A Weight Regularized Relaxation Based Graph Matching Algorithm

Zhi-Yong Liu[1], Hong Qiao[1], and Lei Xu[2]

[1] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{zhiyong.liu,hong.qiao}@ia.ac.cn
[2] Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong
lxu@cse.cuhk.edu.hk

Abstract. In this paper we propose a regularized relaxation based graph matching algorithm. The graph matching problem is formulated as a constrained convex quadratic program, by relaxing the permutation matrix to a doubly stochastic one. To gradually push the doubly stochastic matrix back to a permutation one, a simple weighted concave regular term is added to the convex objective function. The concave regular function is not a concave relaxation of the original matching problem. However, it is shown that such a simple concave regular term has a comparative performance as the concave relaxation of the PATH following algorithm, which works only on undirected graphs. A concave-convex procedure (CCCP) together with the Frank-Wolfe algorithm is adopted to solve the matching problem, and some experimental results witness the state-of-art performance of the proposed algorithm.

Keywords: graph matching, CCCP, Frank-Wolfe algorithm, convex relaxation.

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 usually used to solve the object matching problem [1]. Graph matching involves identifying each vertices pair between two graphs in some optimal way, or inherently finding a *good* permutation matrix between the two adjacency matrices of both graphs. The problem is in nature a NP-hard combinatorial optimization problem with factorial complexity, except for some graphs with special structure, such as the planar graphs which has shown to be of polynomial complexity [2]. Consequently, an exhaustive enumeration based matching algorithm is computationally prohibited, except for some small-scale problems.

To make the problem computationally tractable, many approximate approaches have been proposed, to seek an acceptable trade-off between the complexity and

Y. Zhang et al. (Eds.): IScIDE 2011, LNCS 7202, pp. 9–16, 2012.

matching accuracy. As summarized in [3], approximate matching algorithms can be roughly categorized into three groups, tree search based methods, spectral methods and continuous optimization (relaxation techniques). Here, we concentrate on the relaxation techniques, which involve relaxing the combinatorial matching problem to a continuous one [4–6]. The key point lies in the fact that optimization over a continuous set is usually easier to be approximated than its discrete counterpart. Two well-known relaxation techniques in literature include graduated assignment [5] and PATH following algorithm [6], both of which work directly on adjacency matrices.

Given two graphs $G_D = (V_D, E_D)$ and $G_M = (V_M, E_M)$ where V and E respectively denote the sets of vertices and edges and by formulating the matching problem as

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

where A denotes adjacency matrix, P is a permutation matrix, $A_{P(M)}$ denotes the adjacency matrix of the permutated graph of G_M by P, $\mathcal{P}$ denotes the set of permutation matrix, and $f(\cdot)$ is a cost function which usually takes a convex quadratic form, both gradual assignment and PATH following algorithms relax domain of the objective from set of permutation matrix $\mathcal{P}$ to it convex hull, i.e., set of doubly stochastic matrix $\mathcal{D}$. In this way, eqn. (1) becomes a convex quadratic program. A soft assignment schema controlled by a parameter was introduced by the graduated assignment algorithm to control the non-convexity of the problem, and a double normalization is used to constrain the matrix a doubly stochastic one. As the parameter increases to large enough, a permutation matrix is expected to be obtained though usually a clean-up step is further needed.

Different from the gradual assignment algorithm, the PATH following algorithm introduces a weighted concave relaxation to control the non-convexity of the objective. Specifically, by adopting the square of Frobenius matrix norm as the cost, the objective in eqn. (1) is firstly relaxed as follows

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

where $\mathrm{vec}(P)$ is the column-wise vector representation 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 PATH following algorithm is given by

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

where $\triangle_{ij} = (D_M(i,i) - D_D(j,j))^2$, with D and L denoting the degree and Laplacian matrices of the graph, respectively. The concave relaxation holds the same global minimum as the original matching problem, that is, $\min_{P \in \mathcal{P}} f_c(P) = \min_{P \in \mathcal{P}} f_v(P)$. Based on the convex and concave terms above, the objective function of the PATH following algorithm is given by

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

where $\gamma \in [0, 1]$ controls the non-convexity of the objective: A large γ means that $f_{path}(P, \gamma)$ tends to be convex; by contrast, a small γ makes $f_{path}(P, \gamma)$ concave. Thus, by gradually decreasing γ for 1 to 0, the objective becomes finally a concave one, and its minimization results in a permutation matrix.

However, the PATH following algorithm cannot be used to solve the matching problem between directed graphs because the term in eqn. (3) cannot guarantee to be concave for directed graphs. 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 comparative performance as eqn. (3) on matching accuracy. 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.

2.1 Objective Function

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

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

Then, the graph matching problem is formulated as follows,

$$\begin{aligned} \min \ & f_\gamma(P) = \gamma \text{vec}(P)^T Q \text{vec}(P) - (1-\gamma)\text{vec}(P)^T \text{vec}(P) \\ \text{s.t.} \ & P \in \mathcal{D}. \end{aligned} \tag{6}$$

It is obvious that minimization of the concave term given by eqn. (5) results in an extreme point of $\mathcal{D}$, i.e., a permutation matrix. Thus, by gradually decreasing γ 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 degenerates to the convex relaxation, whose global minimization denoted by P_v can be obtained by such as the Frank-Wolfe algorithm. Actually, a permutation matrix can be directly obtained by an optimal assignment procedure which casts the doubly stochastic matrix P_v to a permutation matrix P_p via

$$P_p = \arg\max_{P \in \mathcal{P}} \langle P_v, P \rangle, \tag{7}$$

where $\langle \cdot \rangle$ denotes an inner product. The assignment can be solved by the Hungarian algorithm [7], with the computational complexity $O(N^3)$. Such a *hard-cut*

operation based graph matching algorithm, named QCV (quadratic convex), may however bring a big error in the result, as to be witnessed by the experimental results in section 3. By contrast, as γ gradually decreases, P is gradually pushed away from P_v in such a way that update of P is guided to approach a permutation matrix with 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 gotten 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.

In contrast to the above simple concave term, the concave relaxation given by eqn. (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 two concave terms have a comparative performance on accuracy (see Tab. 1 and Figs. 1 and 2 for details).

2.2 Algorithm

For each fixed γ, the objective function given by eqn. (6) is a constrained quadratic program which is generally neither convex nor concave, to solve which some local search technique seems to be unavoidable. 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, 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 respective to P^{k+1}, eqn. (8) is formulated as the following constrained convex quadratic program:

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

The Frank-Wolfe algorithm, as a reduced gradient method, is adopted to solve the above constrained convex quadratic program. 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

$$\min \langle \nabla f_{cccp}(P^t), X^t \rangle, \text{s.t. } X^t \in \mathcal{D}, \tag{10}$$

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

$$\nabla f_{cccp}(P) = 2\gamma(A'_D A_D P - A'_D P A_M - A_D P A'_M + P A_M A'_M) - 2(1-\gamma)P^*. \tag{11}$$

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 ε is a small positive constant, return P^{t+1}. Otherwise, let $t = t + 1$ and go back to step 2.

In the algorithm, the linear program in step 2 can be solved by the Hungarian algorithm with a complexity $O(N^3)$, and the line search can be efficiently implemented by for instance a backtracking algorithm [8]. 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 2.1.

Algorithm 2.1. 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 $P^{n+1} \leftarrow \text{CCCP}(P^n, \gamma), \gamma \leftarrow \gamma - \delta\gamma, n \leftarrow n + 1$
return (P^n)

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

3 Experimental Illustrations

Three experiments on synthetic data are conducted to evaluate the proposed algorithm. Five algorithms, including Umeyama's algorithm (U for short) [9], PATH following algorithm [6] (on undirected graphs only), QCV algorithm given by eqn. (7), graduated assignment algorithm [5] (GA for short), and the proposed algorithm are experimentally compared on both undirected and directed graphs.

For problems with small scale ($N = 8$ for instance), an exhaustive search is used to get the ground true solution. All of the algorithms were implemented by Matlab 2009b, with a MEX function to implement the Hungarian algorithm.

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 Tab. 1, from which it is witnessed that the PATH (on undirected graphs only) and our algorithms have a much better performance on accuracy than the U, QCV and GA algorithms. It is also observed that PATH and our algorithm have a comparative performance on accuracy, as echoed by the discussions in section 2.1.

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

graph types	error	OPT	U	QCV	GA	PATH	Ours
undirected graph models	mean	3.5473	8.7955	6.3547	6.3483	4.1323	4.0591
	std	1.0678	3.3016	2.5465	2.1100	1.4218	1.3254
directed graph models	mean	5.5269	11.0624	7.9136	9.6330	-	6.3928
	std	0.8921	2.2747	1.8446	2.1368	-	1.2297

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 is controlled by a noise level. Specifically, given a noise level $\delta \in [0, 1]$ and a randomly generated adjacency matrix A_D, A_M is gotten 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 $r_1 < \delta$, $(A_M)_{ij} \leftarrow 0$; or otherwise, $(A_M)_{ij} \leftarrow (A_M)_{ij} + \delta r_2$.
3. If $(A_M)_{ij} = 0$: if $r_1 < \delta$, $(A_M)_{ij} \leftarrow r_2$.
4. Randomly generate a permutation matrix P, and set $A_M \leftarrow PA_MP^T$.

The noise level 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 two types of graphs are shown in Fig. 1. 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 respective to graph size, on both accuracy and complexity. In the experiment, 10 groups of graph pairs with different sizes are included for comparison, with the size increasing from 5 to 50 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 0.1.

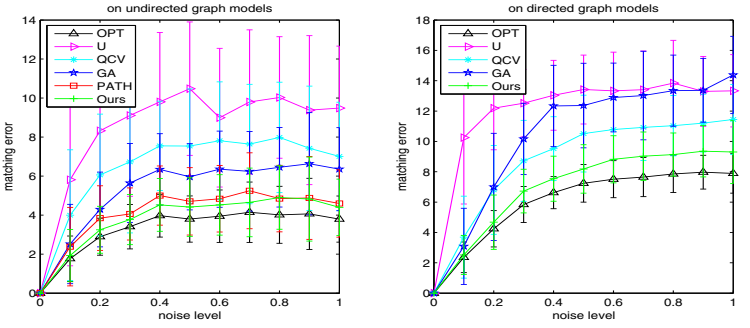

Fig. 1. Changes of the matching error with respective to noise levels, summarized from 100 random runs

On accuracy, similar experimental results as the above two are obtained on all of the 20 groups of graph pairs, as witnessed by Figs. 2. The time-cost of the five algorithms are shown in Fig. 3, in which the slopes of the five curves corresponding to U, QCV, GA, PATH, and our algorithm are around $2.412, 2, 633, 3.135, 2.762$, and 2.737, respectively, which imply that the GA suffers the biggest complexity and U is the simplest one.

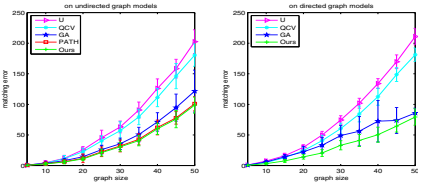

Fig. 2. Changes of the matching error with respective to graph sizes, summarized from 50 random runs on each size

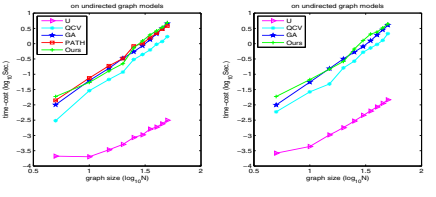

Fig. 3. Time-cost of the five algorithms with respective to the graph size, summarized from 50 random runs on each size

4 Conclusions

In this paper we showed that a simple weighted concave function has a comparative performance to 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 is usually difficult to find. Actually, to the best of our knowledge, only the concave relaxation on undirected graphs without self-loops has been reported in literature [6][1].

[1] Just before finishing the final version of this paper, we reported two type of concave relaxations for directed graph models, see [10, 11].

The CCCP together with the Frank-Wolfe algorithm is then proposed to solve the matching problem. On four different types of graphs, our algorithm showed a state-of-art performance on accuracy.

Acknowledgement. The work described in the paper was supported by a grand from NSFC (grant 60975002), and the National Basic Research Program of China (973 Program) (grant 2009CB825404).

References

1. Eshera, M.A., Fu, K.S.: An image understanding system using attributed symbolic representation and inexact graph-matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(5), 604–618 (1986)
2. Hopcroft, J.E., Wong, J.K.: Linear time algorithm for isomorphism of planar graphs (preliminary report). In: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC 1974, pp. 172–184. ACM, New York (1974)
3. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298 (2004)
4. Xu, L., Oja, E.: Improved Simulated Annealing, Boltzmann Machine and Attributed Graph Matching. In: Almeida, L.B., Wellekens, C.J. (eds.) EURASIP 1990. LNCS, vol. 412, pp. 151–160. Springer, Heidelberg (1990)
5. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(4), 377–388 (1996)
6. Zaslavskiy, M., Bach, F., Vert, J.P.: A path following algorithm for the graph matching problem. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(12), 2227–2242 (2009)
7. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2(1-2), 83–97 (1955)
8. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
9. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(5), 695–703 (1988)
10. Liu, Z.Y., Qiao, H., Xu, L.: An extended path following algorithm for graph matching problem. IEEE Transactions on Pattern Analysis and Machine Intelligence (to appear, 2012)
11. Liu, Z.Y.: Graph matching: a new concave relaxation fuction and algorithm. Automatica Sinica (to appear, 2012)