Fingerprint Template Compression by Solving a Minimum Label k-Node Subtree Problem

Günther R. Raidl and Andreas Chwatal

Institute of Computer Graphics and Algorithms, Vienna University of Technology,
Favoritenstrasse 9–11/186, 1040 Vienna, Austria, {raidl|chwatal}@ads.tuwien.ac.at

Abstract. We present a new approach for strongly compressing a relatively small amount of poorly structured data, as is required when embedding fingerprint template information in images of ID-cards by means of watermarking techniques. The approach is based on the construction of a directed tree spanning a selected part of the data points and a codebook of template arcs used for a compact encoding of relative point positions. The selection of data points, the tree structure, and the codebook are simultaneously optimized by a new exact branch-and-cut approach or, alternatively, a faster *greedy randomized adaptive search procedure* (GRASP) to maximize compression. Experiments indicate that the new method can encode the required information in less space than several standard compression algorithms.

Keywords: Fingerprint recognition, data compression, minimum label spanning tree, branch-and-cut, greedy randomized adaptive search procedure (GRASP)
PACS: 01.30.Cc, 02.10.Ox

INTRODUCTION

We consider a new technique for strongly compressing a relatively small amount of weakly structured data. The application background lies in embedding fingerprint information by means of watermarking techniques [7] in images such as photos on ID-cards as an additional security feature. Since the amount of information that can be reliably embedded in an ID-card's photo without distorting it too much is heavily limited, a strong compression technique is required for shrinking raw fingerprint information [5]. While the compression does not need to be lossless, it is nevertheless crucial not to lose too much precision, so that checking fingerprints against the encoded information is possible with small errors.

Fingerprint matching usually relies on image processing techniques for extracting so-called *minutiae*, which are points of interest such as bifurcations, crossover, and ridge endings [6]. Typically, 15 to 60 minutiae are extracted from a single fingerprint, and for each a discrete type, x and y coordinates and an angle are obtained as attributes. In practice, a reasonably accurate agreement of a relatively low number of minutiae between two fingerprints indicates a match with high probability. We focus here on the selection of a pre-specified number k of all available minutiae in combination with their lossless encoding by means of a small codebook of template arcs and a depending directed tree structure spanning the points. By solving an extension of the well-known minimum label spanning tree problem, we aim at achieving the highest possible compression rate.

In the following, we introduce our approach more formally and present an exact branch-and-cut algorithm as well as a faster heuristic method based on a *greedy randomized adaptive search procedure* (GRASP) [4]. Some information on our modelling and the GRASP approach in particular can also be found in [3]. The current article extends this previous work primarily by introducing the exact branch-and-cut method and addressing its results.

THE TREE-BASED COMPRESSION MODEL

We consider as given input data n d-dimensional points $V = \{v_1, \ldots, v_n\}$ from a discrete domain $\mathbb{D} = \{0, \ldots, \tilde{v}^1 - 1\} \times \ldots \times \{0, \ldots, \tilde{v}^d - 1\}$ corresponding to the minutiae of a fingerprint ($d = 4$ in our case). The domain limits $\tilde{v}^1, \ldots, \tilde{v}^d \in \mathbb{N}$ represent the individual sizes and resolutions of the d dimensions.

We compress these data by selecting a subset of k of these n points and connecting them by an outgoing arborescence, i.e., a directed spanning tree. One point forms the root node and each arc of the tree represents the relative geometric position of its end point in dependence of its starting point. To achieve the compression, we maintain a

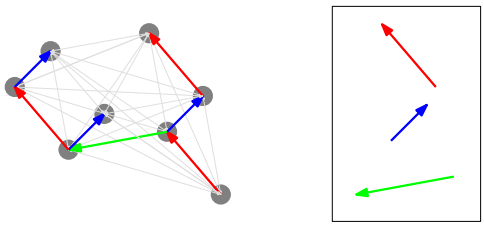

FIGURE 1. Illustration to the encoding of points via a directed spanning tree using a codebook of template arcs; correction vectors are neglected.

limited, small set of appropriately chosen *template arcs* and instead of storing for each tree arc its lengths w.r.t. any of the d dimensions, we encode it more tightly by a simple reference to the most similar template arc plus a correction vector from a small domain. Thus, the set of template arcs acts as a codebook, see also Fig. 1. In order to achieve a compression rate that is as high as possible, we optimize the selection of encoded points, the tree structure, and the set of template arcs at the same time. The domain for the correction vectors is pre-specified, while the number of template arcs is the objective to be minimized. As compressed data, we finally write the template arc set and the tree. The latter is traversed in a depth-first manner, and at each step we store one bit indicating whether a new arc has been traversed to reach a new node or backtracking took place. When following a new arc, a reference to its template arc plus the small correction vector are additionally written.

More formally, we are given a complete directed graph $G = (V,A)$ with the node set V corresponding to the n given points and arc set $A = \{(i, j) \mid i, j \in V \wedge i \neq j\}$. A solution to our problem consists of

1. a vector of template arcs $R = (r_1, \ldots, r_m) \in \mathbb{D}^m$ of arbitrary size m representing the codebook, and
2. a rooted, outgoing tree $T = (V_T, A_T)$ with $V_T \subseteq V$ and $A_T \subseteq A$ connecting precisely $|V_T| = k$ nodes, in which each tree arc $(i, j) \in A_T$ has associated
   - a template arc index $\kappa_{i,j} \in \{1, \ldots, m\}$ and
   - and a correction vector $\delta_{i,j} \in \mathbb{D}'$ from a pre-specified, small domain $\mathbb{D}' \subseteq \mathbb{D}$.

The essential condition for a solution to be feasible is

$$v_j = (v_i + r_{\kappa_{i,j}} + \delta_{i,j}) \bmod \tilde{v} \qquad \forall (i, j) \in A_T, \tag{1}$$

i.e., each tree arc must correspond to the sum of its referenced template arc plus its correction vector. The modulo-calculation is performed in order to always stay within a finite ring, so there is no need for negative values and we do not have to explicitly consider domain boundaries. The major objective now is to find a feasible solution that is minimal with respect to the number of template arcs m, i.e., the codebook size.

REFORMULATION AS A MINIMUM LABEL k-NODE SUBTREE PROBLEM

We approach this goal by first deriving a (large) set R_{cand} of possibly useful *candidate template arcs* and then select the actual codebook as a smallest subset for which a feasible solution exists. The candidate set R_{cand} is determined in such a way that an optimal (minimal) codebook is guaranteed to be contained but obviously redundant template arcs are avoided. For an arbitrary arc $r \in \mathbb{D}$, let $Z(r) \subseteq A$ be the subset of arcs from A for which r can be used as template so that corresponding correction vectors from the limited domain $\mathbb{D}'$ exist in order to fulfill condition (1). In particular, an arc r is considered redundant and not included in R_{cand} when it cannot be used for any arc in A, i.e. $Z(r) = \emptyset$, or it is dominated by some other arc r', i.e. $Z(r) \subset Z(r')$. Furthermore, from all candidate arcs with exactly the same $Z(r)$, only one is kept. We developed a dynamic programming procedure using a k-d tree data structure for efficiently determining R_{cand}.

Having R_{cand} now available, our problem is related to the NP-hard *minimum label spanning tree* (MLST) problem introduced in [1], in which an undirected graph is given, each edge has associated a label of a discrete label set, and a spanning tree whose edges induce a smallest possible subset of the labels is sought. Some more recent work on the MLST problem includes [8, 10]. In our problem, the labels correspond to the candidate template arcs R_{cand}, and differences are that we have to consider the directed case, multiple labels (i.e., candidate template arcs) for each arc in

A, and that not all but only k nodes need to be connected. We therefore refer to our problem as *minimum label k-node subtree* (k-MLST) problem.

A BRANCH-AND-CUT APPROACH

For allowing the optimization to also choose as root of the tree a best suited point, we extend V to V' by including a new artificial root node 0 and extend A to A' by introducing arcs $(0,i),\ \forall i \in V$. For each candidate template arc $r \in R_{\text{cand}}$, we define a variable $y_r \in \{0,1\}$ indicating whether or not the arc is part of the codebook R. Furthermore, we use variables $x_a \in \{0,1\}$, $\forall a \in A'$, indicating which arcs belong to the solution tree, and variables $z_i \in \{0,1\}$, $\forall i \in V$, indicating the k points from V covered by the tree. Let $Y(a)$, $\forall a \in A$, be the set of all template arcs that may be used for representing a, i.e., $Y(a) = \{r \in R_{\text{cand}} \mid a \in Z(r)\}$.

Now, we can model the k-MLST problem by the following integer linear program (ILP):

$$\min \quad m = \sum_{r \in R_{\text{cand}}} y_r \tag{2}$$

$$\text{s.t.} \quad \sum_{r \in Y(a)} y_r \geq x_a \qquad \forall a \in A \tag{3}$$

$$\sum_{i \in V} z_i = k \tag{4}$$

$$\sum_{a \in A} x_a = k - 1 \tag{5}$$

$$\sum_{i \in V} x_{(0,i)} = 1 \tag{6}$$

$$\sum_{a \in \delta^-(S)} x_a \geq z_i \qquad \forall i \in V,\ \forall S \subseteq V,\ i \in S,\ 0 \notin S. \tag{7}$$

Inequality (3) ensures that for each tree arc $a \in A$ at least one valid template arc r is selected for representing it. Precisely k nodes from V must be connected according to (4). Equality 5 forces the number of tree arcs to be $k-1$ (excluding the arc from the artificial root), while (6) guarantees a single node to be directly connected to the artificial root. Finally, we use the connectivity constraints (7), where $\delta^-(S)$ represents the ingoing cut of node set S, to ensure the existence of a path from the root 0 to any node $i \in V$ for which $z_i = 1$, i.e., which is selected for connection.

Note that there are exponentially many connectivity constraints (7) since one exists for every possible subset of nodes from V including node i, $\forall i \in V$. Directly solving the ILP would therefore only be feasible for extremely small problem instances. Instead, we apply branch-and-cut [9] and initially omit the connectivity constraints. At each node of the branch-and-bound tree a cutting plane algorithm is performed and only violated connectivity constraints are identified (separated) and included.

The separation procedure utilizes Cherkassky and Goldberg's implementation of the push-relabel method for the maximum flow problem [2] to perform the required minimum cut computations. The branch-and-cut algorithm has been implemented using CPLEX in version 10.

A GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE

Based on a greedy construction heuristic for the classical MLST problem [1], we first developed a greedy heuristic for our k-MLST problem. Starting from an empty set of selected labels and an empty graph, we iteratively add a label r (template arc) and its induced arcs $Z(r)$ from A until the obtained graph contains a feasible k-node arborescence. The decision on which label to take next is, however, now significantly more difficult than in case of the undirected MLST problem, and we based it on the calculation of upper bounds on the numbers of further labels required to obtain a feasible tree. Having reached a feasible solution, it is finally checked for redundant labels that can be removed without destroying the solution's feasibility. Advanced data structures are used in this procedure in order to avoid repeated time-consuming depth-first searches. The method is fast, but the results are of only moderate quality.

Significantly better solutions, i.e., higher compression rates, can be achieved by extending the greedy heuristic to a greedy randomized adaptive search procedure (GRASP) [4]. The constructive heuristic is iterated and the next label is always selected at random from a restricted set of labels chosen according to the greedy criterion. All the candidate solutions are then further improved by a local search that considers three different neighborhood structures, and the overall best solution is the final result.

EXPERIMENTAL RESULTS

Our branch-and-cut implementation is able to solve the k-MLST problem associated with practical fingerprint instances depending on the number of input minutiae $n = |V|$, parameter k, and the correction vector domains usually within a few minutes up to half an hour on a standard desktop PC. Compared to storing k minutiae in an uncompressed way, our encoded solutions only required about 80–90% of the space. While this does not look as a substantial saving at the first glance, one has to keep in mind that minutiae data in general have very low structure. Comparisons with other compression tools including *compress*, *gzip*, and *bzip2* indicated that these methods almost never achieve factors less than 95% (unnecessary meta-information already removed) and often even increase the required space.

The running times of GRASP are in the order of seconds to very few minutes (depending on the neighborhood structures and number of iterations), and it achieves optimal compression in about 95% of the instances. In the remaining cases, usually only one more label (template arc) is required. GRASP therefore is a meaningful option for practical purposes.

CONCLUSIONS AND FUTURE WORK

The proposed approach is highly promising for the considered fingerprint template compression application. The proposed branch-and-cut algorithm is able to achieve provably best compression rates for all practically-sized instances under our model, while the alternative heuristic GRASP method is significantly faster and also exhibits excellent compression.

Further investigations on larger sets of test instances and with different minutiae matching algorithms are necessary, in particular also in order to get a more detailed understanding on how many minutiae must be encoded to achieve reasonably few false matches and non-matches. Further tuning should allow to shorten running times of both, branch-and-cut and GRASP. An alternative optimization approach based on branch-and-price as well as certain improvements in the representation model (e.g. allowing Steiner nodes) are considered in currently ongoing work.

Last but not least, the proposed compression approach is not restricted to the application of fingerprint template compression. It seems to be a viable option also in several other applications where the aim is to store a relatively small amount of data in a highly compact way and running times for compression are not too critical. Obviously, the decoding of the compressed data can always be done efficiently in time $\Theta(k)$.

ACKNOWLEDGMENTS

This work is supported by the European Marie Curie RTN ADONET under grant 504438.

REFERENCES

1. R.-S. Chang and S.-J. Leu. The minimum labeling spanning trees. *Information Processing Letters*, 63(5):277–282, 1997.
2. B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.
3. A. Chwatal, G. R. Raidl, and O. Dietzel. Compressing fingerprint templates by solving an extended minimum label spanning tree problem. In *Proceedings of MIC2007, the 7th Metaheuristics International Conference*, Montreal, 2007.
4. T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
5. A. Jain and U. Uludag. Hiding fingerprint minutiae in images. In *Third Workshop on Automatic Identification Advanced Technologies*, pages 97–102, Tarrytown, NY, USA, 2002.
6. A. K. Jain and D. Maltoni. *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
7. S. Katzenbeisser and F. A. Petitcolas, editors. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Inc., Norwood, MA, USA, 2000.
8. S. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Proc. Letters*, 66(2):81–85, 1998.
9. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.
10. Y. Xiong, B. Golden, and E. Wasil. A one-parameter genetic algorithm for the minimum labeling spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 9(1):55–60, 2 2005.