Evolutionary Search for Faces from Line Drawings

Jianzhuang Liu, *Senior Member*, *IEEE*, and Xiaoou Tang, *Senior Member*, *IEEE*

Abstract—Single 2D line drawing is a straightforward method to illustrate 3D objects. The faces of an object depicted by a line drawing give very useful information for the reconstruction of its 3D geometry. Two recently proposed methods for face identification from line drawings are based on two steps: finding a set of circuits that may be faces and searching for real faces from the set according to some criteria. The two steps, however, involve two combinatorial problems. The number of the circuits generated in the first step grows exponentially with the number of edges of a line drawing. These circuits are then used as the input to the second combinatorial search step. When dealing with objects having more faces, the combinatorial explosion prevents these methods from finding solutions within feasible time. This paper proposes a new method to tackle the face identification problem by a variable-length genetic algorithm with a novel heuristic and geometric constraints incorporated for local search. The hybrid GA solves the two combinatorial problems simultaneously. Experimental results show that our algorithm can find the faces of a line drawing having more than 30 faces much more efficiently. In addition, simulated annealing for solving the face identification problem is also implemented for comparison.

Index Terms—Three-dimensional object reconstruction, face identification, genetic algorithms, line drawing, minimal edge face phenomenon, simulated annealing.

1 Introduction

In computer vision, an important research area is to develop algorithms that can understand a single 2D line drawing representing an object and reconstruct the 3D geometry of the object. This area belongs to the topics *shape from line drawings* and *shape from contours*. One application of this research is in CAD, where tools are highly desirable that can convert a design sketch into a 3D model directly. An object consists of faces. If the face configuration of an object is known, the complexity of the 3D reconstruction will be reduced significantly. Roughly speaking, the conversion problem can be divided into two subproblems: face identification and 3D geometry reconstruction. In this paper, we study the face identification problem only. For the 3D geometry reconstruction, the reader is referred to references [1], [2], [3], [4], [5], [6], [7].

A 2D line drawing in this paper is defined as the projection of a wireframe object where all the edges (including silhouettes) and vertices of the object are visible and the drawing can be represented by a single connected edge-vertex graph. A line drawing with hidden lines visible makes it possible to reconstruct its complete 3D model. Fig. 1 shows a line drawing of a solid stair model together with its 10 individual faces. Note that crossing points of two or more edges are not vertices and cannot be used to form faces. If the three edges e_1, e_2, and e_3 (hidden lines) are not shown, the stair model is considered as a sheet object with seven faces.

• *The authors are with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. E-mail: {jzliu, xtang}@ie.cuhk.edu.hk.*

Manuscript received 23 July 2004; accepted 30 Nov. 2004; published online 14 Apr. 2005.
Recommended for acceptance by S. Baker.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0377-0704.

Great effort has been made in the interpretation of line drawings for the past two decades (see Section 3). The recent work presented in [8] and [9] can handle a larger range of objects than others in face identification. Both methods include two steps: finding a set of circuits that may be potential faces and searching for real faces from this set. Given a set of circuits, Shpitalni and Lipson used a tree search scheme for finding the real faces [8], while Liu and Lee developed a maximum weight clique finding algorithm for the same goal [9]. It needs to be emphasized that the two steps in each of the two methods correspond to two combinatorial problems. It is well-known that there have been no efficient algorithms for general tree search and maximum weight clique problems. The problems become even more difficult owing to the fact that their inputs are circuits of a line drawing, the number of which is generally exponential in the number of edges of the drawing [10]. In Section 5, it can be seen that the combinatorial explosion prevents the two methods from handling a line drawing having many faces (> 35) within feasible time.

Genetic algorithms (GAs) are a class of probabilistic search algorithms that emulate natural evolutionary process. GAs, if well designed, can often outperform traditional optimization methods. Many successful applications of GAs to various problems have been published in the literature [11], [12], [13], [14]. In this paper, we design an efficient GA with variable-length chromosomes to solve the face identification problem. Geometric constraints on circuits of a line drawing and a heuristic called minimal edge face phenomenon are developed and incorporated into the operations of the GA. The proposed hybrid GA simultaneously tackles the two combinatorial problems involved in the previous methods [8], [9]. Our experiments show that the new algorithm reduces the computational complexity of face identification from exponential to linear with respect to the number of edges in line drawings.

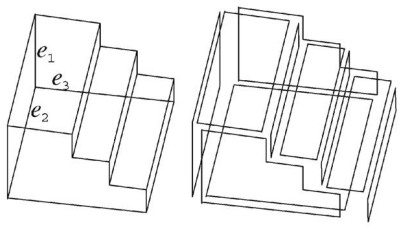

Fig. 1. A line drawing with 10 faces.

The rest of this paper is organized as follows: In Section 2, we briefly describe the technical terms that will be used in this paper. The related work on line drawing interpretation is described in Section 3. The proposed hybrid GA for the face identification is presented in Section 4. Section 5 shows experimental results and comparisons of computational times and memory requirements between the new algorithm and the previous ones. In Section 6, we implement a simulated annealing algorithm and compare it with the GA. Finally, the conclusions are given in Section 7.

2 Terminology

For easier understanding of the technical content of this paper, we first summarize several terms that will be used in the rest of this paper. More detailed descriptions for the graph theory and topology terms can be found in [15], [16].

- **Graph.** A graph is defined to be a set of points (vertices) that are interconnected by a set of lines (edges). A graph G may be written as $G = (V, E)$ with V and E being its vertex set and edge set, respectively. The numbers of vertices and edges are denoted by $|V|$ and $|E|$, respectively.
- ***k*-connected**. A graph is called k-connected if at least k of its vertices and the edges adjacent to them must be removed to make the remaining graph disconnected.
- **Manifold.** A manifold, or more rigorously 2-manifold, is a solid where every point on its surface has a neighborhood topologically equivalent to an open disk in the 2D Euclidean space.
- **Genus.** The genus of a surface can be considered as the number of holes that pass through it completely. The genus of a graph G is the smallest genus of a surface on which G can be embedded.
- **Circuit.** A circuit is a closed trail in a graph where all its vertices except the end vertices are distinct.
- **Potential face.** A potential face is a circuit without edges intersecting.
- **Minimal potential face (MPF).** An MPF is a potential face without a smooth (noncreased) edge (in the drawing) connecting two of its nonadjacent vertices.
- **Degree.** The degree of a vertex v, written $d(v)$, is the number of edges adjacent to v.
- **Vertex rank.** The rank of a vertex v, written $R(v)$, denotes the number of circuits with boundaries passing through v. The boundary of a circuit is the set of vertices and edges of the circuit.
- **Edge rank.** The rank of an edge e, written $R(e)$, denotes the number of circuits with boundaries passing through e.

3 Related Work

We may divide the related work on interpretation of line drawings into three areas: line labeling, 3D reconstruction from multiple views of wireframe models, and face identification and 3D reconstruction from single line drawings with hidden lines visible. Methods for line labeling focus on finding a set of consistent labels from a line drawing without hidden lines in order to test if it is legal and/or on 3D reconstruction based on such a labeled line drawing [17], [18], [19], [20], [21], [22], [23]. Approaches in the second area try to reconstruct a 3D CAD model from its multiple (three, in general) orthographic projections [24], [25], [26]. More information can be found from three orthographic views for the reconstruction task than from a single projected view, which is the premise of the third area. Our work belongs to the third area. We especially focus on face identification from single line drawings with hidden lines visible.

The earliest work in face identification was done by Markowsky and Wesley [27]. Their topologically-driven algorithm can handle only wireframes with straight lines and planar faces and requires the 3D coordinates of vertices to calculate the normals of planes.

Hanrahan [28] and Dutton and Brigham [29] used purely topological methods to find the faces of a drawing. A drawing (graph) is embedded in the plane with a planar embedding algorithm [15]. The resulting regions represent the faces of the corresponding object. Their methods are suitable only for objects of genus 0 whose drawings are 3-connected, due to the requirement of a unique planar embedding for a drawing. Another approach presented by Ganter and Uicker [30] also uses concepts from graph theory. It is based on the spanning tree of the graph of a drawing. The problem with this approach is that it cannot handle objects with holes. Hojnicki and White [31] improved Ganter and Uicker's algorithm by employing better cycle reduction schemes. However, if the number of faces of an object of genus > 0 is unknown in advance, their algorithms still fail when dealing with it.

Agarwal and Waggenspack's method [32] employs a divide-and-conquer strategy to remove stars (tetrahedra, N-sided pyramids, or multiply connected stars) from a drawing. The faces of the drawing are obtained by combining triangles that are created from the stars. Based on a number of properties implied in line drawings representing manifold objects, Liu et al. [33] used a tree search scheme to find the faces of manifolds.

The abovementioned methods for face identification were designed basically for dealing with solid objects, compared with which the following two methods can handle a larger range of objects (e.g., the manifold shown in Fig. 1 and the nonmanifold in Fig. 2a). Because these two methods are most related to the work in this paper, we give more detailed descriptions of them in the following sections.

3.1 Shpitalni and Lipson's Method

Shpitalni and Lipson's face identification method [8] is built upon an observation on face configuration and a basic theorem called the *face adjacency theorem*. The observation, serving as the criterion for the problem, is that, given a line drawing, human beings tend to choose a face configuration

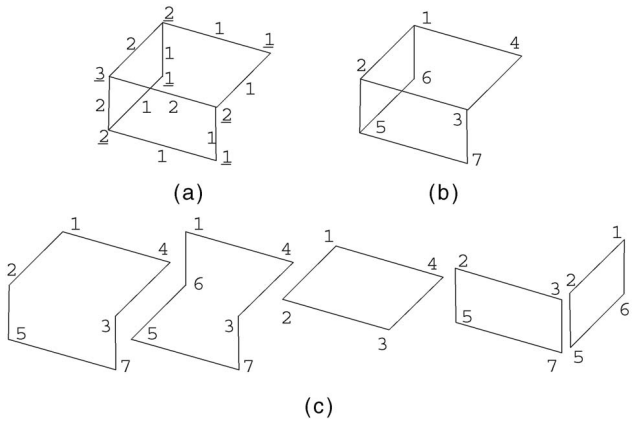

Fig. 2. (a) A 3-face drawing with its maximum ranks shown where $R^+(v)$s are denoted by underlined numbers and $R^+(e)$s by the other numbers. (b) The same drawing with vertices labeled. (c) The potential faces of the drawing with the last four being MPFs.

in which there are as many edges as possible. The face adjacency theorem states that two adjacent planar faces may coexist in the same object if and only if their common edges are collinear. This theorem can be extended to allow nonplanar, smooth faces. The extended theorem states that two adjacent smooth faces may coexist in the same object if and only if their common edges are smooth. The method can be formulated as the following optimization problem:

Definition 1. *Given n MPFs (minimal potential faces) generated from a line drawing, the upper bounds (maximum ranks) $R^+(e)$ and $R^+(v)$ of all the edges and vertices of the drawing, and a binary matrix $B = [b_{ij}]_{n\times n}$ (obtained according to the face adjacency theorem) with $b_{ij} = 1(0)$ denoting that faces i and j can (cannot) coexist in the same object, the face identification problem is to search for subsets xs of the MPFs such that*[1]

$$minimize: g(x) = \sum [R^+(e) - R(e)] + \sum [R^+(v) - R(v)] \quad (1)$$

$$subject\ to: R(e) \le R^+(e), \forall e \quad (2)$$

$$R(v) \le R^+(v), \forall v \quad (3)$$

$$b_{ij} = 1, i \ne j, i, j \in x, \quad (4)$$

where $R(e)$ and $R(v)$ are the respective actual edge and vertex ranks corresponding to x.

In this formulation, (1) implies the criterion and (4) reflects the geometric constraint imposed by the face adjacency theorem. Shpitalni and Lipson calculated the maximum ranks $R^+(e)$ and $R^+(v)$ from a line drawing through an iterative procedure.

The ranks play an important role in Shpitalni and Lipson's method. Since they are also used in the implementation of our GA, we discuss them in more detail. The maximum edge and vertex ranks impose two of the three constraints. Based on the face adjacency theorem, when no two edges meeting at a vertex are collinear, Shpitalni and Lipson gave the following three inequalities and an equation for finding the maximum edge and vertex ranks of a line drawing:

1. In [8], Shpitalni and Lipson presented the objective function in this form $g(x) = \sum |R^+(e) - R(e)| + \sum |R^+(v) - R(v)|$. The two absolute signs are removed in (1) because of the two constraints in (2) and (3).

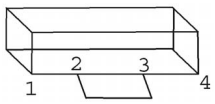

Fig. 3. A smooth entity chain $(1, 2, 3, 4)$ where three faces pass through edge $(2, 3)$.

$$R^+(v) \le \{d(v)[d(v) - 1]\}/2 \quad (5)$$

$$R^+(e) \le \min\{d(v_1), d(v_2)\} - 1 \quad (6)$$

$$R^+(v) = \left\lfloor [\sum R^+(e)]/2 \right\rfloor, \text{for all edges meeting at vertex } v \quad (7)$$

$$R^+(e) \le \min\{R^+(v_1), R^+(v_2)\}, \quad (8)$$

where v_1 and v_2 are two end-vertices of edge e in (6) and (8); $\lfloor a \rfloor$ denotes the largest integer $\le a$; all the ranks are integers. These relations are derived from the local analysis of a general edge and its two end-vertices. Because a face boundary passing through vertex v must pass through two edges meeting at v, the largest number of faces passing through v cannot exceed $C^2_{d(v)}$, the possible combinations of edge pairs at v, which leads to (5). Similarly, since a face passing through edge e also passes through one of the other edges meeting at its end-vertices v_1 or v_2, we have $R^+(e) \le d(v_1) - 1$ and $R^+(e) \le d(v_2) - 1$; thus (6) follows. After the ranks of all the edges meeting at vertex v have been obtained, (7) follows from the fact that a face passing through v also passes through two of these edges. Because a face passing through an edge e also passes through its two end-vertices v_1 and v_2, we have $R^+(e) \le R^+(v_1)$ and $R^+(e) \le R^+(v_2)$, which lead to (8). Shpitalni and Lipson used (5) and (6) to compute the preliminary estimation of the maximum edge and vertex ranks and then applied (7) and (8) iteratively until all the ranks satisfy (5)-(8).

For better understanding of the ranks, potential faces and MPFs, Fig. 2 shows a line drawing having three faces. In this simple example, there are only five potential faces, among which the last four are MPFs, as shown in Fig. 2c. Circuit $(1, 2, 5, 7, 3, 4, 1)$ is not an MPF because there exists an edge connecting vertices 2 and 3 in the drawing. The last three circuits are real faces.

When two edges meeting at a vertex are collinear in a line drawing, two or more faces may share the two edges in a smooth entity chain. For example, for the smooth entity chain $(1, 2, 3, 4)$ in Fig. 3, there are three faces passing through edge $(2, 3)$. To accommodate this case, Shpitalni and Lipson extended (6) to

$$R^+(e) \le \min\left\{\sum d(v_L) - 2n_L, \sum d(v_R) - 2n_R\right\} + 1, \quad (9)$$

where n_L (n_R) is the number of vertices on the left (right) of edge e, and v_L (v_R) denote all the n_L (n_R) vertices along the smooth entity chain on the left (right) of edge e. Then, (9) is used to replace (6) to find the maximum ranks in their iterative procedure.

3.2 Liu and Lee's Method

In [9], Liu and Lee used the same criterion and face adjacency theorem as those in [8] to formulate the problem. They indicated that the maximum ranks of a line drawing can be calculated by

$$R^+(e) = \min\{d(v_1), d(v_2)\} - 1 \quad (10)$$

and (7). When there are smooth entity chains in a line drawing, (10) is replaced by

$$R^+(e) = \min\Big\{\sum d(v_L) - 2n_L, \sum d(v_R) - 2n_R\Big\} + 1. \quad (11)$$

We will use (10) and (11) to compute the maximum edge ranks of a line drawing in the GA implementation. The face identification in [9] is formulated as follows:

Definition 2. *Let $w(i)$ be the number of edges of a face i. Given an edge-vertex graph of a line drawing and the set, denoted by $SMPF$, of the MPFs generated from the graph, the objective of face identification is to find all the sets $x_1, x_2, \ldots, x_s$, where s is the number of sets and $x_k \subset SMPF$, $1 \leq k \leq s$, such that, for every x_k, the sum of $w(i)$, $i \in x_k$, is maximal and the faces in x_k satisfy the face adjacency theorem. In short, the problem is:*

$$maximize : f(x) = \sum_{i \in x} w(i), x \subset SMPF \quad (12)$$

$$subject\ to : b_{ij} = 1, i \neq j, i, j \in x. \quad (13)$$

The two formulations in Definitions 1 and 2 are proven to be equivalent in [9]. Based on Definition 2, Liu and Lee developed a much faster algorithm to find the faces in a drawing.

3.3 Combinatorial Explosion Problem

Each of the two methods in [8] and [9] involves two combinatorial problems. Shpitalni and Lipson generated the MPFs by a circuit space approach and used the A* algorithm to search for the optimal solutions (see Definition 1) on a tree constructed by the MPFs. Liu and Lee employed a depth-first search algorithm to find the MPFs and developed a maximum weight clique finding algorithm to solve the problem in Definition 2.

In general, the number of circuits in a drawing is exponential in the number of edges [10]. For example, the simple 3-step stairs shown in Fig. 1 presents 312 circuits. If the stairs are expanded by only one more step, the number of circuits will increase to 1,114! Although usually only a small fraction of the circuits are MPFs, the number of MPFs still grows exponentially with the number of edges of a line drawing. What is worse is that these exponentially-increasing MPFs are the input of the second combinatorial search algorithms for both methods. There are no known efficient algorithms for general tree search and maximum weight clique problems. Our experiments show that neither method works for line drawings with more than 35 faces within feasible time.

4 A Hybrid GA for Face Identification

In this section, we develop a GA with variable-length chromosomes and a local search heuristic for the face identification. Incorporated into the local search algorithm are the maximum ranks, geometric constraints imposed by the face adjacency theorem, and a novel heuristic called minimal edge face phenomenon. The hybrid GA is designed to find solutions by direct search on a line drawing.

4.1 Outline of Standard GAs

Standard GAs generally have these five basic components [13]:

1. a genetic representation of solutions to a problem,
2. a scheme to create an initial population of solutions,
3. a fitness function to evaluate how good a solution is,
4. genetic operators used to generate offspring during reproduction, and
5. parameters of GAs.

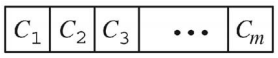

Fig. 4. A chromosome having m genes (MPFs).

A GA searches some solution space by maintaining a population $P(t)$ of *chromosomes* in each generation t. A solution is encoded into a chromosome. The main advantage of maintaining a population is that, with many different solutions available, the search has a greater chance of finding global optima. The overall quality (fitness) of the solutions is improved generation after generation. In each generation, some individuals are selected to undergo stochastic transformations through genetic operators to create new individuals called offspring, $O(t)$. Based on *survival of the fittest*, the better an individual fits its environment, the greater is the likelihood of it being selected. Two commonly used operators are *crossover* and *mutation* [14]. A new generation is formed by the selection of fitter individuals from $P(t)$ and $O(t)$. It is expected that the GA converges to the best solutions after a number of generations. The standard GA is summarized in Algorithm 1.

Algorithm 1. (Standard GA [13])

1. $t \leftarrow 1$
2. Initialize $P(t)$ randomly
3. Evaluate $P(t)$
4. **repeat**
5. Select individuals from $P(t)$
6. Perform mutation and crossover on the selected individuals to generate $O(t)$
7. Form $P(t+1)$ from $P(t)$ and $O(t)$
8. $t \leftarrow t+1$
9. **until** the termination condition is satisfied

The framework of the standard GA is rather general. A simple GA, however, does not usually give satisfactory results for a difficult optimization problem. In the following, we will design a GA to meet the nature of the face identification problem and to solve it efficiently.

4.2 Genetic Representation and Fitness Function

The first step to designing a GA is to encode a solution into a chromosome. A chromosome consists of a number of genes. For the face identification, we use a gene to represent an MPF. The gene stores the number of vertices and the ordered vertex labels of the MPF. Unlike fixed-length chromosomes in most GAs, here the length of a chromosome is variable since the number of the faces of an object is unknown in advance. Fig. 4 shows a chromosome having m genes (MPFs) at some generation.

Given a line drawing with all the edges visible, human beings tend to choose a face configuration in which there are as many edges as possible. This is the criterion for our solution to the face identification problem (which is also the criterion in Shpitalni and Lipson's and Liu and Lee's methods [8], [9]). The MPFs kept in a chromosome give a possible solution to the problem. Therefore, we define a fitness function to evaluate how fit a chromosome is as follows:

Definition 3. *The fitness of a chromosome k is evaluated by*

$$f_k(x) = \sum_{i \in x} w(i), \tag{14}$$

where x is the set of compatible MPFs currently stored in the chromosome, and $w(i)$ denotes the number of edges of a MPF i.

Here, maximizing $f_k(x)$ implies the criterion and the term *compatible* imposes a constraint on the MPFs. Obviously, the more MPFs are added into a chromosome, the higher its fitness is. However, whether or not a new MPF can be added into the chromosome is determined by the geometric constraint, the face adjacency theorem. In other words, all the MPFs in a chromosome must be able to coexist in the same object. We call these MPFs compatible.

The goal of the GA is to evolve chromosomes to be fittest (with the highest fitness value). Comparing Definitions 1, 2, and 3, it is not difficult to see that both the objectives and constraints of them are the same. In Definition 3, the term compatible implies the same constraint as that in (13). These equivalent but different formulations lead to different ways to solve the face identification problem with very different efficiency.

We need MPFs to form chromosomes. From the discussion in Section 3.3, it has been known that the number of MPFs increases exponentially with the number of edges in a line drawing. Therefore, we do not want to generate all the MPFs in a drawing before the GA runs. Instead, we design the GA to have a local search operation to find the MPFs that are most likely to be faces directly from the drawing.

4.3 Minimal Edge Face Phenomenon

A face of an object corresponds to a circuit constructed by some edges in a line drawing. For an edge of a line drawing with all the vertices of degree > 1, we can always find a circuit passing through the edge and having fewest edges. Such a circuit is called a *minimal edge circuit*. Through extensive observation, we have found that it is very likely for a minimal edge circuit to stand for a face. Among all the 1,772 edges of the line drawings given in the experiments in this paper and in the two papers by Shpitalni and Lipson [8] and Liu and Lee [9], there are only 75 edges, passing through which the minimal edge circuits are not faces. For these examples, the percentage of a minimal edge circuit being a face is as high as 95.7 percent. We call this the *minimal edge face phenomenon*.

Consider the edge $(2, 3)$ of the line drawing shown in Fig. 2b. There are two minimal edge circuits, $(2, 1, 4, 3, 2)$ and $(2, 5, 7, 3, 2)$, passing through it. These two circuits represent two faces. In fact, this phenomenon applies to all the edges of the line drawings in Figs. 1, 2, and 3. However, not all such circuits are faces of an object. An example can be seen in the drawing shown in Fig. 5 depicting a block with a hole passing through it. From edge $(3, 4)$, we can find a minimal edge circuit $(3, 2, 1, 6, 5, 4, 3)$ in the drawing, but it is not a face. Is it possible to find all the faces in any drawing simply by searching among minimal edge circuits? The answer is no, since not all faces are minimal edge circuits. For example, the two faces each with eight edges in Fig. 1 cannot be found by this approach, neither can the face $(1, 7, 8, 9, 10, 4, 5, 6, 1)$ in Fig. 5.

Although there are exceptions, the minimal edge face phenomenon can apply to most cases. This heuristic leads to a very effective local search scheme. Together with the GA, it efficiently solves the two combinatorial explosion problems involved in the previous methods. This local search scheme is described in the next section.

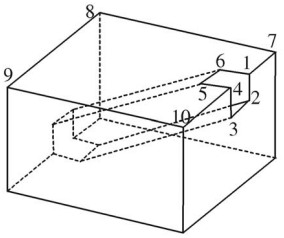

Fig. 5. A block with a hole, where hidden edges are shown in dashed lines for easier observation. The circuit $(3, 2, 1, 6, 5, 4, 3)$ is not a face.

4.4 A Local Search Scheme

Enumerating all the circuits in a drawing ensures that all the faces can be explored. However, doing this will lead to the combinatorial explosion which is what we try to avoid. In order for the GA to form its chromosomes with higher and higher fitness values, we propose a local search scheme to assist the GA's exploration on a line drawing.

Given a set of MPFs stored in a chromosome, the local search scheme tries to extend the chromosome by adding into it as many MPFs as possible. However, a new MPF, if added into the chromosome, has to be compatible with all the existing MPFs in it. Thus, the following two conditions will be used in the local search algorithm.

Condition 1. The new MPF does not cause the rank of any edge e in the line drawing to exceed the maximum edge rank $R^+(e)$.

Condition 2. It can coexist in the line drawing with all the MPFs stored currently in the chromosome.

Both of the conditions are geometric constraints for the new MPF to be compatible with all the other MPFs. The first condition requires that the edge ranks obtained from the group of MPFs should not exceed their corresponding maximum edge ranks. The second condition comes from the face adjacency theorem.

It should be emphasized that Condition 2 implies Condition 1, which was proven by Liu and Lee in [9]. The use of both is for the purpose of making the search algorithm more efficient. Given a line drawing, the maximum edge ranks $R^+(e)$ can be calculated conveniently by (10) and (11) and be used directly during the local search for MPFs, while Condition 2 is tested after an MPF has been found.

In Condition 1, we consider only the maximum edge ranks $R^+(e)$, ignoring the maximum vertex ranks $R^+(v)$. In Definition 1, both edge and vertex ranks are used. At first glance, it seems that if both are used, the search algorithm would be more efficient with one more constraint. However, the following theorem shows that if all the $R^+(e)$ are not exceeded by the edge ranks obtained from a set of circuits, all the $R^+(v)$ will not be exceeded either by the vertex ranks from the same set. Thus, testing whether a maximum vertex rank is exceeded is redundant. It only reduces the efficiency of the algorithm if used.

Theorem 1. *Given a set x of circuits in a line drawing. If the circuits in x satisfy $R(e) \leq R^+(e), \forall e$, then $R(v) \leq R^+(v), \forall v$.*

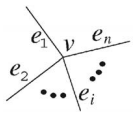

Fig. 6. A vertex v met by n edges.

Proof. As shown in Fig. 6, let all the edges meeting at a vertex v be $e_1, e_2, \ldots, e_n$. As mentioned in Section 3.2, the maximum vertex rank can be obtained by

$$R^+(v) = \left\lfloor \frac{1}{2} \sum_{i=1}^{n} R^+(e_i) \right\rfloor. \quad (15)$$

Recall that $\lfloor a \rfloor$ denotes the largest integer $\leq a$. A circuit in x passing through v also passes through two of these edges. Thus, we have $2R(v) = \sum_{i=1}^{n} R(e_i)$, which leads to

$$2R(v) \leq \sum_{i=1}^{n} R^+(e_i). \quad (16)$$

If $\sum_{i=1}^{n} R^+(e_i)$ is even, then, from (15), $2R^+(v) = \sum_{i=1}^{n} R^+(e_i)$. Hence, $R(v) \leq R^+(v)$. If $\sum_{i=1}^{n} R^+(e_i)$ is odd, again from (15), $2R^+(v) = \sum_{i=1}^{n} R^+(e_i) - 1$. Since $2R(v)$ is even, we have, from (16), that $2R(v) \leq \sum_{i=1}^{n} R^+(e_i) - 1$. Thus, $R(v) \leq R^+(v)$. □

Let us see an example of how edge ranks are used. Consider the drawing shown in Fig. 2b and the MPFs in Fig. 2c (the last four circuits). Suppose a chromosome now contains two MPFs $C_1 = (1, 2, 3, 4, 5)$ and $C_2 = (2, 5, 7, 3, 2)$. If a new MPF $C_3 = (1, 2, 5, 6, 1)$ is found, we can verify that it satisfies the two conditions. For example, for edge $(2, 5) = e_1$, its rank $R(e_1) = 2$ satisfies $R(e_1) \leq R^+(e_1) = 2$. However, if the new MPF is $C_4 = (1, 6, 5, 7, 3, 4, 1)$, we will find that many edge ranks obtained by C_1, C_2, and C_4 exceed their upper bounds. For example, for edge $(1, 4) = e_2$, we have $R(e_2) = 2 > R^+(e_2) = 1$. In this case, C_4 cannot be added into the chromosome.

Now, we discuss how to search for MPFs in order to fill a chromosome. We define a new term first, which will be used in Algorithm 2.

Definition 4. *Given a set x of MPFs currently stored in a chromosome, $RR(e) = R^+(e) - R(e)$ is called the remaining edge rank of an edge e, where $R(e)$ is obtained from the MPFs in x.*

When $x = \emptyset$, all the remaining edge ranks are equal to their corresponding maximum edge ranks. In this case, any MPF is allowed to appear in the chromosome. For an edge e, when $RR(e) = 0$, no new MPF can pass through this edge.

A new MPF is searched from a line drawing based on both the minimal edge face phenomenon and the remaining edge ranks. This local search strategy, when combined with the GA, forms an efficient and effective solution to the face identification problem. Algorithm 2 describes the pseudocode of the local search algorithm.

Algorithm 2. (Extending a chromosome)

[To add more minimal edge MPFs into the set x of MPFs currently stored in the chromosome by searching on the line drawing $G = (V, E)$ with the edges numbered from 1 to $|E|$, given the remaining edge ranks, $RR(e), e = 1, 2, \ldots |E|$.]

1. **procedure** $Extension(x)$
2. Generate a random permutation $(r_1, r_2, \ldots, r_{|E|})$, where $r_i \neq r_j$ for $i \neq j$, and $r_i, r_j, i, j \in \{1, 2, \ldots, |E|\}$
3. **for** $i = 1$ **to** $|E|$ **do**
4. **if** $RR(r_i) > 0$ then
5. **begin**
6. Call $FindingMinimalEdgeMPF(r_i)$
7. **if** there exists a new MPF **and** it can coexist with all the MPFs stored currently in the chromosome **then**
8. Add it to x and update RR
9. **end**

In the algorithm, the search for more MPFs is done by examining the edges one by one, but not in some fixed edge order. Instead, the order is randomly generated in Step 2 when the procedure is called each time. The goal is to reduce the risk for the algorithm to get trapped in local maxima. Step 6 calls another procedure to search for one MPF passing through edge r_i if the remaining rank $RR(r_i) > 0$. When there exists such an MPF, we have to test whether Condition 2 is satisfied (Step 7). If yes, the MPF is added into the chromosome and the remaining edge ranks recorded in the array RR must be updated (Step 8).

$FindingMinimalEdgeMPF()$ is a modified version of Moore's efficient breadth-first search algorithm for finding a shortest path in a graph [15]. Given an edge e, it searches for a MPF passing through e and having minimal edges, which reflects the idea of the minimal edge face phenomenon. The detail and explanation of this algorithm can be found in [34].

4.5 Mutation and Crossover

In this section, we will discuss how to incorporate the local search algorithm into the standard GA. Mutation and crossover are two commonly used operators in GAs. It is worth noting that, after an operation on a chromosome, the MPFs in the chromosome must still be able to coexist with each other in the line drawing.

The process of mutation makes the genes of a selected chromosome undergo random changes with a small mutation rate. The motivation behind it is to introduce a diversity of solutions into the population. In our application of the GA, mutation is performed by simply deleting some genes (MPFs) in a chromosome with a small rate.

Crossover is the main operator in the GA. Its goal is to mate good chromosomes to generate better offspring. With a crossover rate, it operates on two selected chromosomes by combining the genes of the two. We use a single point crossover to produce two children, O_1 and O_2, from two parents, P_1 and P_2, as shown in Fig. 7a. However, the MPFs in O_1 or O_2 may not be able to coexist in the same line drawing. Let us take O_1 as an example. Suppose each of the three MPF pairs, C_2 and C_5', C_2 and C_7', and C_4 and C_7', cannot coexist in the drawing. To maintain valid children, some MPFs must be deleted. Two valid children O_1^1 and O_1^2 from O_1 are shown in Fig. 7b which are shorter and have most compatible MPFs in O_1. Similarly, we may obtain two valid children O_2^1 and O_2^2 from O_2.

It is common in GAs that two parents produce two children in order to maintain the same population size. Here, we keep the two best children out of O_1^1, O_1^2, O_2^1, and O_2^2. From the hybrid GA given in the next section, it can be seen that, in order to obtain fittest chromosomes, we always try to extend every

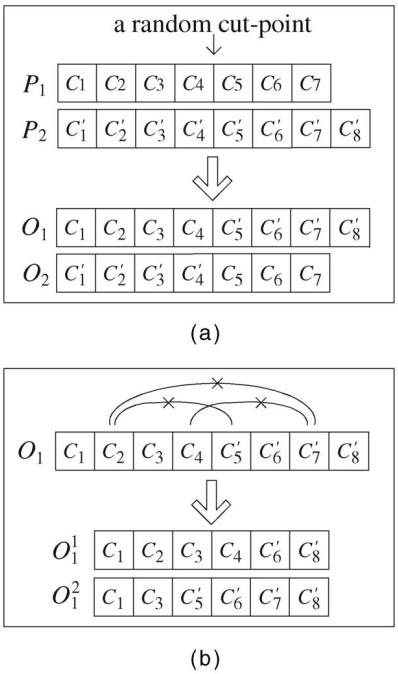

Fig. 7. (a) A single point crossover. (b) Two valid children from O_1.

chromosome so that it contains as many MPFs as possible, with the help of the local search algorithm. Therefore, before evaluating O_1^1, O_1^2, O_2^1, and O_2^2, we extend each of them by calling the procedure $Extension$ listed in Algorithm 2. Then, the two extended children with the highest fitness values remain and the other two are discarded.

4.6 The Hybrid GA

Combining the standard GA with our local search scheme, we formulate the hybrid GA in Algorithm 3, where $MaxGeneration$ denotes the maximum generation the algorithm will reach and $PopulationSize$ is an even integer denoting the size of the population $P(t)$. The local search procedure $Extension$ (Algorithm 2) is incorporated into the standard GA, always trying to add more faces into chromosomes to make them fitter.

Algorithm 3. (Hybrid GA)

1. $t \leftarrow 1$
2. Initialize $P(t)$ by calling $Extension(x)$ with $x = \emptyset$ for each chromosome in $P(t)$
3. Calculate the fitness value of each chromosome in $P(t)$
4. **while** $t \leq MaxGeneration$ **do**
5. **begin**
6. Rank the chromosomes in $P(t)$ linearly based on their fitness values
7. **for** $i = 1$ **to** $PopulationSize/2$ **do**
8. **begin**
9. Select two chromosomes P'_1 and P'_2 from $P(t)$ based on the ranking
10. Perform mutation on P'_1 and P'_2 with a mutation rate, producing two chromosomes P_1 and P_2
11. Perform single-point crossover on P_1 and P_2 with a crossover rate, producing another two chromosomes O_1 and O_2
12. Delete incompatible MPFs in O_1 and O_2, producing four valid children O_1^1, O_1^2, O_2^1, and O_2^2
13. Call $Extension$ to extend O_1^1, O_1^2, O_2^1, and O_2^2 and keep the best two in $O(t)$
14. **end**
15. Form $P(t+1)$ by keeping $PopulationSize/2$ best chromosomes in $P(t)$ and $PopulationSize/2$ best chromosomes in $O(t)$
16. $t \leftarrow t + 1$
17. **end**

5 Experimental Results

In this section, we conduct a set of experiments to demonstrate that our hybrid GA can identify faces from line drawings efficiently. We also compare the efficiency between the GA and Liu and Lee's algorithm [9]. We do not compare with the algorithm proposed by Shpitalni and Lipson [8] because Liu and Lee's algorithm is already faster than Shpitalni and Lipson's. In what follows, LLA and HGA are short for Liu and Lee's algorithm and the hybrid GA, respectively.

All the algorithms are implemented using Visual C++, running on a 1 GHz Pentium III PC. In the HGA, the parameters, population size, maximum generation, mutation rate, and crossover rate, are set to be 50, 15, 0.05, and 0.9, respectively. They are fixed in the experiments unless mentioned otherwise. How these parameters affect the HGA will be discussed at the end of this section.

GAs are a stochastic global optimization technique. It is not guaranteed that a GA will find optimal solutions to a problem. In fact, GAs return only nearly optimal solutions in most applications to combinatorial problems [11], [12], [13], [14]. The likelihood of some GA finding optimal solutions depends on factors like how hard a problem is, how well the GA is designed, and the parameters chosen in the GA. In the following, when we say that the HGA can identify the faces of a line drawing, we mean that, given the above set of parameters, the HGA has an extremely high probability of obtaining the optimal solutions. The rate of failure is negligibly small in practical applications (see more discussion later).

The first set of line drawings for testing the HGA come from all the objects given in the sections of experimental results in the two papers by Shpitalni and Lipson [8] and Liu and Lee [9]. The HGA finds the same faces in each object as the other two algorithms and takes about 0.25 second each. It is not necessary to compare the computational times between the HGA and LLA on these objects with less than 30 faces because the LLA is also fast enough to handle them.

Next, we will show another set of objects each with more than 30 faces. In Fig. 8, four stairs models with increasing faces are shown. The 32 faces in Stairs 1 found by the HGA are also shown. There is one solution for the first three models. But, there are two in Stairs 4. One solution contains MPF 1 and MPF 2 and the other contains MPF 3 and MPF 4 (see Fig. 8). The other 37 faces common in the two solutions are omitted. For the face identification, there may be multiple solutions in a line drawing like Stairs 4. In [8] and [9], when cases of multiple solutions appeared, they used some image regularities to select the most plausible one; for Stair 4, the solution containing MPF 1 and MPF 2 is chosen.

When the HGA is applied to a line drawing having multiple solutions, we find that, in most cases, the most plausible one is found out in the population. In order not to

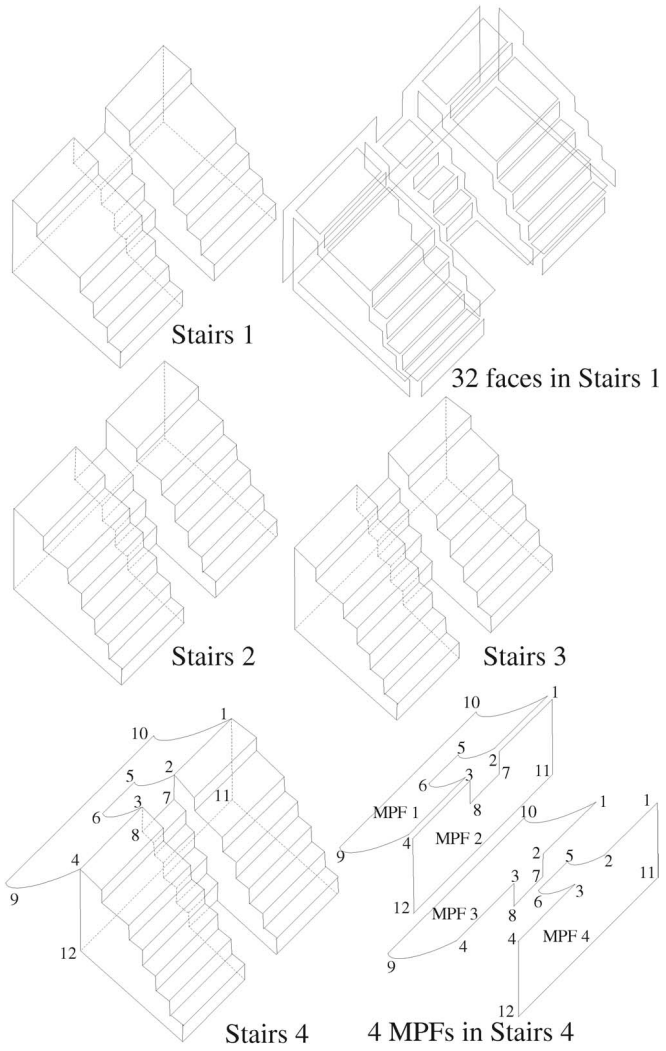

Fig. 8. Four stairs models.

miss it, however, after the execution of the HGA, we run a postprocessing procedure to look for multiple solutions. By examining MPFs 1 and 2 (or MPFs 3 and 4) in Fig. 8, it is not difficult to see that the two MPFs share more than one common edge (here they are edges $(1,2)$ and $(3,4)$). With this observation, even if the solution containing MPFs 1 and 2 is not found out by the HGA, it can still be obtained from the solution containing MPFs 3 and 4 in the following way: Look at MPFs 3 and 4. The two common edges, (1,2) and (3,4), separate them into four sets of edges:

$$S_1 = \{\text{edge}(2,7), \text{edge}(7,8), \text{edge}(8,3)\}$$
$$S_2 = \{\text{edge}(4,9), \text{edge}(9,10), \text{edge}(10,1)\}$$
$$S_3 = \{\text{edge}(2,5), \text{edge}(5,6), \text{edge}(6,3)\}$$
$$S_4 = \{\text{edge}(4,12), \text{edge}(12,11), \text{edge}(11,1)\},$$

where S_1, S_2, and the two common edges form MPF 3, while S_3, S_4, and the two common edges form MPF 4. By exchanging S_1 and S_3, we can obtain two MPFs: One formed by S_2, S_3 and the two common edges and the other formed by S_1, S_4, and the two common edges. These two MPFs are exactly MPFs 1 and 2. Similar analysis applies to more complicated cases. This postprocessing successfully deals with all the line drawings having multiple solutions in the experiments (including those in [8] and [9]). Like [8] and [9], when multiple solutions are obtained from a line drawing, the most plausible one is selected by the image regularities.

TABLE 1
Results for the Four Models in Fig. 8

	Stairs 1	Stairs 2	Stairs 3	Stairs 4
Faces	32	36	38	39
Edges	90	102	108	114
MPFs	2561	8089	16428	48126
Memory in LLA	$>6.5\times10^6$	$>6.5\times10^7$	$>2.6\times10^8$	$>2.3\times10^9$
Memory in HGA	$<2.5\times10^5$	$<2.5\times10^5$	$<2.5\times10^5$	$<2.5\times10^5$
Time taken by LLA	9s	95s	1632s	?
Time taken by HGA	0.35s	0.43s	0.48s	0.50s

Table 1 summarizes the results for the four stairs. It is obvious that the number of MPFs grows exponentially with the number of edges, which causes both memory and computational time taken by the LLA to increase exponentially. Let us consider the memory requirement first. In the LLA (also in Shpitalni and Lipson's algorithm), most memory consumption is due to the generation of the matrix $B = [b_{ij}]_{n\times n}$, with n being the number of MPFs (see Definitions 1 and 2). When there are 48,126 MPFs in Stairs 4, the LLA needs at least $48,126 \times 48,126 \simeq 2.3 \times 10^9$ basic memory units. Thus, the LLA has to allocate a huge memory of 2.3G units to handle Stairs 4. In the HGA, however, the memory requirement does not depend on the number of MPFs. The two largest arrays are used to store the chromosomes in the population $P(t)$ and the offspring $O(t)$. $P(t)$ and $O(t)$ together need a memory of less than

$$(2 \times \text{Population} \times \text{Max\_Length\_Of\_A\_Chromosome} \times \text{Max\_Length\_of\_A\_Circuit}).$$

In our experiments, the maximum length of a chromosome and the maximum length of a circuit are less than 50. Therefore, $P(t)$ and $O(t)$ take less than 2.5×10^5 basic memory units.

The last two rows in Table 1 give the times (in seconds) taken by the two algorithms. It is obvious that the time consumed by the LLA grows exponentially. We do not give the time for the LLA to deal with Stairs 4 because it takes too much time to find the solution (we could not obtain the result after the LLA ran for one day). On the contrary, we are very happy with the HGA. It is much more efficient and its computational time increases approximately linearly with the number of edges of the stairs models.

Fig. 9 displays a convergence process of the HGA when it is run to handle Stairs 4. The average fitness value is

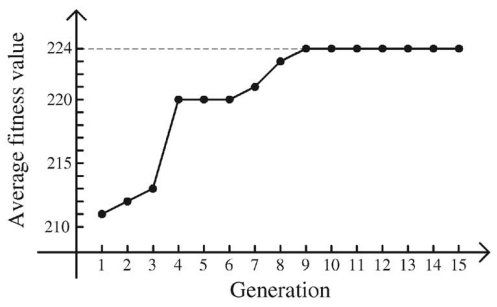

Fig. 9. Convergence process of the HGA for Stairs 4.

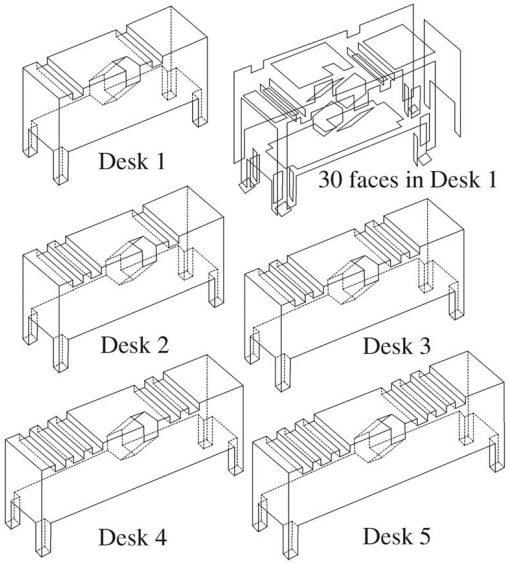

Fig. 10. Five desk models.

TABLE 2
Results for the Five Models in Fig. 10

	Desk 1	Desk 2	Desk 3	Desk 4	Desk 5
Faces	30	34	38	42	46
Edges	90	102	114	126	138
MPFs	3259	14520	65000	307012	1466488
Memory in LLA	$>1.0\times10^{7}$	$>2.1\times10^{8}$	$>4.2\times10^{9}$	$>9.4\times10^{10}$	$>2.1\times10^{12}$
Memory in HGA	$<2.5\times10^{5}$	$<2.5\times10^{5}$	$<2.5\times10^{5}$	$<2.5\times10^{5}$	$<2.5\times10^{5}$
Time taken by LLA	19s	740s	?	?	?
Time taken by HGA	0.37s	0.48s	0.58s	0.70s	0.83s

computed by considering all the chromosomes in $P(t)$ at each generation t. For this example, every chromosome contains a best solution beginning from generation 9.

Fig. 10 shows five desk models, all of which have a hole passing through them. Desks 1-5 have two, three, four, five and six slots on them, respectively. Each of them has four solutions. Our algorithm can find all the solutions. The most plausible one for Desk 1 is also shown in Fig. 10. Table 2 summarizes the results for the five desk models. Again, we see that the combinatorial explosion caused by the number of MPFs takes the LLA 740 seconds to find the faces of Desk 2. For each model of Desks 3-5, the LLA has to spend so much time that we cannot wait for the result. On the other hand, the HGA again exhibits excellent performance both in computational time and memory requirement. Fig. 11 shows the

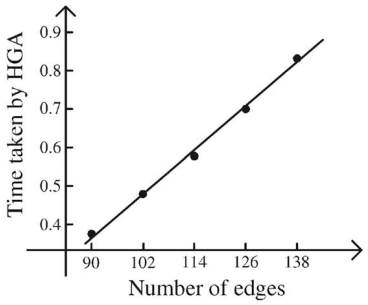

Fig. 11. Time (second) taken by the HGA versus the number of edges for the desk models.

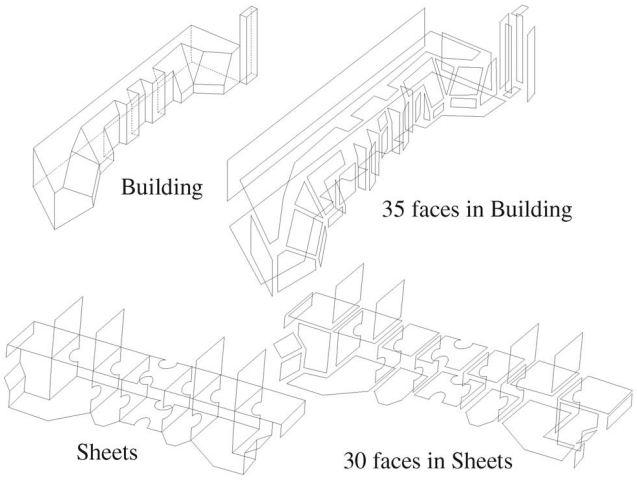

Fig. 12. Two objects and their faces found.

TABLE 3
Results for the Two Models in Fig. 12

	Building	Sheets
Faces	35	30
Edges	89	122
MPFs	8569	8129
Memory in LLA	$>7.3\times10^{7}$	$>6.6\times10^{7}$
Memory in HGA	$<2.5\times10^{5}$	$<2.5\times10^{5}$
Time taken by LLA	96s	137s
Time taken by HGA	0.35s	0.45s

approximately linear relation between the time taken by the HGA and the number of edges for the Desk models.

Two more objects in Fig. 12 are used to test the HGA and LLA. There is one solution for each of the two objects. The faces found by the HGA (or LLA) are also given in the figure. The results are summarized in Table 3, which again demonstrates the significant better performance of the HGA over the LLA.

As we mentioned before, given the set of parameters, (population $= 50$, maximum generation $= 15$, mutation rate $= 0.05$, crossover rate $= 0.9$), the HGA can find the optimal solutions in a very high probability. In fact, for all the line drawings given in the experiments (including those in the two previous papers [8], [9]), the HGA would not fail once if we run the HGA on each object 1,000 times. Now, we study how the success probability varies when some of the parameters are changed.

At first, we find that the HGA is not sensitive to the mutation rate and crossover rate (usually, the former is small and the latter is large). For example, if they are chosen to be 0.1 and 0.8, respectively, the success probability is almost the same. Thus, we consider in the following the changes of the population size and maximum generation only. Here, we use Stairs 4 as the test subject.

Fig. 13 shows how the success probability goes to 1.0 as the population size increases from 10 to 50. For each of the population size, the HGA is run 500 times for Stairs 4 with the maximum generation set at 15. When the population size is set at 10, 20, 30, 40, and 50, the numbers of times that the HGA fails to find the optimal solutions are 186, 70, 43, 2, and 0, respectively. Fig. 14 shows the relation between the

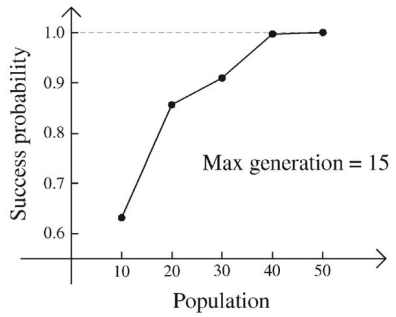

Fig. 13. Success probability versus population size for Stairs 4 with maximum generation $= 15$.

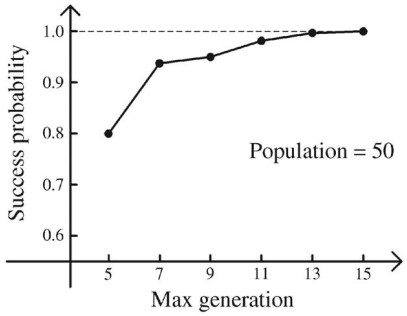

Fig. 14. Success probability versus maximum generation for Stairs 4 with population $= 50$.

success probability and the maximum generation. For each of the maximum generation, the HGA is also run 500 times with the population size set at 50. When the maximum generation is set at 5, 7, 9, 11, 13, and 15, the numbers of the HGA failure are 101, 35, 22, 12, 5, and 0, respectively. These two examples show that, when the population size and maximum generation are chosen to be 50 and 15, the HGA can have a very high probability of finding the optimal solutions for Stairs 4.

Is it possible for the HGA to fail when dealing with a line drawing? The answer is yes for complex line drawings such as Desk 5. We do find that the HGA gets into local optima at an approximate rate of 0.00025 when handling Desk 5. In other words, if the HGA is executed 4,000 times for Desk 5, it would fail once. This very low rate of failure is negligible in practical applications. It demonstrates that the HGA is not only efficient in computational time and memory requirement, but also robust in obtaining optimal solutions.

6 Comparison with Simulated Annealing

While the HGA performs very well on the face identification problem, we may wonder whether there exist other optimization methods that can reach similar performance or better. We have implemented a simulated annealing (SA) algorithm and made comparisons between the two algorithms. The new ideas presented in Section 4 are the key to the efficient implementations of SA, just as they are the key to the HGA.

SA is a well-known heuristic optimization technique and has been successfully applied to many optimization problems [35]. It simulates the slow cooling process of material from over its melting point to its solid state where the lowest energy configuration is expected. In order to implement SA in the traditional minimization framework, we define the following energy function

$$f(x) = -\sum_{i \in x} w(i), \tag{17}$$

where x is the current solution (a set of compatible MPFs) and $w(i)$ denotes the number of edges of a MPF i. In fact, $f(x)$ is simply the negative of the fitness function of a chromosome defined in (14). The detailed description of the theory and various modifications of SA can be found in [35]. Here, we present a SA algorithm (SAA) tailored for face identification.

Algorithm 4. (SAA)

1. Construct an initial solution x_0 by calling $Extension(x)$ with $x = \emptyset$
2. $t \leftarrow T_0$
3. **while** $t > T_{min}$ **do**
4. **begin**
5. $i \leftarrow 1$
6. **while** $i \leq M_t$ **do**
7. **begin**
8. Call $Neighbor(x_0)$ to obtain a neighbor x of x_0
9. **if** $\delta = f(x) - f(x_0) < 0$ **then** $x_0 \leftarrow x$ else
10. **begin**
11. Randomly generate $r \in (0, 1)$
12. **if** $r < e^{-\delta/t}$ **then** $x_0 \leftarrow x$
13. **end**
14. $i \leftarrow i + 1$
15. **end**
16. $t \leftarrow \alpha(t)$
17. **end**

In the SAA, T_0 (T_{min}) is an initial (final) *temperature* high (low) enough, M_t defines the number of iterations executed at each temperature t, and $\alpha(t)$ is a temperature reduction function. The final solution stored in x_0 is obtained when the algorithm ends. All the steps except 1 and 8 are similar to those in the general SA algorithm [35]. Step 1 calls Algorithm 2, $Extension$, described in Section 4.4 to generate a random initial solution. The key of the SAA is in Step 8: randomly selecting a neighbor of the current point x_0 in the solution space. The procedure is given in the following algorithm.

Algorithm 5. (Randomly selecting a neighbor x of x_0)

1. **procedure** $Neighbor(x_0)$
2. Randomly select an edge e with $RR(e) > 0$
3. Randomly search the line drawing for one MPF that passes through e
4. Delete the MPFs in x_0 that are not compatible with the newly-found MPF and put it into x_0, forming a new set x' of MPFs
5. Call $Extension(x')$ to extend x', resulting in x

From the above algorithm, it can be seen that $Extension$ is called again, in which the idea of the minimal edge face phenomenon is embedded to help find faces.

To make an SA algorithm work well in solving an optimization problem, the parameters T_0, T_{min}, M_t, and the

function $\alpha(t)$ must be chosen carefully, and they are usually problem-dependent. After many trial-and-error experiments, we find that the SAA with $T_0 = 300$, $T_{min} = 0.001$, $M_t = 10$, and $\alpha(t) = 0.9t$ can give very good results both in computational efficiency and in finding optimal solutions. With these parameters and the temperature reduction function, the performance of the SAA is close to that of the HGA. If the SAA is executed on each line drawing in the experiments 1,000 times, it would not fail once and each execution takes less than 1 second. More specifically, let us consider the line drawing Desk 5 that has the most faces. The SAA gets into local optima approximately at a rate of 0.00033, that is, it would fail only once if it is run 3,000 times. This rate of failure is low enough in practice. Besides, the time the SAA takes in each execution for Desk 5 is 0.85 second on the average.

Our experiments show that the performance of the HGA is slightly better than that of the SAA. Both algorithms can efficiently find faces from a line drawing with very small rates of failure. The key to making these algorithms perform much better than the previous algorithms in [8] and [9] comes from the local search scheme where the minimal edge face phenomenon and the geometric constraints are incorporated, as well as from the algorithmic designs that avoid the search of all MPFs from a line drawing.

7 Conclusions

A hybrid GA for finding faces from single 2D line drawings has been presented. The faces identified from a line drawing provide important information for the reconstruction of its 3D geometry. From the experiments, it can be seen that the HGA finds the same faces as the previous two algorithms do, but exhibits significantly better performance for objects with faces > 30, both in computational time and memory requirement. The obstacle for the two previous methods to handle such objects is that each of them involves two combinatorial problems: searching for all MPFs from a line drawing and then searching for faces from the MPFs. This combinatorial explosion makes it difficult, if not impossible, for the two algorithms to handle objects with faces > 30.

Our strategy to conquer the combinatorial explosion is to combine the standard GA with a novel local search scheme. The former is well-known for its good global search ability; the latter has a high likelihood of finding faces directly from a line drawing based on the maximum edge ranks, face adjacency theorem, and minimal edge face phenomenon. The amount of memory allocated in our HGA depends on the size of the population, the maximum length of a chromosome, and the maximum length of a circuit. These quantities do not increase much with the increase of edges in a line drawing. For all the line drawings in the experiments, the HGA takes less than one second to deal with each of them on a 1 GHz Pentium III PC. From the two sets of objects given in Figs. 8 and 10, we can see that the computational time of the HGA grows approximately linearly with the number of edges of the drawings, presenting excellent computational performance. The experiments also demonstrate that the HGA is robust in finding optimal solutions.

We have also implemented a simulated annealing algorithm for face identification based on the local search scheme. Similarly to the HGA, the SSA is designed without the need to search for all the MPFs from a line drawing. The HGA works slightly better than the SSA.

Acknowledgments

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong SAR (Project no. AoE/E-01/99).

References

[1] T. Marill, "Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects," *Int'l J. Computer Vision,* vol. 6, no. 2, pp. 147-161, 1991.
[2] Y.G. Leclerc and M.A. Fischler, "An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames," *Int'l J. Computer Vision,* vol. 9, no. 2, pp. 113-136, 1992.
[3] H. Lipson and M. Shpitalni, "Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing," *Computer-Aided Design,* vol. 28, no. 8, pp. 651-663, 1996.
[4] K. Sugihara, "A Necessary and Sufficient Condition for a Picture to Represent a Polyhedral Scene," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 5, pp. 578-586, 1984.
[5] P.E. Debevec, C.J. Yaylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. SIGGRAPH '96 Conf.,* pp. 11-20, 1996.
[6] A. Turner, D. Chapman, and A. Penn, "Sketching Space," *Computers & Graphics,* vol. 24, pp. 869-879, 2000.
[7] A.P. Vicent, P.C. Calleja, and R.R. Martin, "Skewed Mirror Symmetry in the 3D Reconstruction of Polyhedral Models," *J. Winter School on Computer Graphics,* vol. 11, no. 3, pp. 504-511, 2003.
[8] M. Shpitalni and H. Lipson, "Identification of Faces in a 2D Line Drawing Projection of a Wireframe Object," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 10, pp. 1000-1012, Oct. 1996.
[9] J. Liu and Y.T. Lee, "A Graph-Based Method for Face Identification from a Single 2D Line Drawing," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1106-1119, Oct. 2001.
[10] E.M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practices.* Prentice-Hall, 1977.
[11] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization.* New York: John Wiley & Sons, 2000.
[12] K.F. Man, K.S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Designs.* New York: Springer, 1999.
[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* New York: Springer-Verlag, 1996.
[14] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Mass.: Addison-Wesley, 1989.
[15] G. Chartrand and O.R. Oellermann, *Applied and Algorithmic Graph Theory.* New York: McGraw-Hill, 1993.
[16] M. Mantyla, *An Introduction to Solid Modeling.* Computer Science Press, 1988.
[17] D.A. Huffman, "Impossible Objects as Nonsense Sentences," *Machine Intelligence,* B. Meltzer and D. Mitchie, eds., vol. 6, pp. 295-323, 1971.
[18] M.B. Clowes, "On Seeing Things," *Artificial Intelligence,* vol. 2, pp. 79-116, 1971.
[19] D. Waltz, "Understanding Line Drawings of Scenes with Shadows," *Psychology of Computer Vision,* P.H. Winston, ed., pp. 19-91, New York: McGraw-Hill, 1975.
[20] R. Haralick and L. Shapira, "The Consistent Labeling Problem: Part 1," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 1, no. 2, pp. 173-184, 1979.
[21] J. Malik, "Interpreting Line Drawings of Curved Objects," *Int'l J. Computer Vision,* vol. 1, pp. 73-103, 1987.
[22] M.C. Cooper, "Interpretation of Line Drawings of Complex Objects," *Image and Vision Computing,* vol. 11, no. 2, pp. 82-90, 1993.
[23] M.C. Cooper, "The Interpretations of Line Drawings with Contrast Failure and Shadows," *Int'l J. Computer Vision,* vol. 43, no. 2, pp. 75-97, 2001.

[24] R. Lequette, "Automatic Construction of Curvilinear Solid from Wireframe Views," *Computer-Aided Design,* vol. 20, no. 4, pp. 171-180, 1988.
[25] S. Ablameyko, V. Bereishik, A. Gorelik, and S. Medvedev, "3D Object Reconstruction from Engineering Drawing Projections," *Computing & Control Eng. J.,* vol. 10, no. 6, pp. 277-284, 1999.
[26] M.H. Kuo, "Reconstruction of Quadric Surface Solid from Three-View Engineering Drawings," *Computer-Aided Design,* vol. 30, no. 7, pp. 517-527, 1998.
[27] G. Markowsky and M.A. Wesley, "Fleshing Out Wire-Frames," *IBM J. Research and Development,* vol. 24, no. 5, pp. 582-597, 1980.
[28] P.M. Hanrahan, "Creating Volume Models from Edge-Vertex Graphs," *Computer Graphics,* vol. 16, no. 3, pp. 77-84, 1982.
[29] R.D. Dutton and R.C. Brigham, "Efficiently Identifying the Faces of a Solid," *Computer and Graphics in Mechanical Eng.,* vol. 7, no. 2, pp. 143-147, 1983.
[30] M.A. Ganter and J.J. Uicker, "From Wire-Frame to Solid Geometric: Automated Conversion of Data Representations," *Computer in Mechanical Eng.,* vol. 2, no. 2, pp. 40-45, 1983.
[31] J.S. Hojnicki and P.R. White, "Converting CAD Wireframe Data to Surfaced Representations," *Computers in Mechanical Eng.,* pp. 19-25, Mar./Apr. 1988.
[32] S.C. Agarwal and J.W.N. Waggenspack, "Decomposition Method for Extracting Face Topologies from Wireframe Models," *Computer-Aided Design,* vol 24, no. 3, pp. 123-140, 1992.
[33] J. Liu, Y.T. Lee, and W.-K. Cham, "Identifying Faces in a 2D Line Drawing Representing a Manifold Object," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 12, pp. 1579-1593, Dec. 2002.
[34] J. Liu, "Efficient Search of Faces from Complex Line Drawings," technical report, the Multimedia Lab, Dept. of IE, the Chinese Univ. of Hong Kong, 2003.
[35] K.A. Dowsland, "Simulated Annealing," *Modern Heuristic Techniques for Combinatorial Problems,* C.R. Reeves, ed. pp. 20-69, Halsted Press, 1993.

Jianzhuang Liu (M'02-SM'02) received the BE degree from Nanjing University of Posts & Telecommunications, People's Republic of China, in 1983, the ME degree from Beijing University of Posts & Telecommunications, People's Republic of China, in 1987, and the PhD degree from the Chinese University of Hong Kong, in 1997. From 1987 to 1994, he was an academic member in the Department of Electronic Engineering, Xidian University, China. From August 1998 to August 2000, he was a research fellow at the School of Mechanical and Production Engineering, Nanyang Technological University, Singapore. Then he was a postdoctoral fellow at the Chinese University of Hong Kong for several years. He is now an assistant professor in the Department of Information Engineering, the Chinese University of Hong Kong. His research interests include image processing, graphics, computer vision, pattern recognition, and artificial intelligence. He is a senior member of the IEEE.

Xiaoou Tang (S'93-M'96-SM'02) received the BS degree from the University of Science and Technology of China, Hefei, in 1990, and the MS degree from the University of Rochester, Rochester, New York, in 1991. He received the PhD degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is currently an associate professor and the director of Multimedia Lab in the Department of Information Engineering, the Chinese University of Hong Kong. His research interests include computer vision, pattern recognition, and video processing. He is the local chair of the IEEE International Conference on Computer Vision (ICCV) 2005. He is a guest editor of the special issue on underwater image and video processing for the *IEEE Journal of Oceanic Engineering* and the special issue on image and video-based biometrics for the *IEEE Transactions on Circuits and Systems for Video Technology*. He is a senior member of the IEEE.