An Extended Path Following Algorithm for Graph-Matching Problem

Zhi-Yong Liu, Hong Qiao, *Senior Member*, *IEEE*, and Lei Xu, *Fellow*, *IEEE*

Abstract—The path following algorithm was proposed recently to approximately solve the matching problems on undirected graph models and exhibited a state-of-the-art performance on matching accuracy. In this paper, we extend the path following algorithm to the matching problems on directed graph models by proposing a concave relaxation for the problem. Based on the concave and convex relaxations, a series of objective functions are constructed, and the Frank-Wolfe algorithm is then utilized to minimize them. Several experiments on synthetic and real data witness the validity of the extended path following algorithm.

Index Terms—Graph matching, convex relaxation, concave relaxation, directed graph, PATH following algorithm.

1 Introduction

Graph matching plays a central role in many graph-based techniques, such as classification in structural pattern recognition, where patterns are represented by graph models [1], and identification of pairs of homologous proteins, where proteins of each species are organized by a graph [2]. Given two graphs $G_D = (V_D, E_D)$ and $G_M = (V_M, E_M)$, where V and E, respectively, denote the sets of vertices and edges, matching between them is to find a one-to-one mapping $f : V_D \to V_M$ such that some fitness function is minimized. The fitness function can be defined in different ways, such as on the edit distance [3] and Frobenius norm of the difference between adjacency matrices [4]. The problem is, in nature, an NP-hard combinatorial optimization problem with factorial complexity, except for some graph models with special structure, of which a typical example is the planar graphs, which have been shown to be of polynomial complexity [5].

To make the problem computationally tractable, many approximate approaches have been proposed in the last three decades, trying to seek an acceptable tradeoff between the complexity and matching accuracy. The approximate approaches can be categorized from different perspectives, for which an extensive review is referred to [6], for instance. In this paper, we focus on one group of approximate techniques that works directly on the adjacency matrix of the graphs to be matched and involves a relaxation of the combinatory optimization to be a continuous one [7], [8], [4]. With the help of the adjacency matrix, the matching problem can be formulated as identifying a permutation matrix that minimizes some cost function as follows:

$$P = \arg\min_{P\in\mathcal{P}} f\big(A_D, A_{P(M)}\big), A_{P(M)} = PA_MP^T, \tag{1}$$

- Z.-Y. Liu and H. Qiao are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.
E-mail: {zhiyong.liu, hong.qiao}@ia.ac.cn.
- L. Xu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong.
E-mail: lxu@cse.cuhk.edu.hk.

Manuscript received 10 Aug. 2011; revised 19 Dec. 2011; accepted 18 Jan. 2012; published online 30 Jan. 2010.
Recommended for acceptance by D. Weinshall.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-08-0539.
Digital Object Identifier no. 10.1109/TPAMI.2012.45.

where $\mathcal{P}$ denotes the set of permutation matrices, A the adjacency matrix, and $A_{P(M)}$ the permuted adjacency matrix of A_M by P. To avoid the combinatory nature of the problem, one typical approach is to relax the domain of the problem from $\mathcal{P}$ to its convex hull, i.e., the set of doubly stochastic matrices denoted by $\mathcal{D}$. However, optimization of the relaxed problem generally results in a doubly stochastic matrix instead of a permutation one. To finally solve the matching problem, a further step is usually needed to project the doubly stochastic matrix P back to a permutation matrix X. An intuitive approach may resort to solving the following maximal linear assignment problem:

$$X = \arg\max_{X\in\mathcal{P}} \langle X, P\rangle, \tag{2}$$

typically with the help of the Hungarian algorithm [9]. Such a hard-cut type of assignment may, however, introduce a significant additional error. By contrast, starting from some appropriate $P \in \mathcal{D}$, the graduated assignment [8] and PATH following [4] algorithms can converge to a permutation matrix in a smooth way. The graduated assignment algorithm approaches a permutation matrix by increasing a parameter which controls the nonconvexity of the problem, and as the parameter is increased to be large enough, the algorithm finally gets a permutation matrix. On the other hand, a concave relaxation of the original graph-matching problem was introduced by the PATH following algorithm, and by linearly combining the concave and convex relaxations, the PATH following algorithm provides a gradual nonconvexity-like algorithm to minimize the concave relaxation which holds the same minima as the original matching problem. For matching problems on undirected graphs, the PATH following algorithm exhibited an excellent performance on matching accuracy.

However, the PATH following algorithm cannot be used to solve the matching problem on directed graph models since the concave relaxation proposed in [4] can no longer be guaranteed to be concave. In this paper, we propose a concave relaxation for the directed graph model, and thus extend the PATH following algorithm to cover such a type of graphs. A brief review of the PATH following algorithm is given in Section 2. The proposed method is described in detail in Section 3, followed by some experimental demonstrations given in Section 4. Section 5 finally concludes the paper.

2 A Brief Review of PATH Following Algorithm

The PATH following algorithm adopts the Frobenius matrix norm as the cost, and the graph-matching problem is formulated as follows:

$$\begin{aligned} P &= \arg\min_{P\in\mathcal{P}} f(P),\\ f(P) &= \| A_D - PA_MP^T \|_F^2 = \| A_DP - PA_M \|_F^2 \\ &= \mathrm{vec}(P)^T Q \mathrm{vec}(P), \end{aligned} \tag{3}$$

where $\mathrm{vec}(P)$ is the columnwise vector representation of P, and

$$Q = \big(I \otimes A_D - A_M^T \otimes I\big)^T \big(I \otimes A_D - A_M^T \otimes I\big) \in \mathbb{R}^{N^2\times N^2} \tag{4}$$

is a symmetric positive definite matrix, except for the case of graph isomorphism, which makes Q positive semidefinite. Actually, in the algorithm it is unnecessary to utilize the large-size matrix Q in an explicit way.

By relaxing the domain of the problem from $\mathcal{P}$ to $\mathcal{D}$, a convex relaxation of the original objective function is given by

$$F_{vex}(P) = \mathrm{vec}(P)^T Q \mathrm{vec}(P), P \in \mathcal{D}. \tag{5}$$

Once the global minimum of (5) has been found, which is generally a doubly stochastic matrix, a permutation matrix can be gotten by

(2) to solve the matching problem. Such a graph-matching algorithm is referred to as QCV (convex optimization).

Meanwhile, a concave relaxation is figured out on undirected graphs without self-loops as

$$F_{ucav}(P) = -\mathrm{tr}(\triangle P) - 2\mathrm{vec}(P)^T\left(L_M^T \otimes L_D^T\right)\mathrm{vec}(P), P \in \mathcal{D}, \tag{6}$$

where $\triangle_{ij} = (R_{Mii} - R_{Djj})^2$,[1] with R and L denoting the degree and Laplacian matrices of the graph, respectively. The concave relaxation holds the same minima as the original matching problem, i.e.,

$$\min_{P\in\mathcal{P}} F_{ucav}(P) = \min_{P\in\mathcal{P}} F_0(P). \tag{7}$$

Finally, a series of objective functions is constructed as follows to form the PATH following algorithm:

$$F_\gamma(P) = (1-\gamma)F_{vex}(P) + \gamma F_{ucav}(P), P \in \mathcal{D}, \tag{8}$$

where $\gamma \in [0,1]$. The algorithm starts by setting $\gamma = 0$, which makes the objective functions degenerate to the convex relaxation. By gradually increasing γ, the objective becomes generally neither concave nor convex, and the PATH algorithm is to find a local minimum of $F_{\gamma+d\gamma}$ initialized from the local minimum found by minimizing F_γ, by the Frank-Wolfe algorithm. As γ becomes large enough (may not be necessary to be one) to make F_γ a concave function, minimization of F_γ will finally push P to a permutation matrix.

It was shown in [4] that, for the matching problems between two undirected graphs, the PATH following algorithm exhibited a state-of-the-art performance on matching accuracy. It is also noted that though the global minimum of the concave relaxation is the same as the original matching problem, the final result obtained by the PATH following algorithm can be far away from the ground truth solution, due to the large number of local minimums (up to N!) and its local search nature. On the other hand, though (6) is still a relaxation for directed graphs without self-loops, it can no longer be guaranteed to be concave. Below, we will propose a concave relaxation for directed graphs without self-loops, and thus extend the path following algorithm to cover such a type of graph model.

3 Proposed Method

3.1 A Concave Relaxation for Directed Graphs without Self-Loops

For directed graphs without self-loops, $F_{ucav}(P)$ given by (6) still remains a relaxation of the original objective function (3) since the difference between them is a constant with respect to $P \in \mathcal{P}$ [4]. However, (6) cannot guarantee to be a concave function because the term $(L_M^T \otimes L_D^T)$ is not necessarily positive semidefinite for asymmetric L_M^T and L_D^T even if both of them are positive semidefinite [10]. Here, we propose a concave relaxation for directed graphs without self-loops as follows:

$$\begin{aligned} F_{dcav}(P) &= -\mathrm{tr}(\triangle P) - \mathrm{vec}(P)^T\Phi\mathrm{vec}(P), \\ \Phi &= L_M^T \otimes L_D^T + L_M \otimes L_D + \sigma I, P \in \mathcal{D}, \end{aligned} \tag{9}$$

where $\sigma = c - \varsigma$ with c being a small positive constant, and ς, typically a negative number, the smallest eigenvalue of the symmetric matrix

$$L = L_M^T \otimes L_D^T + L_M \otimes L_D. \tag{10}$$

To justify the validity of the concave relaxation, it is necessary to check, first, its concavity and, second, whether $F_{dcav}(P)$ satisfies (7). The fact that all of the eigenvalues of the real symmetric matrix Φ are positive implies that Φ is a positive definite matrix, which guarantees that (9) is a concave function. Then, we justify the second point as follows:

$$\begin{aligned} &\min_{P\in\mathcal{P}} F_{dcav}(P) \\ &= \min_{P\in\mathcal{P}} \left[-\mathrm{tr}(\triangle P) - \mathrm{vec}(P)^T\left(L_M^T \otimes L_D^T + L_M \otimes L_D + \sigma I\right)\mathrm{vec}(P)\right] \\ &= \min_{P\in\mathcal{P}} \left[-\mathrm{tr}(\triangle P) - 2\mathrm{vec}(P)^T\left(L_M^T \otimes L_D^T\right)\mathrm{vec}(P)\right] - \sigma N \\ &= \min_{P\in\mathcal{P}} F_{ucav}(P) = \min_{P\in\mathcal{P}} F_0(P), \end{aligned} \tag{11}$$

thanks to the fact that

$$\mathrm{vec}(P)^T\left(L_M^T \otimes L_D^T\right)\mathrm{vec}(P) = \mathrm{vec}(P)^T(L_M \otimes L_D)\mathrm{vec}(P).$$

Therefore, $F_{dcav}(P)$ is a concave relaxation of the original matching problem.

Finally, based on the concave relaxation given by (9) and the convex relaxation given by (5), a series of objective functions for the matching problem on directed graphs without self-loops is given as follows:

$$F_\gamma(P) = (1-\gamma)F_{vex}(P) + \gamma F_{dcav}(P), P \in \mathcal{D}. \tag{12}$$

In practice, to prevent the concave term from increasing too quickly as γ increases, we may choose a normalized concave relaxation to construct the objective function as

$$F_\gamma(P) = (1-\gamma)F_{vex}(P) + \gamma F_{dcav}(P)/\sigma, P \in \mathcal{D} \tag{13}$$

because σ in $F_{dcav}(P)$ may be quite large, especially for large-size graph models.

A simple illustration of the changes of $F_\gamma(P)$ with respect to γ and P is shown by Fig. 1, which witnesses how the objective function changes smoothly from a convex function to a concave one as γ increases from 0 to 1.

Though the concave relaxation is proposed for the directed graphs without self-loops, it can be extended to the graphs with self-loops by adding just a linear term. Specifically, a symmetric score matrix $C \in \mathbb{R}^{N\times N}$ is first constructed as follows:

$$C_{ij} = (A_{Dii} - A_{Mjj})^2. \tag{14}$$

Then, the matching cost between the edges belonging to self-loops can be figured out as

$$F_s(P) = \mathrm{tr}(PC), P \in \mathcal{P}. \tag{15}$$

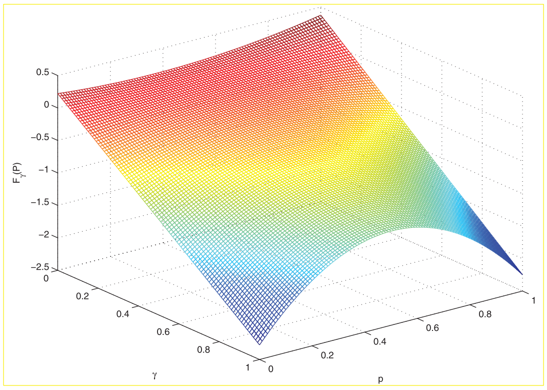

Fig. 1. Changes of the objective function $F_\gamma(P)$ with respect to γ and p with P given by $P = [p, 1-p; 1-p, p]$, where $A_D = [0, 0.1453; 0.3920, 0]$ and $A_M = [0, 0.7561; 0.4454, 0]$.

1. The index i and j should be exchanged in [4].

Thus, similarly to the labeled graph-matching problem discussed in [4], the objective functions for the directed graphs with self-loops can be constructed as follows:

$$F_\gamma(P) = \mathrm{tr}(PC) + (1-\gamma)F_{vex}(P) + \gamma F_{dcav}(P)/\sigma, P \in \mathcal{D}. \quad (16)$$

Consequently, with the two concave relaxations given by (6) and (9), we can actually tackle the matching problems between any types of graph models as long as they can be represented by a real adjacency matrix.

3.2 Estimation of σ

The parameter $\sigma = c - \varsigma$ in (9) needs to be specified. It is usually computationally heavy to figure out the smallest eigenvalue (ς) of the large-size ($N^2 \times N^2$) matrix L. Fortunately, to make (9) a concave relaxation, it is unnecessary to get the exact value of ς, but we just need to estimate one of its lower bounds. A lower bound of the smallest eigenvalue λ_{min} of a real symmetric matrix M whose entries are within the interval $[a, b]$ is given as follows [11]:

$$\lambda_{min}(M) \geq \begin{cases} \begin{cases} n(a-b)/2 & \text{if } n \text{ is even}, \\ \left(na - \sqrt{a^2 + (n^2-1)b^2}\right)/2 & \text{otherwise}, \end{cases} & \text{if } |a| < b, \\ na & \text{otherwise}, \end{cases} \quad (17)$$

where n denotes the size of matrix M and a, b the smallest and largest element of M. It is still computationally demanding ($O(N^4)$) to find the smallest and largest element of the matrix L. To alleviate the computational load, we further relax the value interval $[a, b]$ to be a looser one as follows:

$$\begin{aligned} a &= 2 \times \text{min\_element\_of}\{L_M \otimes L_D\}, \\ b &= 2 \times \text{max\_element\_of}\{L_M \otimes L_D\}, \end{aligned} \quad (18)$$

where the operators $min\_element\_of\{\cdot\}$ and $max\_element\_of\{\cdot\}$ denote finding the minimal and maximal element of the matrix, respectively. Then, to calculate a and b, we just need to find the smallest and largest element of L_M and L_D. Specifically, denoting the smallest and largest elements of L_M and L_D as ζ_M, η_M and ζ_D, η_D respectively, where ζ and η should be negative and positive, respectively, as the weights of the graph edge are usually set as real positive numbers (if not (19) should be modified accordingly), a and b are given by

$$a = 2 \times \min\{\zeta_M \eta_D, \eta_M \zeta_D\}, \; b = 2 \times \max\{\zeta_M \zeta_D, \eta_M \eta_D\}. \quad (19)$$

Though any lower bound of ς can certainly give a concave relaxation by (9), it is still interesting to analyze how the lower bound will affect the performance of the algorithm. Intuitively, a tighter bound might be preferred since a too large σ makes L relatively trivial in the function. An extreme case is as $\sigma \to \infty$, which makes the concave relaxation $F_{dcav}/\sigma \to \mathrm{vec}(P)^T \mathrm{vec}(P)$.

Actually, based on the lower bound provided by (17), we can get a tighter bound for ς by constructing a series of positive definite matrices. Specifically, by partitioning the positive definite matrix

$$\Phi_n = L + \sigma I = \begin{bmatrix} \Phi_{n-1} & d \\ d^T & b \end{bmatrix},$$

where $\sigma = c - \varsigma$ with ς given by (17), the lower bound of the smallest eigenvalue of Φ_n is given by [12]

$$l_n = \frac{1}{2}\left(b + l_{n-1} - \sqrt{(b - l_{n-1})^2 + 4d^T d}\right) \leq \lambda_{min}(\Phi_n), \quad (20)$$

where l_{n-1} denotes the lower bound of Φ_{n-1}. It is not difficult to check that all of $\Phi_{n-1} \cdots \Phi_1$ are symmetric positive definite matrices thanks to the rather general lower bound given by (17). Thus, we can iteratively find l_n starting from l_1. Based on l_n, we can get a tighter lower bound of ς by $\varsigma \geq l_n - \sigma$.

The process involves iterating $N^2 - 1$ times of (20), which involves, on average, a $O((N^2+1)/2)$ complexity. Thus, the total computational complexity is $O(N^4)$, and meanwhile, it in general requires a large storage ($O(N^4)$). Moreover, as will be shown by the experimental results later, the tighter bound just outperforms the loose one very slightly (< 1 percent on average). Thus, in practice we may still prefer to use (17) as the estimation to avoid a heavy computational load, especially on large-size problems.

3.3 Implementation Algorithm

For each fixed γ, the objective function given by (8) is a constrained quadratic program which is generally neither convex nor concave. Just following the path following algorithm, we utilize the Frank-Wolfe algorithm to solve it. Specifically, to minimize $F_\gamma(P)$ based on the currently estimated P^*, the Frank-Wolfe algorithm is comprised of the following four steps:

- Step 1: Initialize $P^0 = P^*$ and let $t = 0$.
- Step 2: Find an extreme point X^t (a permutation matrix) of $\mathcal{D}$ by solving the linear program

$$\min \langle \nabla F_\gamma(P^t), X^t \rangle, \text{s.t.} X^t \in \mathcal{D}, \quad (21)$$

  where $\nabla F_\gamma(P)$ is given by

$$\begin{aligned} \nabla F_\gamma(P) &= (1-\gamma)\nabla F_0(P) + \gamma \nabla F_{dcav}(P)/\sigma, \\ \nabla F_0(P) &= 2\left(A_D^T A_D P - A_D^T P A_M - A_D P A_M^T + P A_M A_M^T\right), \\ \nabla F_{dcav}(P) &= -\triangle^T - 2L_D^T P L_M - 2L_D P L_M^T - 2\sigma P. \end{aligned} \quad (22)$$

- Step 3: Find a step size $\alpha \in [0, 1]$ to minimize $F_\gamma(P^t + \alpha(X^t - P^t))$, and update $P^{t+1} = P^t + \alpha(X^t - P^t)$.
- Step 4: If $\langle \nabla F_\gamma(P^{t+1}), P^{t+1} - X^t \rangle < \varepsilon |F_\gamma(P^{t+1}) + \langle \nabla F_\gamma(P^{t+1}), X^t - P^{t+1} \rangle|$, where ε is a small positive constant, return P^{t+1}. Otherwise, let $t = t + 1$ and go back to step 2.

In the algorithm, the linear program in step 2 can be solved by the Hungarian algorithm with a complexity $O(N^3)$, and the line search can be implemented by for instance the backtracking algorithm [13].

Then, the graph-matching algorithm is summarized by Algorithm 3.1.

```
Algorithm 3.1. GRAPHMATCHING ($A_D$, $A_M$)
$P^n \leftarrow 1_{N\times N}/N, n \leftarrow 0, \gamma \leftarrow 0$
while $\gamma \leq 1 \& P^n \notin \mathcal{P}$
  do $P^{n+1} \leftarrow$ FW $(P^n, \gamma), \gamma \leftarrow \gamma + \delta\gamma, n \leftarrow n + 1$
return $(P^n)$
```

We adopt the schema used by the path following algorithm [4] to determine $\delta\gamma$, that is, to double or divide $\delta\gamma$ by 2 to keep the difference of the objective functions between two successive iterations just below a predefined small const. In the algorithm, once P has become a permutation matrix, which implies that the current objective function becomes actually a concave one, the algorithm is terminated, even if γ has not reached 1.

Storage complexity of the algorithm is $O(N^2)$ and the computational complexity of the algorithm is roughly $O(N^3)$ since the complexities of the Hungarian algorithm and matrix multiplication involved in the algorithm are both $O(N^3)$.

4 Experimental Demonstrations

Four algorithms, Umeyama's spectral algorithm (U for short) [14], graduated assignment [8] (GA for short), QCV algorithm in (5),

TABLE 1
Comparative Experimental Results on 100 Graph Pairs with $N = 8$

	matching error	OPT	U	QCV	GA	EPATH1	EPATH2	EPATH3
uniform graphs	mean	5.4349	10.7466	8.2531	9.4097	6.2838	6.2629	6.2317
	standard deviation	0.9235	2.0955	1.8671	1.8451	1.2257	1.2432	1.1964
	# of optimal matching	100	0	1	0	22	19	21
scale-free graphs	mean	8.71	16.55	12.49	14.69	9.38	9.35	9.31
	standard deviation	1.7191	3.3945	3.1059	3.3385	1.8592	1.6840	1.7735
	# of optimal matching	100	0	14	2	68	69	71

and the extended PATH following algorithm (EPATH for short) are experimentally compared to evaluate their performances on both accuracy and complexity. For the GA algorithm, the measure of compatibility between the edges of the two graphs is given as

$$C_{aibj} = \begin{cases} 0 & \text{if either } G_{Dab} \text{ or } G_{Mij} \text{ is null,} \\ 1 - (G_{Dab} - G_{Mij})^2 & \text{otherwise,} \end{cases}$$

and the parameters are set the same as those in [8]. For the EPATH algorithm, ε is set as 0.001 and the initialization of all of the algorithms except for U is $P^0 = 1_{N\times N}/N$. All of the algorithms were implemented by Matlab 2009b, with a MEX function to implement the Hungarian algorithm.

4.1 On Synthetic Data

Two types of graph models are synthetically generated for the comparison, the uniform graphs and scale free graphs. A uniform graph model is generated as follows: Given a sparsity s, for each off-diagonal entry generate a random number r which is uniformly distributed within $[0, 1]$; if $r > s$, randomly generate its weight $A_{ij} = w \in [0, 1]$, or otherwise $A_{ij} = 0$. The scale free graph whose degree distribution follows a power law $p(k) \propto k^{-\alpha}$ is generated in the same manner as that used in [4], with appropriate modifications specific for directed graph models. Each edge of the scale free graph is assigned a unit weight, that is, its adjacency matrix consists of only 0 and 1.

The first experiment is to compare the four algorithms on the two types of graphs, and for each type, 100 8-node graph pairs are randomly generated. For the uniform graph, the sparsity is set as 0.5, meaning that about half of the off-diagonal entries of the adjacency matrix are zero, and for the scale free graphs, the parameter α is set as 2.5. Also, in the experiment we empirically test three types of σ in the concave relaxation F_{dcav}, i.e., the one gotten by (17) (EPATH1), by (20) (EAPTH2), and the ground true smallest eigenvalue (EPATH3).

The experimental results are listed in Table 1, where OPT denotes the globally minimal matching error found by an exhaustive search. From the experimental results, we can observe two points. First, the EPATH algorithm with any one of the three estimations of σ significantly outperforms the other three algorithms on matching accuracy. Second, although EPATH3 (with the exact smallest eigenvalue) outperforms EPATH2 (with a tight lower bound), and EPATH2 outperforms EPATH3 (with a loose lower bound), the three types of EPATH algorithms exhibit comparable performance as the best one (EPATH3) outperforms the worst one (EPATH1) by less than 1 percent. Considering the computational complexity, we adopt EPATH1 in the subsequent experiments. It is also interesting to observe that the EPATH algorithm and QCV got a better result on scale free graphs than on uniform graphs. For instance, on scale free graphs, the average matching error of EPATH1 has a smaller deviation $((9.38 - 8.71)/8.71 = 7.69\%)$ from the minimal error than the one (15.62 percent) on uniform graphs. But, this point is not obvious for U or GA.

We proceed to compare the four algorithms by graph pairs with the second one generated based on the first one by adding some noises. Specifically, for uniform graphs, given a noise level $\beta \in$

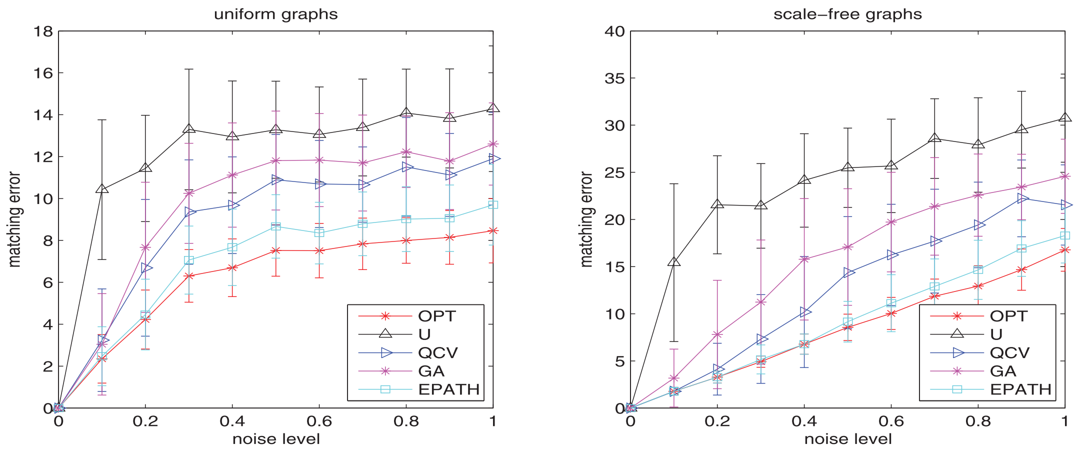

Fig. 2. Changes of the mean and standard deviation of the matching errors with respect to the noise level on the two types of graphs.

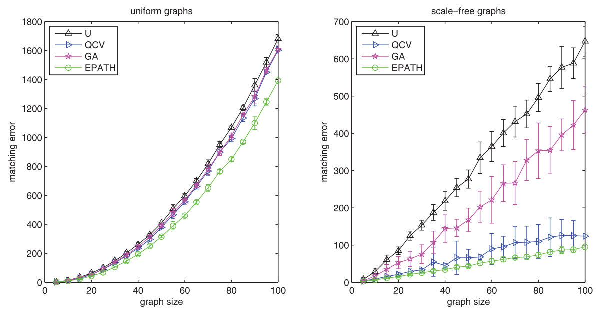

Fig. 3. Matching error of the four algorithms with respect to the graph size.

$[0, 1]$ and a randomly generated adjacency matrix A_D, A_M is generated by the following steps:

1. Set $A_M \leftarrow A_D$, and for each $(A_M)_{ij}, i \neq j$, generate two uniformly distributed random numbers r_1 and $r_2 \in [0, 1]$.
2. If $(A_M)_{ij} > 0$: if $r_1 < \beta$, $(A_M)_{ij} \leftarrow 0$; or otherwise, $(A_M)_{ij} \leftarrow (A_M)_{ij} + \beta r_2$.
3. If $(A_M)_{ij} = 0$: if $r_1 < \beta$, $(A_M)_{ij} \leftarrow r_2$.
4. Generate a random permutation matrix P, and set $A_M \leftarrow P A_M P^T$.

For the scale free graphs, the noise is added by randomly adding βN_D edges into G_D to get G_M, where we denote by N_D the number of edges of G_D.

In the experiment, the noise level β increases gradually from 0 to 1 by a step size 0.1. At each noise level, 100 graph pairs with size $N = 8$ are randomly generated according to the above process. The experimental results are shown in Fig. 2, from which we can summarize some points as follows:

- All four algorithms succeeded in getting the optimal matching in the case of graph isomorphism, i.e., null noise. In such a case, the global minimum of the convex relaxation of QCV and EPATH is exactly the solution of the matching problem.
- The performance of U becomes significantly worse if one of the two graphs to be matched is a little deviated from isomorphic, a 0.1 noise level for instance. This implies that the eigenvalue distance adopted by U is inconsistent with the elementwise distance between the adjacency matrices.
- As the noise level becomes heavier, the performance of GA declines in a quicker way than the QCV and EPATH algorithms.
- EPATH significantly outperforms the other three algorithms when the two graphs to be matched are not isomorphic, especially on a high noise level.

The third experiment is to evaluate the scalability of the four algorithms on both accuracy and complexity. The size of the graphs is increased from 5 to 100 by a step size 5, and on each size 50 graph pairs are generated in the same way as the second experiment with a fixed noise level 0.3 and a sparsity 0.2. The experimental results on accuracy and time cost are shown in Figs. 3 and 4, respectively. It is once again observed that on all scales the performance of EPATH is significantly better than the other three algorithms in terms of matching accuracy. On the scale free graphs, the QCV algorithm also exhibits a promising performance.

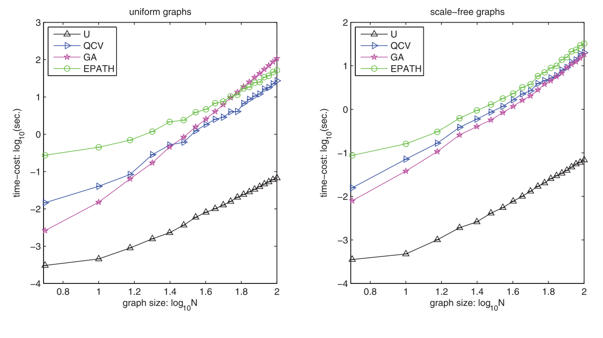

Fig. 4. Time cost of the four algorithms with respect to the graph size.

TABLE 2
Comparative Experimental Results on the QAP Benchmark Data Set

	OPT	U	QCV	GA	EPATH
lipa20a	3683	3925	3902	3909	**3885**
lipa20b	27076	35213	34827	**27076**	32081
lipa30a	13178	13841	13787	13668	**13577**
lipa30b	151426	196088	189496	**151426**	**151426**
lipa40a	31538	32663	32647	32590	**32247**
lipa40b	476581	626004	572039	**476581**	**476581**
lipa50a	62093	64138	63930	63730	**63339**
lipa50b	1210244	1569908	1468492	**1210244**	**1210244**
lipa60a	107218	110196	110075	109809	**109168**
lipa60b	2520135	3305286	3131985	**2520135**	**2520135**
lipa70a	169755	173906	173496	173172	**172200**
lipa70b	4603200	5974833	5576103	**4603200**	**4603200**
lipa80a	253195	258262	258140	258218	**256601**
lipa80b	7783962*	10079359	9703626	**7763962**	**7763962**
lipa90a	360630	367756	367250	366743	**365233**
lipa90b	12490441	16271254	13870571	**12490441**	**12490441**

*There is a typo in the reference [15]. This number should be 7763962.

The slopes of the log curves shown in Fig. 4 reflect the computational complexity of the algorithms with respective to graph size. The slopes corresponding to U, QCV, GA, and EPATH are roughly 2.33 ± 0.05, 3.07 ± 0.08, 3.77 ± 0.3, and 2.97 ± 0.09, respectively, which implies that GA has the worst scalability to matching large graphs. It is somewhat strange to observe that U has an unexpected low complexity $O(N^{2.33})$. This may be due to the fact that the Hungarian algorithm is implemented by a MEX file, which runs significantly faster than its M file counterpart.

4.2 On Quadratic Assignment Problem (QAP) Data

We then evaluate the algorithms on some asymmetric data of the quadratic assignment problem benchmark data set (QABLib) [15]. Actually, minimization of the objective function in (3) is equivalent to maximization of the following objective function:

$$\max_{P \in \mathcal{P}} \operatorname{tr}\left(P^T A_D P A_M^T\right), \tag{23}$$

while QAP is to minimize (23). We then use the EPATH algorithm to solve the QAP on asymmetric data by introducing $A_D = -A_1$, and $A_M = A_2$, where A_1, A_2 denote the QAP data, with the results listed in Table 2,

It is observed that EPATH outperforms the other three algorithms, except for only one data set. It is also somewhat surprising to notice that on the type b data set, the GA and EPATH algorithms can almost always find the optimal solution.

5 Conclusions

The PATH following algorithm was recently proposed to solve the matching problem on undirected graphs and exhibited excellent performance on matching accuracy. In this paper, we extend the algorithm to cover the directed graph models by proposing a concave relaxation for the directed graphs without self-loops. Some experimental comparisons on synthetic and real data witness the validity the proposed method.

Acknowledgments

The authors thanks the anonymous reviewers whose comments greatly improved the manuscript, and Zhi-Yong Liu thanks Dr. Zaslavskiy for some helpful discussions on the QAP. This work was supported by the National Science Foundation of China (NSFC) (grants 60975002, 61033011), and the National Basic Research Program of China (973 Program) (grant 2009CB825404).

References

[1] M.A. Eshera and K.S. Fu, "An Image Understanding System Using Attributed Symbolic Representation and Inexact Graph-Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 5, pp. 604-618, Sept. 1986.

[2] R. Singh, J.B. Xu, and B. Berger, "Global Alignment of Multiple Protein Interaction Networks with Application to Functional Orthology Detection," *Proc. Nat'l Academy of Sciences USA,* vol. 105, no. 35, pp. 12763-12768, 2008.

[3] R. Myer, R.C. Wilson, and E.R. Hancock, "Bayesian Graph Edit Distance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 628-635, June 2000.

[4] M. Zaslavskiy, F. Bach, and J.P. Vert, "A Path Following Algorithm for the Graph Matching Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 12, pp. 2227-2242, Dec. 2009.

[5] J.E. Hopcroft and J.K. Wong, "Linear Time Algorithm for Isomorphism of Planar Graphs (Preliminary Report)," *Proc. Sixth Ann. ACM Symp. Theory of Computing,* pp. 172-184, 1974.

[6] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty Years of Graph Matching in Pattern Recognition," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 18, no. 3, pp. 265-298, 2004.

[7] H. Almohamad and S. Duffuaa, "A Linear Programming Approach for the Weighted Graph Matching Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 5, pp. 522-525, May 1993.

[8] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 377-388, Apr. 1996.

[9] H.W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly,* vol. 2, nos. 1/2, pp. 83-97, 1955.

[10] J.S. Li, "Kronecker Products of Positive Semidefinite Matrices," *J. Math. Research and Exposition,* vol. 17, no. 3, pp. 327-334, 1997.

[11] X.Z. Zhan, "Extremal Eigenvalues of Real Symmetric Matrices with Entries in an Interval," *SIAM J. Matrix Analysis and Applications,* vol. 27, no. 3, pp. 851-860, 2006.

[12] A. Dembo, "Bounds on the Extreme Eigenvalues of Positive-Definite Toeplitz Matrices," *IEEE Trans. Information Theory,* vol. 34, no. 2, pp. 352-355, Mar. 1988.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge Univ. Press, 2004.

[14] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 5, pp. 695-703, Sept. 1988.

[15] R.E. Burkard, S.E. Karisch, and F. Rendl, "Qaplib—A Quadratic Assignment Problem Library," *J. Global Optimization,* vol. 10, no. 4, pp. 391-403, 1997.

[16] L. Xu and E. Oja, "Improved Simulated Annealing, Boltzmann Machine and Attributed Graph Matching," *Proc. EURASIP Workshop 1990 Neural Networks,* G. Goos and J.Hartmanis, eds., pp. 151-160, 1989.