describe and compare some of these algorithms and report computational experiences.

The paper is organized as follows. In the next section we describe the approximation problem of interest and how it translates into a constrained shortest path problem. In Section 3 we study polytopes related to the constrained shortest path problem, and how their vertices and edges are connected to paths in the given acyclic directed graph. Different results on vertices, edges and integrality of polytopes are described. Some of these results lead to algorithms for solving the approximation problem as discussed in Section 4. Other algorithms, e.g. based on dynamic programming, are also presented. Computational results and experiences as well as some examples are given in Section 5.

Some remarks on our notation are in order. $\mathbb{R}$ denotes the set of real numbers. The optimal value of an optimization problem Q is denoted by $v(Q)$. For a finite set A we let $\mathbb{R}^A$ denote the set of real vectors indexed by A (so, by selecting an ordering of the elements of A this set is the Euclidean space $\mathbb{R}^{|A|}$). We use fairly standard graph theoretic notation, see any modern text book in graph theory. For polyhedral theory, we refer to [14], [3] or [12]. $A \Delta B$ denotes the symmetric difference of two sets A and B, i.e., $(A \setminus B) \cup (B \setminus A)$.

2. Approximation and the constrained shortest path problem. We shall describe the curve approximation problem of interest in this paper.

Let $[a, b]$ be a nonempty interval of real numbers. For a set of points $Z = \{\mathbf{z}_i = (t_i, f_i) : i = 0, \ldots, n\} \subset \mathbb{R}^2$ where $a = t_0 < t_1 < \ldots < t_n = b$ we let f_Z denote the linear interpolant of Z defined on the interval $[a, b]$. That is, $f_Z(t_i) = f_i$ for $i = 0, \ldots, n$ and f_Z is linear on each of the subintervals $[t_i, t_{i+1}]$ for $i = 0, \ldots, n-1$. The function f_Z is continuous and piecewise linear (and, clearly, any real-valued function h defined on $[a, b]$ with these two properties is of the form $h = f_Z$ for suitable Z). Each point in Z is called a *breakpoint* of f_Z (even if the left-sided and right-sided derivative of the function coincide at that point). If $S \subseteq Z$ is such that $\mathbf{z}_0, \mathbf{z}_n \in S$ the function f_S is called a *subinterpolant* of f_Z. Thus the subinterpolant interpolates f_Z in some subset S of Z and it may be viewed as an approximation of f_Z.

The *curve approximation problem* (*CAPX*) is the following approximation problem: given a set Z as above and a number $k \leq n$, find a subinterpolant f_S of f_Z with $|S| \leq k + 1$ such that $\|f_Z - f_S\|$ is minimized. Here $\| \cdot \|$ denotes some norm on the vector space $C_{a,b}$ of continuous real-valued functions f defined on $[a, b]$ (and satisfying $f(a) = f(b) = 0$). The subinterpolant has at most k linear segments (i.e., subintervals between consecutive breakpoints). We shall restrict the attention to norms $\| \cdot \|$ that satisfy the following "local additivity" property: if $f \in C_{a,b}$ and $J_1, \ldots, J_s$ is a partition of $[a, b]$ into closed intervals, then we have

$$\|f\| = \|f \cdot I^{J_1}\| + \ldots + \|f \cdot I^{J_s}\| \qquad (2.1)$$

where $t \to I^J(t)$ is the indicator function which is 1 for $t \in J$ and 0 otherwise. For instance, any L_p norm defined by $\|f\|_p = \int_a^b |f(t)| dt$ for $1 \leq p < \infty$ satisfies (2.1) due to the additivity of the integral.

The local additivity property opens up for a reformulation of the (CAPX) problem as discussed next. Let f_S be a subinterpolant of f_Z, say that

$$S = \{(t_{i_0}, f_{i_0}), (t_{i_1}, f_{i_1}), \ldots, (t_{i_s}, f_{i_s})\}$$

where $i_0 = 0$ and $i_s = n$. Then we have

$$\|f_Z - f_S\| = \sum_{j=0}^{s-1} \|(f_Z - f_S) \cdot I^{[t_{i_j}, t_{i_{j+1}}]}\| = \sum_{j=0}^{s-1} c_{i_j, i_{j+1}},$$

where we define

$$c_{j,l} := \|(f_Z - f_S) \cdot I^{[t_{i_j}, t_{i_l}]}\| \quad \text{for } 0 \le j < l \le n. \tag{2.2}$$

This means that the total approximation error is the sum of the local errors between consecutive interpolation points. Furthermore, the information needed to calculate the approximation error for any subinterpolant of f_Z are the numbers $c_{j,l}$ for $0 \le j < l \le n$.

The (CAPX) problem may be transformed into a combinatorial optimization problem as follows. We introduce a directed graph $D = (V, E)$ with node set V and arc set E. V consists of the nodes $v_i = (t_i, f_i)$ for $i = 0, \ldots, n$, that is, the breakpoints of the target function f. For each $1 \le i < j \le n$ we introduce an arc (v_i, v_j) and E consists of all these arcs. The digraph D is clearly acyclic. There is a correspondence between subinterpolants of f and directed paths from v_0 to v_n in D: to a subinterpolant g with breakpoints in t_{i_j} for $j = 0, \ldots, s$ (and $i_0 = 0$, $i_s = n$) we define the path with nodes $v_{i_0}, v_{i_1}, \ldots, v_{i_s}$. Thus each linear segment of g corresponds to an arc in the path. Therefore the number of linear segments of g equals the number of arcs in the path. We define the weight of the arc (v_j, v_l) to be $c_{j,l}$ given in (2.2). We see that the (CAPX) problem is equivalent to the problem

$$\min\{\mathbf{c}(P) : P \in \mathcal{P}(k)\} \tag{2.3}$$

where $\mathcal{P}(k)$ denotes the set of directed v_0v_n-paths in D with at most k arcs and $\mathbf{c}(P) := \sum_{e \in P} c_e$ denotes the weight of the directed path P.

We call the optimization problem (2.3) the *k-constrained shortest path problem.*

This problem may be of interest in other settings as well, but with a different digraph and weights. For instance, an interesting problem in telecommunications is to route traffic between two destinations on a single path which satisfies a "hop constraint". In fact, a bound on the number of arcs in the path reflects a lower bound on the reliability of that path (under standard assumptions of a stochastic graph with independent arc failures). More general models with hop constraints have been studied in [7].

Remark. The CAPX problem may also be presented for curves in higher dimensional spaces; a curve is then a continuous vector-valued function defined on a real interval $[a, b]$. We see that this problem may also be transformed into a CSP problem.

We note that (2.3) is a special case of the shortest path problem with a "time constraint" of the form $\sum_{(i,j) \in P} t_{i,j} \le k$ where $t_{i,j}$ may be interpreted as the time it takes to move from node i to node j in the digraph. This problem is known to be *NP*-hard, see [6]. The CSP problem is polynomially solvable (see [6]) using dynamic programming (see Section 4). The problem is also treated in [13].

We fix some terminology. By the term "path" we hereafter mean a directed path. The cardinality of a path is its number of arcs. The set of all v_iv_j-paths in D is denoted by $\mathcal{P}_{i,j}$ (a path is usually considered as an arc set). We also define $\mathcal{P}_{i,j}(k) := \{P \in \mathcal{P}_{i,j} : |P| \leq k\}$, i.e., this is the set of v_iv_j-paths of cardinality at most k. For simplicity, we write $\mathcal{P} := \mathcal{P}_{0,n}$ and $\mathcal{P}(k) := \mathcal{P}_{0,n}(k)$. For a path P we define its *weight* $c(P) := \sum_{(i,j)\in P} c_{i,j}$.

The augmentation of two internally node-disjoint paths $P \in \mathcal{P}_{i,j}$ and $Q \in \mathcal{P}_{j,l}$ is denoted by (P, Q); this is the path from v_i to v_l obtained by augmenting the sequence of arcs in P by the sequence of arcs in Q.

3. CSP polytopes and integrality. In this section we study polytopes associated with the constrained shortest path problem. From some polyhedral results we also determine a class of graphs for which the CSP problem may be solved as a linear program. This class of graphs is of interest in the curve approximation problem.

Let $\mathbf{A}$ be the node-arc incidence matrix of an acyclic digraph $D = (V, E)$ and recall that $V = \{v_0, \ldots, v_n\}$. Define $\mathbf{b} \in \mathbb{R}^V$ by $b_{v_0} = 1$, $b_{v_n} = -1$ and $b_{v_j} = 0$ for $0 < j < n$. It is well-known that the convex hull of incidence vectors of v_0v_n-paths in D coincides with the polytope

$$M = \{\mathbf{x} \in \mathbb{R}^E : \mathbf{Ax} = \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}. \tag{3.1}$$

This follows from the fact that $\mathbf{A}$ is totally unimodular (each subdeterminant is either -1, 0 or 1) and integrality results for associated polyhedra, see e.g., [14], [12]. We call M the *path polytope*. Thus the shortest path problem corresponds to minimizing some linear function over M. We are interested in the polytope $M(k)$ obtained by intersecting M with the halfspace $\{\mathbf{x} \in \mathbb{R}^E : \mathbf{x}(E) \leq k\}$, i.e.,

$$M(k) = \{\mathbf{x} \in \mathbb{R}^E : \mathbf{Ax} = \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1},\ \mathbf{x}(E) \leq k\}. \tag{3.2}$$

If the underlying graph needs to be indicated we may write $M(D; k)$ for this polytope.

The main goal in this section is to study vertices and integrality of $M(k)$. Note that the integral vectors in $M(k)$ are the incidence of v_0v_n-paths with at most k arcs. Thus, the problem (CSP) is equivalent to the integer linear program

$$\min\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in M(k),\ \mathbf{x} \text{ is integral}\}$$

which explains our interest in $M(k)$. We also remark that the matrix $\mathbf{A}$ augmented with a row $\mathbf{1}$ is not totally unimodular.

Some of the vertices of M are also vertices of $M(k)$, namely the vertices corresponding to paths in $\mathcal{P}(k)$, i.e., v_0v_n-paths of length at most k. However, $M(k)$ may also have other vertices and we shall determine the vertex set of this polytope below.

We define some useful terminology. A *two-terminal* graph is a graph D with two specified distinct nodes ("terminals") u and v in D. We define the *sum* $D_1 + D_2$ where D_i is a two-terminal graph with terminals u_i and v_i for $i = 1, 2$ as the two-terminal graph with node set $V[D_1] \cup V[D_2]$ where nodes v_1 and u_2 are identified and with arc set $E[D_1] \cup E[D_2]$, and, finally, with terminals u_1 and v_2. By repeating this sum operation we may get $D_1 + D_2 + \ldots + D_t$ (this binary operation $+$ is associative on the set of two-terminal graphs).

If P_1 and P_2 are two internally node-disjoint and non-trivial uv-paths in D, we call $P_1 \cup P_2$ a *uv-split*. A two-terminal graph of the form $P_1 + S_1 + P_2 + S_2 + \ldots + P_s + S_s + P_{s+1}$ where, for each i, P_i is a u_iv_i-path and S_i is a v_iu_{i+1}-split is called a

s-split-path from u_1 to v_{s+1}. We here allow trivial paths, i.e., consisting of one node only (but splits are non-trivial). Note that an s-split-path has exactly s splits. By a split-path in a graph D we mean a subgraph of D which is a split-path.

Consider a 1-split-path $Q = P_1 + S_1 + P_2$ where S_1 consists of the two node-disjoint paths C_1 and C_2. Assume that $|C_1| \leq |C_2|$. We may then construct the two paths $P_1 + C_1 + P_2$ and $P_1 + C_2 + P_2$ and we define $k(Q)$ to be the set of integers lying *strictly* between the lengths of these two paths, i.e.,

$$(3.3) \qquad k(Q) = \{|P_1| + |P_2| + |C_1| + 1, \ldots, |P_1| + |P_2| + |C_2| - 1\}$$

Note that $k(Q)$ is empty iff $|C_2| = |C_1|$ or $|C_2| = |C_1| + 1$. We say that Q *covers* an integer k if $k \in k(Q)$. We also let $k(D)$ denote the union of all the sets $k(Q)$ taken over all 1-split-paths Q from v_0 to v_n in the digraph D.

We shall determine the edges of the path polytope M given in (3.1), and for this the following well-known results on adjacency are useful (see [8]); we include a proof for completeness.

LEMMA 3.1. *Let $\mathcal{F}$ be a class of subsets of $\{1, \ldots, n\}$ and define the associated polytope $T = \text{conv}(\{\chi^F : F \in \mathcal{F}\})$. Let $F_1, F_2 \in \mathcal{F}$.*

(i) If there are two sets $F_1', F_2' \in \mathcal{F}$ both distinct from F_1 and F_2 and such that $F_1' \cap F_2' = F_1 \cap F_2$ and $F_1' \cup F_2' = F_1 \cup F_2$, then χ^{F_1} and χ^{F_2} are not adjacent on T.

(ii) If there is no $F \in \mathcal{F}$ distinct from F_1 and F_2 and with $F_1 \cap F_2 \subseteq F \subseteq F_1 \cup F_2$, then χ^{F_1} and χ^{F_2} are adjacent on T.

Proof. First we note that the vertices of T are the vectors χ^F, $F \in \mathcal{F}$; this follows from the fact that $T \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$. χ^{F_1} and χ^{F_2} are adjacent iff there is an objective function (vector) $\mathbf{c} \in \mathbb{R}^n$ such that the optimal vertex solutions of the LP problem $\max\,\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in T\}$ are precisely the points χ^{F_1} and χ^{F_2}.

(i) Assume that $F_1', F_2' \in \mathcal{F}$ are both distinct from F_1 and F_2 and that $F_1' \cap F_2' = F_1 \cap F_2$ and $F_1' \cup F_2' = F_1 \cup F_2$. From this we get

$$1/2 \cdot (\chi^{F_1} + \chi^{F_2}) = 1/2 \cdot (\chi^{F_1'} + \chi^{F_2'}).$$

Thus, if χ^{F_1} and χ^{F_2} are optimal in $\max\,\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in T\}$, then their midpoint is also optimal which again implies that both $\chi^{F_1'}$ and $\chi^{F_2'}$ are optimal. This proves property (i).

(ii) Assume that there is no $F \in \mathcal{F}$ distinct from F_1 and F_2 and with $F_1 \cap F_2 \subseteq F \subseteq F_1 \cup F_2$. Let γ be a "suitably" large number, and define c by $c_j = \gamma$ for $j \in F_1 \cap F_2$, $c_j = -\gamma$ for $j \notin F_1 \cup F_2$, $c_j = |F_2 \setminus F_1|$ for $j \in F_1 \setminus F_2$ and $c_j = |F_1 \setminus F_2|$ for $j \in F_2 \setminus F_1$. We see that if $F \in \mathcal{F}$ is such that χ^F is optimal in $\max\,\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in T\}$ we must have that $F_1 \cap F_2 \subseteq F \subseteq F_1 \cup F_2$. Property (ii) follows directly from this.
□

The edges of the path polytope are described next.

PROPOSITION 3.2. *Consider the two-terminal graph (D, v_0, v_n) and path polytope M in (3.1). Let P_1 and P_2 be two v_0v_n-paths. Then the vertices χ^{P_1} and χ^{P_2} are adjacent on M if and only if $P_1 \Delta P_2$ is a split, or, equivalently, $P_1 \cup P_2$ is a 1-split-path.*

Proof. We first prove the necessity of the condition. Let $P_1, P_2 \in \mathcal{P}$ be distinct paths such that χ^{P_1} and χ^{P_2} are adjacent on M. Since D is acyclic $P_1 \cup P_2$ is an s-split-path for some integer s where each split contains one subpath of P_1 and one subpath of P_2. Assume that $s \geq 2$. Select a split S and consider the two v_0v_n-paths $Q_1 = (P_1 \setminus S) \cup (P_2 \cap S)$ and $Q_2 = (P_2 \setminus S) \cup (P_1 \cap S)$. Then $Q_1 \cap Q_2 = P_1 \cap P_2$ and

$Q_1 \cup Q_2 = P_1 \cup P_2$ and, as $s \geq 2$, both these new paths are distinct from P_1 and P_2. This contradicts that χ^{P_1} and χ^{P_2} are adjacent according to Lemma 3.1. It follows that $s = 1$ (clearly $s \geq 1$ as P_1 and P_2 are distinct) and $P_1 \cup P_2$ is a 1-split-path as desired.

To prove the converse, assume that $P_1 \cup P_2$ is a 1-split-path. We see that the only v_0v_n-paths contained in $P_1 \cup P_2$ are P_1 and P_2, and by Lemma 3.1 χ^{P_1} and χ^{P_2} are adjacent.

$\square$

Consider next the polytope $M(k)$. All integer points in $M(k)$ are also vertices; these points are the incidence vectors of paths in $\mathcal{P}(k)$. However, $M(k)$ may also have other vertices and these are constructed from the 1-split-paths as discussed next.

Consider a 1-split path $Q = P_1 + S + P_2$ such that Q covers k (i.e., $k \in k(Q)$). Assume that $S = C_1 \cup C_2$ with $|C_1| < |C_2|$ (as above C_1 and C_2 are paths) and let $\lambda^Q = (|C_2| + |P_1| + |P_2| - k)/(|C_2| - |C_1|)$. We define the *split-solution* $\mathbf{x}^Q \in \mathbb{R}^E$ by $x^Q_e = 1$ for $e \in P_1 \cup P_2$, $x^Q_e = \lambda^Q$ for $e \in C_1$, $x^Q_e = 1 - \lambda^Q$ for $e \in C_2$, and $x^Q_e = 0$ for all other arcs e. Observe that (i) $\mathbf{0} \leq \mathbf{x}^Q \leq \mathbf{1}$, (ii) $0 < x^Q_e < 1$ if and only $e \in S$, (iii) $\mathbf{A}\mathbf{x}^Q = \mathbf{b}$, and (iv) $\mathbf{x}^Q(E) = k$. Consequently, we have that $\mathbf{x}^Q \in M(k)$. Moreover, $\mathbf{x}^Q$ is a (strict) convex combination of the incidence vectors of the two paths $P_1 + C_1 + P_2$ and $P_1 + C_2 + P_2$.

Proposition 3.3. *The vertex set of $M(k)$ consists of the incidence vectors of each path in $\mathcal{P}(k)$ and the split solution $\mathbf{x}^Q$ for each 1-split-path Q from v_0 to v_n such that Q covers k. In particular, $M(k)$ is integral if and only $k \notin k(D)$. Thus, in this case, we have that*

$$\text{(3.4)} \qquad \text{conv}(\{\chi^P : P \in \mathcal{P}(k)\}) = \{\mathbf{x} \in \mathbb{R}^E : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1},\ \mathbf{x}(E) \leq k\}.$$

Proof. The vertices of $M(k)$ are (i) the vertices of M that satisfy $x(E) \leq k$, and (ii) the points obtained as the intersection of the relative interior of an edge of M with the hyperplane $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}(E) = k\}$. This follows from a general result on the intersection of a polytope and a halfspace, see e.g., [3].

Consider an edge F of M having a relative interior point $\mathbf{x}'$ in the hyperplane $\{\mathbf{x} \in \mathbb{R}^E : \mathbf{x}(E) = k\}$. By Proposition 3.2 there is a split-path $Q = P_1 + S + P_2$ with $S = C_1 \cup C_2$ and $|C_1| < |C_2|$ such that $F = [\chi^{Q_1}, \chi^{Q_2}]$ where $Q_i = P_1 + C_i + P_2$ for $i = 1, 2$. Since $\mathbf{x}'(E) = k$ we get $\chi^{Q_1}(E) < k < \chi^{Q_2}(E)$ so S covers k and $\mathbf{x}'$ must coincide with the split-solution x^Q and the proof is complete.

$\square$

An immediate consequence of the previous result is the following integrality result of interest in curve approximation.

Corollary 3.4. *Assume that D contains no split $S = C_1 \cup C_2$ with $|C_2| - |C_1| \geq 2$. Then $M(k)$ is integral for all k.*

Proof. The condition implies that $k(Q) = \emptyset$ for each split-path Q in D and therefore $k(D) = \emptyset$. The result then then follows from Proposition 3.3.

$\square$

An interesting question is to determine the set K^I of those $k \leq n$ for which $M(k)$ is integral. Let L_* (resp. L^*) be the minimum (resp. maximum) cardinality of a v_0v_n-path in D. From Proposition 3.3 it follows that $K^I = \{L_*, \ldots, n\} \setminus k(D)$. If $k < L_*$ then $M(k) = \emptyset$, and if $k \geq L^*$ then $M(k) = M$. Thus, we are led to a study of some properties of cover sets $k(D)$. We shall see that for many graphs K^I is empty, but there some interesting exceptional cases that are also of interest in curve approximation.

We shall determine the cover set of the graph $D^1 + \ldots + D^m$ where for each $j \leq m$ D^j is a split consisting of two paths of lengths a_i and b_i. Let $D(\mathbf{a}, \mathbf{b})$ denote this graph, where $\mathbf{a} = (a_1, \ldots, a_m)$ and $\mathbf{b} = (b_1, \ldots, b_m)$. We assume that $a_i < b_i$ and that $b_i - a_i \geq b_{i+1} - a_{i+1}$ for $i = 1, \ldots, m-1$ (otherwise we could change the order of the graphs D^i; the cover set is not altered by this operation).

LEMMA 3.5. *Let $D(\mathbf{a}, \mathbf{b})$ be as above and define $r_i = b_i - a_i$ for $i \leq m$, so $L_* = \sum_{j=1}^n a_j$ and $L^* = \sum_{j=1}^n b_j$.*

If all the r_i's are equal, say $r_i = d$ for $i = 1, \ldots, m$, then $k(D) = \{L_* < j < L^* : j \not\equiv 0 \pmod d\}$. Otherwise (i.e., when $r_1 > r_m$) we have that $k(D) = \{L_* + 1, \ldots, L^* - 1\}$.

Proof. Let u_1 and v_m be the terminals in $D(\mathbf{a}, \mathbf{b})$. Each $u_1 v_m$-path P in $D(\mathbf{a}, \mathbf{b})$ is constructed by selecting for each $i \leq m$ either the path of length a_i or the one with length b_i in the graph D_i and then adding all these paths together. Thus there are 2^m different paths. Moreover, there is a one-to-one correspondence between the set of these paths and the set of all subsets of $\{1, \ldots, m\}$: $S \subseteq \{1, \ldots, m\}$ determines those i for which we choose b_i instead of a_i. From Proposition 3.2 it is clear that two paths are adjacent in the path polytope M if and only if they differ for exactly one i, or, equivalently, if $S' = S \cup \{i\}$ where S and S' correspond to the two paths.

For each subset S of $\{1, \ldots, m\}$ we let, as usual, $r(S) := \sum_{j \in S} r_j$. Based on the remarks above we want to determine the numbers s that can be covered in the sense that $r(S) < s < r(S) + r_j$ for some $S \subset \{1, \ldots, m\}$ and some $j \in \{1, \ldots, m\} \setminus S$.

Consider the case when $r_i = d$ for $i = 1, \ldots, m$. For each $S \subset \{1, \ldots, m\}$ we then have $r(S) = d \cdot |S|$ and $r(S) + r_j = d \cdot (|S| + 1)$ for $j \notin S$. Thus it is clear that $k(D)$ consists of all integers lying strictly between L_* and L^* except $d, 2d, \ldots, (m-1)d$.

Finally, consider the case with $r_1 > r_m$. We use the notation $N_t := \{1, \ldots, t\}$ for each natural number t. Let $t \leq m-1$ and consider $S = \{1, \ldots, t\}$ and $j = t+1$. The pair (S, j) then covers each integer lying strictly between $r(N_t)$ and $r(N_{t+1})$. To cover $r(N_t)$ let $S = \{2, \ldots, t, m\}$ so $r(S) = r(N_t) + r_m - r_1$. Since $r_1 > r_m$ we have $r(S) < r(N_t)$. Furthermore, $r(S) + r_1 = r(N_t) + r_m > r(N_t)$ as $r_m \geq 1$, so $r(N_t)$ is covered by the pair $(S, 1)$. This can be done for each $t \in \{1, \ldots, m-1\}$, and therefore $K(D) = \{L_* + 1, \ldots, L^* - 1\}$ as desired.
□

Based on this lemma one may argue that in "most" graphs $k(D)$ becomes very large, which again means that the polytope $M(k)$ has fractional vertices for most values of k. To realize this, one may consider different pairs of $v_0 v_n$-paths in D and apply the lemma to that subgraph, e.g., one may consider a path pair consisting of a shortest and a longest $v_0 v_n$-path. However, there is an interesting class of graphs for which $k(D) = \emptyset$, i.e., $M(k)$ is integral for every k.

Define the *2-graph* $T_n = (V, E)$ by $V = \{v_0, \ldots, v_n\}$ and $E = \{(v_i, v_j) : 0 \leq i < j \leq i + 2 \leq n\}$. To simplify the notation we shall sometimes denote a node v_i by just i. We view T_n as a two terminal graph with terminals 0 and n. The longest $(0, n)$-path in T_n consists of all arcs $(i, i+1)$ for $i = 0, \ldots, n-1$ so it has cardinality n, while the shortest path has cardinality $\lceil n/2 \rceil$.

For 2-graphs the structure of split-paths is very simple. Let $0 \leq i < j \leq n$ and let C_1 and C_2 be two internally node-disjoint ij-paths in T_n. We may assume that $(i, i+1) \in C_1$ (otherwise we rename the paths). If $j - i$ is even, it follows from the disjointness that C_1 contains the nodes $i, i+1, i+3, \ldots, j-3, j-1, j$ (and the corresponding arcs) while C_2 contains the nodes $i, i+2, i+4, \ldots, j-2, j$. Similarly, if $j - i$ is odd, then C_1 contains the nodes $i, i+1, i+3, i+5, \ldots, j-2, j$ and C_2

contains the nodes $i, i+2, i+4, i+6, \ldots, j-3, j-1, j$. In particular, this shows that there is exactly one pair of internally node-disjoint (i,j)-paths in T_n. Furthermore, the cardinalities of these two paths are equal when $j-i$ is odd and they differ by 1 when $j-i$ is even. We call the split $C_1 \cup C_2$ *even* (resp. *odd*) when $j-i$ is even (resp. odd).

Corollary 3.6. *For each $k \leq n$, the polyhedron $M(T_n;k)$ is integral.*

Proof. This follows from the mentioned properties of splits in T_n and Corollary 3.4.
$\square$

Let $\mathbf{A}$ be the node-arc incidence matrix of T_n (and the supply vector $\mathbf{b}$ as usual for the two terminal problem). We note that the matrix $\mathbf{A}$ augmented with the row $\mathbf{1}$ is not totally unimodular, for instance one can find 2×2 submatrices with determinant equal to 2. Thus our integrality result can not be derived directly from standard integrality results for polyhedra associated with totally unimodular matrices.

In Proposition 3.2 we gave a characterization of adjacency on the path polytope. We now consider the same question for the polytope $M(T_n;k)$, which by Corollary 3.6 is the convex hull of the incidence vectors of v_0v_n-paths in T_n with at most k arcs. Geometrically, what happens is that the hyperplane $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}(E) = k\}$ introduces some "new edges" between vertices corresponding to paths of cardinality k.

The structure of split-paths in T_n is also sufficiently simple to allow a complete characterization of the edges of $M(T_n;k)$.

Proposition 3.7. *Let P_1 and P_2 be two $(0,n)$-paths in T_n with $|P_1|, |P_2| \leq k$. Then χ^{P_1} and χ^{P_2} are adjacent on $M(T_n;k)$ if and only if $P_1 \Delta P_2$ is a 1-split-path or a 2-split-path.*

Proof. We shall use the geometrical fact that every edge of the intersection of a polytope P with a halfspace H is either an edge of P lying in H or an edge obtained as the intersection of a two-dimensional face of P with the hyperplane that H defines.

Assume that $\mathbf{x}^1 = \chi^{P_1}$ and $\mathbf{x}^2 = \chi^{P_2}$ are adjacent on $M(T_n;k)$. We shall show the necessity of the condition. If at least one of the two paths has cardinality less than k, then it is easy to see that $\mathbf{x}^1$ and $\mathbf{x}^2$ are adjacent on $M(T_n;k)$ if and only if they are adjacent on $M(T_n)$ which again, by Proposition 3.2 is equivalent to that $P_1 \Delta P_2$ is a split-path. Thus, it remains to consider the case when $|P_1| = |P_2| = k$.

$P_1 \cup P_2$ is an s-split-path for suitable $s \geq 1$ with splits $S^i = C_1^i \cup C_2^i$ (where $C_1^i \subseteq P_1$ and $C_2^i \subseteq P_2$) for $i \leq s$. We say that S^i is of type t if $|C_1^i| - |C_2^i| = t$ for $t = -1, 0, 1$. It follows from our discussion of split-paths for T_n that each S^i is of exactly one of these types. Let q_t be the number of i's for which S^i is of type t.

Using similar arguments to those in the first part of the proof of Proposition 3.2 one can show that $q_0 \in \{0,1\}$. Furthermore, since P_1 and P_2 have equal cardinality we have $q_{-1} = q_1$.

We claim that $q_1 \leq 1$. To see this, assume that $q_1 \geq 2$ and therefore there are two splits S^1 and S^2 of type 1 and two splits S^3 and S^4 of type -1. Let P_1' be obtained from P_1 by "switching" over S^1 and S^3, i.e., replacing $P_1 \cap S^t$ by $P_2 \cap S^t$ for $t = 1,3$. Similarly, let P_2' be obtained from P_2 by switching over S^1 and S^3. Then P_1' and P_2' are both $(0,n)$-paths of cardinality k and by Lemma 3.1 it follows that χ^{P_1} and χ^{P_2} are not adjacent. This proves the claim.

Finally, we observe that if $q_{-1} = q_1 = 1$, then $q_0 = 0$. Otherwise we could make a switch over the split-cycle of type 0 and find another feasible pair of paths violating the necessary adjacency condition of Lemma 3.1.

Thus the only two remaining possibilities are that $P_1 \cup P_2$ is either a 1-split-path or a 2-split-path as desired.

It remains to prove that if $P_1 \cup P_2$ is either a 1-split-path or a 2-split-path, then χ^{P_1} and χ^{P_2} are adjacent. We note that in each of these cases there is no $(0, n)$-path P distinct from P_1 and P_2 and satisfying $P_1 \cap P_2 \subset P_1 \cup P_2$. Thus, the adjacency follows from the second part of Lemma 3.1.

$\square$

This adjacency characterization may be exploited algorithmically for solving the CSP problem in 2-graphs. Starting in any vertex of $M(T_n; k)$ one may check if any adjacent vertex is better by considering 1-splits and 2-splits. If a better vertex is found, one moves to that one; otherwise, the solution is optimal by linear programming theory. In the next section some algorithms for solving CSP are presented.

We now return to the CSP problem in a general acyclic graph D. It is of interest to study the CSP problem using Lagrangian duality. Let $\lambda \geq 0$ and consider the relaxed problem

(3.5) $$\min\{\mathbf{c}^T x + \lambda(\mathbf{x}(E) - k) : \mathbf{x} \in M\}$$

which is called the *Lagrangian subproblem* and denoted by $LR_k(\lambda)$. Note that this is a shortest path problem with a modified cost function. The *Lagrangian dual problem* is to maximize $v(LR_k(\lambda))$ for $\lambda \geq 0$. (Recall that $v(LR_k(\lambda)$ denoted the optimal value of the problem $LR_k(\lambda)$). From the general theory of Lagrangian relaxation (see [12]) it follows that

(3.6) $$v(CSP) \geq v^* = v(LP_k)$$

where

(3.7) $$v^* := \max_{\lambda \geq 0} v(LR_k(\lambda))$$

provides the best lower bound on $v(CSP)$ obtained from Lagrangian duality and LP_k denote the linear programming relaxation of CSP:

$$\min\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in M(k)\}.$$

The equality in (3.6) is due to the fact that the path polytope $M = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b};\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$ is integral. If, furthermore, the polytope $M(k)$ is integral (as is the case for 2-graphs), the inequality in (3.6) may be replaced by an equality.

Let $\mathbf{x}^1, \ldots, \mathbf{x}^N$ denote the incidence vectors of all v_0v_n-paths in D and let $g_k^j(\lambda) = \mathbf{c}^T\mathbf{x}^j + \lambda(\mathbf{x}^j(E) - k)$ be the Lagrangian cost of the solution $\mathbf{x}^j$. Then we have

$$v(LR_k(\lambda)) = \min_{j \leq t} g_k^j(\lambda).$$

Thus the function $v(LR_k(\cdot))$ is the pointwise minimum of a finite (but large) number of affine functions, so it is piecewise linear and concave. The Lagrangian dual problem is therefore to maximize this concave function in the single variable λ.

Consider an optimal solution λ^* of (3.7) and define $J_k^* = \{j \leq N : g_k^j(\lambda^*) = v^*\}$. The graphs of the affine functions g_k^j for $j \in J_k^*$ all go through the point (λ^*, v^*). By optimality of λ^* this solution may be chosen such that $\mathbf{x}^j(E) - k \leq 0$ and $\mathbf{x}^l(E) - k > 0$ for suitable $j, l \in J_k^*$. The solution $\mathbf{x}^j$ is therefore both a feasible solution of CSP and it solves the Lagrangian subproblem $LR(\lambda^*)$. The question of integrality of $M(k)$ may now treated in terms of Lagrangian duality.

PROPOSITION 3.8. *Assume that $v(CSP) > 0$. Then there is an optimal integral solution of the LP problem max $\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in M(k)\}$ if and only if there is an $s \in J_k^*$ with $\mathbf{x}^s(E) = k$.*

Proof. It follows from $v(CSP) > 0$ that any solution $\mathbf{x}^j$ with $\mathbf{c}^T\mathbf{x}^j = 0$ must be such that $\mathbf{x}^j(E) > k$. This implies that an optimal solution λ^* of the Lagrangian dual problem must be strictly positive. Consider an $j \leq N$ such that $\mathbf{x}^j(E) \leq k$, i.e., $\mathbf{x}^j$ is an integral feasible solution of max $\{\mathbf{c}^T\mathbf{x} : x \in M(k)\}$, and therefore also feasible in CSP. For each $j \in J_k^*$ we then obtain from (3.6) that

$$\mathbf{c}^T\mathbf{x}^j \geq v(CSP) \geq v(LP_k) = v(LR_k(\lambda^*)) = \mathbf{c}^T\mathbf{x}^j + \lambda^*(\mathbf{x}^j(E) - k).$$

Thus, $\mathbf{c}^T\mathbf{x}^j = v(LP)$ if and only if $\mathbf{x}^j(E) = k$ (as $\lambda^* > 0$).
$\square$

This description may be carried further to give a similar integrality result for varying k. Let for each $k \geq 0$ and $\lambda \geq 0$

$$g_k(\lambda) = v(LR_k(\lambda)) = \min_{j \leq N} g_k^j(\lambda). \tag{3.8}$$

As noted above g_0 is piecewise linear and concave. Let λ^r, $r = 1, \ldots, m$ denote the breakpoints of g_0. Thus, for each r, g_0 is linear on the interval $[\lambda^r, \lambda^{r+1}]$ and equal to, say, the function $g_0^{t(r)}$. In the breakpoint λ^r there may be several t's such that $g_0^t(\lambda^r) = g_0(\lambda^r)$; we let $T(r)$ denote the set of such t's. Clearly, we have that $t(r-1), t(r) \in T(r)$ for all r (except for $r = 1$ where $\lambda^1 = 0$ and $t(1) \in T(1)$).

The structure of the solutions $\mathbf{x}^t$ corresponding to each of the sets $T(r)$ determines the existence of an integral optimal solution to the problem max $\{\mathbf{c}^T\mathbf{x} : x \in M(k)\}$ as described next.

COROLLARY 3.9. *The set of integers k such that the LP problem max $\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in M(k)\}$ has an integral optimal solution coincides with the set*

$$\{\mathbf{x}^t(E) : t \in \cup_r T(r)\}.$$

Proof. We note that $g_k^t(\lambda) = g_0^t(\lambda) - \lambda k$ and therefore also $g_k(\lambda) = g_0(\lambda) - \lambda k$. The result now follows directly from Proposition 3.8.
$\square$

We remark that both Proposition 3.8 and Corollary 3.9 are general results that hold for a general combinatorial optimization problem with a cardinality constraint $\mathbf{x}(E) \leq k$. See also [9] for a similar discussion. Furthermore we note that the integrality result for the polytope $M(k)$ given in Corollary 3.4 may be derived from Corollary 3.8 using the additional fact that a polytope is integral if and only if each LP problem over that polytope has an integral optimal solution.

4. Algorithms for solving CSP.

In this section we present several algorithms for solving the CSP problem. Some of these algorithms are based on ideas related to adjacency on the polytopes M and $M(k)$ studied in the previous section. We are mainly interested in applications of CSP to the curve approximation problem CAPX and, more specifically, for digraphs that are 2-graphs (see Section 3). We present the basic algorithmic ideas and some implementation details. Numerical results and experiences are given in Section 5.

We have studied four different algorithms, namely

(i) COMB: a combinatorial algorithm;

(ii) LAGR: an algorithm based on Lagrangian relaxation;
(iii) DYNP: a dynamic programming algorithm;
(iv) SIMX: a linear programming algorithm.

We consider problems in an acyclic graph D with nodes $v_0, \ldots, v_n$ and arcs of the form (v_i, v_j) for $i < j$. These graphs are represented by an adjacency list consisting of the inward arcs to each of the nodes.

It should be remarked that the algorithm DYNP is extendible to problems in general graphs.

A combinatorial algorithm. We shall extend some of the results in Section 3 and thereby develop a combinatorial algorithm for solving the CSP problem in 2-graphs. The development consists of two stages. First, we transform the CSP problem into a linear programming problem with half as many variables as the number of arcs in the 2-graph. Next, some adjacency properties of optimal solutions of this new LP problem (for varying k) are derived. These properties are the foundation of the combinatorial algorithm which we denote by COMB. We shall assume that $\lceil n/2 \rceil \leq k \leq n$ so that CSP has feasible solutions.

The starting point is a reformulation of the CSP problem obtained by elimination of certain variables. We call each arc $(i, i{+}1)$ in $D = (V, E)$ a *basic arc* (in CAPX these arcs correspond to the graph of the target function). Let E^b denote the set of basic arcs and define the subgraph $D^r = (V, E^r)$ where $E^r := E \backslash E^b = \{(i, i{+}2) : 0 \leq i \leq n{-}2\}$. We call D^r the *reduced graph.* Define $c^r_{i,i+2} = c_{i,i+2} - c_{i,i+1} - c_{i+1,i+2}$.

Consider the following integer linear programming associated with D^r. It will be denoted by ILP$_k$.

$$(4.1)\qquad \begin{array}{lll} \text{min} & \sum_{i=0}^{n-2} c^r_{i,i+2}\, y_i & \\ \text{subject to} & & \\ \text{(i)} & y_i + y_{i+1} \leq 1 & \text{for } 0 \leq i \leq n-3; \\ \text{(ii)} & \sum_{i=0}^{n-2} y_i \geq n-k; & \\ \text{(iii)} & y_i \in \{0,1\} & \text{for } 0 \leq i \leq n-2. \end{array}$$

This problem has feasible solutions since $\lceil n/2 \rceil \leq k \leq n$. Let $\mathbf{y} = (y_0, \ldots, y_{n-2})$ be the variable vector in this problem.

ILP$_k$ relates to the CSP as follows. Define for each feasible solution $\mathbf{y}$ of (4.1) the vector $\mathbf{x}^y \in \{0,1\}^E$ by $x^y_{i,i+2} = y_i$ and $x^y_{i,i+1} = x_{i+1,i+2} = 1 - y_i$ for all $i \leq n-2$.

LEMMA 4.1. *If* $\mathbf{y}$ *is an optimal solution of problem* ILP_k*, then* $\mathbf{x}^y$ *is an optimal solution of CSP. In particular, we have* $v(CSP) = v(ILP_k) + c(E^b)$.

Proof. Let $\mathbf{x} \in \{0,1\}^E$ be an integral solution in $M(k)$, i.e., $\mathbf{x}$ is the incidence vector of a v_0v_n-path. We see that (i) if $x_{i+1,i+2} = 1$, then $x_{i,i+2} = x_{i+1,i+3} = 0$, and (ii) if $x_{i+1,i+2} = 0$, then $x_{i,i+2} + x_{i+1,i+3} = 1$ (from connectivity). In addition, we clearly have that $x_{i,i+2} + x_{i+1,i+3} \leq 1$, and therefore $x_{i+1,i+2} = 1 - (x_{i,i+2} + x_{i+1,i+3})$. This implies that all the variables corresponding to the basic arcs in D may be eliminated from the CSP problem, and the resulting problem is precisely (4.1).
□

The elimination of variables described in this lemma means that it suffices to solve the smaller linear programming problem ILP$_k$; see the SIMXr algorithm described later.

We remark that the polyhedron defined by the linear system (4.1)(i)–(iii) is integral. In fact, one can show that the coefficient matrix is a network matrix and therefore totally unimodular (see [12] for a definition of network matrix). In particular, this means that the problem ILP$_k$ may be solved by its LP relaxation. We return to this in the next section.

We proceed by giving a combinatorial result concerning optimal solutions of ILP$_m$ as m is decreased.

THEOREM 4.2. *Let* $\mathbf{z}$ *be an optimal solution of ILP$_m$. Then there is an optimal solution* $\mathbf{y}$ *of ILP$_{m-1}$ such that for certain integers l and r with $l \leq r$ the following holds: (i) $y_j = x_j$ when $j < l$ or $j > r$, (ii) $y_j = 1 - x_j$ when $l \leq j \leq r$, and (iii) for $l \leq j \leq r$ the variable y_j is 1 if $j - l$ even and it is 0 if $j - l$ is odd.*

Proof. To prove the result we initially pick an (arbitrary) optimal solution $\mathbf{y}'$ of ILP$_{m-1}$.

Assume first that there exists a $t \leq n$ such that

$$y'_t = 1,\ y'_{t-1} = y'_{t+1} = x_{t-1} = x_t = x_{t+1} = 0. \tag{4.2}$$

Define $\mathbf{y} \in \{0,1\}^{n-1}$ by $y_j = y'_j$ for $j \in \{t-1, t, t+1\}$ and $y_j = x_j$ for all other j's. Note that $\mathbf{y}$ is feasible in ILP$_{m-1}$. It follows from the optimality of $\mathbf{x}$ (in ILP$_m$) that $\mathbf{c}^T\mathbf{y} \leq \mathbf{c}^T\mathbf{y}'$, so $\mathbf{y}$ is also optimal in ILP$_{m-1}$. Furthermore, $\mathbf{y}$ satisfies properties (i)–(iii) in the theorem as desired.

Next we treat the case when there is no $t \leq n$ such that (4.2) holds. Then there exists for each j with $y'_j = 1$ an $i(j) \in \{j-1, j, j+1\}$ such that $x_{i(j)} = 1$. Let $J^{\neq} = \{j \leq n : i(j) \neq j\}$. Note that $x_j = 1 - y'_j$ for each $j \in J^{\neq}$ and $x_j = y'_j$ for $j \in J \setminus J^{\neq}$. The set $J^{\neq}$ is the union of intervals (of integers) I^r, $r \leq m$ with the property that for each I^j, say $I^j = \{s, s+1, \ldots, t\}$ we have $x_{s-1} = y_{s-1} = 0$ and $x_{t+1} = y_{t+1} = 0$.

Consider such an interval I^r with even cardinality. Define $\mathbf{y}''$ as the solution obtained from $\mathbf{y}'$ by letting $y''_j = 1 - y'_j$ for $j \in I^r$ and $y''_j = y'_j$ otherwise. Then $\mathbf{y}''$ is also optimal in ILP$_{m-1}$. In fact, it is feasible (due to the construction of the interval I^r) and it is optimal because the optimality of $\mathbf{x}$ implies that $\mathbf{c}(\{j \in I^r : y'_j = 1\}) \geq \mathbf{c}(\{j \in I^r : x_j = 1\})$. The new solution $\mathbf{y}''$ agrees with $\mathbf{x}$ in more components than $\mathbf{y}'$ did. Repeating this procedure for all intervals I^r with even cardinality we end up with an optimal solution which we denote by $\mathbf{y}'$.

It remains to study the intervals I^r with odd cardinality; let O denote the set of such r's. Define $n^r_x = |\{j \in I^r : x_j = 1\}|$ and $n^r_y = |\{j \in I^r : y'_j = 1\}|$. Then, for each $r \in O$ we have either that (i) $n^r_x = n^r_y + 1$ or (ii) $n^r_x = n^r_y - 1$. Let O_x (resp. O_y) denote the subset of O for which (i) (resp. (ii)) holds. Since $\sum_{j=0}^{n-2} y'_j = \sum_{j=0}^{n-2} x_j$, we obtain

$$|O_y| = |O_x| + 1.$$

Assume that $|O_x| \geq 1$ and let $r \in O_x$. Let also $r' \in O_y$ (which is possible as $|O_y| \geq 2$). We may then construct a new optimal solution $\mathbf{y}''$ of ILP$_{m-1}$) by defining $y''_j = 1 - y'_j$ for $j \in I^r \cup I^{r'}$ and $y''_j = y'_j$ otherwise. Here the feasiblity and optimality may be argued as above. We may repeat this procedure and thereby end up with another optimal solution $\mathbf{y}$ which coincides with $\mathbf{x}$ everywhere except at one interval I^r with $n^r_y = n^r_x + 1$. This solution $\mathbf{y}$ satisfies properties (i)–(iii) in the theorem and the proof is complete.
□

Based on Theorem 4.2 we obtain the following algorithm, denoted by COMB, for solving ILP$_k$ and therefore CSP in 2-graphs: solve ILP$_m$ for $m = n, n-1, \ldots, k$. Note that we start in the infeasible part of the path polytope (with exception of the case $k = n$ in which the CSP problem is an ordinary shortest path problem). This is done because the number of solutions to be scanned is considerably less compared to the alternative approach of solving ILP$_m$ for increasing m.

The initial problem is trivial; $\mathbf{z} = \mathbf{0}$ is an optimal solution.

The general step is to find an optimal solution $\mathbf{y}$ of ILP$_{m-1}$ from the known optimal solution $\mathbf{z}$ of ILP$_m$. If a feasible solution $\mathbf{z} = (z_0, \ldots, z_{n-2})$ of ILP$_m$ is such that $z_{i-1} = 0, z_i = 0, z_{i+1} = 1, z_{i+2} = 0, z_{i+3} = 1, \ldots, z_{j-1} = 1, z_j = 0, z_{j+1} = 0$ for some i and j, then $\{i, i+1, \ldots, j\}$ is referred to as an $\mathbf{z}$-*interval*. If $z_0 = 1$ or $z_{n-2} = 1$ we define the two special $\mathbf{z}$-intervals $\{0, \ldots, j\}$ and $\{i, \ldots, n-2\}$, respectively. We also note that the $\mathbf{z}$-intervals are pairwise disjoint. Let I^z denote all the indexes in $\{0, \ldots, n-2\}$ that are not contained in any $\mathbf{z}$-interval. It is clear that each feasible solution $\mathbf{z}$ can be represented by its $\mathbf{z}$-intervals. In order to obtain $\mathbf{y}$ from $\mathbf{z}$ one checks each solution obtained by either (i) complementing variables in some $\mathbf{z}$-interval or (ii) setting $y_p = 1$ for an $p \in I^z$ and $y_j = z_j$ othervise.

If no more than $n-k$ weigths are negative we can terminate after $n-k$ iterations. Otherwise we need to solve the problems ILP$_{k-i}$ for $i = 1, 2, \ldots$ until $v(ILP_{k-i}) > v(ILP_{k-i+1})$. The optimal solution of ILP$_{k-i+1}$ will then also be optimal in CSP.

The number of candidate solutions to be checked in iteration m is $2m - n - 1$, so the complexity of the algorithm is

$$\sum_{m=n}^{k} (2m - n - 1) = nk - k^2 - 2k$$

If for instance $k = 3n/4$, we get the complexity $3n^2/16$.

This algorithm may also be interpreted in terms of the original CSP problem in the graph D. This can be done via the transformation we used to derive ILP$_k$ from CSP. Assume that $\mathbf{x}$ is optimal in ILP$_m$. Then $\mathbf{x}$ corresponds to a v_0v_n-path $P(m)$ in D of length m. The candidate solutions $\mathbf{y}$ to be checked for optimality in ILP$_{m-1}$ correspond to those paths $P(m-1)$ in D of length m such that $\chi^{P(m)}$ and $\chi^{P(m-1)}$ are adjacent in the path polytope M. In fact, the interval on which $\mathbf{x}$ and $\mathbf{y}$ differ according to Theorem 4.2 corresponds to an even split, and for 2-graphs such a split has a very simple form, see Section 3.

The COMB algorithm has also a nice geometrical interpretation in terms of the "slices" of the path polytope. Consider the slice

$$S_m = \{\mathbf{x} \in M : \mathbf{x}(E) = m\}$$

for $m = 1, \ldots, n$. COMB moves in each iteration from an optimal solution of $\mathbf{c}^T\mathbf{x}$ over S_m to an optimal solution of $\mathbf{c}^T\mathbf{x}$ over S_{m-1}. This move is a nondegenerate simplex pivot, i.e., along an edge of the path polytope M. The decision on which edge to choose is made by checking all of the edges of M joining the current vertex to a vertex in S_{m-1}.

Lagrangian relaxation. The LAGR algorithm is obtained by dualizing the cardinality constraint $\mathbf{x}(E) \leq k$ as described in Section 3. Thus one solves the Lagrangian dual problem (3.7) by solving a sequence of Lagrangian subproblems $LR(\lambda^{(i)})$ (see (3.5)) until convergence of the multiplier sequence is achieved or an optimal integral solution is found. Each subproblem is a shortest path problem with a modified cost function depending on the chosen λ. As the graph is acyclic the shortest path algorithm simplifies (see below).

Since we here consider 2-graphs only, we know that we have equalities in (3.6), i.e., the optimal value of the Lagrangian dual problem equals the optimal value of the CSP problem.

As for other applications of Lagrangian relaxation it is crucial for convergence (and speed) to update the multiplier λ suitably. We tried several adjustment methods and found that binary search performed best (significantly better than, for instance, the standard subgradient method, see [12]). Binary search has guaranteed and fast convergence and is numerically stable.

The binary search procedure keeps track of an interval $[\lambda_l, \lambda_r]$ containing an optimal multiplier λ^* and halves that interval in each iteration. Throughout the iterations we have that the right-sided derivative of g_k in λ^l is nonnegative and that the left-sided derivative of g_k in λ^r is negative (recall that the Lagrange function g_k is defined in (3.8)). This assures that the interval contains a point with 0 as a subgradient, i.e, an optimal λ^*. Based on the input data an initial interval for λ may be determined with the mentioned derivative properties. In each iteration the Lagrangian subproblem is solved for the midpoint $\lambda_m := (\lambda_l + \lambda_r)/2$ and one obtains an optimal solution $\mathbf{x}$ being the incidence vector of some $v_0 v_n$-path. If $\mathbf{x}(E) = k$ we terminate: an optimal solution is found (see Proposition 3.8). If $\mathbf{x}(E) < k$ (resp. $\mathbf{x}(E) > k$) we update the interval by $\lambda_r := \lambda_m$ (resp. $\lambda_l := \lambda_m$). The algorithm also terminates if $\lambda_r - \lambda_l$ is smaller than a specified small tolerance.

The shortest path subproblems are solved by a simple pulling algorithm as described in [13]. The graph is acyclic so the shortest path to node j is obtained as the shortest path to some preceding node i plus the edge (i, j), and for 2-graphs we have $i \in \{j-1, j-2\}$. The algorithm traverses the nodes in the order $j = 1, 2, \ldots, n$ and determines a shortest path to node j, say P_j, by the equation

$$c(P_j) = \min_{i \in \{j-1, j-2\}} (\mathbf{c}(P_i) + c_{i,j} + \lambda_m) \tag{4.3}$$

For each j we let $i(j)$ be a node for which this minimum is obtained; $i(j)$ is called the parentnode of node j. By moving from node n to node 0 via parentnodes a shortest $v_0 v_n$-path P_n is determined (in reverse order). The (computational) complexity of this shortest path algorithm is $O(2n)$ because it visits each edge once. This means that the algorithm for solving the Lagrangian dual has a complexity bound of $O(n \log_2(\epsilon/C))$ assuming that the algorithm is terminated whenever $\lambda_r - \lambda_l \leq \epsilon$ and where C is the length of the initial interval for λ.

Dynamic programming. A natural approach to the CSP problem is to use dynamic programming. This was done in solving the shortest path subproblems in the LAGR algorithm, but it may also be used on the CSP problem directly.

The basic fact is that a shortest path of cardinality at most m to a node j is obtained as a shortest path of cardinality at most $m-1$ to some preceding node i plus the edge (i, j).

Let P_j^m denote a shortest $v_0 v_j$-path of cardinality at most m. The DYPR algorithm is then: for $m = 1, \ldots, k$ calculate the the minimum of the three numbers $\mathbf{c}(P_j^{m-1})$, $\mathbf{c}(P_{j-1}^{m-1}) + c_{j-1,j}$ and $\mathbf{c}(P_{j-2}^{m-1}) + c_{j-2,j}$ and let $\mathbf{c}(P_j^m)$ be this minimum.

A difference compared to (4.3) is that nodes are processed in the opposite order. Furthermore, we only need to visit the nodes i for which there exists path of given cardinality to a higher ordered node, i.e., i must satisfy $i = 2j, 2j+1, \ldots, j$ if $j \leq \lceil n/2 \rceil$ or $i = n, \ldots, j$ if $j > \lceil n/2 \rceil$. As for the algorithm used to solve $LR_k(\lambda)$ we store the parentnode for each P_j^m we obtain. To construct the final path we start in node n and moves to the parentnode u for the path with cardinality at most k. We then move to the parentnode of node u for the path with cardinality at most $k-1$, and continue in this manner until we reach node 0. The number of parentnodes equals $\sum_{i=0}^{\lceil n/2\rceil}(i+1)+\sum_{i=\lceil n/2\rceil+1}^{k}(n-i+1)$ which gives that the algorithm needs storage space of $O(n^2)$. The complexity of the algorithm is of the order $8kn-7/4n^2-4k^2$. For instance, if $k=3n/4$, we get the complexity $2n^2$.

Simplex methods. The polytope $M(k)$ is integral for 2-graphs, so the CSP problem in 2-graphs may be solved as a linear programming problem. Moreover this LP problem is a minimum cost network flow problem with one additional constraint. This structure may be exploited to give a specialized simplex algorithm using basis partitioning although this was not done here. Instead we solved the problem using the general LP-solver CPLEX (see [5]) based on two different approaches. The first approach is based one the original "network flow" formulation $\min\{\mathbf{c}^T\mathbf{x}:\mathbf{x}\in M(k)\}$ and the resulting algorithm is denoted SIMXo. The second approach is to solve the LP relaxation of the "reduced problem" ILP_k. (Recall that the feasible polyhedron of this problem is integral, see Section 3). This algorithm is denoted by SIMXr.

We also considered variants of SIMXo and SIMXr by finding a candidate initial LP basis. For the SIMXr algorithm we use the initial basis obtained by setting all the variables equal to one.

For SIMXo an initial basis is obtained by a greedy algorithm. For simplicity of the presentation we assume that n is even. First we consider the unique v_0v_n-path P of cardinality $\lceil n/2\rceil$; each arc in P is of the form $(i,i+2)$. Using the quicksort algorithm (see [15]) we determine the $k-\lceil n/2\rceil$ arcs $(i,i+2)$ in P having largest value $c^r_{i,i+2}$ (as defined in connection with the problem ILP_k). Each such arc $(i,i+2)$ is then replaced by the two arcs $(i,i+1)$ and $(i+1,i+2)$. The resulting solution P' is a v_0v_n-path P of cardinality k. From P' we construct a spanning tree by adding suitable (basic) arcs, and together with a slack variable for the constraint $\mathbf{x}(E)\leq k$ this represents the initial LP basis.

5. Computational results. In this section we present some test examples to compare the algorithms described in the previous section. All the graphs considered are 2-graphs, see Section 3. In addition to a general comparison of the algorithms for graphs of different sizes we illustrate how the cardinality k influences the run time for a fixed graph.

The test runs were performed on a SGI-Indy workstation with a 100 MHz processor and 32 Mbyte memory. COMB, LAGR and DYNP were written in C++, and the SIMX algorithms were coded in C (in order to call CPLEX, our linear optimizer, see [5]). We tried to make the implementations efficient and as uniform as possible. The run time was measured by the UNIX function *times*. This function has a 60 Hz sample rate, which gives sufficient precision for our purposes. The run time is reported in minutes and seconds.

The test data were obtained by choosing some real-valued functions to be approximated, i.e., we solved CAPX problems (see Section 2). Each target function f was chosen as the linear interpolant of some real-valued function on an interval $[a,b]$. The number of interpolation points were varied so graphs D of different sizes were obtained. In all cases the interpolation points were equally spaced. The arc weights $c_{i,j}$ were determined as follows. We let $c_{i,i+1}=0$ for each i (as f is linear on the corresponding interval) and $c_{i-1,i+1}=|f(t_i)-\frac{1}{2}(f(t_{i-1})+f(t_{i+1}))|$. This corresponds to using the ℓ_1-norm (for the vectors consisting of the function values in the points $t_0,t_1,\ldots,t_n$). This norm was mainly chosen in order to make the calculations of the vector $\mathbf{c}$ easy.

In Figure 1 and Figure 2 the performance of the different algorithms on a wide range of sizes of instances are given. The columns in Figure 1 contains the following information:

v: The optimal value of the CSP instance.
COMB: The combinatorial algorithm.
    CPU: User CPU-time (minutes.seconds).
DYNP: The dynamic programming algorithm.
LAGR: The Lagrangian relaxation algorithm.
    ITER: The number of Lagrangian subproblems.
    ARCS: The number of arcs in the optimal solution.
    GAP: $(v - v^*)/v \cdot 100\%$.

Recall that v^*, as defined in (3.7), is the optimal value of the Lagrangian dual problem.

The columns in Figure 2 contain the additional information:

GR: Greedy algorithm.
    GAP: $(v -$ greedy value$)/v \cdot 100\%$.
SIMXo: LP formulation of CSP solved by CPLEX.
    PIV: The number of simplex pivots.
SIMXo+B: SIMXo is solved with a "good" initial basis.
SIMXr: LP formulation in the reduced graph solved by CPLEX.
SIMXr+B: SIMXr is solved with a "good" initial basis.

			COMB	DYNP	LAGR			
n	k	v	CPU	CPU	CPU	ITER	ARCS	*GAP*
1000	550	1.652	0.00	0.02	0.00	15	550	0
	750	0.390	0.00	0.03	0.00	19	750	0
	900	0.066	0.00	0.04	0.00	61	898	4.55
3000	1550	1.234	0.04	0.20	0.00	14	1550	0
	2000	0.231	0.04	0.28	0.00	20	2000	0
	2500	0.041	0.02	0.34	0.00	20	2500	0
5000	2600	0.572	0.10	0.58	0.00	15	2600	0
	3000	0.182	0.10	M	0.00	47	3000	0
	3500	0.086	0.08	M	0.00	59	3444	9.30
	4000	0.032	0.06	M	0.00	56	4000	0
10000	5500	0.162	0.46	M	0.01	53	5500	0
	7500	0.0022	0.32	M	0.01	58	7496	9.09
30000	15500	2.691	8.40	M	0.01	19	15500	0
	20000	0.247	6.57	M	0.01	15	20000	0
	25000	0.0022	3.31	M	0.04	57	24999	4.55
50000	26000	0.517	21.18	M	0.06	44	26000	0
	30000	0.234	19.50	M	0.08	56	29748	5.55
	40000	Z	11.17	M	0.08	56	38079	

FIG. 5.1. *The algorithms COMB, DYNP and LAGR.*

The entries marked 'M' in the DYNP column in Figure 1 indicates insufficient memory for the given instance. The entry containing 'Z' indicates that the objective function is close to zero ($\mathbf{c}(P) = 1.4 \cdot 10^{-11}$). The entries marked 'T' in Figure 2 indicates that the program was aborted since it exceeded the chosen run-time limit of 45 minutes. Note that the time needed to calculate the initial bases is included in the run times reported, but it is negligible for all the instances.

In the second set of test runs we varied the parameter k occuring in the cardinality

		GR	SIMXo		SIMXo+B		SIMXr		SIMXr+B	
n	k	$\%GAP$	CPU	PIV	CPU	PIV	CPU	PIV	CPU	PIV
1000	550	0.6	0.07	1334	0.04	1546	0.02	1526	0.01	238
	750	0.8	0.05	1054	0.07	1571	0.04	1555	0.01	576
	900	3.0	0.02	488	0.03	635	0.01	671	0.02	838
3000	1550	0.3	0.55	3844	0.38	4584	0.19	4562	0.02	119
	2000	24.2	0.55	3660	0.54	2429	0.30	4882	0.09	1223
	2500	87.8	0.23	1666	0.27	2508	0.15	2462	0.15	2487
5000	2600	0.7	2.33	6483	1.50	7687	0.51	7681	0.08	285
	3000	9.9	2.52	6839	2.13	8367	1.08	7824	0.21	1665
	3500	23.3	2.11	5267	2.29	7791	1.17	7514	0.30	2812
	4000	46.9	1.44	4100	1.46	5494	0.58	5430	0.36	3679
10000	5500	17.3	16.59	12233	8.37	4966	3.52	15832	1.09	2621
	7500	1536.4	13.30	10079	7.56	12767	4.19	12719	2.11	6970
30000	15500	T	T	T	T	T	30.13	46288	4.04	1687
	20000	T	T	T	T	T	37.39	43131	17.33	18632
	25000	T	T	T	T	T	23.34	24857	21.25	24933

Fig. 5.2. *The simplex algorithms.*

constraint $\mathbf{x}(E) \leq k$. This was done by running the algorithms on an instance with 3000 nodes and k in the range $1600, 1700, \ldots, 2900$. The results are shown graphically in Figure 3 where we have plotted the CPU-time for the different algorithms as a function of k.

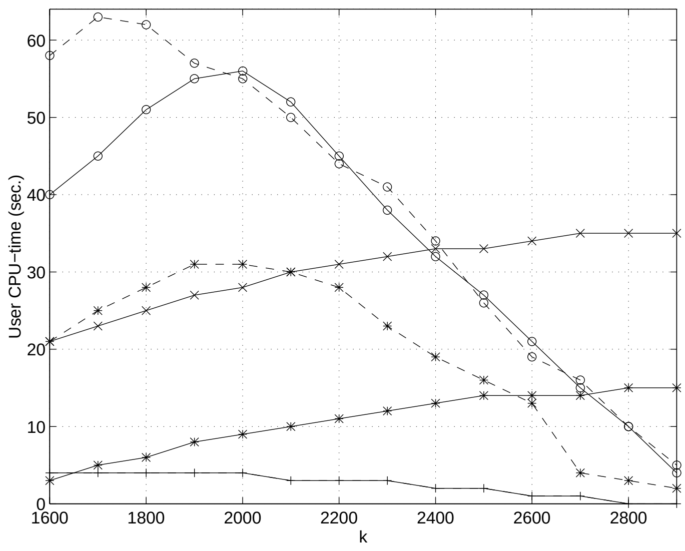

Fig. 5.3. *Run times for a 3000 node instance as a function of k.*

The curve labeled with '+' is the COMB algorithm (it is the bottom curve), the curve labeled with 'x' is the DYNP algorithm, the curves labeled with '$*$' is the SIMXr algorithm and the curves labeled with 'o' is the SIMXo algorithm. For the SIMX algorithms the solid curves indicates the run time when an initial basis is given and the dashed ones are without an initial basis.

We may summarize our computational experiences with these algorithms as follows.

COMB. The combinatorial algorithm has a very good performance, and is able to solve problems of all the reported sizes within a reasonable time. It is slower than the SIMXr algorithm with an initial basis when k is very small. Otherwise it is much faster than the other algorithms that are guaranteed to find an optimal solution.

DYNP. The major weakness of this algorithm is its memory consumption, and as can be seen from Figure 1 the algorithm can only solve small scale problems. However, an advantage of DYNP is that it can be used to solve CSP problems in arbitrary graphs.

LAGR. The LAGR algorithm is the fastest among the implemented algorithms, but as Figure 1 shows it does not always produce an optimal solution. Whether LAGR solves CSP or not depends upon the number of similar arc weights in the given graph (see Section 3 for a discussion of this topic).

SIMXo. CPLEX has poor performance on the SIMXo formulation of the LP-problem, but the run time is reduced rapidly as we allow more points in the solution. Notice that the initial basis only has an effect when k is low, so in that case the greedy solution seems to be pretty close to optimal.

SIMXr. CPLEX is much faster on this LP problem than the preceding one, and SIMXr is able to solve larger problems than SIMXo within our time limit. The run time was considerably reduced by using the greedy solution as an initial basis, even whenever this initial basis was infeasible.

Finally we illustrate an application of our study to a problem from medicine (see the figures at the end of the paper). The data in Figure 4 shows a cross section of a human heart, and were obtained from [16]. The target curve was found by linear interpolation of the data (i.e., a straight line between consecutive data points). In Figure 5 one can see the target curve and an optimal subinterpolant for $k = 183$ (i.e., with 183 straight segments). At this scale the two curves are almost impossible to distinguish. A zoomed picure showing the lower left part of the two cuves may be seen in Figure 6. It should be remarked that the computational time for finding this approximation was less than a hundredth of a second.

6. Conclusions. We have considered a cardinality constrained shortest path problem in acyclic directed graphs. The problem has important applications in curve approximation problems that arise in e.g. computer aided geometric design. The polytope $M(k) = \{\mathbf{x} \in \mathbb{R}^E : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1},\ \mathbf{x}(E) \leq k\}$ was studied. A characterization of all vertices of $M(k)$ was presented, and extended to adjacency and integrality results for certain graphs called 2-graphs. A combinatorial algorithm COMB was developed based on the adjacency descriptions.

The numerical results indicate that several algorithms seem feasible in many applications, but that COMB and a Lagrangian algorithm LAGR are preferable for large scale problems. The COMB algorithm is guaranteed to find an optimal solution while the the Lagrangian algorithm may fail to do this. For curve approximation problems the algorithms may be used to find best approximations with a desired data reduction within reasonable time. It would be interesting to see some of these algorithms, properly extended, in (for instance) some real world geometric design tools. Further work in this area could also be to consider the approximation problem for functions of two variables or even splines. These problems are much harder, but at least some reasonable heuristics for the two-variable problem may be developed based on the algorithms discussed here.

REFERENCES

[1] E. Arge and M. Dæhlen, *Data reduction of piecewise linear curves*, Tech. Report STF33 A94042, SINTEF, Norway, 1994.

[2] V. Bhaskaran, B. K. Natarajan, and K. Konstantinides, *Optimal piecewise-linear compression of images*, in Data Compression, J. Storer and M. Cohn, eds., IEEE Computer Society Press, 1993, ch. 31, pp. 168–177.

[3] A. Brøndsted, *An introduction to convex polytopes*, Springer, New York, 1983.

[4] R. E. Burkard, H. W. Hamacher, and G. Rote, *Sandwich approximation of univariate convex functions with an application to separable convex programming*, Naval Research Logistics, 38 (1991), pp. 911–924.

[5] CPLEX, *Using the cplex callable library*, tech. report, CPLEX Optimization, Inc., 1994.

[6] M. Garey and D. Johnson, *Computers and intractability. A guide to the theory of NP-completeness*, W.H. Freeman and company, 1979.

[7] L. Gouveia, *Using variable redefinition for computing minimum spanning and steiner trees with hop constraints*, Tech. Report 2, Faculdade de Ciéncias da Universidade de Lisboa, Centro de investigação operacional, Lisboa, Portugal, 1996.

[8] D. Hausmann, *Adjacency on polytopes in combinatorial optimization*, Verlag Anton Hain, 1980.

[9] E. Houssaine and L. Wolsey, *Lot-sizing polyhedra with a cardinality constraint*, Operations Research Letters, 11 (1992), pp. 13 – 18.

[10] H. Imai and M. Iri, *Computational-geometric methods for polygonal approximations of a curve*, Computer Vision, Graphics, and Image Processing, 36 (1986), pp. 31–41.

[11] T. Lyche and K. Mørken, *A data-reduction strategy for splines with applications to the approximation of functions and data*, IMA Journal of Numerical Analysis, 8 (1988), pp. 185–208.

[12] G. Nemhauser and L. Wolsey, *Integer and combinatorial optimization*, Wiley, 1988.

[13] J. B. O. R. K. Ahuja, T. L. Magnanti, *Network flows: theory, algorithms, and applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[14] A. Schrijver, *Theory of linear and integer programming*, Wiley, Chichester, 1986.

[15] N. Wirth, *Algorithms + datastructures = programs*, Prentice Hall, 1976.

[16] D. K. Zyck, *Hjertets elektriske aktivitet modellert ved hjelp av elementmetoden*, master's thesis, University of Oslo, 1994.

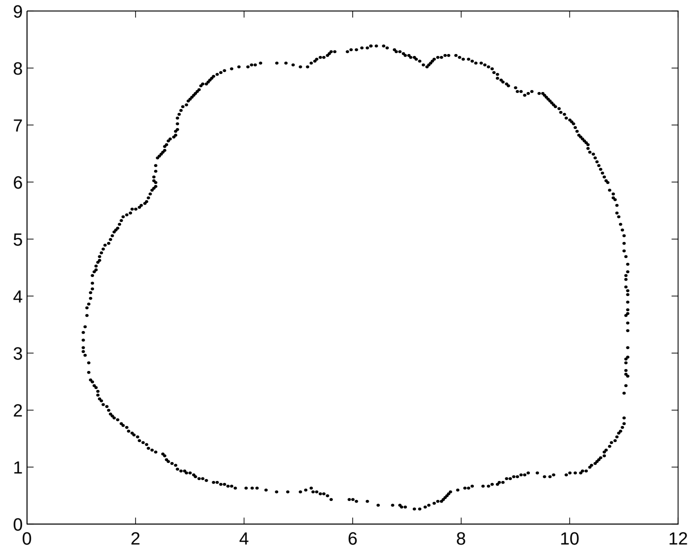

Fig. 6.1. *The 363 data points.*

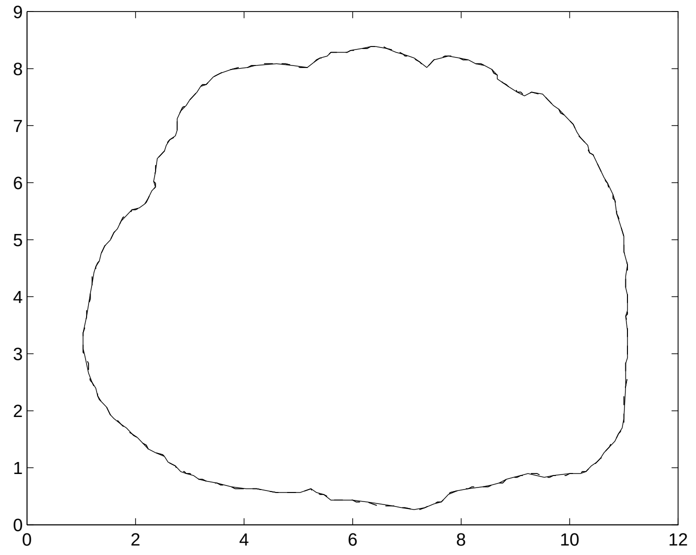

Fig. 6.2. *The original curve and an optimal solution with* $k = 183$.

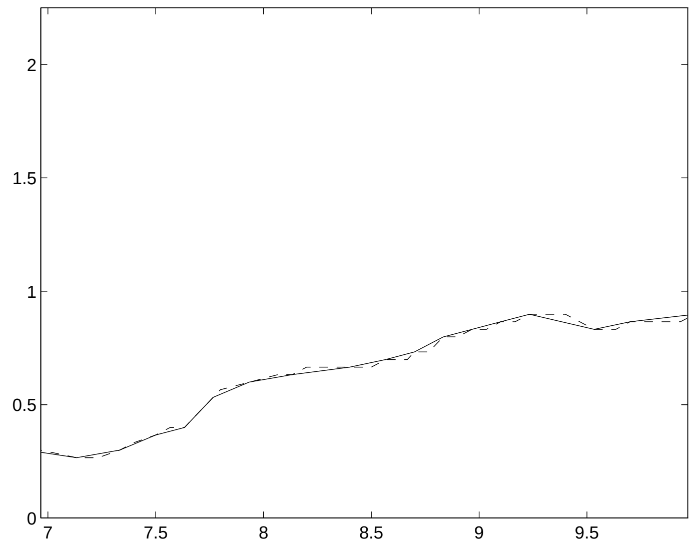

Fig. 6.3. *Zoom of lower left corner.*