Neurocomputing 134 (2014) 140–148

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# A graph matching algorithm based on concavely regularized convex relaxation

Zhi-Yong Liu [a,*], Hong Qiao [a], Li-Hao Jia [a], Lei Xu [b]

[a] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[b] Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China

ARTICLE INFO

*Article history:*
Received 29 May 2012
Received in revised form
25 October 2012
Accepted 31 December 2012
Available online 23 January 2014

*Keywords:*
Graph matching
Concave–convex procedure
Concave regularization
Frank–Wolfe algorithm
Convex relaxation

ABSTRACT

In this paper we propose a concavely regularized convex relaxation based graph matching algorithm. The graph matching problem is firstly formulated as a constrained convex quadratic program by relaxing the feasible set from the permutation matrices to doubly stochastic matrices. To gradually push the doubly stochastic matrix back to be a permutation one, an objective function is constructed by adding a simple weighted concave regularization to the convex relaxation. By gradually increasing the weight of the concave term, minimization of the objective function will gradually push the doubly stochastic matrix back to be a permutation one. A concave–convex procedure (CCCP) together with the Frank–Wolfe algorithm is adopted to minimize the objective function. The algorithm can be used on any types of graphs and exhibits a comparable performance as the PATH following algorithm, a state-of-the-art graph matching algorithm but applicable only on undirected graphs.



## 1. Introduction

Graph matching plays a central role in many graph based techniques. For instance, graph is frequently used as the structural representation of objects in computer vision and pattern recognition, and consequently the graph matching algorithm is commonly used to solve the object matching problem [5,1]. Graph matching involves identifying each vertex pair between graphs in some optimal way, or inherently finding a *good* permutation matrix between the two adjacency matrices of both graphs.[1] The problem is in nature a NP-hard combinatorial optimization problem with a factorial complexity, except for some graphs with special structure, such as the planar graphs, which has shown to be of polynomial complexity [13]. Therefore, an exhaustive search algorithm is computationally prohibited in practice, except for some small scale problems.

To make the problem computationally tractable, many approximate approaches have been proposed, trying to seek a good trade-off between the complexity and matching accuracy. As summarized in [4], approximate matching algorithms can be roughly categorized into three groups, tree search based methods, spectral methods and continuous optimization (relaxation techniques). Tree search methods [22,6] are based on some simplifications of the depth-first search, for instance. Their performances depend largely on the problem nature, i.e., graph structure. The spectral methods [23,25] have their roots in the fact that the eigenvalues of the adjacency matrices of two isomorphic graphs are identical to each other. Unfortunately, the converse conclusion may be quite wrong, that is, two graphs with identical eigenvalues may be far from isomorphic. This might make the spectral methods result in a quite poor matching when the two graphs are not isomorphic.

Relaxation techniques involve relaxing the combinatorial matching problem to be a continuous one. The key point lies in the fact that optimization over a continuous set is usually easier to be approximated than its discrete counterpart. Specifically, the graph matching problem involves relaxing the set of permutation matrices, denoted by $\mathcal{P}$, to its convex hull, i.e., the set of doubly stochastic matrices denoted by $\mathcal{D}$. Typical relaxation techniques in the literature include for instance relaxation labeling [7,17], graduated assignment [9] and PATH following algorithm [28]. The relaxation labeling assigns each vertex of one graph with a probabilistic discrete label, and updates the label based on some measures, such as the vertex connectivity [7] or edit distance [17]. A common problem faced by the relaxation techniques is the backprojection which involves projecting the continuous solution found by the relaxed problem back to be a discrete one. Intuitively, given a $P_d \in \mathcal{D}$, the backprojection can be accomplished by a maximal linear assignment schema as given in (7), which is commonly employed by the relaxation labeling. However, such a linear projection may introduce a significant additional error. A soft assignment schema controlled by a parameter was introduced by

* Corresponding author.
*E-mail address:* zhiyong.liu@ia.ac.cn (Z.-Y. Liu).

[1] In this paper we consider only the equal-sized graph matching problem.

0925-2312/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.065

the graduated assignment algorithm to control the non-convexity of the problem [9]. As the parameter increases to be large enough, a permutation matrix is expected to be obtained though usually a clean-up step is further needed.

Different from the graduated assignment algorithm, the PATH following algorithm introduces a weighted linear combination of convex and concave relaxations to gradually get the discrete solution. Specifically, given the two graphs $G_D=(V_D,E_D)$ and $G_M=(V_M,E_M)$ to be matched where $V$ and $E$, respectively, denote the sets of vertices and edges, it adopts the following square of Frobenius matrix norm as the objective function:

$$f(P)=\|A_D-PA_MP^T\|_F^2=\mathrm{tr}(A_D-PA_MP^T)^T(A_D-PA_MP^T),P\in\mathcal{P} \tag{1}$$

where $A_D$ and $A_M$ denote the adjacency matrices of $G_D$ and $G_M$, respectively, $\mathcal{P}$ denotes the set of permutation matrix. By taking advantage of $P\in\mathcal{P}$, a convex relaxation of (1) can be found as follows [28]:

$$f_v(P)=\mathrm{vec}(P)^TQ\ \mathrm{vec}(P),\quad P\in\mathcal{D}, \tag{2}$$

where vec($P$) creates a column vector from the matrix $P$ by stacking the column vectors of $P$, and $Q=(I\otimes A_D-A_M^T\otimes I)^T(I\otimes A_D-A_M^T\otimes I)\in\mathbb{R}^{N^2\times N^2}$ is a symmetric definite positive matrix. The concave relaxation introduced by the PATH following algorithm is given by

$$f_c(p)=-\mathrm{tr}(\Delta P)-2\ \mathrm{vec}(P)^T(L_M^T\otimes L_D^T)\ \mathrm{vec}(P),P\in\mathcal{D}, \tag{3}$$

where $\Delta_{ij}=(D_M(i,i)-D_D(j,j))^2$, with $D$ and $L$ denoting the degree and Laplacian matrices of the graph, respectively. The concave relaxation holds the same minima as the original matching problem, but it is applicable only on undirected graphs without self-loops. Based on the convex and concave terms above, the objective function of the PATH following algorithm is given by

$$f_{path}(\gamma,P)=\gamma f_v(P)+(1-\gamma)f_c(P),\quad P\in\mathcal{D}, \tag{4}$$

where $\gamma\in[0,1]$ controls the non-convexity of the objective: a large $\gamma$ means that $f_{path}(P,\gamma)$ tends to be convex; by contrast, a small $\gamma$ makes $f_{path}(P,\gamma)$ concave. Thus, by gradually decreasing $\gamma$ from 1 to 0, the objective becomes finally a concave one, and its minimization results in a permutation matrix. On equal-sized graph matching problems the PATH following algorithm exhibited a state-of-the-art performance in terms of both accuracy and complexity [28].

However, the PATH following algorithm cannot be used to solve the matching problem between directed graphs because the term in Eq. (3) can no longer guarantee to be concave. In this paper we introduce a much simpler concave term which can be applied on both directed and undirected graphs. Though the simple concave term is not a relaxation of the original matching problem, it is shown that it has a comparable performance as Eq. (3) on matching accuracy.[2] Moreover, instead of directly using the Frank–Wolfe algorithm, we firstly adopt the concave–convex procedure (CCCP) [27] to decompose the objective into a sequential constrained convex quadratic program, which is then solved by the Frank–Wolfe algorithm [8], avoiding the trouble of line search on a non-convex function. Section 2 is devoted to the proposed method, some experimental illustrations and discussions are given in Section 3, and finally Section 4 concludes the paper.

## 2. Proposed method

The objective function for the graph matching problem is firstly proposed, and then the CCCP together with Frank–Wolfe algorithm is proposed to minimize the objective, followed by an efficient initialization given by simplicial decomposition.

### 2.1. Objective function

The proposed objective function takes a similar form as Eq. (4), with the same convex relaxation but with a different concave term. To make the algorithm applicable for matching problems on both directed and undirected graphs, we propose to use the following concave term:

$$f_c(P)=-\mathrm{vec}(P)^T\ \mathrm{vec}(P),\quad P\in\mathcal{D}. \tag{5}$$

Then, similar to Eq. (4) an objective function of the graph matching problem is formulated as follows:

$$\text{min. } f_\gamma(P)=\gamma\ \mathrm{vec}(P)^TQ\ \mathrm{vec}(P)-(1-\gamma)\ \mathrm{vec}(P)^T\ \mathrm{vec}(P),\quad \text{s.t. } P\in\mathcal{D}. \tag{6}$$

It is obvious that minimization of the concave term given by Eq. (5) results in an extreme point of $\mathcal{D}$, i.e., a permutation matrix. Thus, by gradually decreasing $\gamma$ from 1 to 0, minimization of the objective will make $P$ gradually converge to a permutation matrix.

At the beginning when $\gamma=1$, the objective function (6) degenerates to the convex relaxation, whose global minimization denoted by $P_v$ can be obtained by the Frank–Wolfe algorithm (here we adopt the simplicial decomposition as discussed below). Actually, a permutation matrix can be directly obtained by an optimal linear assignment procedure which casts the doubly stochastic matrix $P_v$ to be a permutation matrix $P_p$ via

$$P_p=\arg\max_{P\in\mathcal{P}}\ \mathrm{tr}\ P_v^TP. \tag{7}$$

The assignment can be solved by the Hungarian algorithm [14], with a computational complexity $O(N^3)$. Such a *hard-cut* operation based graph matching algorithm, named QCV (quadratic convex), may however bring a big error into the final result, as to be witnessed by the experimental results in Section 3. By contrast, as $\gamma$ gradually decreases, $P$ is gradually pushed away from $P_v$ in the way that update of $P$ is guided to approach a permutation matrix with a smaller matching error. This point can be intuitively understood in the following way. During the convergence process the update direction of $P$ comprises two parts, $g_v(P)$ and $g_c(P)$, the directions provided by the convex and concave terms, respectively. Guidance from $g_v(P)$ is to minimize the increase of the convex term, which, if can be globally minimized during the whole process, is equal to the difference between the best matching error and the global minimization of the convex relaxation got by $P_v$. On the other hand, $g_c(P)$ provides no informative search direction since any permutation matrix gives the same global minimum for the concave term. Thus, in the global minimization sense it is under the guidance of $g_c(P)$ that $P$ is expected to approach a permutation matrix with a relatively small matching error.

To get an intuitive feel about the process, a simple example on matching two directed graphs with self-loops and with $N=3$ is given as shown in Fig. 1. $P$ is parameterized as $P=[b,a,1-a-b;d,c,1-c-d;1-a-c,1-b-d,a+b+c+d-1]$ with the constraints $a\in[0,1],d\in[0,1],b\in[0,\pi],c\in[0,\pi]$ where $\pi=1-\max\{a,d\}$. In Fig. 1 the objective function is plotted by changing $b$ and $c$ with fixed $a$ and $d$ at their current estimations. As $\gamma=1$, $P$ converges from the initial $1_{3\times3}/3$ to $a=0.462,d=0.305,b=0.374,c=0.451$, based on which the QCV gets the result as $P_{\in\mathcal{P}}=[0,1,0;1,0,0;0,0,1]$. As the algorithm proceeds, it is illustrated by the figure how $P$ gradually approaches another solution with a smaller matching error, i.e., $P=I_3$, where $A_D=[0.496,0.302,0.826;0.179,0.390,0.876;0.037,0.998,0.999]$ and $A_M=[0.652,0.505,0.498;0.117,0.936,0.839;0.760,0.403,0.970]$.

[2] In our subsequent works we show theoretically that the simple concave term realizes exactly a concave relaxation.

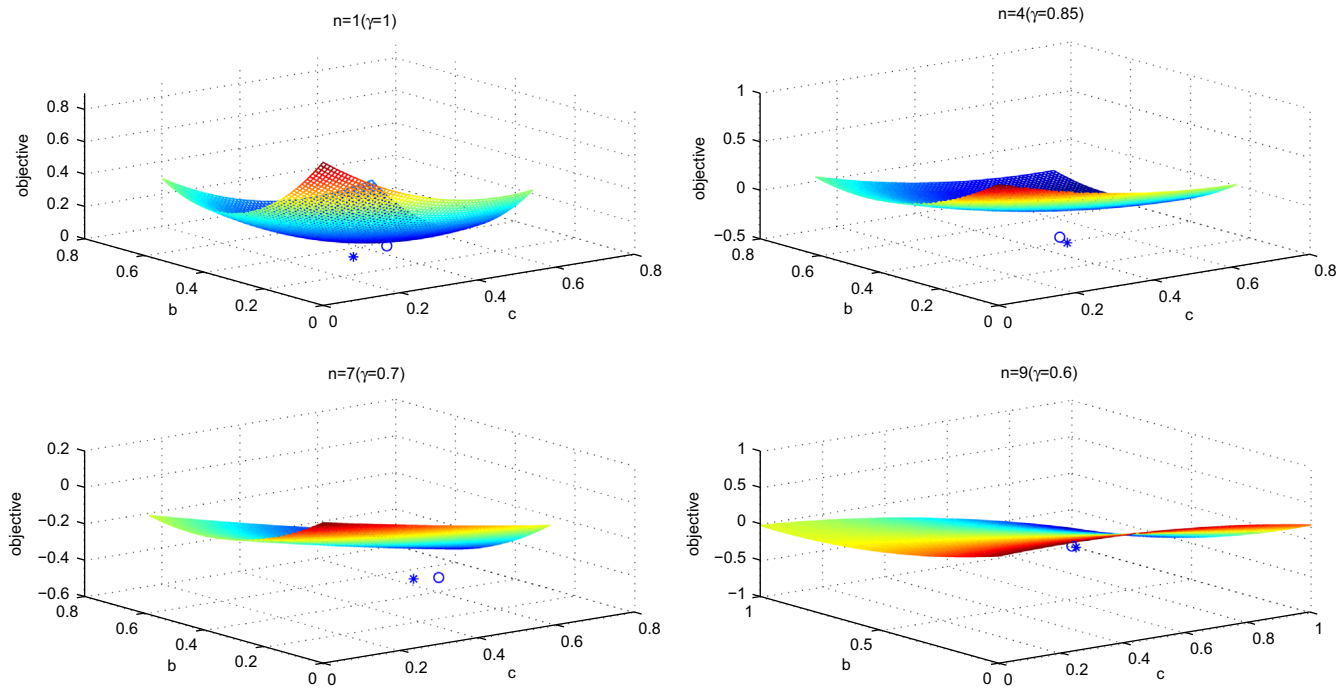



**Fig. 1.** Convergence illustration of the proposed algorithm on matching two directed graphs with self-loops with adjacency matrices. In each step ∗ denotes the starting point and ∘ denotes the convergence point.

In contrast to the above simple concave term, the concave relaxation given by Eq. (3) also provides a search direction $g_c(P)$ which also provides a meaningful guidance for the update of $P$ in the global optimization sense. However, starting from $P_v$, the search direction $g_c(P)$ provided by the concave relaxation is the same as $g_v(P)$, i.e., the direction from $P_v$ to the global optimal point. Therefore, $g_c(P)$ is somewhat redundant to $g_v(P)$, and $g_c(P)$ just strengthens $g_v(P)$ but does not provide additional useful guidance. This is to some extent confirmed by the experimental comparisons in Section 3 which witnesses that, on matching undirected graphs, the simple concave term (5) has a comparable or even a slightly better performance than the concave relaxation.

### 2.2. Algorithm

For each fixed $\gamma$, the objective function given by Eq. (6) is a constrained quadratic program which is generally neither convex nor concave. Unlike the PATH following algorithm which adopts directly the Frank–Wolfe algorithm to minimize the objective function, here we firstly utilize the concave–convex procedure (CCCP) to decompose the objective function into a sequential constrained convex quadratic program, which is then solved by the Frank–Wolfe algorithm.

The CCCP algorithm consists of sequentially minimizing the following constrained convex function:

$$f_{k+1}(P^{k+1}) = f_v(P^{k+1}) + \mathrm{vec}(P^{k+1})^T \nabla f_c(P^k), P^{k+1} \in \mathcal{D}, \tag{8}$$

where $P^{k+1}$ denotes the $P$ to be found in step $k$, and $f_v$ and $f_c$ the convex and concave terms, respectively. Since $\nabla f_c(P^k) = -2(1-\gamma)\,\mathrm{vec}(P^k)$ is a constant with respect to $P^{k+1}$, Eq. (8) is formulated as the following constrained convex quadratic program:

$$\begin{aligned} \min.\quad & f_{cccp}(P) = \gamma \mathrm{vec}(P)^T Q \mathrm{vec}(P) - 2(1-\gamma)\mathrm{vec}(P^k)^T \mathrm{vec}(P), \\ \text{s.t.}\quad & P \in \mathcal{D}. \end{aligned} \tag{9}$$

It showed [19] that the positive definite transformation matrix $TM(P)$ [24] of the CCCP algorithm takes the forms

$$\begin{aligned} TM(P^k) &\approx \left[ I - \left( \frac{\partial^2 f_{cav}}{\partial^2 P} \right) \left( \frac{\partial^2 f_{vex}}{\partial^2 P} \right)^{-1}_{P=P^k} \right] [-H(P^k)]^{-1} \\ &= \left[ I - \frac{-(1-\gamma)Q^{-1}}{\gamma} \right] [-H(P^k)]^{-1} \end{aligned} \tag{10}$$

where $H(P^k)$ denotes the Hessian matrix of the objective $E$. Because $Q$ is positive definite, all of the eigenvalues of $Q^{-1}$ are also positive, which implies that the CCCP algorithm used here is expected to enjoy a superlinear convergence rate.

The Frank–Wolfe algorithm is then adopted to solve the constrained convex quadratic program (9). Specifically, it comprises the following four steps:

*Step* 1: Initialize $P^0 = P^*$ and let $t=0$, where $P^*$ denotes the result obtained by the previous CCCP loop.

*Step* 2: Find an extreme point $X^t$ (a permutation matrix) of $\mathcal{D}$ by solving the linear program

$$\text{min. tr } \nabla f_{cccp}(P^t)^T X^t, \quad \text{s.t. } X^t \in \mathcal{D}, \tag{11}$$

where $\nabla f_{cccp}(P)$ is given by

$$\begin{aligned} \nabla f_{cccp}(P) = {} & 2\gamma(A_D^T A_D P - A_D^T P A_M - A_D P A_M^T + P A_M A_M^T) \\ & - 2(1-\gamma)P^*. \end{aligned} \tag{12}$$

*Step* 3: Find a step size $\alpha \in [0,1]$ to minimize $f_{cccp}(P^t + \alpha(X^t - P^t))$, and update $P^{t+1} = P^t + \alpha(X^t - P^t)$.

*Step* 4: If $|\langle \nabla f_{cccp}(P^t), X^t - P^t \rangle| < \varepsilon |f_{cccp}(P^t) + \langle \nabla f_{cccp}(P^t), X^t - P^t \rangle|$ where $\varepsilon$ 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 efficiently implemented by the backtracking algorithm [3]. The stopping criterion in step 4 is applicable, thanks to the convexity of the objective function $f_{cccp}$.

Finally, the graph matching algorithm is summarized by Algorithm 1.

**Algorithm 1.** GraphMatching $(A_D, A_M)$.

$P^n \leftarrow 1_{N\times N}/N, n\leftarrow 0, \gamma\leftarrow 1$
**while** $\gamma \geq 0 \& P \notin \mathcal{P}$
  **do** $P^{n+1} \leftarrow \mathrm{CCCP}(P^n, \gamma), \gamma\leftarrow\gamma-\delta\gamma, n\leftarrow n+1$
**return** $(P^n)$

In the algorithm $\delta\gamma$ is the step size of $\gamma$, and once $P$ becomes a permutation matrix which implies that the current objective becomes a concave one, the algorithm is terminated, even if $\gamma$ has not reached zero. Each main loop of the graph matching algorithm is called as a CCCP algorithm, and each CCCP loop is further called as a Frank–Wolfe algorithm. Thus, $P^t$ and $P^k$ in the Frank–Wolfe and CCCP algorithms should be formally denoted as $P^{nkt}$ and $P^{nk}$ with the three superscripts $n$, $k$, and $t$ denoting the processes of the main loop, CCCP loop and Frank–Wolfe loop, respectively.

### 2.3. An efficient initialization based on simplicial decomposition

It is well known that the Frank–Wolfe algorithm suffers a sub-linear convergence rate due to the fact that the reduced search direction of the Frank–Wolfe algorithm tends to be perpendicular to the steepest descent direction as the iteration proceeds [15,10]. Below we adopt the simplicial decomposition to efficiently solve the convex relaxation, i.e., initialization of the objective function as $\gamma=1$.

The constraint $P\in\mathcal{D}$ can be written in the form of a linear equality together with a inequality constraint as

$$C\,\mathrm{vec}(P) = \mathbf{1}_{2N-1},\quad \mathrm{vec}(P)\geq \mathbf{0}_{N^2}, \tag{13}$$

where $C$ is an appropriate constant matrix to constrain each column and row of $P$ a unit vector. Thus, the constrained program in Eq. (9) satisfies the following three conditions: (1) the feasible region is convex and compact; (2) the objective function is convex; and (3) the constraints are linear. These cause the program to be effectively solved by the simplicial decomposition algorithm [12,21,15,10], which, as a generalization of the Frank–Wolfe algorithm [8], is expected to converge in a much faster way. Specifically, the simplicial decomposition algorithm decomposes the program into two parts, subproblem and master problem, and solves the original program by iterating the two parts. Given the current estimation of $P^t$ at the $t$th iteration, the subproblem solves exactly the same linear program given by (11).

The master problem involves solving the following quadratic convex problem over a $r-1$ dimensional simplex:

$$\text{min. } h(\beta)=f_{cccp}(W\beta),\quad \text{s.t. } \sum_{i=1}^{r}\beta_i=1, \beta_i\geq 0, i=1,2,\ldots,r, \tag{14}$$

where $W\in\mathbb{R}^{N^2\times r}$ consists of a subset of extreme points of $\mathcal{D}$ that have been found previously by the subproblem, together with the current estimation $P^t$, i.e.,

$$W=[\mathrm{vec}(X^1),\ldots,\mathrm{vec}(X^t),\mathrm{vec}(P^t)].$$

More specifically, by defining $S\Delta qW^TQW$ we have

$$h(\beta)=f_{cccp}(W\beta)=\gamma\beta^TS\beta-2(1-\gamma)\,\mathrm{vec}(P^*)^TW\beta \tag{15}$$

where $S\in\mathbb{R}^{r\times r}$ is a symmetric positive definite matrix. Then, $P^{t+1}$ in the next iteration is given by

$$P^{t+1}=W\beta. \tag{16}$$

Each entry of $S$ is simply given by

$$\begin{aligned}S_{ij}=S_{ji}&=W_i^TQW_j\\&=\mathrm{vec}(A_D\,\mathrm{uvec}(W_i)-\mathrm{uvec}(W_i)A_M)^T\,\mathrm{vec}(A_D\,\mathrm{uvec}(W_j)\\&\quad-\mathrm{uvec}(W_j)A_M),\end{aligned} \tag{17}$$

where $\mathrm{uvec}(\cdot)$ is a converse operator of $\mathrm{vec}(\cdot)$, and $W_i$ denotes the $i$th column vector of $W$. Just like Eq. (12), it does not involve the big-size $Q$ explicitly.

A key parameter in the simplicial decomposition algorithm is $r$, the number of extreme points needed to be combined to get the final solution. An upper bound of $R$ was suggested for $r$ [11], which is usually much smaller than $N^2$, and it was shown that there is a tradeoff between $R$ and the number of iterations to get the optimal solution. $R=3\sqrt{N}$ is suggested in all of our experiments.

A much smaller $R$ implies that the master problem can be efficiently solved by a second-order technique, such as the projected Newton method [2,20]. By transferring the simplex constraint to be a bound region, the global minimum of the master problem can be efficiently reached by the projected Newton algorithm with a super-linear convergence, usually needing only 3–5 iterations to converge. A detailed implementation of the projected Newton method on the master problem is given in Appendix A.

The simplicial decomposition algorithm is finally summarized as follows:

*Step* 1: Let $P^0=P^*, [W_s]^0=\emptyset, [W_x]^0=\{P^0\}, t=0$.
*Step* 2: Get $X^t$ by solving the subproblem in Eq. (11) by the Hungarian algorithm.

*Step* 3.1: If $|[W_s]^t|=R$, remove the element with the minimal weight $(\beta_i)$ in $[W_s]^t$.
*Step* 3.2: Let $[W_s]^{t+1}=[W_s]^t\cup X^t$,
$[W_x]^{t+1}=\{P^t\}, W^{t+1}=[W_s]^{t+1}\cup[W_x]^{t+1}$.

*Step* 4: Get $P^{t+1}$ by solving the master problem in Eq. (14) by the projected Newton algorithm. If $|\mathrm{vec}(P^{t+1}-P^t)|<\varepsilon_P$ or $|f(P^{t+1})-f(P^t)|<\varepsilon_f|f(P^t)|$, terminate the algorithm and output $P^{t+1}$ as the solution; otherwise, let $t=t+1$ and go back to step 2.

In the algorithm $\varepsilon_P$ and $\varepsilon_f$ are two small positive constants to control the precision of the algorithm.

## 3. Experimental illustrations

Two series of experiments on both synthetic data and real feature correspondence were conducted to evaluate the proposed algorithm. Five algorithms including Umeyama's algorithm (U for short) [23], PATH following algorithm [28] (on undirected graphs only), QCV algorithm given by Eq. (7), graduated assignment algorithm [9] (GA for short), and the proposed algorithm (Ours) were experimentally compared. All of the algorithms were implemented by Matlab 2009b, with a MEX function to implement the Hungarian algorithm.

### 3.1. On synthetic data

Two types of synthetic graphs, uniform graphs and scale-free graphs, were synthetically generated for the comparison. For problems with small scale ($N$=8 for instance), an exhaustive search was applied to get the optimal matching. The degree distribution of a uniform graph follows a uniform distribution, which is generated as follows: given a sparsity $s$, for each entry of the adjacency matrix 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 [28], by setting $\alpha=1.5$. Each edge of the scale free graph is assigned a unit weight, that is, its adjacency matrix comprises only 0 and 1.

The first experiment is to simulate the scenario of graph matching without any prior. In the experiment, 100 pairs of graphs with size $N=8$ are randomly generated by the following procedure: for each entry $A_{ij}$ ($A_{ji}=A_{ij}$ in the case of undirected graph) randomly generate a uniformly distributed number $r\in[0,1]$; if $r>0.5$ (meaning that sparsity of the graph is around 0.5), randomly generate its weight $A_{ij}=w\in[0,1]$, or otherwise $A_{ij}=0$. The first experimental results are listed in Table 1 from which it is witnessed that the PATH (on only undirected graphs) and our algorithms have much better performances on accuracy than the U, QCV and GA algorithms. Besides, PATH and our algorithm have a comparable performance on accuracy, as echoed by the discussions in Section 2.1.

The second experiment is to evaluate the noise robustness of the algorithms. In the experiment, the second graph in a graph pair was generated based on the first one by adding some noises which are controlled by a noise level. Specifically, for uniform graphs, given a noise level $\rho\in[0,1]$ 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 $(A_M)_{ij}$, randomly generate two variables $r_1$ and $r_2\in[0,1]$.
2. If $(A_M)_{ij}>0$: if $r_1<\rho$, $(A_M)_{ij}\leftarrow 0$; or otherwise, $(A_M)_{ij}\leftarrow (A_M)_{ij}+\rho r_2$.
3. If $(A_M)_{ij}=0$: if $r_1<\rho$, $(A_M)_{ij}\leftarrow r_2$.
4. Randomly generate a permutation matrix $P$, and set $A_M\leftarrow PA_MP^T$.

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

The noise level $\rho$ increases from 0 to 1 by a step size 0.1. On each noise level, 100 graph pairs with $N=8$ are generated according to the above process. The experimental results on the four types of graphs are shown in Fig. 2. Similar to the first experimental result, the PATH and our algorithm outperform significantly the other three algorithms.

The third experiment is to evaluate the scalability of the five algorithms with respect to the graph size, on both accuracy and

**Table 1**
Comparative experimental results on four types of graphs with $N=8$, summarized from 100 random runs.

| Graph types | Error | OPT | U | GA | QCV | PATH | Ours |
|---|---|---|---|---|---|---|---|
| Undirected uniform | Mean | 3.229 | 8.255 | 6.476 | 8.741 | 3.776 | 3.737 |
| | Std | 1.055 | 2.947 | 1.892 | 2.367 | 1.289 | 1.354 |
| Directed uniform | Mean | 5.565 | 11.26 | 9.619 | 9.300 | – | 6.447 |
| | Std | 0.948 | 2.573 | 1.833 | 2.095 | – | 1.258 |
| Undirected scale-free | Mean | 10.20 | 23.66 | 13.13 | 21.13 | 10.73 | 10.60 |
| | Std | 3.689 | 4.957 | 5.056 | 5.424 | 3.644 | 4.050 |
| Directed scale-free | Mean | 12.23 | 24.83 | 18.96 | 21.43 | – | 13.76 |
| | Std | 1.356 | 3.374 | 3.056 | 3.158 | – | 2.238 |

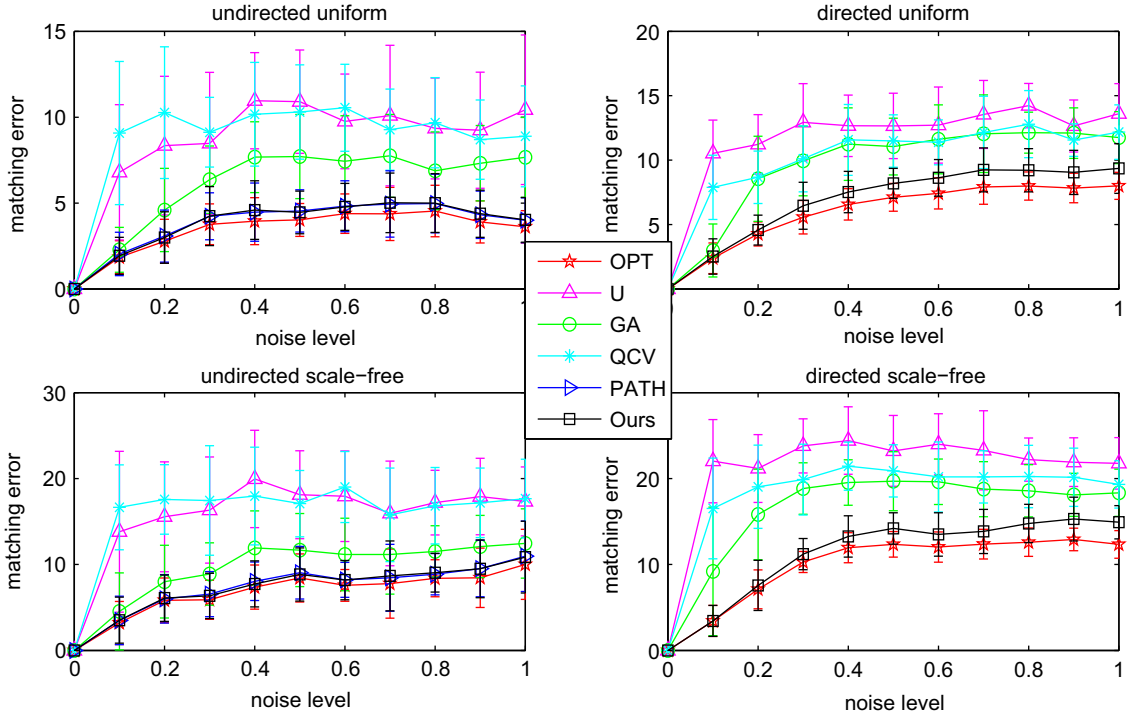



**Fig. 2.** Changes of the matching error with respect to noise levels, summarized from 100 random runs. Left: undirected graphs without self-loops. Right: undirected graphs with self-loops.

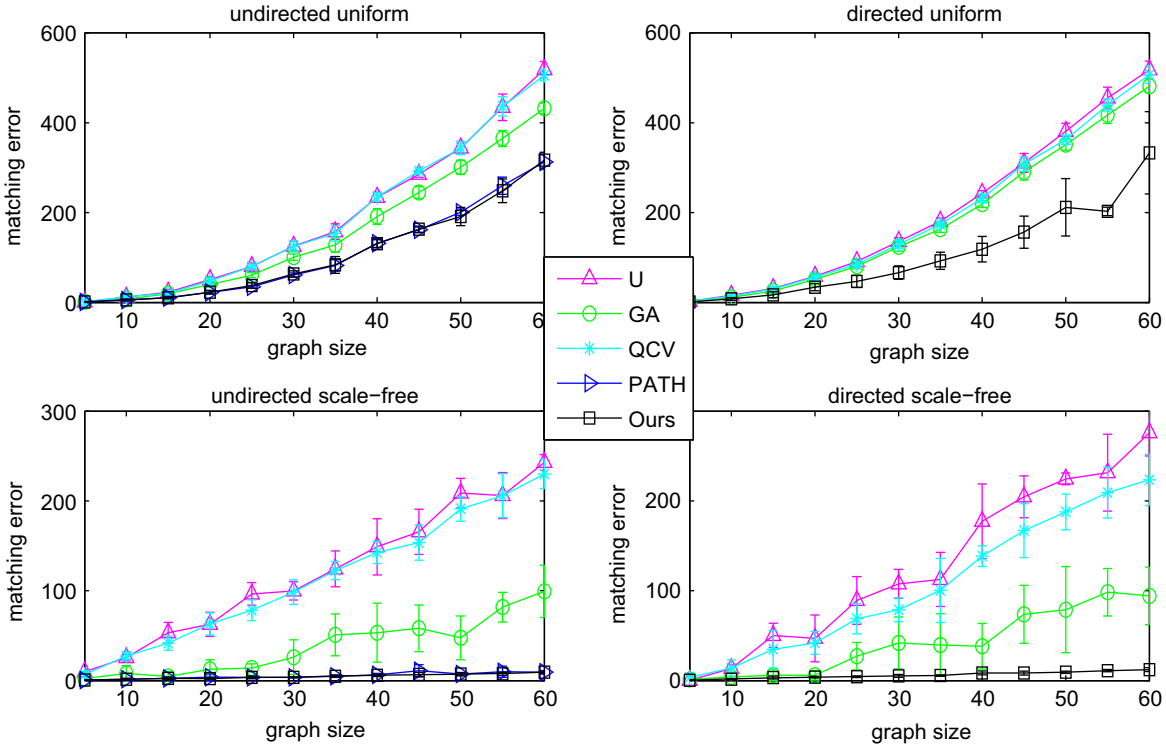



**Fig. 3.** Changes of the matching error with respect to graph sizes, summarized from 100 random runs. Left: undirected graphs without self-loops. Right: undirected graphs with self-loops.

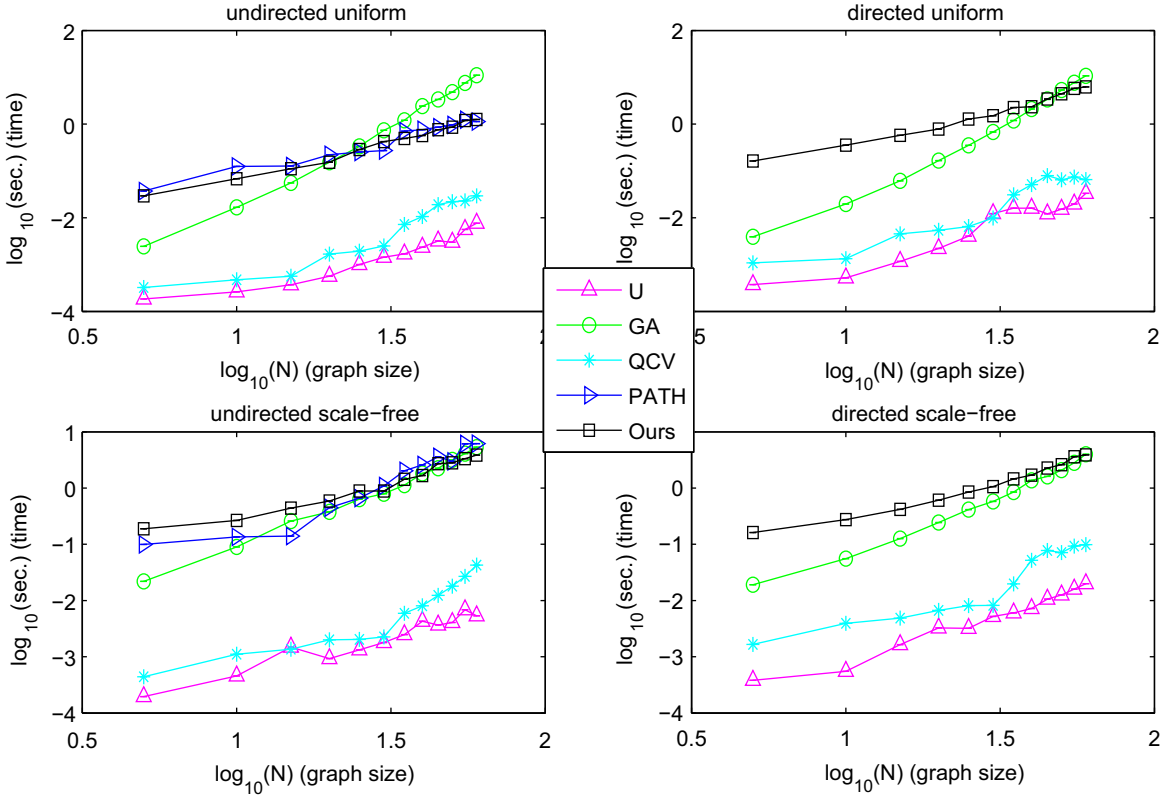



**Fig. 4.** Changes of the matching error with respect to graph sizes, summarized from 100 random runs. Left: directed graphs without self-loops. Right: directed graphs with self-loops.

complexity. In the experiment, 12 groups of graph pairs with different sizes are included for comparison with the size increasing from 5 to 60 by a step size 5. In each group, 50 graph pairs are generated in the same way as the second experiment with a noise level 0.2.

On accuracy, similar experimental results as the above two are obtained on all of the 12 groups of graph pairs, as witnessed by Fig. 3. On complexity, the time-cost of the five algorithms is shown in Fig. 4 in which the slopes of the five curves corresponding to U,

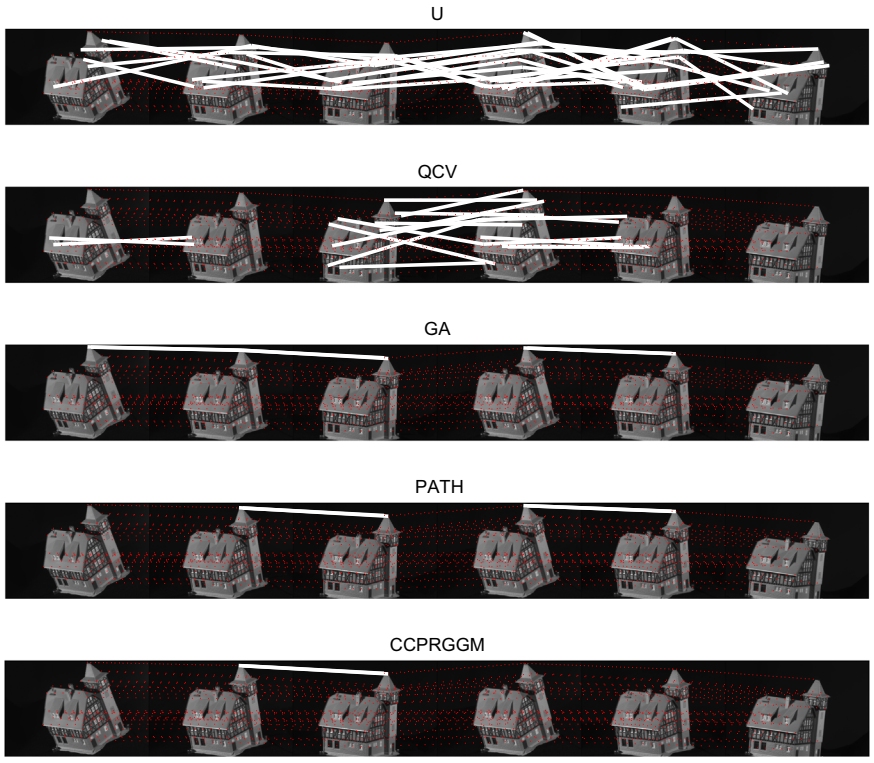



**Fig. 5.** Five feature correspondence results by the five graph matching algorithms, with 47, 19, 6, 4, and 2 mismatched pairs (out of 125), which are indicated by white thicker lines.

QCV, GA, PATH, and our algorithm are around $2.412 \pm 0.032$, $3.033 \pm 0.06$, $4.135 \pm 0.19$, $2.962 \pm 0.08$, and $2.937 \pm 0.07$, respectively, which imply that the GA suffers the biggest complexity and U is the simplest one. On the other hand, the complexity of QCV, PATH and our algorithm is all around $O(N^3)$.

### 3.2. On feature correspondence

Feature correspondence (or point pattern matching) is a fundamental problem in pattern recognition and computer vision. The objective function of feature correspondence usually involves two terms, the unary term related to the appearance cues and the pairwise term related to the geometric relationship [16,26,18]. To evaluate the proposed graph matching algorithm, we focus on the pairwise term, which in the experiment is formulated by the normalized square of the difference between the distances between two feature locations as follows:

$$F(P) = \| A_D - P A_M P^T \|_F^2,$$

$$A_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \| l_i - l_j \|_F / \max_{mn} A_{mn} & \text{otherwise,} \end{cases}$$

where $l_i = (x_i, y_i)^T$ denotes the location of the feature $i$. It takes exactly the same form as the graph matching problem in (1), and thus can be solved by the proposed algorithm.

We adopt the CMU hotel sequence data for the comparison, where we choose 6 frames, i.e., the frames 1, 21, 41, 61, 81 and 101 with a 20-frame interval, as shown in Fig. 5. For each frame we manually marked the same 25 feature points (typically the corner points), and totally perform $C_6^2 = 15$ matchings between the 6 frames and take the number of mismatched feature pairs (the total number is 375) as the evaluation criterion. The matching results are listed in Table 2, and five typical matching results are shown in Fig. 5, which shows similar results as on the synthetic data.

**Table 2**
Comparative experimental results of the five algorithms on pairwise matching

| Results | U | QCV | GA | PATH | Ours |
|---|---|---|---|---|---|
| Average matching error | 39.3772 | 4.1764 | 1.3976 | 0.9657 | 0.9627 |
| # of mismatched feature pairs | 137 | 49 | 13 | 8 | 8 |

## 4. Conclusions

In this paper we showed that together with the convex relaxation, a very simple concave function has a comparable performance with the concave relaxation for the graph matching problem. The point is that the simple concave function can be utilized on matching different types of graphs, but by contrast, the concave relaxation of PATH following algorithm can be used only on undirected graphs. On four different types of graphs, our algorithm showed a state-of-art performance on accuracy.

## Acknowledgments

This work was supported by National Science Foundation of China (NSFC) (Grants 61375005, 60975002, 61033011, 61210009), and the National Basic Research Program of China (973 Program) (Grant 2009CB825404).

## Appendix A. Master problem: a projected Newton algorithm

To use the projected Newton method to solve the master problem in Eq. (14), we first transform the simplex to a bound

constraint [2]. Given a feasible $\beta$, find

$$l = \arg\max_i \{\beta_i | i = 1, 2, \ldots, r\}. \tag{A.1}$$

Then, a variable $\lambda$ is introduced in the following way:

$$\lambda_i = \begin{cases} \beta_i & \text{if } i \neq l \\ 1 & \text{otherwise} \end{cases} \text{ or } \beta = T\lambda, \tag{A.2}$$

where $T_{ij} = 1$ if $i=j$, or otherwise,

$$T_{ij} = \begin{cases} -1 & \text{if } i = l, \\ 0 & \text{otherwise.} \end{cases}$$

With the above transformation, the master problem in Eq. (14) is rewritten as

$$\begin{aligned} \text{min.} \quad & g(\lambda) = \gamma \lambda^T \Psi \lambda - 2(1-\gamma)\, \text{vec}(P^*)^T W T \lambda \\ \text{s.t.} \quad & \lambda_i \geq 0, \forall i \neq l, \ \ \lambda_l = 1, \ \ \lambda_l - \sum_{i \neq l} \lambda_i \geq 0. \end{aligned} \tag{A.3}$$

where $\Psi \Delta q T^T S T$ is a symmetric positive definite matrix with $S$ given by Eq. (17). Since the last constraint, i.e, $\lambda_l - \sum_{i \neq l} \lambda_i \geq 0$, is by construction inactive at the point $\lambda = T^{-1}\beta$, it will be ignored in the iteration of the projected Newton algorithm. The algorithm iterates the following two steps to convergence:

*Step* 1: Get $\lambda^{\text{m}}$ by Eq. (A.2), and set $\overline{\lambda_i^m} = y_i^m$ if $y_i^m \geq 0$; or $\overline{\lambda_i^m} = 0$ otherwise, where $y^m = \lambda^m - \alpha D^m \nabla g(\lambda^m)$.

*Step* 2: Update $\lambda_i^{m+1} = 1$ if $i = l^m$; or $\lambda_i^{m+1} = \overline{\lambda_i^m}$ otherwise, and reconstruct $\beta^{\text{m}+1}$ by $\beta^{m+1} = T^m \lambda^{m+1}$.

There are two terms in the algorithms needed to be specified, i.e., $D^m$ and $\alpha$ in step 1.

In implementation, the elements of $\lambda$ are divided into *free* and *restricted* subsets. The *restricted* subset is defined as

$$\Im^m = \{i | \lambda_i^m \leq \varsigma, \nabla g(\lambda_i^m) > 0\}, \tag{A.4}$$

and the *free* subset denoted by $\wp^{\text{m}}$ contains the remainders, where $\varsigma$ is a small positive number. Then, $D^m$ is defined as follows:

$$[D^m]_{ij} = \begin{cases} 0 & i \neq j \text{ and either } i \in \Im^m \text{ or } j \in \Im^m \\ 0 & \text{if } i = l^m \text{ or } j = l^m \\ [\nabla^2 g(\lambda^m)]_{ij}^{-1} & \text{else,} \end{cases} \tag{A.5}$$

where the gradient and Hessian of the objective are given as follows:

$$\nabla g(\lambda^m) = 2\gamma \Psi^m \lambda^m - 2(1-\gamma)(T^m)^T W^T \text{ vec}(P^*), \tag{A.6}$$

$$\nabla^2 g(\lambda^m) = \gamma \Psi^m. \tag{A.7}$$

By setting $\alpha \Delta q \tau \sigma^m$ for some $\tau \in (0, 0.5), \sigma \in (0, 1)$, $\alpha$ is found by finding the smallest nonnegative integer $m$ that satisfies both

$$1 - \sum_{i \neq l} \lambda_i^{m+1} \geq 0 \tag{A.8}$$

and

$$g(\lambda^{m+1}) \leq g(\lambda^m) + \tau \sigma^m \nabla g(\lambda^m)^T (\lambda^{m+1} - \lambda^m).$$

It is worth adding some discussions on step 1, where $\overline{\lambda^m}$ is actually got by $\overline{\lambda^m} = \arg\min_{\lambda \in \Omega} (\lambda - y^m)^T (\lambda - y^m) = \mathfrak{P}[y^m]$, a projection of the unconstrained direction under the standard Euclidean norm instead of being with the metric is defined by the Hessian matrix. Each element by the projection $\mathfrak{P}[y^m]$ is simply given by the componentwise median of $\mathfrak{P}[y_i^m] = \{v_i^m, y_i^m, u_i^m\}$, where $v_i^m$ denotes the lower bound (0 for $i \neq l^m$), and $u_i^m$ upper bound ($+\infty$ for $i \neq l^m$). Moreover, as discussed in [20], as $\varepsilon$ in Eq. (A.4) is small enough, we can just ignore all of the elements in $\Im^{\text{m}}$ that remain zero during the iterations, and consequently save computational load.

## References


[1] A. Berg, T. Berg, J. Malik, Shape matching and object recognition using low distortion correspondences, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2005) 26–33.

[2] D.P. Bertsekas, Projected Newton methods for optimization problems with simple constraints, SIAM J. Control Optim. 20 (1982) 221–246.

[3] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, U.K., 2004.

[4] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, Int. J. Pattern Recognit. Artif. Intell. 18 (2004) 265–298.

[5] M.A. Eshera, K.S. Fu, An image understanding system using attributed symbolic representation and inexact graph-matching, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1986) 604–618.

[6] M.L. Fernandez, G. Valiente, A graph distance metric combining maximum common subgraph and minimum common supergraph, Pattern Recognit. Lett. 22 (2001) 753–758.

[7] M.A. Fischler, R.A. Elschlager, The representation and matching of pictorial structures, IEEE Trans. Comput. C 22 (1973) 67–92.

[8] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval Res. Logist. Q. 3 (1956) 95–110.

[9] S. Gold, A. Rangarajan, A graduated assignment algorithm for graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 18 (1996) 377–388.

[10] M. Guignard, A. Ahlatcioglu, The convex hull relaxation for nonlinear integer programs with convex objective and linear constraints, in: Proceedings of the European Workshop on Mixed Integer Nonlinear Programming, 2010, pp. 149–158.

[11] D.W. Hearn, S. Lawphongpanich, J.A. Ventura, Restricted simplicial decomposition: computation and extensions, Math. Program. Study 31 (1987) 99–118.

[12] B.V. Hohenbalken, Simplicial decomposition in nonlinear programming algorithms, Math. Program. 13 (1977) 49–68.

[13] J.E. Hopcroft, J.K. Wong, Linear time algorithm for isomorphism of planar graphs (preliminary report), in: Proceedings of the 6th annual ACM symposium on Theory of computing, STOC'74, ACM, New York, NY, USA, 1974, pp. 172–184.

[14] H.W. Kuhn, The Hungarian method for the assignment problem, Naval Res. Logist. Q. 2 (1955) 83–97.

[15] T. Larsson, M. Patriksson, Simplicial decomposition with disaggregated representation for the traffic assignment problem, Transp. Sci. 26 (1992) 4–17.

[16] H. Liu, S. Yan, Common visual pattern discovery via spatially coherent correspondences, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1609–1616.

[17] R. Myer, R.C. Wilson, E.R. Hancock, Bayesian graph edit distance, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 628–635.

[18] G. Philbin, J. Sivic, A. Zisserman, Geometric latent Dirichlet allocation on a matching graph for large-scale image datasets, Int. J. Comput. Vis. 95 (2011) 138–153.

[19] R. Salakhutdinov, S.T. Roweis, Z. Ghahramani, On the convergence of bound optimization algorithms, in: Proceedings of Uncertainty in Artificial Intelligence, pp. 509–516.

[20] M. Schmidt, D. Kim, S. Sra, Projected Newton-type methods in machine learning, in: S. Sra, S. Nowozin, S.J. Wright (Eds.), Optimization for Machine Learning, MIT Press, Cambridge, MA, USA, 2011.

[21] C.M. Shetty, M.B. Dyaa, A decomposition procedure for convex quadratic programs, Naval Res. Logist. Q. 35 (1988) 111–118.

[22] W.H. Tsai, K.S. Fu, Error-correcting isomorphisms of attributed relational graphs for pattern analysis, IEEE Trans. Syst. Man Cybern. 9 (1979) 757–768.

[23] S. Umeyama, An eigendecomposition approach to weighted graph matching problems, IEEE Trans. Pattern Anal. Mach. Intell. 10 (1988) 695–703.

[24] L. Xu, M.I. Jordan, On convergence properties of the EM algorithm for Gaussian mixtures, Neural Comput. 8 (1996) 129–151.

[25] L. Xu, I. King, A PCA approach for fast retrieval of structural patterns in attributed graphs, IEEE Trans. Syst. Man Cybern. B: Cybern. 31 (2001) 812–817.

[26] Y. Yang, F. Nie, D. Xu, J. Luo, Y. Zhuang, Y. Pan, A multimedia retrieval framework based on semi-supervised ranking and relevance feedback, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2012) 723–742.

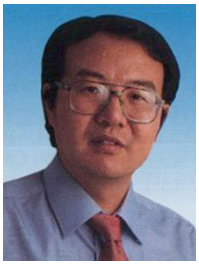
[27] A.L. Yuille, A. Rangarajan, The concave–convex procedure, Neural Comput. 15 (2003) 915–936.

[28] M. Zaslavskiy, F. Bach, J.P. Vert, A path following algorithm for the graph matching problem, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2009) 2227–2242.


**Zhi-Yong Liu** is a professor at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include machine learning, pattern recognition, computer vision, and bioinformatics.

**Hong Qiao** is a professor at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. Her research interests include robotics, machine learning, and computer vision.

**Li-Hao Jia** is currently a postdoctoral researcher at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include vision perception, sparse representation and their applications to robotics.



**Lei Xu** is a chair professor at the Department of Computer Science and Engineering, Chinese University of Hong Kong, Statin, Hong Kong. His research interests include statistical learning, computer vision, and bioinformatics.