# Incorporating Discrete Constraints Into Random Walk-Based Graph Matching 

Xu Yang, Zhi-Yong Liu, Senior Member, IEEE, and Hong Qiao, Senior Member, IEEE


#### Abstract

Graph matching is a fundamental problem in theoretical computer science and artificial intelligence, and lays the foundation for many computer vision and machine learning tasks. Approximate algorithms are necessary for graph matching due to its NP-complete nature. Inspired by the usage in network-related tasks, random walk is generalized to graph matching as a type of approximate algorithm. However, it may be inappropriate for the previous random walk-based graph matching algorithms to utilize continuous techniques without considering the discrete property. In this paper, we propose a novel random walk-based graph matching algorithm by incorporating both continuous and discrete constraints in the optimization process. Specifically, after interpreting graph matching by random walk, the continuous constraints are directly embedded in the random walk constraint in each iteration. Further, both the assignment matrix (vector) and the pairwise similarity measure between graphs are iteratively updated according the discrete constraints, which automatically leads the continuous solution to the discrete domain. Comparisons on both synthetic and real-world data demonstrate the effectiveness of the proposed algorithm.


Index Terms-Feature correspondence, feature extraction, graph matching, image registration, image representation.

## I. Introduction

GRAPH matching is a fundamental problem in theoretical computer science and artificial intelligence, such as chemical compound classification [1], [2], biomedical and biological applications [3], [4], network-related problems [5], etc. It lays the foundation for many computer vision and pattern analysis tasks, including gesture recognition [6], feature correspondence [7], facial landmark point localization [8], and computer-aided design [9].

[^0]Most graph matching problems are known to be NPcomplete. ${ }^{1}$ Therefore researchers have introduced various types of algorithms to approximately solve them, as briefly surveyed below. A group of methods relax the graph matching problem to convex optimization formulations, for which the optimal solution can be found in polynomial time. For instance, [10] approximates graph matching by a linear programming problem, and [11] and [12] adopt the semidefinite programming formulations. A major disadvantage of these methods is that they usually involve a hard-cut projection of the continuous solution to the discrete domain, which may introduce significant additional error [13]. Another group of methods approximately solve the problem by deterministic annealing, which can usually result in a suboptimal solution in reasonable time. A typical instance is the famous graduated assignment (GA) method [14], which is still considered to be a state-of-the-art algorithm [15], [16]. Besides, the recently proposed path following algorithms [13], [17]-[19] also belong to the deterministic annealing. By introducing the convex relaxation and concave relaxation of the original objective function, they push the continuous solution gradually to the discrete domain following a path, and usually result in a better discrete solution than that obtained by the hard-cut projection. Another group of methods explore the spectral property of matrices associated with the graphs. The first spectral algorithm for graph matching is proposed by Umeyama [20], which involves the spectral decomposition of the adjacency matrix for either graph. Another important spectral algorithm [21] aims at the rank-one approximation of an affinity matrix which precomputes and stores the pairwise similarity between graphs. This algorithm is further improved by many works [15], [22], [23]. Particularly, the work [23] introduces a discrete (integer) projection scheme by utilizing the conditional gradient ascent [24], [25] method, which significantly improves the matching accuracy. Other types of approximate algorithms include those based on dual decomposition [26], based on game theory [27], etc.

Random walk has been successfully applied to many network-related tasks, e.g., Google's Pagerank algorithm [28] in ranking problems, which witness its efficacy and efficiency. Therefore some researchers also tried to use random walk to approximately solve the graph matching problem. Two typical random walk-based graph matching algorithms [29], [30]

[^1]are introduced in Section II-A. Briefly, [29] origins from the difference minimization between adjacency matrices associated with the two graphs, and presents their random walk algorithm by proving its connection with the stationary distributions of the two graphs. The other algorithm [30] directly targets at the stationary distribution of the association graph of the two graphs, and presents their random walk method by further incorporating the continuous matching constraints, which is realized by the Sinkhorn method [31] in a personalized Pagerank way. Some other random walk-based graph matching algorithms include [32] and [33]. Basically speaking, random walk is applied to the graph matching algorithm by taking advantages of the techniques and explanations of random walk in ranking problems.

However, there is an important difference between the ranking problems and graph matching that the latter one is in nature a combinatorial optimization problem with specific discrete constraints. For instance, the commonly used bilinear objective function is adopted by many graph matching algorithms [18], [21], [22], [30], and from this perspective what makes graph matching different from other problems involving bilinear objective functions is its specific discrete constraints. Therefore, it may be inappropriate for the previous random walk algorithms to utilize continuous techniques without considering the discrete property. On the other hand, as surveyed above, some methods [13], [18], [19], [23] significantly improve the matching performance by introducing the graduated projection to the discrete domain instead of the hardcut projection, which also reflects the necessity of considering the discrete property in the optimization process.

Based on the above observations, in this paper we propose a novel random walk-based graph matching algorithm by incorporating both continuous and discrete constraints in the optimization process. Specifically, after interpreting graph matching by random walk, the continuous constraints are directly embedded in the random walk constraint in each iteration. Further, to get a better discrete matching solution, both the assignment matrix (vector) and the pairwise similarity measure between graphs are iteratively updated according the discrete constraints, which automatically leads the continuous solution to the discrete domain.

The remainder of this paper is organized as follows. The preliminaries for graph matching and random walk are first presented in Section II, together with detailed introductions to two representative random walk methods for graph matching. Then the main contributions of this paper are proposed in Section III, and experimentally verified in Section IV. Finally, Section V concludes this paper.

## II. Preliminaries: Graph Matching and Random Walk

Please refer to Table I in the Appendix for the explorations of notations used in this paper.

A graph $\mathcal{G}=\{V, E\}$ of size $M$ is defined by a finite vertex set $V=\{1,2, \ldots, M\}$ and an edge set $E \subseteq V \times V$. The labeled weighted graph is further defined by assigning an appearance descriptor [34]-[36] vector $l_{i_{G}} \in \mathbb{R}^{d_{l} \times 1}$ as a
label to the vertex $i_{G}$, and assigning a nonnegative pairwise descriptor vector $w_{i_{G} j_{G}} \in \mathbb{R}^{d_{w} \times 1}$ as a weight to the edge $i_{G j_{G}}$. For instance, when using vertices to represent the SIFT key points in an image, $l_{i_{G}}$ could be the appearance descriptor of size $d_{l}=128$ around the $i_{G}$ th point, and $w_{i_{G} j_{G}}$ could be a vector of size $d_{w}=2$ consisting of the distance descriptor and orientation descriptor between the points $i_{G}$ and $j_{G}$.

Association graph of size $M N$, or modular product of two graphs $\mathcal{G}$ of size $M$ and $\mathcal{H}$ of size $N$, has long been an important concept in graph matching, especially in subgraph matching problems. For instance, the exact graph matching problem finding the maximum common subgraph of two graphs, can be reduced to the problem of finding the maximum clique in their association graph [3]. In the proposed algorithm, we use a type of quasi-association graph $\mathcal{A}$, which certain inexact weighted graph matching algorithms [14], [18], [21], [30] also use or implicitly use. Similar to the traditional association graph, in $\mathcal{A}$ each assignment $\left(i_{G}, i_{H}\right)$ is treated as a vertex, and if $i_{G}$ and $j_{G}$ are adjacent in $\mathcal{G}$, and $i_{H}$ and $j_{H}$ are adjacent in $\mathcal{H}$, the vertices $\left(i_{G}, i_{H}\right)$ and $\left(j_{G}, j_{H}\right)$ are then adjacent in $\mathcal{A}$. Differently, in the traditional association graph, the vertex $\left(i_{G}, i_{H}\right)$ is also considered to be adjacent with $\left(j_{G}, j_{H}\right)$ if neither $i_{G}$ is adjacent with $i_{H}$ nor $j_{G}$ is adjacent with $j_{H}$, while in the quasi-association graph $\mathcal{A}$ the vertex $\left(i_{G}, i_{H}\right)$ is considered to be nonadjacent with $\left(j_{G}, j_{H}\right)$ if either $i_{G}$ is nonadjacent $j_{G}$ or $i_{H}$ is nonadjacent with $j_{H}$. One major advantage of this modification is that $\mathcal{A}$ is sparse if $\mathcal{G}$ and $\mathcal{H}$ are sparse. Note $\mathcal{A}$ is also a labeled weighted graph, where its label $l_{\left(i_{G}, i_{H}\right)}$ and weight $w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}$ are usually non-negative real values which, respectively, measure the affinity between $l_{i_{G}}$ and $l_{i_{H}}$, and the consistency between $w_{i_{G} j_{G}}$ and $w_{i_{H} j_{H}}$.

It is common in studying and exploring graphs to use matrices naturally associated with those graphs. The "weighted" adjacency matrix $\mathbf{G} \in \mathbb{R}^{M \times M}$ recording the adjacency and weights of edges is one most common type of matrix associated with the weighted graph. Specifically, $\mathbf{G}_{i_{G}, j_{G}}=w_{i_{G} j_{G}}$ if $i_{G}$ and $j$ are adjacent, and $\mathbf{G}_{i_{G}, j_{G}}=0$ otherwise. Hereafter the weighted adjacency matrix is abbreviated by adjacency matrix. The adjacency matrix associated with the quasi-association graph $\mathcal{A}$ is denoted by $\mathbf{A} \in \mathbb{R}^{M N \times M N}$, and defined by

$$
\begin{align*}
\mathbf{A}_{i, j} & =\mathbf{A}_{\left(i_{G}-1\right) N+i_{H},\left(j_{G}-1\right) N+j_{H}}  \tag{1}\\
& = \begin{cases}w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}, & \text { if vertices }\left(i_{G}, i_{H}\right) \text { and } \\
& \left(j_{G}, j_{H}\right) \text { are adjacent } \\
0, & \text { otherwise } .\end{cases} \tag{2}
\end{align*}
$$

The walk matrix $\mathbf{W}^{G} \in \mathbb{R}^{M \times M}$ associated with the graph $\mathcal{G}$ is closely related to its adjacency matrix $\mathbf{G}$. It is column stochastic by

$$
\begin{equation*}
\mathbf{W}_{i_{G}, j_{G}}^{G}=\frac{\mathbf{G}_{i_{G}, j_{G}}}{\sum_{i_{G}=1}^{M} \mathbf{G}_{i_{G}, j_{G}}} \tag{3}
\end{equation*}
$$

where $\mathbf{W}_{i_{G}, j_{G}}^{G}$ can be explained by the relative importance of $i_{G}$ from the perspective of $j_{G}$. And it can be used to encode the dynamics of a random walk on $\mathcal{G}$ [37].

A random walk is a stochastic process where a walker starts at a random vertex $j_{G}$, walks randomly to another vertex $i_{G}$ with the probability $\mathbf{W}_{i_{G}, j_{G}}^{G}$, and so on. Let $\mathbf{p}_{j_{G}}^{t} \in[0,1]$ denote
the probability of the walker on $j_{G}$ at the step $t$, and then at the step $t+1$ the probability on $i_{G}$ is

$$
\begin{equation*}
\mathbf{p}_{i_{G}}^{t+1}=\sum_{j_{G}} \mathbf{W}_{i_{G}, j_{G}}^{G} \mathbf{p}_{j_{G}}^{t} \tag{4}
\end{equation*}
$$

In matrix form, it is

$$
\begin{equation*}
\mathbf{p}^{t+1}=\mathbf{W}^{G} \mathbf{p}^{t} \tag{5}
\end{equation*}
$$

where the vector $\mathbf{p} \in[0,1]^{M}$ denotes the probability distribution over all the vertices in $\mathcal{G}$. Such an updating (5) is exactly the well-known power iteration [38]. At a certain step $t^{\prime}$, when the probability distribution does not change anymore, i.e., $\mathbf{p}^{t^{\prime}+1}=\mathbf{p}^{t^{\prime}}$, the stationary distribution $p^{*}=\mathbf{p}^{t^{\prime}}$ of the random walk is obtained. In this paper, the random walk on the quasi-association graph $\mathcal{A}$ is utilized, and its walk matrix $\mathbf{W} \in[0,1]^{M N \times M N}$ can be given following (3) by ${ }^{2}$

$$
\begin{equation*}
\mathbf{W}_{i, j}=\frac{\mathbf{A}_{i, j}}{\sum_{i=1}^{M N} \mathbf{A}_{i, j}} \tag{6}
\end{equation*}
$$

For further reference, the updating (5) is correspondingly rewritten by

$$
\begin{equation*}
\mathbf{p}^{t+1}=\mathbf{W} \mathbf{p}^{t} \tag{7}
\end{equation*}
$$

where the vector $\mathbf{p} \in[0,1]^{M N}$ is then the probability distribution over all the vertices of $\mathcal{A}$, i.e., all the possible assignments.

Matching two graphs $\mathcal{G}$ and $\mathcal{H}$ is to find an appropriate set of assignments between their vertices, where the assignments can be also represented by a matrix $\mathbf{X} \in\{0,1\}^{M \times N}$ by indicating the assignment between $i_{G}$ to $i_{H}$ with $\mathbf{X}_{i_{G}, i_{H}}=1$. If the one-toone mapping assumption is further considered, the assignment matrix $\mathbf{X}$ becomes a partial permutation matrix, that is
$\mathbf{X} \in \mathcal{D}:=\left\{\mathbf{X} \mid \sum_{i} \mathbf{X}_{i, j} \leq 1, \sum_{j} \mathbf{X}_{i, j}=1, \mathbf{X}_{i, j} \in\{0,1\}\right\}$.
The row-wise replica of $\mathbf{X}$ is denoted by $\mathbf{x}$.

## A. Existing Random Walk Methods

Before presenting our algorithm in the next section, two representative random walk algorithms for graph matching are first introduced based on the above mathematical notations, which are respectively, proposed by Gori et al. [29] and Cho et al. [30].

Specifically, to get $\mathbf{X}$, [29] origins from the difference minimization between two adjacency matrices $\mathbf{G}$ and permutated $\mathbf{H}$, i.e., $\min \left(\mathbf{G}, \mathbf{X H} \mathbf{X}^{T}\right)$. Under certain conditions it can be deduced that the difference between $p_{G}^{*}$ and $\mathbf{X} p_{H}^{*}$ is correspondingly minimized, where $p^{*}$ denotes the stationary distribution explained above. To make the vice-versa still hold, [29] instead minimizes the difference between $Z_{G}^{*}$ and $\mathbf{X} Z_{H}^{*}$, where the full rank matrix $Z^{*}$ named by the graph discrete spectrum matrix is made up of a series of $p^{*}$ with varying

[^2]model parameters. Then the matching result $\mathbf{X}$ is obtained by well developed bipartite graph matching techniques.

Based on the quasi-association graph $\mathcal{A}$, [30] first develops an affinity preserving random walk algorithm, which pursues the stationary distribution $\mathbf{p}^{*}$ of $\mathcal{A}$. Then the affinity preserving random walk algorithm is further improved by incorporating the continuous matching constraints in each iteration based on the Sinkhorn method [31] in a personalized Pagerank way. The latter algorithm is named by the reweighted random walk matching algorithm (RRWM), which achieves the state-of-theart performance.

Since the proposed method also adopts the quasiassociation graph $\mathcal{A}$ as in [30], it is obviously different from [29]. And a major feature distinguishing the proposed method from [30] is that it incorporates the discrete graph matching constraints in the optimization process, by iteratively updating the assignment probability vector and the pairwise similarity measure to lead the continuous solution automatically to the discrete domain. Another difference is that in the proposed method the continuous constraints are directly embedded in the random walk constraint in each iteration, rather than building an additional, continuous constraints-related term in [30].

## III. Proposed Method

Based on the above preliminaries, the contributions of this paper are presented in this section, with Section III-A applying random walk to graph matching and Section III-B incorporating graph matching constraints into the scheme.

## A. Graph Matching by Random Walk

Given two labeled weighted graphs $\mathcal{G}$ of size $M$ and $\mathcal{H}$ of size $N$, their quasi-association graph $\mathcal{A}$ is constructed as in Section II, and the adjacency matrix $\mathbf{A}$ associated with $\mathcal{A}$ is given by (1). Based on $\mathbf{A}$, the walk matrix $\mathbf{W}$ is given by (6). Then graph matching can be formulated by random walk as follows.

Assuming that each assignment $\left(i_{G}, i_{H}\right)$ is associated with an initial probability $P^{0}\left(\left(i_{G}, i_{H}\right)\right)$, iteratively the probability for an assignment $\left(i_{G}, i_{H}\right)$ at the step $t+1$ can be given by

$$
\begin{equation*}
P^{t+1}\left(\left(i_{G}, i_{H}\right)\right)=\sum_{\left(j_{G}, j_{H}\right)} P^{t}\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right) P^{t}\left(\left(j_{G}, j_{H}\right)\right) \tag{9}
\end{equation*}
$$

where $P\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right)$ is the conditional probability of the assignment $\left(i_{G}, i_{H}\right)$ given the assignment $\left(j_{G}, j_{H}\right)$. Since each entry $\mathbf{W}_{i, j}=\mathbf{W}_{\left(i_{G}-1\right) N+i_{H},\left(j_{G}-1\right) N+j_{H}}$ in the walk matrix can be regraded as the belief of the assignment $\left(i_{G}, i_{H}\right)$ from the perspective of the assignment $\left(j_{G}, j_{H}\right)$, in probabilistic manner it is equivalent to the conditional probability $P\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right)$, that is

$$
\begin{align*}
P^{t}\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right) & =\mathbf{W}_{\left(i_{G}-1\right) N+i_{H}},\left(j_{G}-1\right) N+j_{H}  \tag{10}\\
& =\mathbf{W}_{i, j} \tag{11}
\end{align*}
$$

Accordingly, given the probability distribution $\mathbf{p}$ over all the assignments as in Section II, there is

$$
\begin{equation*}
P^{t}\left(\left(i_{G}, i_{H}\right)\right)=\mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{t}=\mathbf{p}_{i}^{t} \tag{12}
\end{equation*}
$$

Denoting the $\left(i_{G}-1\right) N+i_{H}$ th row in $\mathbf{W}$ by $\mathbf{W}_{\left(i_{G}-1\right) N+i_{H},:}$, based on (10) and (12), the updating (9) can be equivalently written as

$$
\begin{equation*}
\mathbf{p}_{i}^{t+1}=\mathbf{W}_{i,:} \mathbf{p}^{t} \tag{13}
\end{equation*}
$$

which is exactly the random walk iteration (7). The stationary distribution $\mathbf{p}^{*}$ is obtained until the convergence of $\mathbf{p}$, which can be regarded as the softened matching result. Note the graph matching constraints are abandoned in the formulation, and only the random walk constraint is used, that is

$$
\begin{equation*}
\mathbf{p}^{T} \mathbf{1}=1, \mathbf{p}_{i} \in[0,1] \tag{14}
\end{equation*}
$$

By the above formulation, random walk is directly applied to the graph matching problem. An explicit shortcoming of such a straightforward generalization is that only the pairwise information stored in $\mathbf{W}$ is used, while the appearance cues, i.e., the vertex labels, are not considered. Therefore inspired by the personalized Pagerank algorithm, (7) is extended by

$$
\begin{equation*}
\mathbf{p}^{t+1}=(1-\alpha) \mathbf{W} \mathbf{p}^{t}+\alpha \mathbf{q} \tag{15}
\end{equation*}
$$

The vector $\mathbf{q} \in[0,1]^{M N}$ is related to the appearance cues, which is also a probability distribution over all the assignments, that is

$$
\begin{equation*}
\mathbf{q}^{T} \mathbf{1}=1, \mathbf{q}_{i} \in[0,1] \tag{16}
\end{equation*}
$$

And a feasible construction of $\mathbf{q}$ is

$$
\begin{equation*}
\mathbf{q}_{i}=\mathbf{q}_{\left(i_{G}-1\right) N+i_{H}}=\frac{l_{\left(i_{G}, i_{H}\right)}}{\sum_{\left(i_{G}, i_{H}\right)} l_{\left(i_{G}, i_{H}\right)}} \tag{17}
\end{equation*}
$$

where the label $l_{\left(i_{G}, i_{H}\right)}$ of vertex $\left(i_{G}, i_{H}\right)$ measures the affinity between $l_{i_{G}}$ and $l_{i_{H}}$ as introduced in Section II.

Note since the diagonal entries in $\mathbf{A}$ are all zeros, for convenience some graph matching algorithms [15], [21], [23] replace the zeros by the vertex similarity measures (i.e., $l_{i_{G}, i_{H}}$ ), and name the modified $\mathbf{A}$ by the affinity matrix. Following these algorithms, the previous random walk algorithm [30] constructs $\mathbf{A}$ in the same way, and further utilizes the modified $\mathbf{A}$ in the stochastic normalization of $\mathbf{W}$ as follows:

$$
\begin{align*}
\mathbf{W}_{\left(i_{G}-1\right) N+i_{H},\left(i_{G}-1\right) N+i_{H}} & =\frac{\mathbf{A}_{\left(i_{G}-1\right) N+i_{H},\left(i_{G}-1\right) N+i_{H}}}{\sum_{i=1}^{M N} \mathbf{A}_{i,\left(j_{G}-1\right) N+j_{H}}}  \tag{18}\\
& =\frac{l_{\left(i_{G}, i_{H}\right)}}{l_{\left(i_{G}, i_{H}\right)}+\sum_{\left(i_{G}, i_{H}\right)} w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}} \tag{19}
\end{align*}
$$

and

$$
\begin{align*}
\mathbf{W}_{\left(i_{G}-1\right) N+i_{H},\left(j_{G}-1\right) N+j_{H}} & =\frac{\mathbf{A}_{\left(i_{G}-1\right) N+i_{H},\left(j_{G}-1\right) N+j_{H}}}{\sum_{i=1}^{M N} \mathbf{A}_{i,\left(j_{G}-1\right) N+j_{H}}}  \tag{20}\\
& =\frac{w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}}{l_{\left(i_{G}, i_{H}\right)}+\sum_{\left(j_{G}, j_{H}\right)} w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}} . \tag{21}
\end{align*}
$$

However, the influence of such a modification on $\mathbf{W}$ for a random walk seems not well justified. Besides, because of different physical meanings, there is also a scaling problem between the vertex similarity $l_{\left(i_{G}, i_{H}\right)}$ and the edge similarity $w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}$ in [30]. Instead in this paper, we use the extension (15) to incorporate the appearance cues.

An advantage of the extension (15) is that it is convenient to incorporate the matching prior, which may be either obtained by elementary matching techniques, or reflect the preference of assignments. Denoting the matching prior by a probability distribution $\mathbf{r} \in[0,1]^{M N}$ over all the assignments, it is straightforward that the updating (15) becomes

$$
\begin{align*}
\mathbf{p}^{t+1}= & \left(1-\alpha_{1}-\alpha_{2}\right) \mathbf{W} \mathbf{p}^{t}+\alpha_{1} \mathbf{q}+\alpha_{2} \mathbf{r}  \tag{22}\\
= & \left(1-\left(\alpha_{1}+\alpha_{2}\right)\right) \mathbf{W} \mathbf{p}^{t} \\
& +\left(\alpha_{1}+\alpha_{2}\right)\left(\frac{\alpha_{1}}{\alpha_{1}+\alpha_{2}} \mathbf{q}+\frac{\alpha_{2}}{\alpha_{1}+\alpha_{2}} \mathbf{r}\right)  \tag{23}\\
= & \left(1-\alpha^{\prime}\right) \mathbf{W} \mathbf{p}^{t}+\alpha^{\prime} \mathbf{q}^{\prime} \tag{24}
\end{align*}
$$

where

$$
\begin{align*}
\alpha^{\prime} & =\alpha_{1}+\alpha_{2}  \tag{25}\\
\mathbf{q}^{\prime} & =\frac{\alpha_{1}}{\alpha_{1}+\alpha_{2}} \mathbf{q}+\frac{\alpha_{2}}{\alpha_{1}+\alpha_{2}} \mathbf{r} \tag{26}
\end{align*}
$$

The vector $\mathbf{q}^{\prime}$ is still a probability distribution over all the assignments, and therefore (22) actually uses the same formulation with (15).

Another advantage of the extension (15) is about the convergence. For a random walk on a graph, the stationary distribution is actually the primary right eigenvector of the walk matrix, which corresponds to the largest eigenvalue $\lambda_{1}=1$ [37]. The convergence of the scheme and the rate of convergence are closely related to the second largest eigenvalue $\lambda_{2}$ in magnitude. Specifically, if $\lambda_{2}<1$, the iteration would converge to a unique stationary distribution $\mathbf{p}^{*}$, and the rate of convergence is positively related to $1-\left|\lambda_{2}\right|$. By (15), the real walk matrix utilized is $(1-\alpha) \mathbf{W}+\alpha \mathbf{1} \mathbf{q}^{T}$, and it has been proved that $\lambda_{2}<1-\alpha$ [39]. Hence the convergence of the scheme can be guaranteed.

However, as explained in Section II, graph matching is a combinatorial optimization problem with specific discrete constraints. In the above scheme, the graph matching constraints are ignored in the optimization process. Below we show how to incorporate both continuous and discrete constraints in the optimization process.

## B. Incorporation of Graph Matching Constraints

In this section, we show how to introduce graph matching constraints into the above random walk scheme. First, the continuous constraints are directly embedded in the random walk constraint in each iteration. Then both the assignment probability vector $\mathbf{p}$ and the walk matrix $\mathbf{W}$ are iteratively updated according the discrete constraints, which automatically leads the continuous solution to the discrete domain.

1) Incorporation of Continuous Constraints: As mentioned in Section I, approximate algorithms are necessary for the NP-complete graph matching problems. Quite a number of approximate algorithms [13], [21]-[23], [30] involve relaxing the discrete constraints to the continuous constraints. And for those algorithms utilizing the discrete domain $\mathcal{D}$ [defined in (8)], a most reasonable continuous relaxation may be

$$
\begin{equation*}
\mathbf{X} \in \mathcal{C}:=\left\{\mathbf{X} \mid \sum_{i} \mathbf{X}_{i, j} \leq 1, \sum_{j} \mathbf{X}_{i, j}=1, \mathbf{X}_{i, j} \in[0,1]\right\} \tag{27}
\end{equation*}
$$

In the vector form, $\mathcal{C}$ can be rewritten as

$$
\begin{equation*}
\left\{\mathbf{x} \mid\left(\mathbf{1}_{M}^{T} \otimes \mathbf{I}_{N}\right) \mathbf{x} \leq \mathbf{1}_{N},\left(\mathbf{I}_{M} \otimes \mathbf{1}_{N}^{T}\right) \mathbf{x}=\mathbf{1}_{M}, \mathbf{x}_{i} \in[0,1]\right\} \tag{28}
\end{equation*}
$$

where $\mathbf{1}_{M}$ denotes an all-one vector of size $M, \mathbf{I}_{N}$ denotes the identity matrix of size $N$, and $\otimes$ denotes the Kronecker product of two matrices. The reasonability is reflected in the doubly stochastic form of the continuous assignment matrix $\mathbf{X}$ in $\mathcal{C}$, which is interpretable in a probabilistic manner, i.e., $\mathbf{X}_{i_{G}, i_{H}}$ denotes the probability of $i_{G}$ in $\mathcal{G}$ being assigned to $i_{H}$ in $\mathcal{H}$ under the one-to-one matching assumption. Besides, some algorithms [18], [19] also take advantage of the property that $\mathcal{C}$ is the convex hull of $\mathcal{D}$ with $\mathcal{D}$ as its extreme point set. Therefore, the incorporation of continuous constraints $\mathcal{C}$ in the optimization process may improve the performance. Thanks to the fact that for $\mathbf{x}_{i} \in \mathcal{C}, \sum_{i} \mathbf{x}_{i}=M$ is always a constant, in the optimization process $\mathbf{x}$ can be constrained by embedding (28) in the random walk constraint (14) as follows:

$$
\begin{array}{r}
\mathcal{C}_{1}:=\left\{\mathbf{p} \left\lvert\,\left(\mathbf{1}_{M}^{T} \otimes \mathbf{I}_{N}\right) \mathbf{p} \leq \frac{\mathbf{1}_{N}}{M}\right.,\left(\mathbf{I}_{M} \otimes \mathbf{1}_{N}^{T}\right) \mathbf{p}=\frac{\mathbf{1}_{M}}{M}\right. \\
\left.\mathbf{p}_{i} \in\left[0, \frac{1}{M}\right]\right\} . \tag{29}
\end{array}
$$

Unfortunately, after the updating of $\mathbf{p}$ by (15), though the random walk constraint (14) still hold, the combined constraints (29) are broken. Therefore $\mathbf{p}$ is normalized based on the Sinkhorn method [31] (see steps 5 and 6 in Algorithm 1). Note except the updating step by (15), in the whole Algorithm 1 including the incorporation of discrete constraints, there is always $\mathbf{p} \in \mathcal{C}_{1}$.
2) Incorporation of Discrete Constraints-p: Though the continuous constraints are incorporated in the optimization process, still only a continues matching result $\mathbf{p}^{*}$ rather than a discrete one is obtained. And as mentioned in Section I, a hard-cut projection of $\mathbf{p}^{*}$ directly to the discrete domain, may introduce significant additional error [13], while the graduated projection-based methods [13], [19], [23] would improve the matching performance. Therefore below the discrete constraints are further incorporated in the optimization process to lead the continuous solution gradually to the discrete domain, which consists the updating of both $\mathbf{p}$ and $\mathbf{W}$. The common basic idea is to enhance the more probable assignments and at the same time weaken the less probable assignments in each iteration.

Specifically, the updating of $\mathbf{p}$ consists of the computation of the projection direction $\mathbf{e}$, and the search of the step size $s$. First, corresponding to (29), a discrete domain is defined by

$$
\begin{array}{r}
\mathcal{D}_{1}:=\left\{\mathbf{p} \left\lvert\,\left(\mathbf{1}_{M}^{T} \otimes \mathbf{I}_{N}\right) \mathbf{p} \leq \frac{\mathbf{1}_{N}}{M}\right.,\left(\mathbf{I}_{M} \otimes \mathbf{1}_{N}^{T}\right) \mathbf{p}=\frac{\mathbf{1}_{M}}{M}\right. \\
\left.\mathbf{p}_{i}=\left\{0, \frac{1}{M}\right\}\right\} \tag{30}
\end{array}
$$

which is the same with $\mathcal{D}$ except for a constant multiplication factor $M$. Then the projection direction $\mathbf{e}$ is given by

$$
\begin{equation*}
\mathbf{e}=\mathbf{y}-\mathbf{p}, \mathbf{y} \in \mathcal{D}_{1} \tag{31}
\end{equation*}
$$

which is a direction from the current point $\mathbf{p}$ to a discrete point $\mathbf{y}$. Particularly, the discrete point $\mathbf{y}$ is obtained by the
projection of the doubly stochasticized, exponentially inflated $\mathbf{p}$, rather than the direct projection of $\mathbf{p}$. More specifically, the exponential operator is applied to $\mathbf{p}$ as follows:

$$
\begin{equation*}
\mathbf{p}^{\prime}=\exp \left(\frac{\mathbf{p}}{\operatorname{mean}(\mathbf{p})}\right)=\exp (M N \mathbf{p}) \tag{32}
\end{equation*}
$$

and then $\mathbf{p}^{\prime}$ is doubly stochasticized also by the Sinkhorn method (steps 7.2 and 7.3 in Algorithm 1). These operations are intended to emphasize the more reliable assignments and depress the less reliable ones. Then $\mathbf{y}$ is obtained by finding the nearest point in $\mathcal{D}_{1}$ to $\mathbf{p}^{\prime}$, which can be formulated by the following linear assignment problem:

$$
\begin{align*}
& \mathbf{y}=\arg \max _{\mathbf{y}} \mathbf{y}^{T} \mathbf{p}^{\prime} \\
& \text { s.t. } \mathbf{y} \in \mathcal{D}_{1} \tag{33}
\end{align*}
$$

and further be solved by for instance the efficient Kuhn-Munkres algorithm [40]. For the step size $s$, since the proposed algorithm does not explicitly involve an objective function, we choose to find the point $\mathbf{p}+s \mathbf{e}, s \in[0,1]$ maximizing

$$
\begin{equation*}
(\mathbf{p}+s \mathbf{e})^{T} \mathbf{A}(\mathbf{p}+s \mathbf{e}) \tag{34}
\end{equation*}
$$

which, similar as in [18] and [23], is a quadratic optimization problem with one variable $s$. Therefore, the step size can be found by exact line search as follows:

$$
s= \begin{cases}s^{\prime}, & \text { if } 0 \leq s^{\prime} \leq 1  \tag{35}\\ 0, & \text { if } s^{\prime}<0 \\ 1, & \text { if } s^{\prime}>1\end{cases}
$$

where

$$
\begin{equation*}
s^{\prime}=\frac{\mathbf{e}^{T}\left(\mathbf{A}+\mathbf{A}^{T}\right) \mathbf{p}}{2 \mathbf{e}^{T} \mathbf{A} \mathbf{e}} \tag{36}
\end{equation*}
$$

Then in each iteration, the continuous solution is updated toward the discrete domain $\mathcal{D}_{1}$ by

$$
\begin{equation*}
\mathbf{p}=\mathbf{p}+s \mathbf{e} \tag{37}
\end{equation*}
$$

3) Incorporation of Discrete Constraints-W: Reconsidering (9), heretofore the conditional probability $P^{t}\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right)$ is treated as an invariant. Below based on the updating of assignment probabilities (i.e., $\mathbf{p}$ ), the conditional probabilities (i.e., $\mathbf{W}$ ) are also updated in the optimization process, which could in return help to push $\mathbf{p}$ to discrete domain $\mathcal{D}_{1}$. The operation also origins from the idea that more probable assignments should be enhanced while the less probable ones should be weakened. Specifically, if at the step $t+1$ the probability $P\left(\left(i_{G}, i_{H}\right)\right)$ (i.e., $\left.\mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}\right)$ increases, then the conditional probabilities $P\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right), \quad \forall\left(j_{G}, j_{H}\right)$ (i.e., $\left.\mathbf{W}_{\left(i_{G}-1\right) N+i_{H},:}\right)$ should also be increased, and similarly if $P\left(\left(i_{G}, i_{H}\right)\right)$ decreases, then $P\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right), \forall\left(j_{G}, j_{H}\right)$ should also be decreased. In matrix form $\mathbf{W}$ is updated by

$$
\begin{equation*}
\mathbf{W}^{t+1}=\mathbf{E} \mathbf{W}^{t} \tag{38}
\end{equation*}
$$

where $\mathbf{E}$ is a diagonal matrix defined by

$$
\begin{equation*}
\mathbf{E}_{i, i}=\frac{\mathbf{p}_{i}^{t+1}}{\mathbf{p}_{i}^{t}} \tag{39}
\end{equation*}
$$

## Algorithm 1 Random Walk-Based Graph Matching

Input: Two labeled weighted graphs $\mathcal{G}$ and $\mathcal{H}$
1: Initialize the walk matrix $\mathbf{W}^{0}$ by (1) and (6), and the assignment probability vector by $\mathbf{p}^{0}=\frac{1_{M N}}{M N}$
2: Build the probability vector $\mathbf{q}$ by (17), and initialize the factor $\alpha$
3: repeat
4: Update $\mathbf{p}^{t+1}$ based on $\mathbf{W}^{t}$ and $\mathbf{p}^{t}$ by (15), i.e., $\mathbf{p}^{t+1}=(1-\alpha) \mathbf{W} \mathbf{p}^{t}+\alpha \mathbf{q}$
5: Repeat (incorporation of the continuous constraints)
5.1: $\quad \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{t+1}=\frac{\mathbf{p}_{\left(G_{G}-1\right) N+i_{H}}^{t+1}}{N \sum_{i_{G}=1}^{M} \mathbf{1}_{\left(i_{G}-1\right) N+i_{H}}^{t+1}}$
5.2: $\quad \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{t+1}=\frac{\mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{t+1}}{M \sum_{i_{H}=1}^{N} \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{t+1}}$

6: Until iteration maximum is reached
7: Further Update $\mathbf{p}^{t+1}$ (Incorporation of the discrete constraints: $\mathbf{p}$ )
7.1: Compute the exponentially inflated $\mathbf{p}^{\prime}$ by (32)
7.2: Repeat (Doubly stochasticization of $\mathbf{p}^{\prime}$ )
7.2.1: $\quad \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{\prime}=\frac{\mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{\prime}}{N \sum_{i_{G}^{\prime}=1}^{M} \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{\prime}}$
7.2.2: $\quad \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{\prime}=\frac{\mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{\prime}}{M \sum_{i_{H}=1}^{N} \mathbf{p}_{\left(i_{G}-1\right) N+i_{H}}^{\prime}}$
7.3: Until iteration maximum is reached
7.4: Compute the projection direction $\mathbf{e}$ by (31), i.e., $\mathbf{e}=\mathbf{y}-\mathbf{p}^{t+1}$, where $\mathbf{y} \in \mathcal{D}_{1}$ is obtained by (33)
7.5: Find the step size $s$ by (35)
7.6: Update $\mathbf{p}^{t+1}$ by (37), i.e., $\mathbf{p}^{t+1}=\mathbf{p}^{t+1}+s \mathbf{e}$
8: Update $\mathbf{W}^{t+1}$ (Incorporation of the discrete constraints: W) by (38), i.e., $\mathbf{W}^{t+1}=\mathbf{E W}^{t}$, where $\mathbf{E}$ is given by (39)
9: until $\mathbf{p}^{t+1}=\mathbf{p}^{t}$ and $\mathbf{p}^{t+1} \in \mathcal{D}_{1}$
10: Multiply $\mathbf{p}^{t+1}$ by $M$ to get the assignment vector $\mathbf{x}$ Output: An assignment vector $\mathbf{x}$

After the updating by (38), $\mathbf{W}^{t+1}$ is usually not in the standard walk matrix form, i.e., usually $\sum_{i=1}^{M N} \mathbf{W}_{i, j}^{t+1} \neq 1, \forall j$. Therefore, $\mathbf{W}^{t+1}$ is further normalized to make it column stochastic simply by

$$
\begin{equation*}
\mathbf{W}_{i, j}^{t+1}=\frac{\mathbf{W}_{i, j}^{t+1}}{\sum_{i=1}^{M N} \mathbf{W}_{i, j}^{t+1}} \tag{40}
\end{equation*}
$$

Note that in the implementation, the updating (38) and normalization (40) are only performed when there are no zero entries in both $\mathbf{p}^{t+1}$ and $\mathbf{p}^{t}$.

The whole scheme is summarized in Algorithm 1. The computational complexity is $\max \mathcal{O}(p q), \mathcal{O}\left(N^{3}\right)$ where $p$ and $q$ denote the edges numbers and $N$ denotes the vertex number. Specifically, $\mathcal{O}(p q)$ are related to operations on $\mathbf{W}$, such as $\mathbf{W p}$ or EW. And $\mathcal{O}\left(N^{3}\right)$ are mostly related to operations on $\mathbf{p}$, such as the linear assignment problem (33).

Below some related works are discussed from the algorithmic perspective. First, the proposed method is closely related to the existing random walk-based graph matching algorithms [29], [30], especially [30], as discussed in

Section II-A. The proposed algorithm is also related to the algorithms [13], [19], [23] which involve the graduated projection of the continuous solution to the discrete domain. The difference is that they typically build explicit objective functions, and take advantage of the conditional gradient method, or known as the Frank-Wolfe algorithm ${ }^{3}$ in the objective function optimization. Besides, the proposed algorithm is inspired by those algorithms [15], [18], [41] involving the updating of matrices encoding pairwise relations. These matrices include, for instance the affinity matrix [15], [18] as in (1), or the dissimilarity matrix [41]. And the proposed method generalizes their thoughts to the random walk-based graph matching. Some state-of-the-arts among these algorithms will be experimentally compared with below.

## IV. Experiments

In this section, the proposed algorithm is experimentally evaluated by comparing it with the following algorithms: the RRWM [30], the spectral matching technique (SM) [21], the GA algorithm [14], the integer projected fixed point method (IPFP) [23], and the probabilistic SM (PSM) [15]. The proposed algorithm is denoted by OUR. All the algorithms for comparison are reimplemented by us using the MATLAB software except IPFP, for which the MATLAB codes ${ }^{4}$ are publicly available. For RRWM, we use the same parameters suggested by Cho et al. [30]. For GA, the parameters are fixed to be same as in [18]. For IPFP, we follow [23] to use a flat, uniform continuous solution to initialize the assignment vector. For OUR, we empirically fixed $\alpha=0.05$ in all the experiments.

The comparisons are carried out on a synthetic point dataset, two benchmark graph matching datasets, i.e., the House sequence ${ }^{5}$ and the real-world Car \& Motorbike image datasets, ${ }^{6}$ and a handwritten Chinese character image dataset. On all the datasets, the matching accuracy is used as the comparison criterion, which is the ratio between the number of correct assignments and the number of ground truth assignments.

## A. Synthetic Points

The proposed method is first evaluated on 3-D synthetic spatial points. The two point sets are generated as follows. First randomly generate a point set $H=\left\{h_{j}\right\}_{j=1}^{N}, h_{j} \sim U[0,1]^{1 \times 3}$ by uniform sampling in a 3-D cubic; randomly generate a ground truth assignment vector $\mathbf{x}^{g t} \in \mathcal{D}$; get the second point set $G=\left\{g_{i}\right\}_{i=1}^{N}$ based on $H$ and $\mathbf{x}$ by

$$
\begin{align*}
& g_{i}=h_{j}+\eta, \eta \sim N(0, \sigma) \\
& \text { if } \mathbf{x}_{(i-1) N+j}^{g t}=1 \tag{41}
\end{align*}
$$

where $\eta$ is the additive noise. Each point set is represented by a graph, and the graph structure is built in a sparse manner by setting the edge density to be 0.1 . The two graph structures are different by disturbing the second one following the way

[^3]

Fig. 1. Three-dimensional synthetic point matching results. The upper plot shows the comparison with respect to graph size, and the lower plot shows the comparison with respect to noise level.


Fig. 2. Running time comparison in 3-D synthetic point matching.
in [13], that is to randomly add and remove $(1 / 2) \sigma p$ edges in the second graph, where $p$ denotes the total edge number. In this experiment, the globally normalized length of an edge is used as its 1-D edge weight, that is

$$
\begin{equation*}
w_{i_{G} j_{G}}=\frac{d_{i_{G} j_{G}}}{\max _{i_{G} \neq j_{G}} d_{i_{G} j_{G}}} \tag{42}
\end{equation*}
$$



Img. no. $=60$
Img. no. $=80$
Img. no. $=100$
Fig. 3. House samples. The image number is provided below each image, which is abbreviated by img. no. The ground truth points are shown in blue.


Fig. 4. House matching results.
where $d_{i_{G} j_{G}}$ denotes the length of the edge $i_{G} j_{G}$. And further, the edge weight $w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}$ in the quasi-association graph $\mathcal{A}$ is defined by

$$
\begin{equation*}
w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}=\exp \left(-\frac{\left(w_{i_{G} j_{G}}-w_{i_{H} j_{H}}\right)^{2}}{\sigma^{L}}\right) \tag{43}
\end{equation*}
$$

where the kernel width parameter $\sigma^{L}$ is empirically fixed as 0.15 . The vertex appearance descriptor is not considered to fairly compared the pairwise matching performance of different algorithms, and correspondingly in the proposed algorithm $\mathbf{q}$ in (15) is set to be a flat distribution, that is

$$
\begin{equation*}
\mathbf{q}=\frac{\mathbf{1}_{M N}}{M N} \tag{44}
\end{equation*}
$$

The performance is first compared with respect to the graph size, which are increased from $N=30$ to $N=60$ by a step size 3 with $M=N-5$ and $\sigma=0.05$. It is also compared with respect to $\sigma$, which is increased from $\sigma=0$ to $\sigma=0.1$ by a step size 0.01 with $M=40$ and $N=45$. The comparison results are illustrated in Fig. 1. It can be observed that OUR and GA achieve comparable performances, which are better than the other algorithms. Note that intuitively the matching accuracy should decrease as the graph size increases because larger problem size implies more local optimal points. By contrast, in the upper plot of Fig. 1, the matching accuracies are


Fig. 5. Typical House matching instances. The img. no. sep. is 50. Green/red lines, respectively, denote correct/incorrect assignments, and yellow lines denote graph structures.


Fig. 6. Running time comparison in House matching.
positively related with the graph size, which is mainly because that by our settings the outlier ratio decreases as the graph size increases.

The running time is compared with respect to the graph size, which is increased from $N=10$ to $N=60$ by a step size 5 with $M=N-5$ and $\sigma=0.05$. The results are shown in Fig. 2 in a logarithmic manner, by which the slope rates empirically indicate the computational complexities. Specifically, the slope rates of RRWM, SM, and IPFP are around $3.8 \pm 0.3$, and those of GA, PGM, and OUR are around $4.2 \pm 0.3$. The proposed method involves a comparable computational complexity with the other algorithms, which is about $\mathcal{O}\left(N^{4}\right)$. For a specified edge density, the graph contains $p=\mathcal{O}\left(M^{2}\right)$ edges. Theoretically, the computational complexity $\max \left(\mathcal{O}(p q), \mathcal{O}\left(N^{3}\right)\right)$ analyzed in Section III-B2 would be $\mathcal{O}\left(M^{2} N^{2}\right)$, similar to the above experimental results.

## B. House Sequence

The House sequence consists of 111 images, which are sequentially sampled from a video clip of rotating artificial House model. Some image samples are illustrated in Fig. 3. Basically, a larger separation between the sequence numbers of two images implies a more difficult matching task. Each image is manually labeled with 30 ground truth points as in [7].


Fig. 7. Car and Motorbike samples. The key points, including both ground truth points and outliers, are shown in blue. The outlier number in either image pair is 10 .


Fig. 8. Car and Motorbike matching results.
The graph structure is constructed by the Delaunay triangulation method. The edge weight and vertex label are set to be the same with the first experiment.

The matching accuracies of different algorithms are compared with respect to the image number separation (abbreviated by img. no. sep.), which is increased from 0 to 100


Fig. 9. Typical Car and Motorbike matching instances. Green/red lines, respectively, denote correct/incorrect assignments, and yellow lines denote graph structures.
with a step size of 10 . For each img. no. sep., 10 matching pairs of images are randomly selected without repetition, and there are totally 110 matching pairs. The matching results are shown in Fig. 4, from which we can observe that generally the accuracies of different algorithm decrease as the img. no. sep. increases, and OUR outperforms the other algorithms. Some matching instances by different algorithms are illustrated in Fig. 5.

The running time of different algorithms is also compared, and graph size is increased from 10 to 30 by a step size 2 . For each step, 30 matching pairs are randomly selected, and the img. no. sep. is fixed to be 50 . The results are shown in Fig. 6 also in a logarithmic manner. It can be observed that the slope rates are a little smaller than those in Fig. 2. The reduction may be due to the Delaunay triangulation method in building the graph structures, by which the edge number $p$ increases linearly with respect to the vertex number $M$.

## C. Car and Motorbike Dataset

The experimental comparisons are also carried out on the real-world Car and Motorbike image dataset, which consists of 30 Car image pairs and 20 Motorbike image pairs. Some samples of these image pair are shown in Fig. 7. In each image pair, a certain number (ranging from 15 to 52) of key points are manually labeled, and are associated with ground truth assignments. Similarly to the above experiment, Delaunay triangulation is used to construct the graph structure. Differently, in this experiment two types of pairwise information from both
the length and orientation aspects are used to compute the 2-D edge weight $w_{i_{G} j_{G}}$, of which the two entries are, respectively, denoted by $w_{i_{G} j_{G}}^{L}$ and $w_{i_{G} j_{G}}^{O}$ for convenience. The entry $w_{i_{G} j_{G}}^{L}$ is similarly defined as in (42), and $w_{i_{G} j_{G}}^{O}$ is defined by

$$
\begin{equation*}
w_{i_{G} j_{G}}^{O}=\frac{2 \angle i_{G j_{G}}}{\pi} \tag{45}
\end{equation*}
$$

where $\angle_{i_{G} j_{G}}$ denotes the acute angle between the edge $i_{G} j_{G}$ and the horizontal axis. Then the edge weight $w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}$ in $\mathcal{A}$ is defined by

$$
\begin{align*}
& w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}=0.5 \exp \left(-\frac{\left(w_{i_{G} j_{G}}^{L}-w_{i_{H} j_{H}}^{L}\right)^{2}}{\sigma^{L}}\right) \\
&+0.5 \exp \left(-\frac{\left(w_{i_{G} j_{G}}^{O}-w_{i_{H} j_{H}}^{O}\right)^{2}}{\sigma^{O}}\right) \tag{46}
\end{align*}
$$

and we empirically set the kernel width parameters to be $\sigma^{L}=$ 0.15 and $\sigma^{O}=0.2$. The label $l_{i_{G}}$ of vertex $i_{G}$ is obtained by the shape context descriptor [42], and based on it the vertex label $l_{\left(i_{G}, i_{H}\right)}$ in $\mathcal{A}$ is defined by

$$
\begin{equation*}
l_{\left(i_{G}, i_{H}\right)}=\exp \left(\frac{\left\|l_{i_{G}}-l_{i_{H}}\right\|^{2}}{\sigma^{V}}\right) \tag{47}
\end{equation*}
$$

where $\|\cdot\|$ denote the Euclidean norm and the $\sigma^{V}$ is the kernel width parameter set to be 1 . Then $\mathbf{q}$ is obtained by (17).

The matching accuracies of different algorithms are compared with respect to the outlier number, which is increased


Fig. 10. Handwritten Chinese character samples.


Fig. 11. Handwritten Chinese character matching results.
from 0 to 20 with a step size of 2 . The matching results are shown in Fig. 8, and we can observe that OUR outperform the other algorithms on the real-world image dataset. Some matching instances by different algorithms are illustrated in Fig. 9.

## D. Handwritten Chinese Character Images

The proposed algorithm is also applied to the handwritten Chinese character images. As shown in Fig. 10, there are four characters in the dataset and each one consists of ten image samples written by different persons fetched from [43]. We follow the settings in [41]. Specifically, for the first and third characters 28 ground truth points are manually labeled, and for the second and fourth characters 23 ground truth points are manually labeled. Besides, 10 outliers are randomly labeled in each image. The graph structure is built by taking into account both the character skeleton and randomness. The performance


Fig. 12. Typical Handwritten Chinese character matching instances by OUR. Green/red lines, respectively, denote correct/incorrect assignments, and yellow lines denote graph structures.

TABLE I
Some Notations Used in This Paper and Their Explanations

| Notations | Explanations |
| :---: | :---: |
| $\mathbb{R}$ | The real number space |
| $\mathbb{R}^{M \times N}$ | The real matrix space of size $M \times N$ |
| $\mathbb{R}^{M}$ | The real vector space of size $M$ |
| $\mathcal{A}, \mathcal{G}$, etc. | The labeled weighted graphs |
| $\mathbf{A}, \mathbf{G}$, etc. | The matrices |
| $\mathbf{A}_{i,:}, \mathbf{A}_{:, j}$ | The $i$ th row, and the $j$ th column of $\mathbf{A}$ |
| $\mathbf{A}_{i, j}$ | An entry in A |
| $\mathbf{x}, \mathbf{p}$, etc. | The vectors |
| $\mathbf{x}_{i}$ | An entry in $\mathbf{x}$ |
| $i_{G}, i_{H}$, etc. | The vertices |
| $\left(i_{G}, i_{H}\right)$ | An assignment between vertices $i_{G}$ in $\mathcal{G}$ and $i_{H}$ in $\mathcal{H}$, or a vertex in a (quasi-)association graph $\mathcal{A}$ |
| $i_{G} j_{G},\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)$ | The edges, where $i_{G} j_{G}$ is an edge between vertices $i_{G}$ and $j_{G}$, and $\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)$ is an edge between vertices $\left(i_{G}, i_{H}\right)$ and $\left(j_{G}, j_{H}\right)$ in a (quasi-)association graph $\mathcal{A}$ |
| $l_{i}, l_{\left(i_{G}, i_{H}\right)}$ | The vertex attributes |
| $w_{i_{G} j_{G}}, w_{\left(i_{G}, i_{H}\right)\left(j_{G}, j_{H}\right)}$, | The edge attributes |
| $P\left(\left(i_{G}, i_{H}\right)\right)$ | The probability of the assignment ( $i_{G}, i_{H}$ ) |
| $P\left(\left(i_{G}, i_{H}\right) \mid\left(j_{G}, j_{H}\right)\right)$ | The conditional probability |

is compared with respect to outlier number which is increased from 0 to 10 by a step size 1. The results are shown in Fig. 11, and some matching instances are given in Fig. 12.

## V. Conclusion

This paper proposes a novel random walk-based graph matching algorithm by incorporating both continuous and discrete constraints in the optimization process, of which the effectiveness is validated by experimental comparisons on benchmark graph matching datasets. And it also demonstrates the necessity of considering the combinatorial optimization nature of the graph matching problem when generalizing efficient continuous methods to it.

## Appendix

## Notations

Some notations used in this paper and their explanations are described in Table I.

## AcKNOWLEDGMENT

The authors would like to thank the anonymous reviewers whose comments have greatly improved this paper.

## REFERENCES

[1] H. L. Shang, Y. D. Tao, Y. Gao, C. Zhang, and X. L. Wang, "An improved invariant for matching molecular graphs based on VF2 algorithm," IEEE Trans. Syst., Man, Cybern., Syst., vol. 45, no. 1, pp. 122-128, Jan. 2015.
[2] A. Smalter, J. Huan, and G. Lushington, "GPM: A graph pattern matching kernel with diffusion for chemical compound classification," in Proc. IEEE Int. Conf. BioInform. BioEng., Athens, Greece, 2008, pp. 1-6.
[3] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," Int. J. Pattern Recognit. Artif. Intell., vol. 18, no. 3, pp. 265-298, 2004.
[4] M. S. Amin, R. L. Finley, and H. M. Jamil, "Top-k similar graph matching using tram in biological networks," IEEE/ACM Trans. Comput. Biol. Bioinform., vol. 9, no. 6, pp. 1790-1804, Nov./Dec. 2012.
[5] J. Leng and P. Jiang, "Mining and matching relationships from interaction contexts in a social manufacturing paradigm," IEEE Trans. Syst., Man, Cybern., Syst., vol. 47, no. 2, pp. 276-288, Feb. 2017, doi: 10.1109/TSMC.2016.2623630.
[6] B. W. Miners, O. A. Basir, and M. S. Kamel, "Understanding hand gestures using approximate graph matching," IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 35, no. 2, pp. 239-248, Mar. 2005.
[7] X. Yang, H. Qiao, and Z.-Y. Liu, "Feature correspondence based on directed structural model matching," Image Vis. Comput., vol. 33, pp. 57-67, Jan. 2015.
[8] F. Zhou, J. Brandt, and Z. Lin, "Exemplar-based graph matching for robust facial landmark localization," in Proc. IEEE Int. Conf. Comput. Vis., Sydney, NSW, Australia, 2013, pp. 1025-1032.
[9] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Fast graph matching for detecting CAD image components," in Proc. Int. Conf. Pattern Recognit., vol. 2. Barcelona, Spain, 2000, pp. 1034-1037.
[10] H. A. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, no. 5, pp. 522-525, May 1993.
[11] P. H. S. Torr, "Solving Markov random fields using semi definite programming," in Proc. Artif. Intell. Statist., Key West, FL, USA, 2003.
[12] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in Proc. Energy Minimization Methods Comput. Vis. Pattern Recognit., St. Augustine, FL, USA, 2005, pp. 171-186.
[13] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 12, pp. 2227-2242, Dec. 2009.
[14] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," IEEE Trans. Pattern Anal. Mach. Intell., vol. 18, no. 4, pp. 377-388, Apr. 1996.
[15] A. Egozi, Y. Keller, and H. Guterman, "A probabilistic approach to spectral graph matching," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 1, pp. 18-27, Jan. 2013.
[16] Y. Tian et al., "On the convergence of graph matching: Graduated assignment revisited," in Proc. Eur. Conf. Comput. Vis., Florence, Italy, 2012, pp. 821-835.
[17] Z.-Y. Liu, H. Qiao, X. Yang, and S. C. H. Hoi, "Graph matching by simplified convex-concave relaxation procedure," Int. J. Comput. Vis., vol. 109, no. 3, pp. 169-186, 2014.
[18] F. Zhou and F. De la Torre, "Factorized graph matching," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Providence, RI, USA, 2012, pp. 127-134.
[19] Z. Y. Liu and H. Qiao, "GNCCP-Graduated nonconvexity and concavity procedure," IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 6, pp. 1258-1267, Jun. 2014.
[20] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," IEEE Trans. Pattern Anal. Mach. Intell., vol. 10, no. 5, pp. 695-703, Sep. 1988.
[21] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in Proc. IEEE Int. Conf. Comput. Vis., Beijing, China, 2005, pp. 1482-1489.
[22] T. Cour, P. Srinivasan, and J. B. Shi, "Balanced graph matching," in Proc. Adv. Neural Inf. Process. Syst., 2006, pp. 313-320.
[23] M. Leordeanu, R. Sukthankar, and M. Hebert, "Unsupervised learning for graph matching," Int. J. Comput. Vis., vol. 96, no. 1, pp. 28-45, 2012.
[24] M. Frank and P. Wolfe, "An algorithm for quadratic programming," Naval Res. Logistics Quart., vol. 3, nos. 1-2, pp. 95-110, 1956.
[25] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in Proc. Int. Conf. Mach. Learn., Atlanta, GA, USA, 2013, pp. 427-435.
[26] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in Proc. Eur. Conf. Comput. Vis., Marseilles, France, 2008, pp. 596-609.
[27] A. Albarelli, S. R. Bulò, A. Torsello, and M. Pelillo, "Matching as a non-cooperative game," in Proc. IEEE Int. Conf. Comput. Vis., Kyoto, Japan, 2009, pp. 1319-1326.
[28] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual Web search engine," Comput. Netw., vol. 56, no. 18, pp. 3825-3833, 2012.
[29] M. Gori, M. Maggini, and L. Sarti, "Exact and approximate graph matching using random walks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 7, pp. 1100-1111, Jul. 2005.
[30] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in Proc. Eur. Conf. Comput. Vis., Heraklion, Greece, 2010, pp. 492-505.
[31] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," Pac. J. Math., vol. 21, no. 2, pp. 343-348, 1967.
[32] D. Emms, S. Severini, R. C. Wilson, and E. R. Hancock, "Coined quantum walks lift the cospectrality of graphs and trees," Pattern Recognit., vol. 42, no. 9, pp. 1988-2002, 2009.
[33] A. Robles-Kelly and E. R. Hancock, "String edit distance, random walks and graph matching," Int. J. Pattern Recognit. Artif. Intell., vol. 18, no. 3, pp. 315-327, 2004.
[34] D. G. Lowe, "Object recognition from local scale-invariant features," in Proc. IEEE Int. Conf. Comput. Vis., 1999, pp. 1150-1157.
[35] S. Belongie and J. Malik, "Matching with shape contexts," in Proc. IEEE Workshop Content-Based Access Image Video Libraries, Hilton Head Island, SC, USA, 2000, pp. 20-26.
[36] W. Kang and Q. Wu, "Pose-invariant hand shape recognition based on finger geometry," IEEE Trans. Syst., Man, Cybern., Syst., vol. 44, no. 11, pp. 1510-1521, Nov. 2014.
[37] D. A. Spielman, "Spectral graph theory and its applications," in Proc. 48th Annu. IEEE Symp. Found. Comput. Sci., 2007, pp. 29-38.
[38] G. H. Golub and C. F. Van Loan, Matrix Computations, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.
[39] T. H. Haveliwala, S. D. Kamvar, and A. D. Kamvar, "The second eigenvalue of the Google matrix," Tech. Rep., Stanford Univ., Stanford, CA, USA, 2003.
[40] H. W. Kuhn, "The Hungarian method for the assignment problem," Naval Res. Logistics Quart., vol. 2, nos. 1-2, pp. 83-97, 1955.
[41] X. Yang, H. Qiao, and Z.-Y. Liu, "Outlier robust point correspondence based on GNCCP," Pattern Recognit. Lett., vol. 55, pp. 8-14, Apr. 2015.
[42] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 4, pp. 509-522, Apr. 2002.
[43] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "CASIA online and offline Chinese handwriting databases," in Proc. Int. Conf. Document Anal. Recognit., 2011, pp. 37-41.

Xu Yang, photograph and biography not available at the time of publication.

Zhi-Yong Liu, photograph and biography not available at the time of publication.

Hong Qiao, photograph and biography not available at the time of publication.


[^0]:    Manuscript received January 11, 2017; accepted March 30, 2017. This work was supported in part by the National Key Research and Development Plan of China under Grant 2016YFC0300801, in part by the National Natural Science Foundation of China under Grant 61503383, Grant 61633009, Grant U1613213, Grant 61375005, and Grant 61210009, and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDB02080003. This paper was recommended by Associate Editor J. A. Lozano. (Corresponding author: Xu Yang.)
    X. Yang is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xu.yang@ia.ac.cn).
    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, also with the Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China.
    Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.
    Digital Object Identifier 10.1109/TSMC.2017.2693029

[^1]:    ${ }^{1}$ Almost all the graph matching problems are NP-complete except graph isomorphism [3], which is a GI-hard problem, i.e., neither known to be NP-complete nor known to have solution in polynomial time.

[^2]:    ${ }^{2}$ In [30], to preserve the original affinity relations, the walk matrix construction can also be constructed by adding an absorbing node as $\mathbf{W}=$ $\left[\begin{array}{cc}\mathbf{A} / d_{\max } & \mathbf{0} \\ 1-\mathbf{d}^{T} / d_{\max } & 1\end{array}\right]$ where $\mathbf{d}_{j}=\sum_{i} \mathbf{A}_{i, j}$ and $d_{\max }=\max _{j} \mathbf{d}_{j}$.

[^3]:    ${ }^{3}$ Despite not explicitly mentioned in [23], their optimization method is actually a generalization of the Frank-Wolfe algorithm.
    ${ }^{4}$ Available at https://sites.google.com/site/mariusleordeanu/
    ${ }^{5}$ Available at http://vasc.ri.cmu.edu/idb/html/motion/house/
    ${ }^{6}$ Available at https://sites.google.com/site/graphmatchingmethods/

