DOI: https://doi.org/10.15407/kvt191.01.019

UDC 519.1

V.M. KYYKO, PhD (Engineering),
Senior Researcher of Pattern Recognition Department
e-mail: vkiiko@gmail.com
International Research and Training Center
for Information Technologies and Systems
of the National Academy of Sciences of Ukraine
and Ministry of Education and Science of Ukraine,
Acad. Glushkov av., 40, Kiev, 03187, Ukraine

MAXIMUM MATCHING IN WEIGHTED BIPARTITE GRAPHS

Introduction. *The most important algorithms for bipartite graphs maximum matching are observed. These algorithms either find maximum matching in non-weighted bipartite graph (e.g. Hopcroft and Karp's algorithm — $O(m\sqrt{n})$) or choose among all matchings with maximum size one having maximal cost (e.g. Edmonds and Karp's algorithm- $O(mn + n^2 \log n)$). Provided that, in praxis new target settings and algorithms for finding maximum matching in bipartite graphs are also desirable.*

The purpose *of the article is to consider a new task setting and algorithms for maximum matching in weighted bipartite graphs as well as using these algorithms in fingerprint recognition.*

Methods*. Modified versions of finding maximum matching M in graph by searching and augmentation of M-augmenting paths are used.*

Results*. Weighted bipartite graph $G = (V, E)$ with a cost function $ce : E \to \{0,1\}$, that associates each edge with one of two possible values (e.g. 0 or 1) is considered. Maximum matching in the graph in new setting consists in finding among all matchings containing maximum number of edges with weight 1, one having maximal cardinality. Two algorithms with complexity $O(m\sqrt{n})$ being modified versions of the Hopcroft-Karp algorithm are proposed. Examples of using these algorithms for removing gaps of lines and finding true correspondence of minutiae in fingerprint recognition are considered.*

Conclusions*. Proposed algorithms find maximum matching in input bipartite graph among all matchings having maximal cardinality in given subset of this graph edges. Using of proposed algorithms leads to increasing processing speed and reliability of fingerprint recognition.*

Key words: *maximum matching, bipartite graph, images.*

INTRODUCTION

Some recognition tasks are reduced to maximum matching in bipartite graphs. Let $G=(V,E)$ be an input bipartite graph, maximum matching in this graph is a set $M \subseteq E$ of maximum size such that no two edges in M have a common vertex.

Algorithms [2–5] for finding maximum matching in non-weighted bipartite graphs are based on searching *M*- augmenting paths in these graphs. Algorithm of Kuhn [2] has complexity $O(mn)$ and consists of $O(n)$ phases hereinafter m,n are numbers of graph edges and vertices. Kim and Chwa's algorithm [3] is similar to [2], but the search for shortest M - augmenting paths is performed in parallel with complexity $O(n\log n\log\log n)$ using $O(n^3/\log n)$ processors. Hopcroft and Karp's algorithm [5] has lesser complexity $O(m\sqrt{n})$ in comparison with [2], because in each phase it searches for all disjoint shortest M - augmenting paths. When the bipartite graph is dense, an algorithm [4] with complexity $O(n^{1.5}\sqrt{m/\log n})$ may be preferable to use. The maximum matching problem in bipartite graphs can be reduced also to a maximum flow problem in graphs that can be solved in $O(m\sqrt{n})$ time using Dinic's algorithm [6].

Algorithms [7–11] for finding maximum matching in weighted bipartite graphs choose among all maximum matchings one having maximum cost, i.e. maximum sum of matching edges weights. Therewith Kuhn's algorithm [7–9] has complexity $O(mn^2)$ and Edmonds and Karp's algorithm [10] has complexity $O(mn+n^2\log n)$ using Fibonacci heap [11].

The purpose of the article is to consider a new task setting and algorithms for maximum matching in weighted bipartite graphs as well as usage of these algorithms in fingerprints recognition. Below target settings of maximum matching in bipartite graphs, two algorithms for maximum matching in weighted bipartite graphs in new setting and examples of usage these algorithms in fingerprints recognition are considered.

TARGET SETTINGS OF MAXIMUM MATCHING IN BIPARTITE GRAPHS

Before target settings consideration let us introduce the main definitions.

Definition1. A bipartite graph $G=(V=V1\cup V2,E)$ is a graph whose vertices can be divided into two disjoint sets $V1$ and $V2$ such that every edge connects a vertex in $V1$ to vertex in $V2$.

Definition2. A graph G is weighted if cost function $ce: E\to R$ is defined, that associates each edge with a real value.

Definition3. A subset $M\subset E$ is called a matching in graph G if no vertex is incident to more than one edge in M.

Definition4. A matching of maximum cardinality is called a maximum matching.

Definition5. A vertex in graph G is called M-free if it is incident with no edge in M.

Definition6. An M-augmenting path in G is a path whose original and terminal vertices are both M-free and whose edges are alternatively in $E\setminus M$ and M.

Definition7. Problem 1 — find one maximum matching in graph G.

Definition8. Problem 2 — among all maximum matchings, choose one having maximum cost.

Definition9. Problem 3 — among all matchings having maximum cost, choose one having maximum cardinality.

In this paper three target settings of maximum matching searching in bipartite graphs are considered, the first two of which are known. The first setting (problem1) consists in finding maximum matching M in non-weighted graph that can be done by finding all M-augmenting paths in this graph. After finding the next M-augmenting path p new matching with size $|M|+1$ is formed by excluding from previous matching all even edges and adding all odd edges in p, that means symmetric difference operation of sets M and $p: M = M \oplus p$. Not finding next M- augmenting path means finding of maximum matching because Berge's theorem [1] says that matching M in graph G is maximum, if and only if G contains no M-augmenting path.

Kuhn's algorithm [2] for finding maximum matching M in non-weighted graph checks for each M-free (free) vertex $v \in V1$ whether there is M-augmenting path in graph that originates in this vertex and increases size of matching if such path exists. This algorithm consists of $O(n)$ phases, in each of which only one M-augmenting path is searched and augmented. Therefore, the algorithm has complexity $O(mn)$. Hopcroft and Karp's algorithm [5] in each phase finds all M-augmenting shortest paths, performs lesser number of phases ($O\sqrt{n}$) in comparison with [2] and as a result has lesser complexity $O(m\sqrt{n})$.

The second setting (problem2) is due to the fact that some maximum matchings can be in graph and it is necessary to find among these matchings one having maximum cost. The third setting (problem3) is a new one and, in some sense, opposite to the second setting — it is necessary to find among all matchings having maximum cost one that is maximum in cardinality.

In this paper two algorithms for problem3 solution are proposed in the case when weight of each edge takes only one of two values, for example 0 and 1. Let MA be matchings in graph G, having maximal number of edges with weight that equals to 1. Proposed algorithms find in set MA those matching having maximum cardinality.

TWO ALGORITHMS FOR MAXIMUM MATCHING IN BIPARTITE GRAPH

Let $ce: E \to \{0,1\}$ be a cost function that associates each edge $(v_1 \in V1, v_2 \in V2)$ in bipartite graph $G = (V, E)$ with value that equals to 0 or 1. For example, cost function can be defined as $ce(v_1, v_2) = cv(v_1) \vee cv(v_2), (v_1, v_2) \in E$ or $ce(v_1, v_2) = cv(v_1) \wedge cv(v_2), (v_1, v_2) \in E$, if given cost function $cv: V \to \{0,1\}$ that associates each vertex with weight 0 or 1. Edge having weight that equals to 1 can be called basic or preferable. Stated above problem3 using such cost function ce is equivalent to following: find among all matchings having maximum number of basic edges one being maximum in cardinality. Let us consider two algorithms for solving this task. The first algorithm consists of the following five phases.

1. Subgraph $G1 \subset G$ is detected which consists of basic edges and vertices incident to these edges.

2. Searching of maximum matching $M1$ in subgraph $G1$ using Hopcroft and Karp's algorithm [5] is carried out.

3. Subgraph $G2 \subset G$ is formed by deletion in graph G matching $M1$ and all edges having common vertex with at least one edge from this matching.

4. Searching of maximum matching $M2$ in subgraph $G2$ is carried out.

5. Searching of maximum matching $M3$ in graph G is carried out with fulfillment of the following two conditions: 1) before searching $M3 = M1 \cup M2$ and 2) only those $M3$ - augmenting paths are searched and augmented, which do not reduce number of basic edges $e \in M3$.

Three following statements are valid.

Statement1. *Number of basic edges in any maximum matching M in graph G does not exceed size of matching $M1$.*

Statement2. *Conjunction of matchings $M1 \subset G1$ and $M2 \subset G2$ is also matching and $|M1|+|M2|$ does not exceed size of maximum matching M in graph G.*

Statement3. *Size of output matching $M3$ equals to size of maximum matching M in graph G if there is no $M3$ -augmenting path that decreases number of basic edges in $M3$ in final phase of the algorithm.*

Let us consider an example of the algorithm work in bipartite graph that consists of ten vertices and ten edges, six basic edges of which are marked by cross marks (Fig. 1, a). In the first phase of the algorithm subgraph $G1 \subset G$ is constructed (Fig. 1, b) that consists of six basic edges and eight vertices incident to these edges. After the first two phases of the algorithm [5] in this subgraph matchings that consist respectively of three (Fig. 1, b) and four (Fig. 1, c) edges (shown by thick lines in Fig. 1) are received. Graph $G2$ in this example is empty, that is why matchings $M1$ (Fig. 1, c) and $M3$ (Fig. 1, d) are the same.

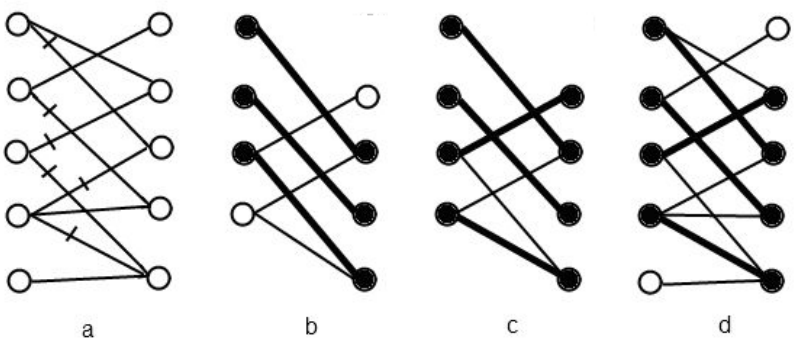

Fig. 1. Input graph G (a), matchings in graph $G1$ after the first (b) and the second (c) phases, matching $M3$ in graph G as result of the algorithm (d).

The second algorithm is modified version of Hopcroft and Karp's algorithm. The algorithm begins with empty matching M and in turn increases this matching in each of the next phases. In the next i-th phase all M-augmenting vertex-disjoint paths having the length that equals to i are searched. These paths in preference increase and at least do not reduce number of basic edges in M. After finding next M-augmenting path p operation $M = M \oplus p$ is carried out which increases $|M|$ and changes set of basic edges $M1 \subset M$ if path p contains these edges. The algorithm's structure is presented below.

Input. Bipartite graph $G = (V, E)$, cost function $ce : E \to \{0,1\}$, subset of basic edges $\{e \in E \mid ce(e) = 1\} \subseteq E$, matching $M = 0$, set of M-augmenting paths $P = 0$.

Output. Matching $M \subseteq E$ that contains maximum number of basic edges.

Repeate. Find $(P_1, P_2, ..., P_k)$, where $P_i, i = 1,2,...,k$ is a set of all vertex-disjoint M-augmenting paths, such that each path $p \in P_i$ has length i and carrying out $M = M \oplus p$ increases or at least not decreases number of basic edges in matching M.

$P = (P_1 \cup P_2 \cup ... P_k)$

$M = M \oplus P$

Until $P \neq 0$.

Let us consider the implementation of the algorithm in more detail. Let H be layered directed acyclic graph; L_i — set of vertices in i-th layer in graph H; t — number of final layer in H; $in(w), w \in H$ — number of input edges in vertex w; $nmax = |M|$ — size of matching M; $n_2(p), n_1(p)$ — numbers of basic edges in path $p \in H$, thereafter covered and not covered by matching M; $dif(p) = n_1(p) - n_2(p)$ — weight of path p; $mdif(w), w \in H$ — maximal weight of paths terminated in vertex w.

Each phase of the algorithm consists of the following two parts: breadth-first (BFS) and depth-first search (DFS). Breadth-first search constructs graph H in which vertices of input graph G are located in layers. Therewith initial layer contains only free vertices $v \in V1$. All the following layers are formed by alternate adding either vertices $v \in V2$, connected with vertices of preceding layer by edges $e \notin M$, or vertices $v \in V1$, connected with vertices of preceding layer by edges $e \in M$. In the process of BFS unlike [5] values $mdif(w), w \in H$ are defined using dynamic programming. Graph H construction is carried out up to one of the following conditions is valid: either next layer contains free vertex $v \in V2$ or is empty (stopping of algorithm).

Let us consider an example of the algorithm work in input bipartite graph all ten edges of which are basic (Fig. 2, a). Matching consisting of three edges is received after the first phase of the algorithm work (Fig. 2, b). Four-layer graph H, constructed in the second phase, is presented in Fig. 2,c in which graph vertices are numerated (1–10), free vertices are shown by circles and other vertices (i.e. covered by matching) — by black spheres. The edges in graph H that belong to M - augmenting paths are shown by thick lines (Fig. 2, c). In result of the sec-

ond phase maximum matching consisting of five edges is received (Fig. 2, d).

DFS finds all vertex-disjoint paths in graph H which have length t and do not reduce the number of basic edges in matching M. Searching of each path starts in free vertex u of the last layer given $mdif(u) >= 0$ and propagates per edges to non-used during DFS vertices in the preceding layer. Ending this search in free vertex of initial layer means that M-augmenting path is traced and will be thereafter augmented.

In the example, presented in Fig. 2, graph G has only basic edges and all paths in graph H terminated in free vertices six or seven have the same weight $dif(p) = 1$ (Fig. 2, c). There are four such paths but only two of them have no common vertices and are augmented.

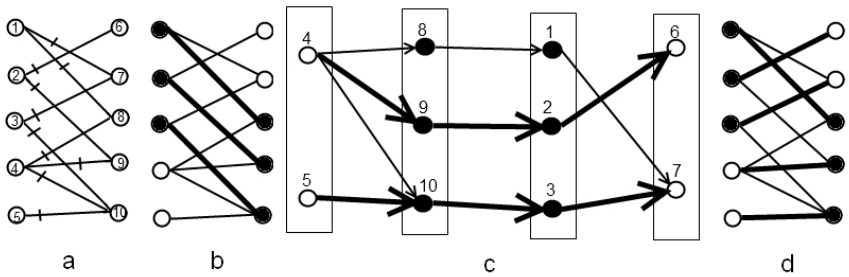

Fig. 2. Input graph G (a), matching (three edges) afterwards the first phase of algorithm (b), four-layered graph H in the second phase (c) and maximum matching (d) after two phases.

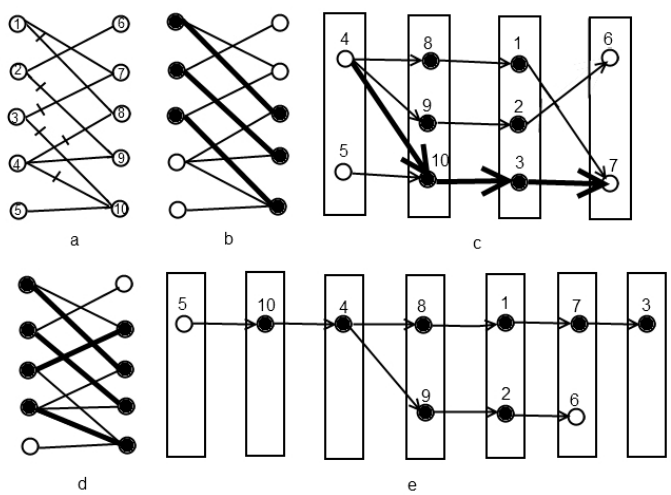

Fig. 3. Input graph G (a), matching (three edges) afterwards the first phase of algorithm (b), graph H in the second phase (c), output matching (d) as the result of the second phase and seven-layered graph H in the third phase (e), that contains no M-augmenting paths.

In the next example presented in Fig. 3 only six edges in graph G are basic. There are four paths in the second phase (Fig. 3, c) having free vertices in terminal points (4-10-3-7, 4-8-1-7, 4-9-2-6 and 5-10-3-7) and path (4-10-3-7) has maximal weight that equals to 1. Other three ways (with weights 0, -1, 0) have common vertices with this path and therefore are not used. There is only one path in graph H in the next (third) phase having free terminal vertex (Fig. 3, e). The weight of this path is equal to -2 and it is not used because its augmentation will decrease number of basic edges in matching. Graph H in this phase has seven layers because the next eighth layer is empty. Output result of algorithm's work is the matching that consists of four edges (Fig. 3, d).

The algorithm may be expressed in the following pseudocode.

Repeate
$t = 0$; BFS;
if ($t > 0$)DFS;
Until $t > 0$

BFS - Constructing graph H by breadth-first search:

1. Add to L_0 all free vertices from $V1$. Let $i = nmax = 0$, $mdif(w) = 0 \mid w \in L_0$;

2. Form next layers of graph H until recurrent layer either is empty or contains free vertices $u \in V2$:

Repeate
for (all vertices $u \in L_i$)
for (all vertices w adjacent to u using unmatched edges)
if (no layer containing w)
add w to L_{i+1}; let $mdif(w) = mdif(u)$, $in(w) = 1$;
else if ($w \in L_{i+1}$)
let $mdif(w) = max(mdif(w), mdif(u) + ce(u,w))$,
$in(w) = in(w) + 1$;
if (L_{i+1} contains free vertex and $mdif(u) >= 0$) let $t = i + 1$;
else if (L_{i+1} is empty) $t = -t$;
else
for (all vertices $u \in L_{i+1}$)
for (all vertices w adjacent to u using matched edges)
if (no layer containing w)
add w to L_{i+2}; let
$mdif(w) = mdif(u) - 1$;
$mdif(w) = max(mdif(w), mdif(u) - ce(u,w))$;
let $i = i + 2$;
Until ($t = 0$)

DFS - searching and augmentation of M -augmenting paths in graph H :

for (all free vertices $u \in L_{i+1}$, ordered by value $mdif(u)$)

if (there is path p from $u \in L_{i+1} \mid mdif(u) \geq 0$ to one of free vertices $w \in L_0$)

delete each vertex v and its outgoing edges in p. Under deletion each edge $(v,vout)$ let $in(vout) = in(vout) - 1$ (if $in(vout) = 0$, delete $vout$ and its outgoing edges). Delete from M all edges $e \in p$ and add other edges $e \notin M$ from this path. Let $nmax = nmax + 1$.

The algorithm finds matching M in graph G that contains maximum number of basic edges. This matching is maximum if in final phase there are no M -augmenting paths augmentation of which reduces number of basic edges in M. Proposed algorithms have complexity $O(m\sqrt{n})$.

MAXIMUM MATCHING IN FINGERPRINT RECOGNITION

Algorithms for comparing fingerprints based on detection and matching of minutiae (papillary lines ends and bifurcations) are considered to be still the best, with the highest matching capability. Using these algorithms a simple rigid transformation (translating, rotating and scaling) is searched at first to align the input fingerprint Q with the template fingerprint T, each represented by its minutiae pattern. After aligning similarity measure or binary decision of whether two fingerprint images are from the same finger or not must be determined and returned. In two roughly aligned images every two minutiae $(q \in Q, p \in T)$ may be considered as matched (corresponding with one another) only if their distance (location and angle) is in tolerance box under known transformation. Not only pairs but often overlapped groups of closely located minutiae occur, that is why finding true minutia matching usually is not a simple task. In [12, 13] matching is searched using rather simple locally optimal algorithm, which may not so often find true matching of minutiae because of its locality. In [14] Ford-Fulkerson algorithm for finding optimal global matching is used provided by T. Cormen et.al. [15]. This algorithm finds the maximum flow in non weighted bipartite graph with complexity $O(nm)$, vertices $v \in V1$ ($v \in V2$) of which correspond to minutiae in the first (second) image and each edge — to pair of vertices closely spaced after aligning (registration). In [16] finding of minutiae matching is based on searching of matching with minimal cost in weighted bipartite graph using algorithm [17] with complexity $O(n^3)$. Algorithms [14, 16] have worse complexity in comparison with [5] that restricts their using in praxis. Another possible drawback of algorithms [12–14] is that they do not use validity data of minutiae which may be formed in process of their detection in fingerprints. In the issue spurious minutiae, having the same influence as true minutiae, may lead to errors in recognition.

One of challenges for [12–14, 16] and other minutiae-based algoritms consists in missing and spurious minutiae in fingerprint images. To reduce effect of these minutiae we propose to classify minutiae in two types (reliable and doubtful) and use weighted bipartite graph $G = (Q \cup T, E)$ with cost function $ce : E \to \{0,1\}$,

where Q and T are sets of minutiae in input I_Q and template I_T images. Therewith weight of edge $e(q \in Q, p \in T)$ equals to 1 (reliable edge) if the following two conditions are satisfied: 1) minutiae corresponding to vertex q is considered to be reliable, because it is detected using comparatively rigorous rules and 2) distance (coordinates and angle) of minutiae q and p after aligning is in certain tolerance box. Minutiae in template image T are considered as reliable.

Maximum matching M_{QT} in bipartite graph can be searched using one of two proposed algorithms. Similarity (matching) degree of two compared images (0–100%) can be defined as follows:

$$sim(I_Q, I_T) = 100(kne_1 + ne_2)/n, \quad (1)$$

where coefficient $1 \le k \le 2$; ne_1, ne_2 are numbers of edges in M_{QT}, having weight 1 and 0; n is total number of minutiae in intersection region of these two images.

In Fig. 4 an example of closely spaced minutiae set in images I_Q (four points) and I_T (two points) after registration is shown: reliable (doubtful) minutiae are marked by circls (squares) and their angles are shown by arrows. All these minutiae fall into the same tolerance box shown in Fig. 4 by big circle. That is why every two minutiae in this set can be matched but the task is to find optimal matching. Similarity degree of this minutiae set defined by (1) takes the following values depending on choosing minutiae matched pairs: 66% (two pairs of matched reliable minutiae: $(q_2, p_1),(q_3, p_2)$), 50% ($(q_1, p_1),(q_2, p_2)$) and 33% ($(q_1, p_1),(q_4, p_2)$). Therewith maximal similarity (66%) corresponds to preferential matching of reliable minutiae using proposed algorithms. In general tolerance boxes for reliable and doubtful minutiae may have different size. Searching maximal similarity degree depending on reliability of matched minutiae seems to be positive feature in comparison with [12–14, 16].

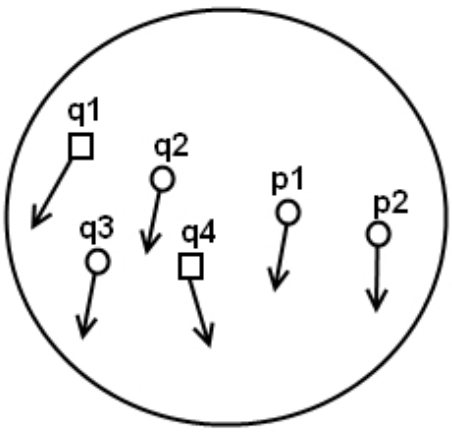

Fig. 4. An example of closely spaced minutiae after registration in two compared fingerprint images I_Q (four points q) and I_T (two points p).

The second example corresponds to removing of noise gaps in lines and curves in vectorized image, in particular fingerprint image. Vectorized image represents graph vertices of which correspond to ends and crossings of lines and edges — to line or curve segments in image. Errors in gaps removing lead to spurious minutiae and hereupon to errors in fingerprints recognition. Task of gaps removing can be reduced to searching of maximum matching in weighted bipartite graph vertices of which correspond to ends of lines in image. Each edge in this graph connects two vertices if distance (location and angle) of corresponding end points is in certain tolerance box b. Therewith edge has weight that equals to 1 (basic edge) if distance of corresponding end points is in tolerance box, size of which is essentially smaller than size of b and equals to 0 otherwise. Searching of maximum matching in this graph can be carried out using one of proposed above algorithms.

Examples indicate that using of the proposed algorithms leads to increasing processing speed and reliability of fingerprint recognition. Testing software for fingerprints recognition using developed algorithms will be the next step of research.

CONCLUSIONS

New setting for finding maximum matching in bipartite graph which set of edges is splitted into two subsets is proposed: find maximum matching in this graph among all matchings having maximum cardinality in one of these subsets. Two algorithms for searching maximum matching in this setting are also proposed, which are based on using Hopcroft and Karp's algorithm and have complexity $O(m\sqrt{n})$. These algorithms can be modified by splitting set of edges in bipartite graph more than in two subsets. Preliminary analysis indicates that most likely using of proposed algorithms leads to the increasing processing speed and reliability of fingerprint recognition.

REFERENCES

1. C. Berge. Two theorems in graph theory. *In Proc. National Academy Sciences, USA*. 1957. P. 842–844.
2. J.A. Bondy and U.S.R. Murty. Graph theory wih applications. *Mac Millan, New York*, 1976.
3. T. Kim and K.Y. Chwa. An $O(n \log n \log\log n)$ parallel maximum matching algorithm for bipartite graphs. *Inf. Proc. Letters*. 1987. 24(1), P.15–17.
4. H. Act, N.Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Inf. Proc. Letters*. 1991. 37, P. 237–240.
5. J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal Comput*. 1973. 2(4), P. 225–231.
6. E.A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl*. 1970. 11(5). P. 1277–1280.
7. H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist., Quart*. 1955. 2. P. 83–97.
8. H.W. Kuhn. Variants of the Hungarian method for the assignment problem. *Naval Res. Logist., Quart.* 1956. 3. P. 253–258.
9. J. Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math*. 1957, P. 32–38.

10. J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. of the Assoc. for Comput. Mach*. 1972. 19(2), P. 248–264.
11. M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in imroved network optimization algorithms. *In 25th FOCS*. 1984. P. 338–346.
12. H.V. Gasparian, A.A. Kirakosian. The comparison system of fingerprints by local features. *Vestnik of RAU*, *Natural Science*, *Physics and Mathematics*. 2006. P. 85–91. (in Russian).
13. A.S. Rykanov. Analysis of fingerprint authentification and verification methods. *Systems for information processing*. 6(87). 2010. P. 164–181. (in Russian).
14. Chengfeng Wang, Marina Gavrilova, Yuan Luo and Jon Rokne. An efficient algorithm for fingerprint matching. *ICPR*. 1. 2006. P. 1034–1037.
15. T. Cormen, C. Leiserson, R. Rivest and C. Stein. Introduction to algorithm. The MIT Press, 2002.
16. V.M. Kyyko, V.V. Matsello. Fingerprints recognition based on searching of corresponding points. *Control systems and machines*. № 3. 2005. P. 36–41 (in Russian).
17. R. Jonker R., A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing 38*. 1987. P. 325–340.

Received 24.11.2017

ЛИТЕРАТУРА

1. C. Berge. Two theorems in graph theory. *In Proc. National Academy of Sciences, USA*. 1957. pp. 842–844,
2. J.A. Bondy and U.S.R. Murty. Graph theory wih applications. *Mac Millan, New York*, 1976.
3. T. Kim and K.Y. Chwa. An $O(n \log n \log\log n)$ parallel maximum matching algorithm for bipartite graphs. *Inf. Proc. Letters*, 24(1), pp.15–17, 1987.
4. H. Act, N. Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m / \log n})$, *Inf. Proc. Letters*, 1991. 37, pp. 237–240.
5. J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal Comput.* 2(4), 1973. pp. 225–231,
6. E.A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl*. 1970. 11(5). P. 1277–1280.
7. H.W. Kuhn. The Hungarian method for the assignment problem. Naval *Res. Logist., Quart.* 1955. 2, pp. 83–97,
8. H.W. Kuhn. Variants of the Hungarian method for the assignment problem. *Naval Res. Logist. Quart.* 1956. 3, pp. 253–258,
9. J. Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math.* 1957. pp. 32–38,
10. J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. of the Assoc. for Comput. Mach.* 19(2), 1972. pp. 248–264.
11. M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in imroved network optimization algorithms. *In 25th FOCS*, 1984. pp. 338–346.
12. Гаспарян А.В., Киракосян А.А. Система сравнения отпечатков пальцев по локальным признакам. *Вестник РАУ. Серия физико-математические и естественные науки*. 2006. С. 85–91.
13. Рыканов А.С. Анализ методов распознавания отпечатков пальца. *Системи обробки інформації*. 2010. Вип. 6 (87). С. 164–171.
14. Chengfeng Wang, Marina Gavrilova, Yuan Luo and Jon Rokne. An efficient algorithm for fingerprint matching. *ICPR*, 1. 2006. P. 1034–1037.
15. T. Cormen, C. Leiserson, R. Rivest and C. Stein. Introduction to algorithm. The MIT Press, 2002.
16. Кийко В.М., Мацелло В.В. Сравнение изображений отпечатков пальцев на основе поиска соответствия особых точек. Управляющие системы и машины, 2015, 3, стр. 36–41.
17. R.Jonker R., A.Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing 38, 1987, pp. 325–340.

Получено 24.11.2017

В.М. Кийко, канд. техн. наук, старш. наук. співроб.
відд. розпізнавання образів
e-mail: vkiiko@gmail.com
Міжнародний науково-навчальний центр інформаційних
технологій та систем НАН України і МОН України,
пр. Акад. Глушкова, 40, м. Київ, 03187, Україна

ЗНАХОДЖЕННЯ НАЙБІЛЬШОГО ПАРОСПОЛУЧЕННЯ НА ЗВАЖЕНОМУ ДВОДОЛЬНОМУ ГРАФОВІ

Вступ. Розглянуто найвідоміші алгоритми пошуку найбільшого паросполучення на дводольному графові, які або виконують пошук найбільшого паросполучення на незваженому дводольному графові, або вибирають із множини найбільших паросполучень одне, яке має найбільшу суму вагів ребер. Ці алгоритми активно вживаються при розв'язанні різних оптимізаційних задач, але на практиці є також потреба в інших постановках і алгоритмах пошуку найбільшого паросполучення на дводольних графах.

Мета статті — розглянути нові постановки завдання щодо знаходження найбільшого паросполучення на зваженому дводольному графові, алгоритми розв'язання цієї задачі, а також використання цих алгоритмів при розпізнаванні папілярних зображень.

Методи. Використовуються модифіковані версіїї пошуку найбільшого паросполучення M на дводольному графові на основі пошуку та аугментації M - збільшуючих шляхів на цьому графові.

Результати. Розглянуто нову постановку пошуку найбільшого паросполучення на зваженому дводольному графові, ваги ребер якого набувають два значення (наприклад, 0 і 1): знайти всі паросполучення, що мають найбільшу кількість ребер з вагою 1, і вибрати серед них таке паросполучення, що має найбільшу кількість ребер. Запропоновано два алгоритми пошуку найбільшого паросполучення у цій постановці зі складністю $O(m\sqrt{n})$. Розглянуто приклади вживання цих алгоритмів для усунення розривів ліній та пошуку відповідності особливих точок на порівнюваних відбитках пальців при розв'язанні задачі розпізнавання папілярних зображень.

Висновки. Запропоновано нові алгоритми пошуку найбільшого паросполучення на зваженому дводольному графові. Використання запропонованих алгоритмів призводить до пришвидчення та зростання надійності розпізнавання зображень відбитків пальців.

Ключові слова*: найбільше паросполучення, дводольний граф, зображення.*

В.М. Кийко, канд. техн. наук, старш. науч. сотр.
отд. распознавания образов
e-mail: vkiiko@gmail.com
Международный научно-учебный центр информационных
технологий и систем НАН Украины и МОН Украины,
пр. Академика Глушкова, 40, г. Киев, 03187, Украина

НАХОЖДЕНИЕ НАИБОЛЬШЕГО ПАРОСОЧЕТАНИЯ
НА ВЗВЕШЕННОМ ДВУДОЛЬНОМ ГРАФЕ

Рассмотрена новая постановка задачи нахождения наибольшего паросочетания на взвешенном двудольном графе, веса ребер которого принимают два значения (например, 0 и 1): найти все паросочетания, содержащие наибольшее количество ребер с весом 1, и выбрать среди них наибольшее по размеру паросочетание. Предложены два алгоритма поиска наибольшего паросочетания на двудольном графе в новой постановке со сложностью $O(m\sqrt{n})$. Рассмотрены примеры применения этих алгоритмов для устранения разрывов линий и поиска соответствия особых точек на двух сравниваемых отпечатках пальцев при решении задачи распознавания папиллярных изображений.

Ключевые слова: *наибольшее паросочетание, двудольный граф, изображения*.