Int J Comput Vis
DOI 10.1007/s11263-014-0766-9

Sequential Monte Carlo for Maximum Weight Subgraphs with Application to Solving Image Jigsaw Puzzles

Nagesh Adluru · Xingwei Yang · Longin Jan Latecki

Received: 19 August 2012 / Accepted: 9 September 2014

Abstract We consider a problem of finding maximum weight subgraphs (MWS) that satisfy hard constraints in a weighted graph. The constraints specify the graph nodes that must belong to the solution as well as mutual exclusions of graph nodes, i.e., pairs of nodes that cannot belong to the same solution. Our main contribution is a novel inference approach for solving this problem in a sequential monte carlo (SMC) sampling framework. Usually in an SMC framework there is a natural ordering of the states of the samples. The order typically depends on observations about the states or on the annealing setup used. In many applications (e.g., image jigsaw puzzle problems), all observations (e.g., puzzle pieces) are given at once and it is hard to define a natural ordering. Therefore, we relax the assumption of having ordered observations about states and propose a novel SMC algorithm for obtaining maximum *a posteriori* estimate of a high-dimensional posterior distribution. This is achieved by exploring different orders of states and selecting the most informative permutations in each step of the sampling. Our experimental results demonstrate that the proposed inference framework significantly outperforms loopy belief propagation in solving the image jigsaw puzzle problem. In particular, our inference quadruples the accuracy of the puzzle assembly compared to that of loopy belief propagation.

Keywords Sequential Monte Carlo · Particle filtering · Sampling importance resampling · Maximum weight clique · Jigsaw puzzle problem · Graph search · Graph matching · QAP

Communicated by Hiroshi Ishikawa.

N. Adluru
University of Wisconsin, Madison, WI, USA
e-mail: nagesh.adluru@gmail.com

X. Yang
Machine Learning Science Group at Amazon.com, Seattle, WA, USA
e-mail: happyyxw@gmail.com

L. J. Latecki (✉)
Temple University, Philadelphia, PA, USA
e-mail: latecki@temple.edu

1 Introduction

Any correspondence problem can be viewed as an instance of a more general problem of finding maximum weight subgraphs (MWSs) in a graph thanks to the formulation in Horaud and Skordas (1989). An *association graph* is defined as a weighted graph $G = (V, E, a)$, where $V = P \times Q$ is a vertex set, $E \subseteq V \times V$ is a set of edges, and $a : E \to \mathbb{R}_{\geq 0}$ is the weight function. Hence each vertex $v_i \in V$ is a correspondence $v_i = (p_i, q_i)$, where $p_i \in P$ and $q_i \in Q$, i.e., p_i is an index of an element in P and q_i is an index of an element in Q that form the vertex v_i.

A specific example is an image jigsaw puzzle problem. With reference to Fig. 1, given a set of puzzle pieces P, shown in (b), and a board with square puzzle locations Q, shown in (c), the goal is to assign the puzzle pieces to "correct" locations. The original image in (a) is not given and hence such a prior is also not available.

Our jigsaw puzzle problem formulation follows Cho et al. (2010) in that the goal is to build the original, unknown image from non-overlapping square patches. This formulation is different from most of the previous approaches Kong and Kimia (2001); Goldberg et al. (2002); Radack and Badler (1982); Wolfson et al. (1988), where the shape of the puzzle pieces is utilized. Since our puzzle pieces all have the same shape of a square, the affinities among the puzzle pieces are

Published online: 10 October 2014

Fig. 1 The goal is to build the original image (**a**) given the jigsaw puzzle pieces (**b**). The original image is not known, thus, it needs to be estimated given the observations shown in (**b**). The empty *squares* in (**c**) form possible locations for the puzzle pieces in (**b**)

less reliable making our problem more challenging. Since the original image is not given, we also do not assume any priors on the target image layout. This is different from Cho et al. (2010), where such priors are also considered. As shown in Demaine and Demaine (2007) the jigsaw puzzle problem is NP-complete if the pairwise affinities among jigsaw pieces are unreliable.

Another example of a correspondence problem is finding a set of corresponding feature points between two images, which belongs to fundamental problems in computer vision. Due to its importance, there exist a huge number of papers addressing the correspondence problem. Many existing methods formulate the correspondence problems as problems of minimizing an energy function of a Markov random field (Maciel and Costeira 2003; Caetano et al. 2006; Jiang et al. 2007; Georgescu and Meer 2004; Cross and Hancock 1998; Zaslavskiy et al. 2009).

In the case of the jigsaw puzzle problem, P is a set of puzzle pieces and Q is a set of board locations. We identify the set of puzzle pieces and the set of board locations with their indices $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$ respectively. Hence $V = P \times Q$ is composed of pairs $v_i = (p_i, q_i)$, where p_i is an index of a puzzle piece that is assigned to a board location q_i. For example, the assignment of puzzle piece 3 to board location a shown in Fig. 1c, can be regarded as $v_1 = (p_1, q_1) = (3, a)$. Since we assume that $|P| = |Q| = n$, the graph has $|V| = n^2$ nodes. With each puzzle piece p_j, we also have an associated image I_j depicted on that piece. A solution to the jigsaw puzzle problem is a subset of exactly n pairs $v_j = (p_j, q_j)$. The jigsaw puzzle problem is an instance of the quadratic assignment problem (QAP), which is one of fundamental combinatorial optimization problems (Burkard et al. 1998). QAP is also known to be NP-hard (Sahni and Gonzalez 1976).

Here we formulate the QAP, and in particular, the jigsaw puzzle problem, as a problem of finding maximum weight subgraph (MWS) in a weighted association graph. In this formulation, a solution is a subset of the graph vertices, which represent selected assignment pairs. An important property of the MWS problem we consider is the existence of hard constraints that each solution must satisfy. In the case of the jigsaw puzzle, these are one-to-one matching constraints. They ensure that no puzzle piece is assigned to two different locations, and no board location has two puzzle pieces on it.

The main contribution of this paper is a novel inference framework for solving the MWS problem with hard constraints on a weighted graph (not necessarily an association graph), which is summarized in the algorithm in Fig. 2. We show that the proposed algorithm is an instance of a family of algorithms called sequential monte carlo methods. Our algorithm is inspired by the algorithmic aspects of the particle filtering (PF) framework and introduces permutations of states in the samples when exploring a high-dimensional posterior. We also prove that the algorithm in Fig. 2 approximates the solution of the constrained MWS problem with any precision if the number of particles is sufficiently large. The actual number of particles needed for achieving MAP in a particular instance depends on how close to the posterior are the sequential proposals (each permutation of state considers a different proposal in each iteration).

PF is a recursive Bayesian filter that belongs to Sequential Monte Carlo (SMC) methods. The classical PF framework has been developed for sequential state estimation like tracking (Isard and Blake 1998; Khan et al. 2004; Smith et al. 2005) or robot localization (Thrun 2002; Fox et al. 2000). There, the observations arrive sequentially and are indexed by their time stamps. Then the posterior density over the corresponding hidden states is recursively estimated. In many applications, e.g., image jigsaw puzzle problems, all observations (e.g., puzzle pieces) are given at once without any particular order. Therefore, we relax the assumption of having ordered observations and extend the PF framework to estimate the posterior density by exploring different orders of observations by selecting the most informative permutations of observations. We thus obtain the SMC algorithm with state permutations. It significantly broadens the scope of applications of the PF inference. One of our key ideas is the fact that it is possible to extend the importance sam-

pling from the proposal distribution so that different particles explore the state space along different dimensions. Then the particle weighting and resampling allow us to automatically determine most informative orders of observations (as permutations of state space dimensions).

It is possible to apply the classical PF framework as stochastic optimization to solve this problem by utilizing a fix order of states. However, by doing so, we would have selected an arbitrary order, and the puzzle construction may fail because of the selected order as can be seen in Sect. 6.2, Figs. 6 and 7. The classical approach would require an extremely large number of particles to overcome such limitations. Our framework on the other hand works with a relatively small number of particles.

From the point of view of graph search, the proposed algorithm utilizes a mixture of depth first and breadth first search for finding MWSs. When the weight distribution of particles is informative (has one or more clear peaks), the PF search acts like a depth first search but it may explore more than one graph regions simultaneously. In contrast, if the weight distribution of particles is close to uniform the PF search acts like a breadth first search.

The presented experimental results focus on the image jigsaw puzzle problem. We compare the solutions obtained by the proposed algorithm to the solutions of the loopy belief propagation under identical settings on the dataset from Cho et al. (2010). In particular, we use exactly the same dissimilarity-based compatibility of puzzle pieces. The proposed PF inference significantly outperforms the loopy belief propagation in all evaluation measures. The main measure is the accuracy of the label assignment, where the difference is most significant. The accuracy using loopy belief propagation is 23.7 % while that using the proposed SMC inference is over 95.3 % for puzzles with 108 pieces.

The rest of the paper is organized as follows. The problem of finding MWSs that satisfy hard constraints is introduced in Sect. 2. Then it is restated as a maximization problem of a probability density function (pdf) on a random field in Sect. 3. The proposed algorithm is also introduced in Sect. 3. After an overview of the PF preliminaries in Sect. 4.1, the extension to SMC with state permutations is described in Sect. 4.2. The proposed SMC algorithm with state permutations is formulated in Sect. 4, where we also prove it is able to approximate the target pdf with any precision. Section 4.3 formally relates our algorithm with state permutations to PF proposal and PF weight functions. Based on this fact, we prove in Sect. 4.6 that our algorithm approximates the solution of the constrained MWS problem with any precision if the number of particles is sufficiently large. Related problems and approaches are described in Sect. 5. Section 6.1 provides implementation details related to solving the image jigsaw puzzle problem as a particular instance of the constrained MWS problem. Section 6.2 presents experimental results and compares them to Cho et al. (2010). Section 6.3 presents additional experiments on graph matching and quadratic assignment problems.

2 Constrained Maximum Weight Subgraphs

A weighted graph G is defined as $G = (V, E, a)$, where $V = \{v_1, \ldots, v_m\}$ is the vertex set, m is the number of vertices, $E \subseteq V \times V$, and $a : E \to \mathbb{R}_{\geq 0}$ is the weight function. Vertices in G correspond to data points, edge weights between different vertices represent the strength of their relationships, and self-edge weight respects importance of a vertex. As is customary, we represent the graph G with the corresponding weighted adjacency matrix, more specifically, an $m \times m$ symmetric matrix $A = (a_{ij})$, where $a_{ij} = a(v_i, v_j)$ if $(v_i, v_j) \in E$, and $a_{ij} = 0$ otherwise. A may be indefinite.

With every vertex v_i there is associated an observation z_i. We denote with $Z = \{z_1, \ldots, z_m\}$ the set of observations associated with graph vertices. We assume that the affinity matrix A depends on the observations Z. The observations are not necessarily different, i.e., two different graph nodes may have the same observations. For example, in the jigsaw puzzle problem, the observation z_i is the image I_{p_i} on the puzzle piece p_i, where $v_i = (p_i, q_i)$. Since q_i varies over all board locations, all vertices related to the same puzzle piece have the same observation.

As is often the case, we identify the vertex set V with its index set, i.e., $V = \{v_1, \ldots, v_m\} = \{1, \ldots, m\}$. For any subset $T \subseteq V$, G_T denotes a subgraph of G with vertex set $V_T = \{v_i, i \in T\}$ and edge set $E_T = \{(v_i, v_j) \mid (v_i, v_j) \in E, i \in T, j \in T\}$. The total weight of subgraph G_T is defined as

$$f(G_T) = \sum_{i \in T, j \in T} a_{ij}. \tag{1}$$

We can express T by an indicator vector $\mathbf{x} = (x_1, \ldots, x_m) \in \{0, 1\}^m$ such that $x_i = 1$ if $i \in T$ and $x_i = 0$ otherwise. Then $f(G_T)$ can be represented in a quadratic form $f(G_T) = f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$.

A *neighborhood* of vertex set T in graph G is given by a set of adjacent vertices that are not in T:

$$\mathcal{N}(T) = \{v \in V \mid \exists_{u \in T}\ A(v, u) > 0 \text{ and } v \notin T\}. \tag{2}$$

Of course, the neighborhood can be further restricted to a small number of nearest neighbors.

We are also given a symmetric relation $M \subseteq V \times V$ between vertices of the graph. We call M a *mutex* (short for mutual exclusion) relation. If $M(i, j) = 1$ then the two vertices i, j cannot belong to the same MWS. $M(i, i) = 0$ for all vertices i. In other words, mutex represents incompatible vertices that cannot be selected together. Formally, this

requirement can be expressed as a constraint on the indicator vector $\mathbf{x} \in \{0, 1\}^m$: if $M(i, j) = 1$, then $x_i + x_j \leq 1$.

We also define a set of indices of vertices that are incompatible with a vertex set $T \subset V$:

$$\text{mutex}(T) = \{j \in V \,|\, \exists_{i \in T}\ M(i, j) = 1\},$$

and a set of compatible vertices as vertices that can be added to T without violating the mutex constraints:

$$\text{com}(T) = V \setminus (\text{mutex}(T) \cup T).$$

Given a set $U \subseteq V$ of initial vertices that must be selected as part of the solution, we consider the following maximization problem

$$\begin{aligned} &\underset{x \in \{0,1\}^m}{\text{maximize}}\ \ f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} \quad \text{subject to} \\ &\text{(C1)}\ \forall i \in U\ \ x_i = 1 \ \text{ and } \ \text{(C2)}\ \mathbf{x}^T M \mathbf{x} = 0. \end{aligned} \tag{3}$$

Constraint (C1) ensures that the initial vertices $U \subseteq V$ are selected as part of the solution and (C2) ensures that all mutex constraints are satisfied. We assume that the problem (3) is well-defined in that there exists $\mathbf{x}$ that satisfies the three constraints (C1, C2).

The goal of (3) is to select a subset of vertices of graph G such that f is maximized and the constraints (C1, C2) are satisfied. Since f is the sum of pairwise affinities of the elements of the selected subset, the larger is the subset, the larger is the value of f. However, the size of the subset is limited by mutex constraints (C2).

A global maximum of (3) is called a *constrained maximum weight subgraph* (CMWS) of graph G. Since two vertices i, j such that $M(i, j) = 1$ cannot belong to CMWS, it makes sense to set $A(i, j) = 0$, which makes matrix A sparser. However, even if $A(i, j) = 0$, vertices i, j may both belong to the same maximal clique. Hence constraint (C2) is stronger than setting $A(i, j) = 0$. (Of course, setting $A(i, j) = -\infty$ guarantees that mutex constraints are satisfied, but it is equivalent to our constraints (C2).)

The problem (3) is a combinatorial optimization problem and is NP-hard Asahiro et al. (2002). Therefore, the discrete assignment $\mathbf{x} \in \{0, 1\}^m$ is usually relaxed to $\mathbf{x} \in [0, 1]^m$, i.e., each coordinate x_i of x is relaxed to a continuous variable in the interval [0, 1], for example, this is done in similar problems of finding dense subgraphs in Cho et al. (2010); Pavan and Pelillo (2007); Liu et al. (2010); Sontag et al. (2010). The relaxed problem can be solved with quadratic programming (QP) for which many solvers exist. However, a solution of the relaxed problem is usually not guaranteed to satisfy constraints (C1, C2). For example, the jigsaw puzzle solutions obtained by loopy belief propagation (Cho et al. 2010) often violate the constraint (C2) as demonstrated in our experimental results: one can observe in the second row in Fig. 5 that some puzzle pieces are assigned to several board locations, although (Cho et al. 2010) utilizes an explicit penalty to prevent this from happening. Another difficulty is related to discretization of the relaxed solution in order to obtain the final discrete assignment. For these reasons, and since for our application, it is very important that the constraints are satisfied, we treat (C1, C2) as hard constraints that cannot be violated, and solve problem (3) directly.

In general, our observation is that the proposed method performs extremely well when the graph node potentials are local, i.e., each graph node is only linked to a small number of other nodes or equivalently the affinity matrix A is sparse, as is the case for the correspondence graph of the image jigsaw puzzle problem. In contrast, when the graph node potentials are more global, i.e., many nodes have large numbers of neighbors, then we expect the relaxed methods to perform better.

If G is the association graph of the jigsaw puzzle problem, then the weight function $A(i, j)$ measures the compatibility of two assignments $v_i = (p_i, q_i)$ and $v_j = (p_j, q_j)$ for $i \neq j$:

- If q_i and q_j are adjacent board squares, i.e., they have a side in common, then $A(i, j)$ is proportional to the similarity of two images z_i and z_j on puzzle pieces p_i and p_j along their common side.
- If q_i and q_j are not adjacent board locations, then $A(i, j) = 0$.

In the special case when $i = j$, $A(i, i)$ measures the compatibility of assigning the puzzle piece p_i to board location q_i. Since we do not assume any prior on the image to be constructed, we set $A(i, i) = 0$ for $i = 1, \ldots, m$. However, in the problem of matching feature points between two images, it makes sense to define $A(i, i)$ as a similarity of texture around points p_i and q_i, e.g., the similarity of their SIFT features (Lowe 2004).

The hard constraints in (3) have the following form for the jigsaw puzzle problem: (C1) expresses an initial assignment, in particular, in all our experiments, $U = \{v_1\} = \{(p_1, q_1)\}$, where q_1 is the top left square and p_1 is the puzzle piece that correctly corresponds to the top left square. We observe that the selected location in the top left corner is usually less informative than locations in the middle of the puzzle board. (C2) simply ensures one-to-one correspondence between puzzle pieces and board locations.

3 SMC Algorithm for Constrained Maximum Weight Subgraphs

In this section we express (3) as maximization problem on a random field and introduce a novel SMC based algorithm for solving it.

By associating a random variable (RV) X_i with each vertex $i \in V$ of graph G, we introduce a random field with the neighborhood structure of graph G. Each RV can be assigned either 1 or 0, where $X_i = 1$ means that the vertex v_i is selected as part of the solution. The conditional probability of the assignment of values to all RVs is denoted as

$$P(X_1 = x_1, \ldots, X_m = x_m \mid Z) = p(\mathbf{x}|Z), \tag{4}$$

where $\mathbf{x} = x_{1:m} = (x_1, \ldots, x_m) \in \{0, 1\}^m$ is an indicator vector, and $Z = \{z_1, \ldots, z_m\}$ is a set of observations associated with graph vertices.

In Sect. 4.3, we define $p(\mathbf{x}|Z)$ so that vector $\mathbf{x}$ at which it obtains its maximum value approximates the maximum of f in (3). This allows us to focus on maximizing (4). Thus, our goal becomes to find values $\widehat{x_i}$ of coordinates x_i of the indicator vector $\mathbf{x}$ such that $\widehat{\mathbf{x}}$ satisfies constraints (C1, C2) and

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}\in\{0,1\}^m}{\operatorname{argmax}}\ p(\mathbf{x}|Z). \tag{5}$$

We define a *particle* (i) at time $t-1$ for $2 \le t \le m$ as vector $x^{(i)}_{1:t-1} = (x^{(i)}_1, \ldots, x^{(i)}_{t-1}) \in \{0, 1\}^{t-1}$. The 'particle' is a sample from the state space defined by the collection of indicator variables i.e. $\{0, 1\}^t$. The dimensionality increases with each iteration t until $t = m$. We have an associated weight with every particle $w(x^{(i)}_{1:t-1}) \ge 0$. The goal of the SMC algorithm is to recursively extend the particles starting at $t = 1$ until $t = m$ so that the weighted particles $\{x^{(i)}_{1:m}, w(x^{(i)}_{1:m})\}$ for $i = 1, \ldots, N$ represent samples from the target distribution (4), where N is the number of particles. Finally, we take the particle with the largest weight as the solution of (5).

The coordinates with value one of vector $x^{(i)}_{1:t-1}$ determine the subset of selected graph vertices for particle (i). Therefore, in our approach we select a subset of vertices $V^{(i)}_{t-1} = \{j_1, \ldots, j_l\} \subset V$ such that $x^{(i)}_j = 1$ if and only if $j = j_1, \ldots, j_l$.

The proposed algorithm for solving (5) is presented in Fig. 2. For simplicity of presentation we assume that $U = \{v_1\}$ in (C1), i.e., we assume that the first vertex must belong to the maximum weight clique.

In the proposal step, each particle (i) is multiplied to many follower particles, where each follower is obtained by adding one more vertex s that satisfies mutex constraints (C2).

We observe that if $t < m$, then $x^{(i)}_{1:t} \notin \{0, 1\}^m$ but we can obtain a solution to Eq. (4) by extending $x^{(i)}_{1:t}$ to $x^{(i)}_{1:m} \in \{0, 1\}^m$, where coordinates of $x^{(i)}_{1:m}$ not present in $x^{(i)}_{1:t}$ are set to zero. As can be seen from the above description the vectors of all obtained particles satisfy the three constraints (C1,C2). The algorithm in Fig. 2 contains an application depended constant γ.

0) **Initialization:** For $t = 1$ and for all particles $i = 1, \ldots, N$, set $V^{(i)}_1 = \{v_1\}$ and $w(V^{(i)}_1) = \exp \frac{A(1,1)}{\gamma}$, where $\gamma > 0$ is a constant.

For $t = 2, \ldots, m$ execute the steps 1)-3):

1) **Proposal:** For every particle $i = 1, \ldots, N$ and for every $s \in \mathcal{N}(V^{(i)}_{t-1}) \cap \mathrm{com}(V^{(i)}_{t-1})$, the followers (i,s) of particle (i) are given by

$$V^{(i,s)}_t = V^{(i)}_{t-1} \cup \{s\}. \tag{6}$$

If $\mathcal{N}(V^{(i)}_{t-1}) \cap \mathrm{com}(V^{(i)}_{t-1}) = \emptyset$, we set $V^{(i,s)}_t = V^{(i)}_{t-1}$.

2) **Importance weighting/evaluation:** An individual importance weight is assigned to each follower particle $V^{(i,s)}_t$ according to

$$w(V^{(i,s)}_t) = w(V^{(i)}_{t-1}) \cdot \exp \frac{A(s,s) + 2\sum_{k=1}^{l} A(s, j_k)}{\gamma}, \tag{7}$$

where $V^{(i)}_{t-1} = \{j_1, \ldots, j_l\} \subset V$.

3) **Resampling:** Sample with replacement N new particles form the current set of particles

$$\{(V^{(i,s)}_t, w(V^{(i,s)}_t)\}. \tag{8}$$

according to their normalized weights. We obtain a set of new particles $\{V^{(i)}_t\}_{i=1}^N$ that inherit the (un-normalized) weights from their parents.

4) **Maximization:** For $t = m$, the particle with the highest weight is the solution to (5).

Fig. 2 SMC Algorithm for Constrained MWS

Section 4 presents important modules needed to verify the correctness of the algorithm in Fig. 2. After introducing the PF preliminaries and SMC with state permutations in Sects. 4.1 and 4.2, respectively, we show that the proposed algorithm is a special instance of the SMC framework. Finally, in Sect. 4.6 we prove that that the particle with largest weight obtained by the algorithm approximates the solution of the constrained MWS problem in (3) with any precision if a sufficiently large number of particles is used.

4 Theory Behind the Algorithm

4.1 Particle Filter Preliminaries

In this section we review preliminary facts about the classic PF. They will be utilized in the following sections when we introduce the proposed framework for the main algorithm.

Given a sequence of RVs $(X_1, \ldots, X_m)$ and a corresponding sequence of observations $Z = (z_1, \ldots, z_m)$, i.e., here the RVs and the observations are ordered. The goal is to find values of these RVs that maximize the posterior distribution

$$P(X_1 = x_1, \ldots, X_m = x_m \mid Z) = p(x_{1:m} \mid Z),$$

where $x_{1:m} = (x_1, \ldots, x_m) \in \mathcal{X}^m$ is a state space vector representing the possible values of the RVs. We also know that each state x_t has a corresponding observation z_t for $t = 1, \ldots, m$. Thus, the goal is to find the values $\widehat{x}_t$ of states x_t such that

$$\widehat{x}_{1:m} = \underset{x_{1:m}}{\operatorname{argmax}}\ p(x_{1:m} \mid Z). \tag{9}$$

Although we will solve (9) in its general formulation, we should keep in mind that our actual goal is to derive a method for solving (5). In the PF framework it is a simple task to ensure that constraints (C1,C2) are satisfied. Since each particle carries a partial selection of graph vertices, we only need to check whether these vertices satisfy the constraints (C1,C2) for each particle. We can easily ensure this when generating the proposal distribution as is done in the algorithm in Fig. 2.

Equation (9) can be solved by approximating the posterior distribution with a finite number of samples in the framework of Bayesian Importance Sampling. Since it is usually difficult to draw samples from the probability density function (pdf) $p(x_{1:m}|Z)$, samples are drawn from a proposal pdf q, $x_{1:m}^{(i)} \sim q(x_{1:m}|Z)$ for $i = 1, \ldots, N$. Then the approximation is given by

$$p(x_{1:m}|Z) \approx \sum_{i=1}^{N} w^{(i)} \delta_{x_{1:m}^{(i)}}(x_{1:m}), \tag{10}$$

where $\delta_{x_{1:m}^{(i)}}(x_{1:m})$ denotes the delta-Dirac mass located at $x_{1:m}^{(i)}$ and

$$w^{(i)} = \frac{p(x_{1:m}^{(i)}|Z)}{q(x_{1:m}^{(i)}|Z)} \tag{11}$$

are importance weights of the samples. Typically the sample $x_{1:m}^{(i)}$ with the largest weight $w^{(i)}$ is then taken as the solution of (9).

Since it is still computationally intractable to draw samples from q due to high dimensionality of $x_{1:m}$, Sequential Importance Sampling is usually utilized. In the classical PF approaches, samples are generated recursively following the order of dimensions in state vector $x_{1:m} = (x_1, \ldots, x_m)$:

$$x_t^{(i)} \sim q_t(x|x_{1:t-1}, Z) = q_t(x|x_{1:t-1}, z_{1:t}) \tag{12}$$

for $t = 1, \ldots, m$, and the particles are built sequentially $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, x_t^{(i)})$ for $i = 1, \ldots, N$. The subscript t in q_t indicates from which dimension of the state vector the samples are generated. Since q factorizes as

$$q(x_{1:m}|Z) = q_1(x_1|Z) \prod_{t=2}^{m} q_t(x_t|x_{1:t-1}, Z), \tag{13}$$

we obtain that $x_{1:m}^{(i)} \sim q(x_{1:m}|Z)$. In other words, by sampling recursively $x_t^{(i)}$ from each dimension t according to (12) we obtain a sample from $q(x_{1:m}|Z)$ at $t = m$.

Since at a given iteration we have a *partial* state sample $x_{1:t}^{(i)}$ for $t < m$, we also need an evaluation procedure of this partial state sample. For this we observe that the weights can be recursively updated according to Thrun et al. (2005):

$$w(x_{1:t}^{(i)}) = \frac{p(z_t|x_{1:t}^{(i)}, z_{1:t-1}) p(x_t^{(i)}|x_{1:t-1}^{(i)})}{q_t(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t})}\, w(x_{1:t-1}^{(i)}). \tag{14}$$

The above equation is derived from (11) using Bayes rule. Consequently, when $t = m$, the weight $w(x_{1:m}^{(i)})$ of particle (i) recursively updated according to (14) is equal to $w^{(i)}$ (defined in (11)). Hence, at $t = m$, we obtain a set of weighted (importance) samples from $p(x_{1:m}|Z)$, which is formally stated in the following theorem Crisan and Doucet (2002):

Theorem 1 *Under reasonable assumptions on the sampling* (12) *and weighting functions* (14) *given in Crisan and Doucet (2002), $p(x_{1:m}|Z)$ can be approximated with weighted samples $\{x_{1:m}^{(i)}, w(x_{1:m}^{(i)})\}_{i=1}^{N}$ with any precision if N is sufficiently large. Thus, the convergence in* (15) *is almost sure:*

$$p(x_{1:m}|Z) = \lim_{N \to \infty} \sum_{i=1}^{N} w(x_{1:m}^{(i)}) \delta_{x_{1:m}^{(i)}}(x_{1:m}). \tag{15}$$

In many applications, the weight equation (14) is simplified by making a common assumption that $q_t(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t}) = p(x_t^{(i)}|x_{1:t-1}^{(i)})$, i.e., we take as the proposal distribution the conditional pdf of the state at time t conditioned on the current state vector $x_{1:t-1}^{(i)}$. This assumption simplifies the recursive weight update to

$$w(x_{1:t}^{(i)}) = w(x_{1:t-1}^{(i)}) p(z_t|x_{1:t}^{(i)}, z_{1:t-1}), \tag{16}$$

and implies that the samples are generated from

$$x_t^{(i)} \sim p_t(x|x_{1:t-1}^{(i)}). \tag{17}$$

Analogous to (12), p_t in (17) indicates the dimension of the state space from which the samples are generated.

We summarize the derived **standard PF algorithm** in Fig. 3. This procedure is called Sampling Importance Resampling (SIR). Resampling is an important part of any PF algorithm, since resampling prevents weight degeneration of particles (Thrun et al. 2005). Usually the weights of new particles after resampling are equal and set to $1/N$. However, it is indicated in Chen (2003) (p. 28) that the performance might

For every time step $t = 2, \ldots, m$ execute the following three steps:
1) **Importance sampling / proposal:** For every particle $i = 1, \ldots, N$ sample followers of particle (i) according to (17) (a special case of (12)) and set $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, x_t^{(i)})$.
2) **Importance weighting / evaluation:** An importance weight is assigned to each particle $x_{1:t}^{(i)}$ according to (16) (a special case of (14)).
3) **Resampling:** Sample with replacement N new particles form the current set of N particles

$$\{x_{1:t}^{(i)} | i = 1, \ldots, N\}$$

according to their weights. We obtain a set of new particles $x_{1:t}^{(i)}$ for $i = 1, \ldots, N$.

Fig. 3 Standard PF Algorithm

be improved if the resampled particles retain their weights or some variants of those. The details justifying such a heuristic can be found in Liu et al. (2001). Therefore, in our approach the new particles simply inherit weights from their parents.

We observe that from the point of view of finding subgraphs, the presented PF algorithm always searches the graph in the same order, which is simply the order of indices of graph nodes. While the fixed order is natural in tracking scenarios, where the order is determined by the time stamps, usually there is no such natural order of graph vertices. As an example consider the correspondence graph G of assigning 6 puzzle pieces to six board locations illustrated in Fig. 1. G has 36 vertices and each vertex is a pair (*puzzle piece index, location index*). If the vertex order happens to be so that the first vertex is $v_1 = (p_1, q_1) = (3, a)$, the second vertex is $v_2 = (p_2, q_2) = (5, f)$, and the third vertex is $v_3 = (p_3, q_3) = (1, b)$, where the puzzle pieces are numbered as in Fig. 1b, then Fig. 1c illustrates the state of the particle $x_{1:3}^{(i)} = (1, 0, 1)$, representing the following value assignments to RVs: $X_1 = 1, X_2 = 0, X_3 = 1$ This means the first and the third vertices are selected, and the second is not selected. Of course, from the point of view of solving the jigsaw puzzle, this means that puzzle pieces numbered 3 and 1 are assigned locations (a) and (b), correspondingly. The problem is that v_1 is not related to v_2, therefore, the value assignment to X_2 is not influenced by the assigned value to X_1. In contrast, the value assignment to X_3 is influenced by the assigned value to X_1. Intuitively, we would like to dynamically determine the order of RVs so that their value assignment is influenced by the already assigned values.

4.2 Extension to Permuted States

The key idea of the proposed approach is to explore different orders of the states $(x_{i_1}, \ldots, x_{i_m})$ such that the corresponding sequences of observations $(z_{i_1}, \ldots, z_{i_m})$ is most informative. This way we are able to utilize the most informative observations first. To achieve this we modify the proposal so that the importance sampling is performed for every dimension not yet represented by the current particle.

To formally define the proposed sampling rule, we need to explicitly represent different orders of states with a permutation $\sigma : \{1, \ldots, m\} \to \{1, \ldots, m\}$. When $t < m$, we actually have an injection $\sigma : \{1, \ldots, t\} \to \{1, \ldots, m\}$, but we will still call it a permutation, since it is a permutation restricted to a subset. We use the shorthand notation $\sigma(1:t)$ to denote $(\sigma(1), \sigma(2), \ldots, \sigma(t))$ for $t \leq m$. Each particle (i) now can have a different permutation $\sigma^{(i)}$ of RV (or state dimensions) represented by vector $x_{\sigma(1:t)}^{(i)}$. Observe that a sequence of states $x_{\sigma(1:t-1)}$ visited before time t may be any subsequence $(i_1, \ldots, i_{t-1})$ of $t-1$ different indices in $\{1, \ldots, m\}$.

We define $\overline{\sigma^{(i)}(1:t-1)} = \{1, \ldots, m\} \setminus \sigma^{(i)}(1:t-1)$, i.e., the indices in $1:m$ that are not present in $\sigma^{(i)}(1:t-1)$ for $t \leq m$. We are now ready to formulate the proposed importance sampling. At each iteration $t \leq m$, for each particle (i) and for each $s \in \overline{\sigma^{(i)}(1:t-1)}$, we sample

$$x_s^{(i)} \sim p_s(x | x_{\sigma(1:t-1)}^{(i)}). \tag{18}$$

The subscript s at the posterior pdf p_s indicates that we sample values for state s. We generate at least one sample for each state $s \in \overline{\sigma^{(i)}(1:t-1)}$. This means that the single particle $x_{\sigma(1:t-1)}^{(i)}$ is multiplied and extended to several follower particles $x_{\sigma(1:t-1),s}^{(i)}$. Consequently, at iteration $t < m$ particle (i) may have at least $m - t + 1$ followers. Each follower is a sample from a different coordinate of the state vector. In contrast, in the standard application of rule (17), at each iteration t particle (i) has followers samples from coordinate $t+1$. We do not make any Markov assumption in (18), i.e., the new state $x_s^{(i)}$ depends on all previous states $x_{\sigma(1:t-1)}^{(i)}$ for each particle (i). Figure 4 summarizes the proposed SMC with state permutations (SMCSP) algorithm.

We observe that the particle weight evaluation in (20) is analogous to (16) in that the conditional probability of observation z_s is a function of two corresponding sequences of observations and states plus the state x_s. The key difference is that each particle may have a different order of RVs represented by the permutation $\sigma^{(i)}(1:t-1)$.

Sampling more than one follower of each particle and reducing the number of followers by resampling is known in the SMC literature as prior boosting (Gordon et al. 1993; Carpenter et al. 1999). It is used to capture multi-modal likelihood regions. The resampling in our framework plays an additional and a very crucial role. It selects the most informative orders of states. Since the weights of $w(x_{\sigma(1:t)}^{(i,s)})$ are determined by the corresponding order of observations $z_{\sigma^{(i)}(1:t-1)}$, and the resampling uses the weights to selects new particles $x_{\sigma(1:t)}^{(i)}$, the resampling determines the order of state dimensions. Consequently, the order of state dimensions is heavily determined by their corresponding observations, and this order may be different for each particle (i), i.e., each particle

For every time step $t = 2, \ldots, m$ execute the following three steps:
1) **Importance sampling / proposal:** For every particle $i = 1, \ldots, N$ sample values $x_s^{(i)}$ of particle (i) from each dimension $s \in \overline{\sigma^{(i)}(1:t-1)}$ according to (18), which we repeat here for completeness,

$$x_s^{(i)} \sim p_s(x|x_{\sigma^{(i)}(1:t-1)}^{(i)}), \tag{19}$$

and set $x_{\sigma^{(i,s)}(1:t)}^{(i,s)} = (x_{\sigma^{(i)}(1:t-1)}^{(i)}, x_s^{(i)})$ and $\sigma^{(i,s)}(t) = s$, which means that $\sigma^{(i,s)}(1:t) = (\sigma(1:t-1), s)$.
2) **Importance weighting/evaluation:** An individual importance weight is assigned to each follower particle $x_{\sigma^{(i,s)}(1:t)}^{(i,s)}$ according to

$$w(x_{\sigma^{(i,s)}(1:t)}^{(i,s)}) = w(x_{\sigma^{(i)}(1:t-1)}^{(i)}) p(z_s|x_{\sigma^{(i,s)}(1:t)}^{(i,s)}, z_{\sigma^{(i)}(1:t-1)}), \tag{20}$$

3) **Resampling:** Normalize the particle weights to sum to one and sample with replacement N new particles form the current set of particles

$$\{x_{\sigma^{(i,s)}(1:t)}^{(i,s)} | \, i = 1, \ldots, N, \; s \in \overline{\sigma^{(i)}(1:t-1)}\}. \tag{21}$$

according to the weights. We obtain a set of new particles $\{x_{\sigma^{(i)}(1:t)}^{(i)}\}_{i=1}^N$ that inherit their weights from their parents.

Fig. 4 SMC with state permutations (SMCSP)

may have a different order of dimensions $\sigma^{(i)}(1:m)$. This is in strong contrast to the classical SMC, where observations are considered only in one order. Another difference is that we have a finite dimension of the state space, while classical SMC methods usually deal with infinite dimensional state space representing time.

Therefore, at $t = m$ all state dimensions are present in each sample $x_{\sigma(1:m)}^{(i)}$. Hence we can reorder the sequence of state dimensions $\sigma^{(i)}(1:m)$ to form the original order $1:m$ by applying the inverse permutation $\left(\sigma^{(i)}\right)^{-1}$ and obtain $x_{1:m}^{(i)} = x_{\sigma^{-1}\sigma(1:m)}^{(i)}$, i.e., the state values are sorted according to the original state indices $1:m$ in each sample (i). This is the key idea in our proof of the following theorem.

Theorem 2 *Under reasonable assumptions on the sampling (19) and weighting functions (20) given in Crisan and Doucet (2002), $p(x_{1:m}|Z)$ can be approximated with weighted samples $\{x_{1:m}^{(i)}, w(x_{\sigma(1:m)}^{(i)})\}_{i=1}^N$ with any precision if N is sufficiently large. Thus, the convergence in (22) is almost sure:*

$$p(x_{1:m}|Z) = \lim_{N\to\infty} \sum_{i=1}^{N} w\left(x_{\sigma(1:m)}^{(i)}\right) \delta_{x_{1:m}^{(i)}}(x_{1:m}). \tag{22}$$

Proof By Theorem 1, we only need to show that $\{x_{1:m}^{(i)}, w(x_{\sigma(1:m)}^{(i)})\}_{i=1}^N$ represent weighted samples from $p(x_{1:m}|Z)$.

The key observation is that p and q are probabilities on joint distribution of m random variables, and as such the order of the random variables is not relevant, e.g., $p(x_2, x_3, x_1|Z) = p(x_1, x_2, x_3|Z)$. This follows from the fact that a joint probability is defined as the probability of the intersection of the sets representing events corresponding to the value assignments of the random variables, and set intersection is independent of the order of sets. Consequently, we have for every permutation σ

$$p(x_{\sigma(1:m)}|Z) = p(x_{1:m}|Z) \tag{23}$$
$$q(x_{\sigma(1:m)}|Z) = q(x_{1:m}|Z) \tag{24}$$

According to the proposed importance sampling (19), $x_{\sigma(1:m)}^{(i)}$ is a sample from $q(x_{\sigma(1:m)}|Z)$. Consequently, by (24), $x_{1:m}^{(i)} = x_{\sigma^{-1}\sigma(1:m)}^{(i)}$ is a sample from $q(x_{1:m}^{(i)}|Z)$ for each particle (i).

By the weight recursion in (20), and by (23) and (24)

$$w\left(x_{\sigma(1:m)}^{(i)}\right) = \frac{p(x_{\sigma(1:m)}^{(i)}|Z)}{q(x_{\sigma(1:m)}^{(i)}|Z)} = \frac{p(x_{1:m}^{(i)}|Z)}{q(x_{1:m}^{(i)}|Z)}. \tag{25}$$

Thus $\{x_{1:m}^{(i)}, w(x_{\sigma(1:m)}^{(i)})\}_{i=1}^N$ represent weighted samples from $p(x_{1:m}|Z)$.

We would like to note that although the theorem guarantees convergence of the samples to the posterior as $N \to \infty$, in practice, since we are interested only in MAP, we only need much smaller N. In fact empirically in our experimental results we achieve that by around $N = 200$. We would also like to note that because of our interest in the MAP, we are not concerned with the mixing time of the sampling procedure which in turn depends on the spectral gaps of the transition matrix or proposal functions (Montenegro and Tetali 2006). Since each particle will be exploring different permutations of the state space, the actual number of particles needed in achieving MAP depends on how many of the proposal functions are close to the posterior. In practice it suffices even if a small proportion of those are good. Accurately predicting or achieving precise bounds on the number of particles needed in such cases would involve characterizing and analyzing such "distances" between proposal and the posterior. Except for special instances of QAP such as Koopmans-Beckmann QAP, for which a polynomial time approximation scheme exists (Arora et al. 1996), it is NP-hard even to approximate the QAP (Sahni and Gonzalez 1976). Hence such analysis of the behavior is left outside the scope of our current work as it not clear how to even verify (much less to estimate) such distances given that the problem is NP-hard. That is not only that there is no known polynomial time algorithm for the problem but it is not even in NP (i.e. we can not verify the solution in polynomial time). Based on our experimental work (both jigsaw puzzle and synthetic experiments) the empirical evidence suggests that in practice $200 \leq N \leq 1{,}000$ works well for $m \sim 100$. Hence N in practice can be much smaller than $m!$.

4.3 SMC Algorithm for Constrained MWS as Instance of SMCSP

The goal of this section is to show that the SMC Algorithm for Constrained MWS in Fig. 2 is an instance of SMCSP algorithm in Fig. 4. Hence Theorem 2 applies to SMC Algorithm for Constrained MWS.

Due to Theorem 2, we need to define the proposal distribution $p_s(x|x^{(i)}_{\sigma(1:t-1)})$ in (19) and $p(z_s|x^{(i,s)}_{\sigma(1:t)}, z_{\sigma^{(i)}(1:t-1)})$ in the importance weight formula (20) in order to approximate our target distribution $p(x_{1:m}|Z)$ with particles according to (22). Both are defined in this section.

4.4 Proposal

We recall that x in $p_s(x|x^{(i)}_{\sigma(1:t-1)})$ can have either value one or zero. Since $x = 0$ means not selecting vertex s, which does not provide much information for our goal of finding maximal cliques, we set $p_s(x = 0|x^{(i)}_{\sigma(1:t-1)}) = 0$. Since in this case, we must have $p_s(x = 1|x^{(i)}_{\sigma(1:t-1)}) = 1$, we obtain that $x^{(i)}_s = 1$ if $x^{(i)}_s \sim p_s(x|x^{(i)}_{\sigma(1:t-1)})$.
Consequently, the sampling becomes deterministic. The other important consequence is that each $x^{(i)}_{\sigma(1:t)}$ is just a sequence of ones, i.e., $V^{(i)}_{\sigma(1:t)} = \{\sigma(1), \ldots, \sigma(t)\} \subset V$ is the sequence of selected vertices of particle $x^{(i)}_{\sigma(1:t)}$.

We define $p_s(x|x^{(i)}_{\sigma(1:t-1)})$ to ensure that constraints are satisfied (C1, C2). We simply ensure that (C1) is satisfied by the initialization. In order to ensure that (C2) is satisfied we set $p_s(x|x^{(i)}_{\sigma(1:t-1)}) = 0$ if $s \notin \text{com}(V^{(i)}_{t-1})$. We also set $p_s(x|x^{(i)}_{\sigma(1:t-1)}) = 0$ if $|V^{(i)}_{t-1}| = m$, i.e., particle (i) already has maximal possible number of selected vertices.

Finally, we set $p_s(x|x^{(i)}_{\sigma(1:t-1)}) = 0$ if $s \notin \mathcal{N}(V^{(i)}_{t-1})$ in order to ensure computational efficiency. Simply extending a given particle with a vertex that is not related to its current vertices does not bring any useful information, therefore, we do not allow such extensions. Hence a selected subgraph is always connected.

To summarize, the above definition of the proposal implies that the proposal is deterministic and for $s = 1, \ldots, m$ the followers of particle (i) are given by

$$x^{(i,s)}_{\sigma(1:t)} = \left(x^{(i)}_{\sigma(1:t-1)}, x_s\right), \tag{26}$$

where $x_s = 1$ and $s \in \mathcal{N}(V^{(i)}_{t-1}) \cap \text{com}(V^{(i)}_{t-1})$. Clearly, all such followers $x^{(i,s)}_{\sigma(1:t)}$ or equivalently their corresponding sets of selected vertices

$$V^{(i,s)}_t = \{\sigma(1), \ldots, \sigma(t-1), s\} \subset V$$

satisfy constraints (C1, C2). We obtain that the proposal in Fig. 2 is an instance of the proposal in (19).

4.5 Importance Weight

According to (20), we need to define $p(z_s|x^{(i,s)}_{\sigma(1:t)}, z^{(i)}_{\sigma(1:t-1)})$, where we recall that $\sigma^{(i,s)}(t) = s$. This means we need to define the probability of observation z_s conditioned on just selected vertex s (having observation z_s) and the current configuration of vertices $V^{(i)}_{t-1} = \{\sigma^{(i)}(1), \ldots, \sigma^{(i)}(t-1)\}$ and their corresponding observations $\{z^{(i)}_{\sigma(1)}, \ldots, z^{(i)}_{\sigma(t-1)}\}$. We define

$$p\left(z_s|x^{(i,s)}_{\sigma(1:t)}, z^{(i)}_{\sigma(1:t-1)}\right) = \exp \frac{A(s,s) + 2\sum_{k=1}^{t-1} A\left(s, \sigma^{(i)}(k)\right)}{\gamma}, \tag{27}$$

where we recall that $A(s, \sigma^{(i)}(k))$ measures how the observation z_s fits the observation $z^{(i)}_{\sigma(k)}$ for $k = 1, \ldots, t-1$. For example, for the jigsaw puzzle problem $v_s = (p_s, q_s)$, and (27) is proportional to how well the image z_s of puzzle piece p_s fits to the images of already placed puzzle pieces that are adjacent to board location q_s. Since a given board square can have at most 4 other adjacent squares, at most 4 terms in the sum in (27) are nonzero. $A(s,s) = 0$ for all $s = 1, \ldots, m$, since we have no jigsaw puzzle image prior. We obtain that the importance weight update in Fig. 2 is an instance of the weight update in (20).

Therefore, the SMC Algorithm for Constrained MWS is an instance of SMCSP in Fig. 4. This implies that Theorem 2 applies to the SMC Algorithm for Constrained MWS.

4.6 SMC Algorithm for Constrained MWS Approximates MWSs

Theorem 3 *The particle with the maximum weight obtained by SMC Algorithm for Constrained MWS in Fig. 2 approximates the solution of constrained MWS problem (3) with any precision if the number of particles N is sufficiently large.*

Proof Let $\{(x^{(i)}_{\sigma(1:t)}, w(x^{(i)}_{\sigma(1:t)}))\}_{i=1}^N$ be a weighted particle obtained by the algorithm in Fig. 2. Since it is a special instance of the PF with state permutations algorithm in Sect. 4.2, Theorem 2 applies to it, and we obtain that the set of weighted particles approximates the target distribution $p(x_{1:m}|Z)$ with any precision for sufficiently large N. Hence the particle with the largest weight approximates the maximum of $p(x_{1:m}|Z)$. Finally, by Lemma 4 (below), this particle approximates the maximum of the constrained MWS problem (3).

We denote with $[x^{(i,s)}_{\sigma(1:t)}]$ a vector $x^{(i,s)}_{\sigma(1:t)}$ padded with zeros to get a vector of dimension m. Directly from the definition of f in (1), we obtain

$$f\left([x^{(i,s)}_{\sigma(1:t)}]\right) = f\left([x^{(i)}_{\sigma(1:t-1)}]\right) + A(s,s) + 2\sum_{k=1}^{t-1} A\left(s, \sigma^{(i)}(k)\right). \quad (28)$$

We can view $f([x^{(i,s)}_{\sigma(1:t)}]) - f([x^{(i)}_{\sigma(1:t-1)}])$ as the gain in the target function f obtained after assigning value one to RV X_s, i.e., after adding vertex s to the current selection of vertices in particle. Hence (27) is the exponent of the gain.

Lemma 4 *At every iteration t of the algorithm in Fig. 2 it holds*

$$\log w\left(x^{(i)}_{\sigma(1:t)}\right) = f\left([x^{(i)}_{\sigma(1:t)}]\right) - t\log\gamma. \quad (29)$$

Proof We prove the identity by induction. Due to the initialization it holds for $t = 1$:

$$w\left(x^{(i)}_{\sigma(1)}\right) = exp\frac{A(1,1)}{\gamma} = \exp\frac{f\left([x^{(i)}_{\sigma(1)}]\right)}{\gamma} \quad (30)$$

Hence $\log(w(x^{(i)}_{\sigma(1)})) = f([x^{(i)}_{\sigma(1)}]) - \log\gamma$. Let us assume the identity holds for $t-1$:

$$\log\left(w(x^{(i)}_{\sigma(1:t-1)})\right) = f\left([x^{(i)}_{\sigma(1:t-1)}]\right) - (t-1)\log\gamma. \quad (31)$$

We show the identity for t. By (20) and (27), we have

$$\begin{aligned}\log\left(w(x^{(i)}_{\sigma(1:t)}\right) &= \log w\left(x^{(i)}_{\sigma(1:t-1)}\right) \\ &+ \log p\left(z_s|x^{(i,s)}_{\sigma(1:t)}, z^{(i)}_{\sigma(1:t-1)}\right) \\ &= \log w\left(x^{(i)}_{\sigma(1:t-1)}\right) + A(s,s) + 2\sum_{k=1}^{t-1} A\left(s,\sigma^{(i)}(k)\right) \\ &- \log\gamma\end{aligned}$$

and by the induction hypothesis (31) and by (28)

$$\begin{aligned}&= f\left([x^{(i)}_{\sigma(1:t-1)}]\right) - (t-1)\log\gamma + A(s,s) \\ &+ 2\sum_{k=1}^{t-1} A\left(s,\sigma^{(i)}(k)\right) - \log\gamma = f\left([x^{(i,s)}_{\sigma(1:t)}]\right) - t\log\gamma\end{aligned}$$

5 Related Work

The first work on jigsaw puzzle problem was reported in Freeman and Garder (1964). Since shape is an important clue for accurate pairwise relation, many methods Kong and Kimia (2001); Goldberg et al. (2002); Radack and Badler (1982); Wolfson et al. (1988) focussed on matching distinct shapes among jigsaw pieces to solve the problem. The pairwise relations among jigsaw pieces are measured by the fitness of shapes. There also exist approaches that consider both the shape and image content (Makridis and Papamarkos 2006; Nielsen et al. 2008; Yao and Shao 2003). Most methods solve the problem with a greedy algorithm and report results on just one or few images. Our problem formulation follows Cho et al. (2010). All puzzle pieces have the same shape of a square, hence only image content is considered. The key difference of our approach as compared to Cho et al. (2010) lies in the inference framework. While Cho et al. (2010) uses loopy belief propagation, we propose a novel inference framework based on PF. As reported in Sect. 6.2, we are able to quadruple the accuracy of the puzzle assembly of Cho et al. (2010).

The recent paper by Pomeranz et al. (2011) also follows the image jigsaw puzzle formulation in Cho et al. (2010), but does not use the same same affinity relations between puzzle patches. They focus on image content analysis of partially build puzzles and use a greedy algorithm for puzzle construction.

Particle filters (PF) belongs to SMC methods for model estimation based on simulation. There is large number of articles published on PF and we refer to two excellent books Doucet et al. (2001), Liue (2001) for an overview. PF is a powerful inference framework that is utilized in many applications. One of the leading examples is the progress in robot localization and mapping based on PF (Thrun et al. 2005). Classical examples of PF applications in computer vision are contour tracking (Isard and Blake 1996, 1998) and object detection (Ioffe and Forsyth 2001). All these approaches utilize PF in the classical tracking/filtering scenario with a predefined order of states and observations.

A preliminary version of the proposed algorithm with state permutations was published by the authors in a conference paper Yang et al. (2011), where it was directly applied to solving the jigsaw puzzle problem. In this paper we consider a more general problem of finding MWSs that satisfy hard constraints. While (Yang et al. 2011) considers direct assignment of the jigsaw puzzle pieces to board locations, here we cast this problem as a vertex selection problem in an association graph and solve it as a MWS problem.

The MWS problem is more general, since it is not restricted to association graphs. Our algorithm for solving this problem is introduced in Sect. 3. In comparison to Yang et al. (2011), Sects. 2, 3, 4.3, 4.6, and 6.1 are new, and they contain novel theoretical and algorithmic considerations. Moreover, even if focused on solving the image jigsaw puzzle problem, the Constrained Maximum Weight Clique PF Algorithm, introduced here, differs significantly from the algorithm in Yang et al. (2011). In particular, here our proposal is deterministic and particle weights are computed differently. This leads to significant performance improvement as compared to Yang et al. (2011), e.g., the accuracy of the label

Table 1 Comparisons with Suh et al. (2012)

	Suh et al. (2012)	Ours
Conceptual	There are two well-defined graphs which should be matched	Do not need well-defined individual graphs
Technical	Constraint(s) are simply assertions that one node has only one match	Unary inclusion and and mutual exclusion constraints
Technical	The proposal uses the association matrix has a normalization constant Z and a practical constant α for implementing the transition kernel	The proposal is independent of the association matrix and is simply driven by the constraints and is deterministic avoiding additional parameters.
Technical	Each particle is extended by only one follower	Each particle is extended by multiple followers based on the permutation of the previous iteration
Technical	No explicit notion of importance weighting step	Clearly uses the association matrix for computing importance weights
Technical	Do not retain the weights from the previous step	Retains the importance weights after resampling thus remembering iterations much further into the past
Technical	While at each iteration t there are exactly t matches and the maximum number of matches is $\min(n^P, n^Q)$, it is unclear how long the sampling procedure will need to be run for	The algorithm stops after $t = m$ iterations The performance comes from the richer exploration of the search space at each step using the permuted states and multiple followers for each particle
Conceptual	Standard SMC framework	A non-trivial modification to the SMC framework by introducing permuted states

assignment is increased from 69 % to over 95 % for puzzles with 108 pieces, which is over 25 %.

We have also formulated object detection in images as a direct assignment problem of edge fragments to model contours in two conference papers Lu et al. (2009); Yang and Latecki (2010) and solved this problem with two different variants the PF algorithm in Yang et al. (2011).

Relationship to Other Heuristic Search Frameworks. Beam search (Bisiani 1987) is a very generic class of heuristic search algorithms so that almost any search for combinatorial NP-hard problem can be called as an instance of beam search or dynamic programming (Tillmann and Ney 2003). Many such beam search algorithms mainly targeted optimizing memory efficiency (Furcy and Koenig 2005 and references therein) and not necessarily "navigation" adaptability in search space, which can be viewed as our main contribution.

There are many sampling algorithms like Gibbs sampler, Hot Coupling (Hamze and de Freitas 2005), Tree sampling, Swendsen–Wang sampling etc. But most of them assume restrictive conditional independence. Hamze et. al. proposed a very generic importance sampling method called Large Flip Importance Sampling (LFIS) to sample from the posterior (Hamze and de Freitas 2007). The main motivation for their approach comes from N-Fold Way (NFW, Bortz et al. 1975) and Tabu search (Fred 1989) where they use heuristics to improve the sampling of the exponential state space using memory and heuristics to design good moves in the state space. Since the moves are no-longer MCMC in the traditional sense they introduce importance weights to the distinct states visited by N copies of the sampler. Independently we had discovered a similar strategy using particle filters with static observations in Lu et al. (2009). In this paper we combined the strengths of both the approaches and presented an improved SMC that employs better navigational strategy to explore state space using permutations so as to compute MAP in an efficient way.

Suh et al. (2012) is the closest framework to our work in terms of formulating the quadratic optimization formulation and using SMC. Table 1 below compares and contrasts our work with Suh et al. (2012).

Relationship to Graph Matching Our objective function (Eq. 3) is syntactically similar to a Quadratic Assignment Problem (QAP) and *can* be used to solve graph matching problem by creating an association graph as presented in Suh et al. (2012). Hence we have only *one* graph and are seeking to find the MWS that satisfies constraints (see Eq. 3).

We observe that not all graph matching formulations in particular, the ones in Umeyama (1988) and in Almohamad and Duffuaa (1993); Zaslavskiy et al. (2009), can be used to find the MWS in a single graph. The key reason being that this class of graph matching algorithms minimize the following objective function which expects two well-defined graphs ($A_1, A_2 \in \mathbb{R}^{m\times m}$), with equal number of vertices.

$$\underset{P\in\mathbb{R}^{m\times m}}{\operatorname{argmin}} \; ||A_1 - P A_2 P^T||. \tag{32}$$

Although the assumption of having equal number of vertices can be relaxed by using dummy vertices, the search space has to be *square permutation matrices* which makes it a more restrictive problem than finding MWS. To emphasize how such differences matter, we would like to note that some algo-

rithms such as Quadratic Convex Relaxation (QCV) perform the search in the space of approximate permutation matrices and project the final result back to space of P.[1] Hence the space in which search is performed matters. Although the final resulting permutation matrix P^* *cannot* be used to obtain a weighted sub-graph in either A_1 or A_2, it can be used to obtain a weighted sub-graph in the "association" graph. We would also like to note that both our and (Suh et al. 2012) type formulations of association graph do not expect equal number of vertices either thus reducing one more choice of dealing with dummy vertices.

In general to match two graphs A_1 and A_2, they can be composed into the association graph $\mathscr{A}$, which corresponds to matrix A in Eq. (3). Examples of such composition are given in formulas (36), (37) and (41), (42) in the experimental subsection (Sect. 6.3). However, usually matrix $\mathscr{A}$ cannot be decomposed into two sub-matrices A_1 and A_2. For any problem with only one adjacency matrix ($\mathscr{A}$) which can not be naturally decomposed into two different graphs A_1, A_2, our SMCSP procedure can be a natural algorithm. Two real world examples that are naturally mapped to MWS but not to graph matching are presented below:

1. Dyer et al. (1985) introduce the problem of finding maximum weighted planar subgraph as an important real world problem of deciding which facilities should be located adjacently. They propose to solve the problem by finding a subgraph which maximizes the sum of edge weights thus maximizing the closeness ratings of the facilities. This is a real world example of the MWS which cannot be mapped to a graph matching problem because we do not have a *partitioning* of the nodes so as to perform any type of matching. Our algorithm can be applied to this method directly with appropriate adjustments to C1 and C2 which in turn only affect the proposal distribution in our SMCSP algorithm.
2. Another real world example is a Steiner tree problem where the goal is to finding the minimum edge weighted sub*tree*. Steiner tree problem models real world problems of VLSI design, wire length estimation and network routing (second paragraph of Sect. 1 of Chlebík and Chlebíková (2008)). Again in this case as well it is unclear a priori how to partition the nodes in the graphs so as to perform any sort of matching. Our SMCSP algorithm can be applied again to this problem by appropriately modifying C1 and C2 to ensure avoiding simple cycles in the resultant subgraph.

There are several other real world subgraph problems which are either equivalent to or very closely related to the MWS (sometimes called heaviest subgraphs in the literature (Hassin and Rubinstein 1994; Macambira 2002; Vassilevska et al. 2010; Álvarez-Miranda et al. 2013; Williams and Williams 2013; Rysz et al. 2013). In general as long as one can not partition the nodes (e.g., pixels and labels, boys and girls etc.), a notion of matching is not well defined but the notion of a subgraph is and hence MWS (or any other subgraph algorithm) can be adapted while matching algorithms cannot be. Analyzing the full behavior (either theoretical or empirical) of our algorithm in such problems however is outside the scope of our current paper.

We would like to note that graph matching is an overloaded term. While some instances of graph matching problem (specifically with quadratic terms) can be represented using a QAP formulation, in general the complexity of graph matching is not as simple to understand as the complexity of solving a QAP. For example the complexity of graph isomorphism is not fully characterized i.e. it is not known to be either in P or to be NP-complete where as the decision version of a QAP is NP-complete. The *sub*graph isomorphism on the other hand is known to be NP-complete.

Finally we would like to note that in the case of combinatorial NP-hard problems it is hard for one class of algorithms to be uniformly superior to all other approaches on different classes of problems or generic search problems. The term "graphs" can encapsulate many mathematical objects, for example something as fundamental as the real line, finite automata etc. and different problem domains benefit from different types of representations. Graphical view points of objects occurring in real world problems have huge benefits but are not universally effective for all problems. Similar is the situation with the algorithmic view points of graph matching and MWS. Each has its own benefits and limitations. However in the application setting such as ours, the SMC framework with permuted states, presented here yields excellent experimental performance in the jigsaw puzzle problem and is expected to perform well in other applications where we do not need two individually well-defined graphs but can model the application as MWS.

6 Experiments

6.1 Jigsaw Puzzle Details

In order to apply the proposed SMC Algorithm for Constrained MWS in Fig. 2 to solve the jigsaw puzzle problem, it remains to define the affinity matrix A.

In the case of the jigsaw puzzle problem, the set of vertices $V = P \times Q$ of graph G is composed of pairs $v_i = (p_i, q_i)$ representing (*puzzle piece index, board location*). Because we have the same number of board locations as the puzzle pieces, the number of graph nodes $m = n^2$, where $n = |P| =$

[1] Relaxations of P are not necessarily equivalent to relaxations of $\mathbf{x}$.

$|Q|$. The obtained MWS has n nodes, since (C2) represents here one-to-one constraints between two sets of n elements.

The observation z_i associated with a vertex v_i is simply the image depicted on the puzzle piece represented by this vertex, i.e., it is a digital image of size $K \times K$ represented by a $K \times K \times 3$ matrix of pixel color values. The set of observations is $Z = \{z_1, \ldots, z_m\}$.

Our goal now is to define the affinity matrix A of graph G representing the compatibility of the puzzle piece images of adjacent puzzle pieces. Formally, A is a $m \times m$ matrix, but actually, we only need to store a significantly smaller matrix of size $n \times n \times 4$ with the third dimension being an adjacency type, since two puzzle pieces can be adjacent in four different ways: left/right, right/left, top/bottom, and bottom/top, which we denote with LR, RL, TB, and BT. Thus, we abstract here from the actual location on the board of two puzzle pieces.

In order to be able to compare our experimental results to the results in Cho et al. (2010) we define A following the definitions in Cho et al. (2010). They first define an image dissimilarity measure D. Given two images z_j and z_i on puzzle pieces p_j and p_i, D measures their dissimilarity by summing the squared LAB color differences along their boundary, e.g., the left/right (LR) dissimilarity is defined as

$$D\Big(p_j, p_i, LR\Big) = \sum_{k=1}^{K} \sum_{c=1}^{3} \Big(z_j(k, u, c) - z_i(k, v, c)\Big)^2, \quad (33)$$

where u indexes the last column of z_j and v indexes the first column of z_i.

Finally, we are ready to define the affinity between two graph nodes $v_i = (p_i, q_i)$ and $v_j = (p_j, q_j)$. Let us first assume that squares q_i and q_j are LR adjacent. Then

$$A(i, j) = \exp\left(-\frac{D(p_j, p_i, LR)}{2\delta^2}\right), \quad (34)$$

where δ is adaptively set as the difference between the smallest and the second smallest D values between puzzle piece p_i and all other pieces in P, see Cho et al. (2010) for more details. The affinity for RL, TB, and BT adjacent squares is analogous.

If squares q_i and q_j are not adjacent, then we set $A(i, j) = 0$. Finally $A(i, i) = 0$ for all vertices $i \in V$, i.e., we do not use any whole puzzle image priors.

6.2 Image Jigsaw Puzzle Results

We compare the image jigsaw puzzle solutions obtained by the proposed algorithm to the solutions of the loopy belief propagation used in Cho et al. (2010) under identical settings. We used the software released by the authors of Cho et al. (2010) to obtain their results and also to compute the affinities defined in Sect. 6.1 used in our approach. The results are compared on the dataset provided in Cho et al. (2010), which we call MIT Dataset. It is composed of 20 images. We also compare the results to our previous approach published in the conference paper Yang et al. (2011).

The experimental results in Cho et al. (2010) are conducted in two different settings: with and without any prior on the target image layout. In Cho et al. (2008) the prior of the image layout is given by a low resolution version of the original image. Cho et al. (2010) utilizes a statistics of the possible image layout as prior. We focus on the results without any prior of the image layout. Consequently, we focus on a harder problem, since we only use the pairwise relations between the image patches, given by pair-wise compatibilities of located puzzle pieces as defined in Sect. 6.1.

We use three types of evaluation methods introduced in Cho et al. (2010). Each method focuses on different aspects of the quality of the obtained puzzle solutions. The most natural and strictest one is **Direct Comparison**. It simply computes the percentage of correctly placed puzzle pieces, i.e., for a puzzle with n pieces, Direct Comparison is the number of correct solution pairs divided by n. A less stricter measure is **Cluster Comparison**. It tolerates an assignment error as long as the puzzle piece is assigned to a location that belongs to a similar puzzle piece. The puzzle pieces are first clustered into groups of similar pieces. Moreover, due to lack of prior knowledge of target image, the reconstructed image may be shifted compared to the ground truth image. Therefore, a third measure called **Neighbor Comparison** is used to evaluate the label consistency of adjacent puzzle pieces independent of their grid location, e.g., the location of two adjacent puzzle pieces is considered correct if two puzzle pieces are left-right neighbors in the ground truth image and they are also left-right neighbors in the inferred image. Neighbor Comparison is the fraction of correct adjacent puzzle pieces. This measurement does not penalize the accuracy as long as the adjacent patches in original image are adjacent in the reconstructed image.

The results on the MIT Dataset are shown in Table 2. The proposed algorithm significantly outperforms the loopy belief propagation in all three performance measures. Moreover, the reconstruction accuracy (according to the most natural measure, Direct Comparison) of the original images by our algorithm is improved by more than four times. As compared to our previous approach in Yang et al. (2011), the proposed algorithm increased the Direct Comparison score by over 25 %.

Effect of initialization The three methods are initialized with one anchor patch, i.e., with one correct (puzzle piece, grid location) pair. We always assign a correct image patch to the puzzle piece at the top left corner of the image. We selected the top left corner, since this is usually one of the less informative puzzle pieces. In this experiment we divide each test

Table 2 Comparison of the best scores on MIT Dataset with 108 puzzle pieces, i.e., the image on each puzzle piece is of size 56 × 56 pixels

	Cho et al. (2010)	Yang et al. (2011)	Our algorithm	Random *location* initialization	Dropping C1
Direct Comparison	0.2366	0.6921	**0.9523**	0.7929	0.0110
Cluster Comparison	0.4657	0.7810	**0.9889**	0.8689	0.2531
Neighbor Comparison	0.6628	0.8620	**0.9443**	0.8856	0.6777

The last two columns show the performance of our algorithm under two different settings of initialization
We can observe that C1 plays a very useful role although the actual location (top-left, bottom-right etc.) of the initialization does not affect the performance much

Fig. 5 *First row* the original images. *Second row* our solutions. *Third row* the jigsaw puzzle solutions of Cho et al. (2010)

image into 108 square patches resulting in $n = 108$ puzzle pieces. Following the experimental setting in Cho et al. (2010), each experiment was repeated three times and best performance is reported for each of the three methods. In all our experiments, γ is set to maximal affinity times the number of patches, i.e., $\gamma = n \cdot \max_{i,j} A(i, j)$.

In principle, the initialization constraint (C1) is not needed for our SMCSP framework, but it provides a very useful initialization step in practice. The navigation through search space itself does not heavily rely on C1 but rather relies on C2. The lack of initialization affects the performance of the algorithm. We randomly matched different puzzle pieces and locations and repeated the experiment 10 different times. The results (last two columns of Table 2) as expected are not as good as when using C1 but comparable to Cho et al. (2010) under cluster and neighbor comparisons. However the *location* (e.g. top-left, right, center etc.) of the initial ground-truth does not affect the performance which is still orders of magnitude better than that of Cho et al. (2010). Hence one can change the location of the initial puzzle as long as the initial puzzle is in the correct location (i.e. C1 is satisfied).

In order to demonstrate that the considered image jigsaw puzzle problem is also very challenging to humans, we show some example results in Fig. 5. There, we show the original images, but we would like to emphasize that the original images are not used during the inference. Figure 5 also demonstrates that the reconstructed images obtained by the proposed algorithm compare very favorably to the results of Cho et al. (2010). We observe many repeated patches in the results of Cho et al. (2010). Often the same patch is assigned to many locations, and some patches are not assigned at all. This fact demonstrates that the loopy belief propagation inference cannot enforce its solutions to satisfy global mutex constraints.

Table 3 Results (direct comparison / cluster comparison/ neighbor comparison) of our algorithm for different numbers of particles

No. Particles	Max score	Mean score
200	0.9088 / 0.9528 / 0.9250	0.8392 / 0.8980 / 0.8967
400	0.9194 / 0.9657 / 0.9344	0.8495 / 0.9137 / 0.9098
600	0.9463 / 0.9843 / 0.9405	0.8926 / 0.9461 / 0.9212
800	0.9500 / 0.9931 / 0.9482	0.8856 / 0.9426 / 0.9236
1000	0.9523 / 0.9889 / 0.9443	0.9218 / 0.9682 / 0.9330

The only case when the proposed approach does not yield good solutions is when repeated patches are present, i.e., patches that are nearly perfectly identical. This is the case for the image in the last column in Fig. 5. Although the image in the first column seems to contain repeated patterns, they are not identical, so that our approach has no problems with this image. Of course, this limitation is due to the local nature of the puzzle patch affinities, and could be addressed by considering more global relation among the patches as is done in Pomeranz et al. (2011). However, we want to stress that our main focus is on evaluating the proposed SMC inference, and we view the challenging problem of the image jigsaw puzzle as formulated in Cho et al. (2010) as an excellent testbed for evaluating random fields inference methods.

Our best result reported in Table 2 is obtained for N =1,000 particles. As illustrated in Table 3 this seems to be a sufficient number of particles for this experiment. The performance increase from N = 800 to N =1,000 particles is minor, and the difference between best score and the average scores for N =1,000 particles is small. Our average computing time for one image with 1,000 particles is 90 s in a mixed Matlab/C++ implementation on a Windows PC Core i7 with 3.40 GHz.

The largest puzzle considered in Cho et al. (2010) contains 432 puzzle pieces. Since the size of whole images did not change, the image patches on the puzzle pieces are of size 28×28 pixels, which significantly reduces the discriminative power of the patch affinities. They are only based on color differences of 28 pixel pairs along one common edge. Therefore, the performance of Cho et al. (2010) drops significantly. It is about 0.10 / 0.30 / 0.55 (Direct Comparison / Cluster Comparison / Neighbor Comparison). We estimated it from the graph in Fig. 8 in Cho et al. (2010). Our performance also dropped to 0.50 / 0.65 / 0.69 with 800 particles. However, we observe that our result for direct comparison is still five times better than the result of loopy belief propagation in Cho et al. (2010).

In order to demonstrate the dynamic of the proposed PF inference, we show reconstructed images of the best particle at different iterations in Fig. 6. As stated above it is also possible to apply the standard PF algorithm, in which all particles follow the same order, to the jigsaw puzzle problem. Figure 7 illustrates the results when all particle follow the TV scan order. As in Fig. 6 we show the best particle at selected iterations. Fig. 7 clearly demonstrates that the standard PF algorithm is unable to provide a solution to the challenging jigsaw puzzle problem when executed with the same number of particles as the proposed algorithm.

Time Complexity For a given image jigsaw puzzle with n pieces, the time complexity of the proposed algorithm in Fig. 2 is $O(n^2N)$, where N is the number of particles, as we now show.

The time complexity of a single iteration t for one particle is bounded from above by the size of the neighborhood of the particle. We first observe that each particle at iteration t has exactly t vertices. Since each vertex is a pair (puzzle piece index, board location), and each board location can have at most 4 adjacent board locations, each node can have at most $4(n-t)$ neighbors, where $n-t$ bounds the number of puzzle pieces. Hence the neighborhood size is bounded by $t \cdot 4(n-t)$. Since the maximum of $t \cdot 4(n-t)$ over $t = 1, \ldots, n$ is of order n, the neighborhood size of each particle is $O(n)$. Since the number of iterations is n, the time complexity for one particle is $O(n^2)$. We obtain that the time complexity of the proposed algorithm in Fig. 2 is $O(n^2N)$.

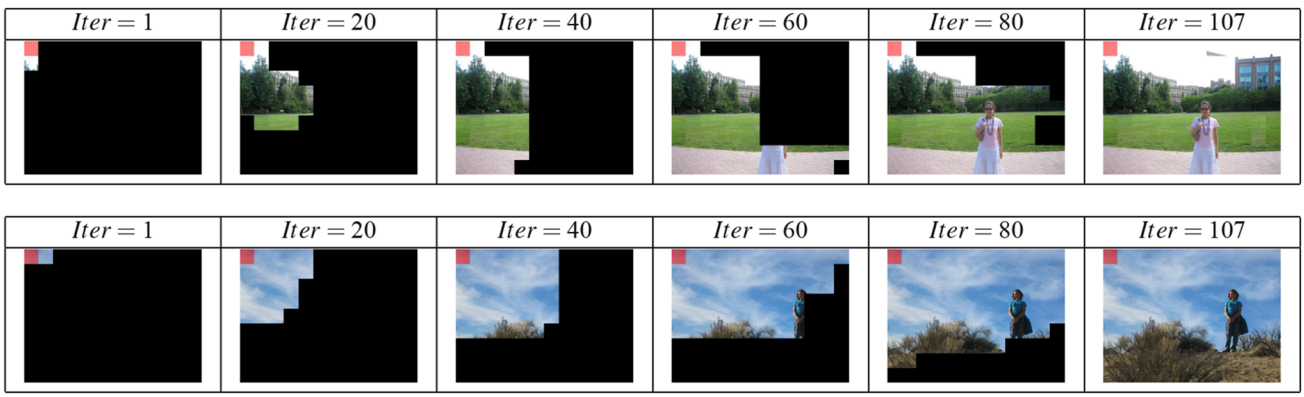

Fig. 6 The reconstructed images of the best particles of our algorithm at different iterations

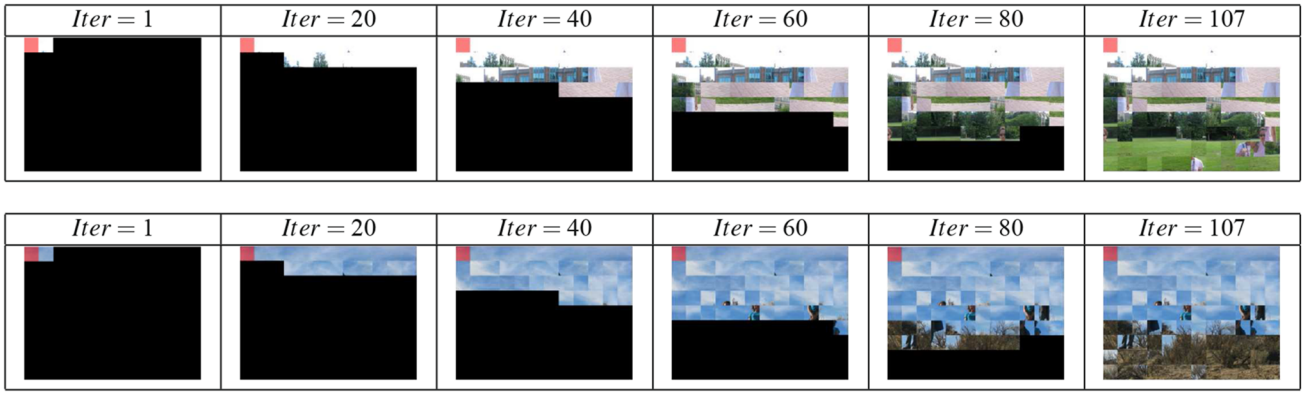

Fig. 7 The reconstructed images of the best particles at different iterations with the standard PF algorithm in which all particle follow the TV scan order

6.3 Graph Matching Experiments Using Synthetic Graphs

Although as discussed Sect. 5, the two formulations (graph matching and MWS) have some important differences (search space, utilizing the optimal solution), since our SMCSP framework *can* be used for graph matching, we compare its performance to that of two different graph matching packages using synthetic graphs. We compare to a total of thirteen different graph matching algorithms.

We compare the performance of our method to the factorized graph matching (FGM) package[2] in which nine different algorithms (including the most recent Zhou and De la Torre 2013) listed below are available.

1. Graduated Assignment (GA) Gold and Rangarajan (1996)
2. Probabilistic Graph Matching (PM) Zass and Shashua (2008)
3. Spectral Matching (SM) Leordeanu and Hebert (2005)
4. Spectral Matching with Affine Constraints (SMAC) Cour et al. (2007)
5. Integer Projected Fixed Point method initialized with solution used for FGM-U (item 8 below) (IPFP-U) Leordeanu et al. (2009)
6. Integer Projected Fixed Point method initialized with SM (IPFP-S) Leordeanu et al. (2009)
7. Re-weighted Random Walk Matching (RRWM) Cho et al. (2010)
8. Factorized Graph Matching for Undirected graphs (FGM-U) Zhou and De la Torre (2012)
9. Factorized Graph Matching for Undirected graphs (FGM-D) Zhou and De la Torre (2013)

The package allows to generate a random synthetic graph with a pre-defined edge density and number of nodes. We use their default settings (number of nodes=10, edge density=0.5) and generate 20 different random graphs and perform the graph matching. Their formulation is similar to that of Suh et al. (2012) and ours. So we can directly apply our algorithm to their association graphs. The Fig. 8 shows the results of the average objective value (the higher the better) and the error bars. We can observe that SMCSP outperforms many of the classical graph matching algorithms and is only outperformed by the latest work Zhou and De la Torre (2013), even when we reduce the number of particles to 200.

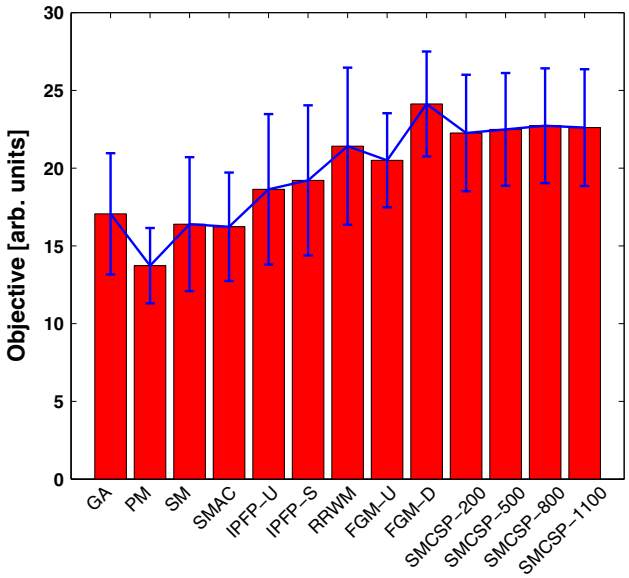

Fig. 8 The average performance (with error bars) as measured by the objective value achieved by the nine different algorithms available in the FGM package and our SMCSP using 200, 500, 800 and 1,100 particles. The averages and error bars are estimated using 20 random synthetic graphs with fixed number of nodes (=10) and edge density (=0.5)

The other package we compare to is the `GraphM`[3] package which implements four different algorithms viz. Umeyama

[2] http://www.f-zhou.com/gm.html.

[3] http://cbio.ensmp.fr/graphm/.

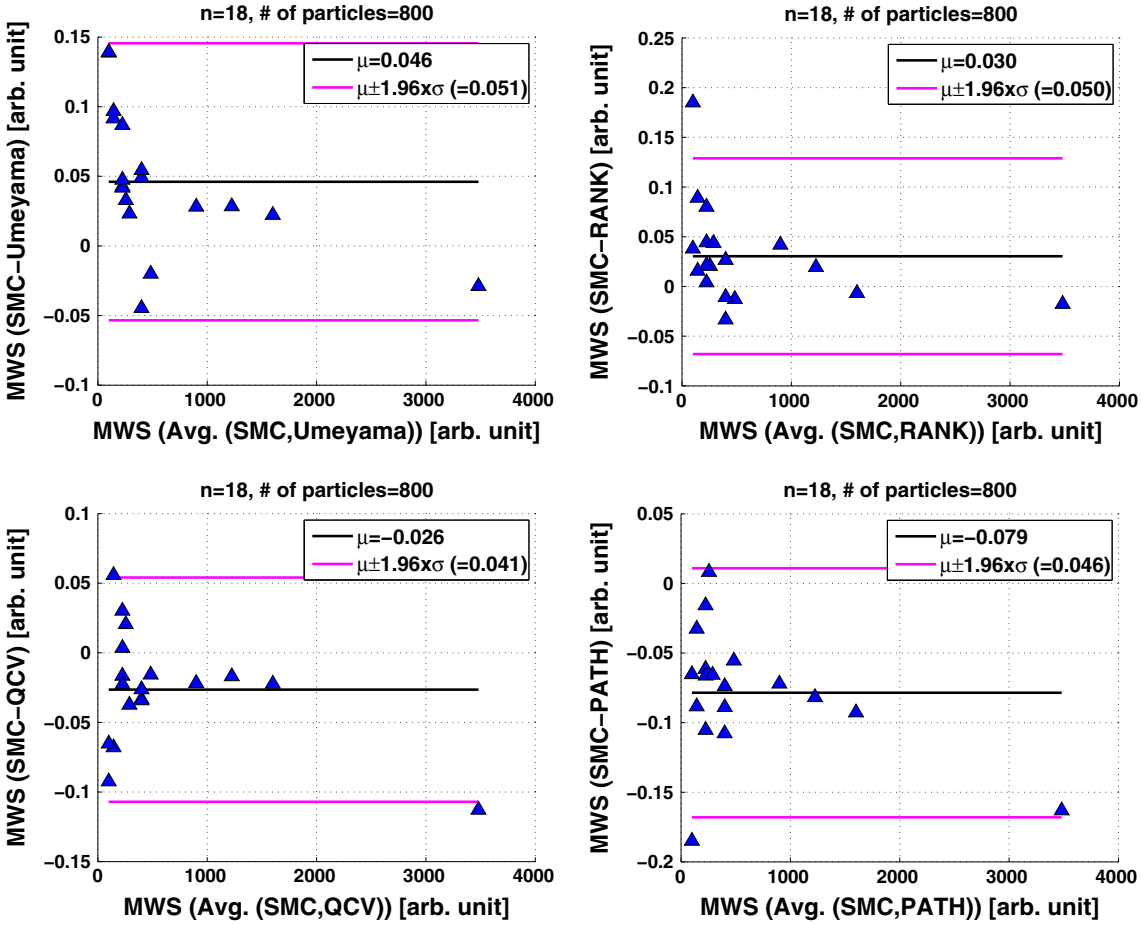

Fig. 9 Bland–Altman plots comparing the performance of our SMC method to Umeyama, RANK, QCV and PATH algorithms in computing the MWS. On average we can observe that the SMCSP algorithm performs better than Umeyama and RANK and slightly worse than QCV and PATH. As in the case of GDist we can observe that the performance is made worse by extremal graphs

(1988), PATH, QCV Zaslavskiy et al. (2009) and RANK Singh et al. (2007). We report the data generated from the following two specific experiments.

1. Utilize SMCSP-MWS to perform graph matching by using an association graph.
2. Utilize the resulting optimal permutation (P^*) from the four other graph matching algorithms and compute the weight of the subgraph in the association graph.

Since the formulation of graph matching in this setting is not the same as ours, the key implementation detail in utilizing SMCSP-MWS algorithm for matching two graphs defined using adjacency matrices A_1 and A_2 is in constructing an association graph ($\mathscr{A}$) such that finding an MWS in $\mathscr{A}$ would result in P^*. Equation (35) presents the objective function which is a generalized and normalized form of Eq. (32) that is optimized in the `GraphM` package.

$$\underset{P\in\mathbb{R}^{m\times m}}{\operatorname{argmin}}\ (1-\alpha)\frac{||A_1 - PA_2P^T||^2}{||A_1||^2+||A_2||^2} - \alpha\frac{\mathrm{tr}(C^TP)}{||C||}. \tag{35}$$

$C(i,j)$ denotes the score of matching vertex $i \in A_1$ to vertex $j \in A_2$ and α is used to weigh the node-matching and edge-matching. We now describe the construction of $\mathscr{A}$. Since our framework does not require the graphs to have the same number of vertices in general, if A_1 and A_2 have m_1 and m_2 number of vertices respectively, $\mathscr{A} \in \mathbb{R}^{m_1m_2\times m_1m_2}$. Since $\mathscr{A}$ requires non negative entries, we set

$$\mathscr{A}(ij,ij) = \alpha\exp\left(\frac{C(i,j)}{||C||}\right),\ \text{for diagonal elements} \tag{36}$$

$$\mathscr{A}(ij,ab) = (1-\alpha)\exp\left(-\frac{(A_1(i,a)-A_2(j,b))^2}{||A_1||^2+||A_2||^2}\right),\ \text{for off}-\text{diagonal elements.} \tag{37}$$

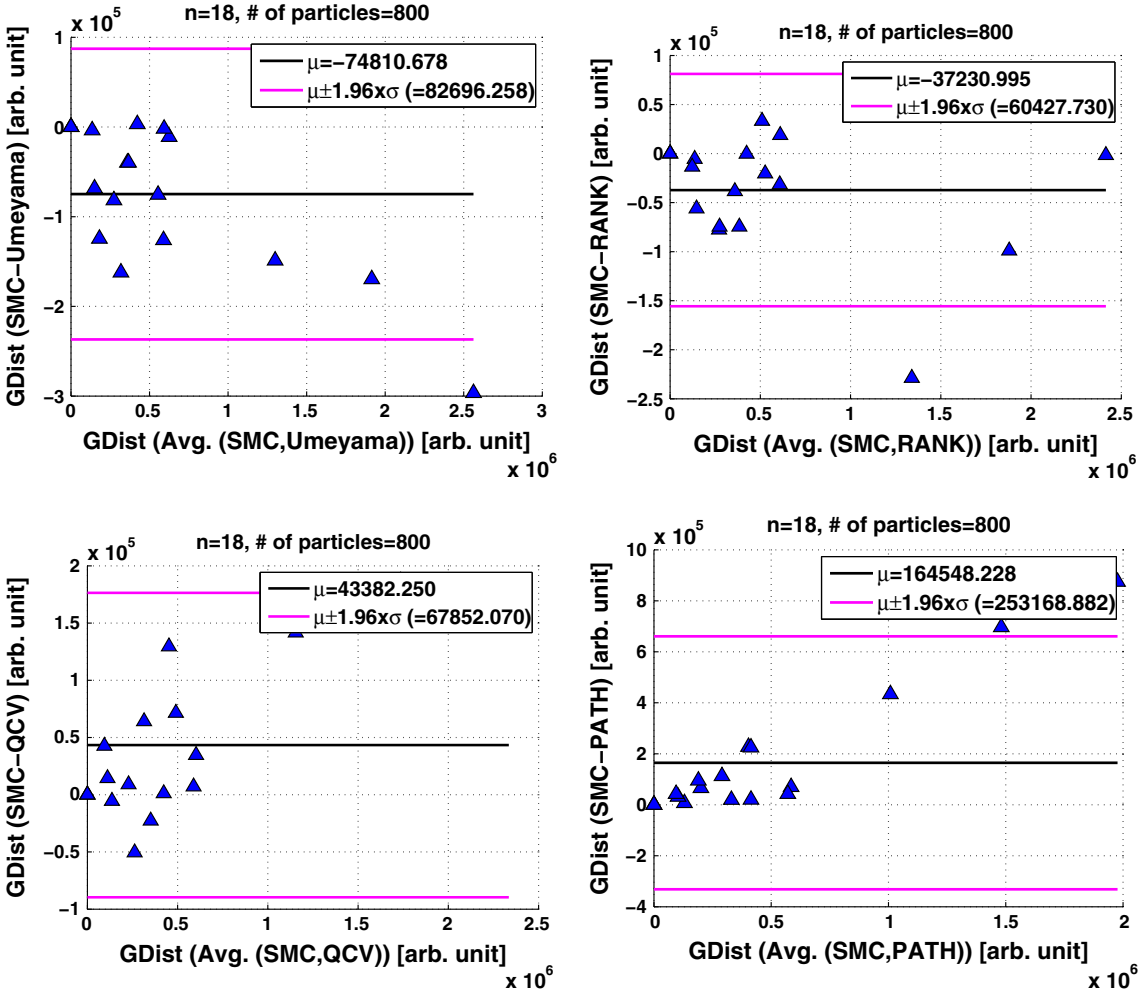

Fig. 10 Bland-Altman plots comparing the performance of the SMCSP method to Umeyama, RANK, QCV and PATH algorithms in computing the GDist with out using any dummy nodes. We can notice the improvements in the performance of the algorithm when compared to the μs (represented by *dark lines*) obtained using the dummy nodes in Fig. 11

The following objective is then optimized using our SMCSP framework.

$$\underset{\mathbf{x}\in\{0,1\}^{m_1 m_2}}{\text{argmax}} \quad \mathbf{x}^T \mathscr{A} \mathbf{x}. \tag{38}$$

After the optimization, $P(i, j)$ is set to 1 if node ij of $\mathscr{A}$ is selected in the MWS solution i.e. if $\mathbf{x}^*(ij) = 1$. Similarly $\mathbf{x}(ij)$ is set to 1 if $P^*(i, j)$ obtained from the other algorithms is 1. For the other algorithms, based on the `GraphM` package, we use the settings in which dummy nodes play a minimal role i.e. just add $|N-M|$ isolated dummy nodes to the smaller graph with zero values in the corresponding entries of C.

The constraint (C1) is empty and (C2) is designed to enforce one-one correspondence. Hence our algorithm picks $n = \min(m_1, m_2)$ nodes as part of the MWS, i.e $\mathbf{x}^*$ has exactly n ones. For simplicity of comparison with other methods we include the dummy nodes in our experiments for the most part but also report some results without using dummy nodes.

We performed 18 synthetic graph matching experiments available in the `GraphM` package which provides the values for A_1s, A_2s and Cs. Since for the majority of the graphs C is not provided we set $\alpha = 0$ (as such the diagonal elements of $\mathscr{A}$ are also set to 0) for our experiments. Since (a) both Eq. (35) and Eq. (43) are NP-hard, that is not only are they computationally hard but the solutions can not even be verified efficiently in polynomial time (unless P = NP), (b) the synthetic graphs are fixed and not randomly generated (as those in FGM package), we resort to comparing the performance between the different algorithms using Bland-Altman (BA) plots Altman and Bland (1983). BA plots are used to capture the relationship between the magnitude of a measure and difference using two methods of obtaining the measure. The X-axis shows the average of the measure using the two methods and the Y-axis plots the difference between the measure.

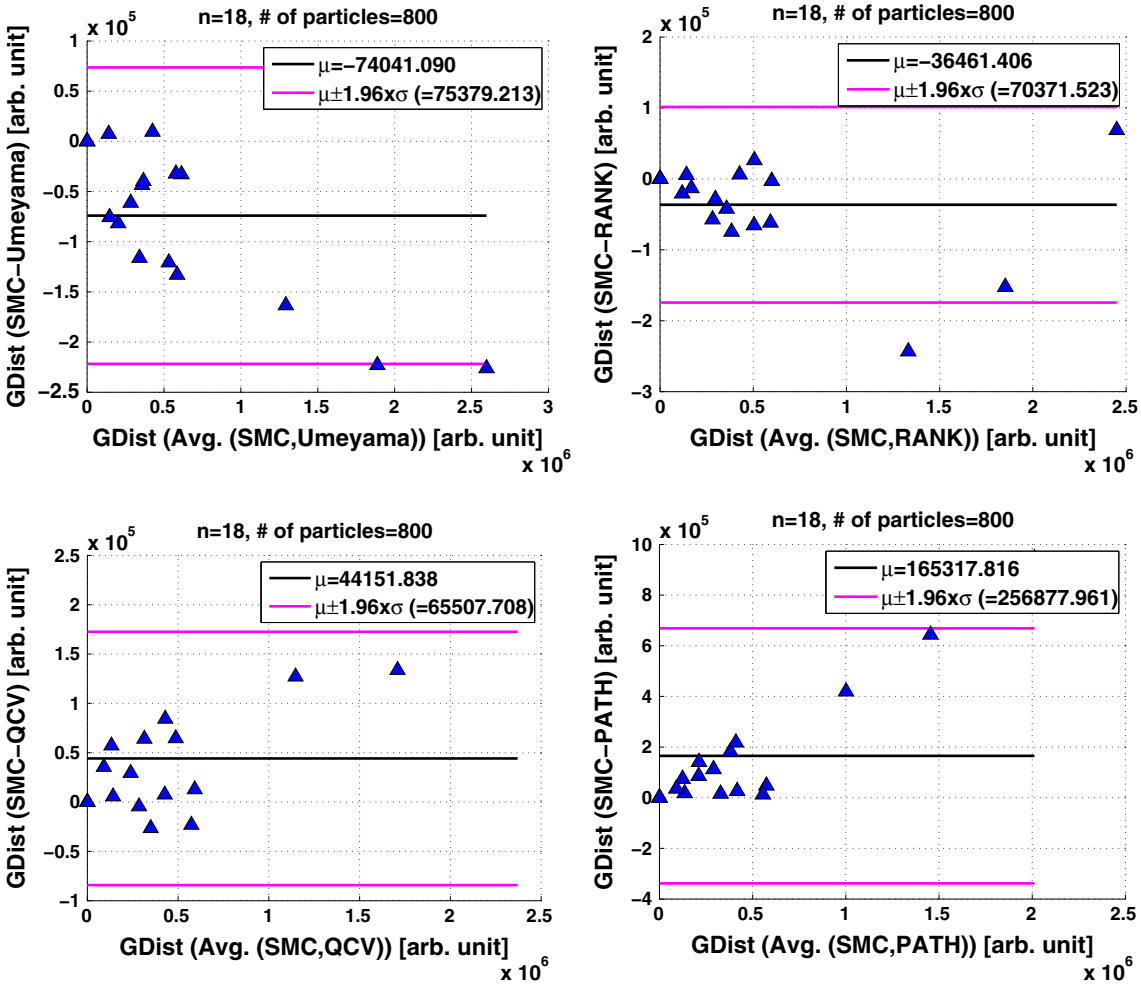

Fig. 11 Bland–Altman plots comparing the performance of our SMC method to Umeyama, RANK, QCV and PATH algorithms in computing the GDist. We can observe that on average (represented by the *dark line*) our SMC method performs better compared to Umeyama and RANK but slightly worse than QCV and PATH. But if we notice carefully in the case of QCV and PATH the average performance is made worse by a couple of extremal graphs that have large GDist

The mean bias or difference and the 95 % deviations of the difference (magenta lines) computed using the entire sample are also plotted. If there is no systematic relationship between the magnitude and difference the scatter looks uniform and random within the standard deviation lines. The bias of a method can be inferred based on mean bias (dark line).

Figure 11 shows the BA plots for GDist ($||A_1 - P^* A_2 P^{*T}||_F^2$), which does not have the second term involving C since $\alpha = 0$. Figure 9 shows the same for MWS ($\mathbf{x}^{*T} \mathscr{A} \mathbf{x}^*$). We can observe that (on average) our SMC method performs better than Umeyama and RANK but *slightly* worse than QCV and PATH. But if we notice carefully in the case of QCV and PATH the average performance is made worse by a couple of extremal graphs that have large GDist. As in the case of GDist we can observe that the performance is made worse by extremal graphs even for MWS. The slight performance difference is only in an average sense and not in a uniform sense as is to be expected with such problems. Figure 10 shows the BA plots for GDist when not using dummy nodes in the SMCSP. We can notice the improvements in the performance of the algorithm when compared to the μs obtained using the dummy nodes in Fig. 11. We do not show BA plots for MWS without dummy nodes because MWS is computed on $\mathscr{A}$ and the other algorithms require dummy nodes according to their current implementations in the `GraphM` package.

In addition to the above graph matching experiments we also tested the performance of the SMCSP algorithm on the QAPLib instances[4]. The QAP objective is

[4] http://www.seas.upenn.edu/qaplib/inst.html.

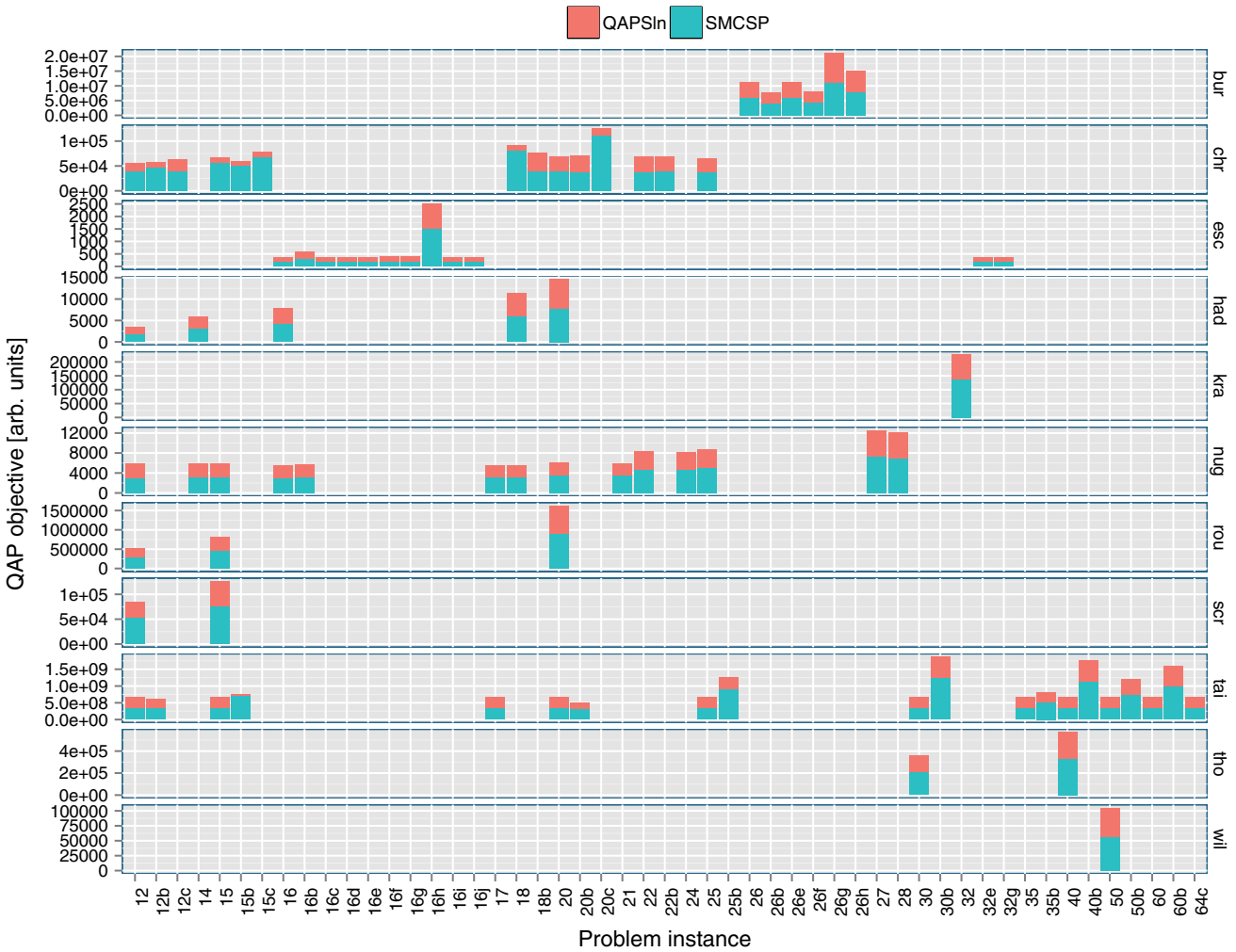

Fig. 12 The performance of SMCSP on QAPLib instances. The Y-axis shows the QAP objective values (Eq. (40)) obtained by our algorithm (SMCSP) and the reported values (QAPSln) on the QAPLib website. The values are in arbitrary units. The closer the sizes of the *red* and *green bars*, the better is the performance of our algorithm. The facets are used to show the clusters of problem instances given by different groups of authors. The *right* side of the facet shows the group name which is the first three characters of the name of the first author of the group. The numerical values of the X-axis ticks show the size of the QAP problem instance i.e. the value of m

$$\min_P \mathtt{tr}\left(FPDP^T\right) \tag{39}$$

$$= \min_P \sum_{i,j} F(i,j)D(P(i),P(j)), \tag{40}$$

where F and D are called the "flow" and "distance" matrices and P is a permutation matrix. The key implementation detail is exactly as before when we ran our algorithm on the instances from `GraphM` package except now we have the F and D square matrices instead of A_1, A_2 and C. If F and D are in $\mathbb{R}^{m\times m}$, then the entries in $\mathscr{A} \in \mathbb{R}^{m^2}$ are computed as,

$$\mathscr{A}(ij,ij) = \exp\left(-F(i,j)\right), \text{ for diagonal elements} \tag{41}$$

$$\mathscr{A}(ia,jb) = \exp\left(-(D(i,a)+D(j,b))\right), \quad \text{for off} - \text{diagnoal elements.} \tag{42}$$

Then we solve the MWS objective (reproduced below for convenience),

$$\underset{\mathbf{x}\in\{0,1\}^{m_1 m_2}}{\text{argmax}} \ \mathbf{x}^T\mathscr{A}\mathbf{x}. \tag{43}$$

As before, after the SMCSP optimization, $P(i,j)$ is set to 1 if node ij of $\mathscr{A}$ is selected in the MWS solution i.e. if $\mathbf{x}^*(ij) = 1$. C1$=\phi$ and C2 is designed to enforce one-one correspondence so that $\mathbf{x}^*$ has exactly m ones.

The QAPLib is a collection of heterogeneous set of problems collected from different groups of authors. In addition to providing the F and D matrices, the collection presents either optimal or heuristic solutions reported by the groups. We present the performance of our algorithm on 80 problem instances from the library. The SMCSP based objective values (bluish green bars) are compared against the reported

objective values (red bars) as shown in Fig. 12. Except for instances from the group 'tai' whose size ≥ 30, all the solutions are reported to be optimal so the best we can hope for SMCSP is to match the size of the red bars. For those specific large instances of 'tai' the reported solutions are from special implementations of tabu search (Taillard 1991, 1995; Misevicius 2005, 2012).

We can observe that in almost all the 80 instances our SMCSP algorithm matches the reported objective values quite closely. It performs somewhat poorly on some problem instances from the group 'chr'. After further investigation into that class of instances we noticed that they are special set of QAP instances where one matrix is the adjacency matrix of a weighted tree and the other of a complete graph. Exploiting that special structure Christofides and Benavent (1989) devised exact QAP algorithm. Our SMCSP could in principle take full advantage of any such structure but we just ran our algorithm 'as is' using only 800 particles consistently for all the problem instances. Even then the results of our algorithm are comparable to best possible results, which is remarkable considering the variety of all the instances and their sizes. The Matlab scripts used in our project are made publicly available[5].

7 Conclusions

We introduce a novel inference framework for solving maximum weight subgraph problems with hard constraints. Our key contribution is an extension of the SMC framework to work with unordered observations. Weighted particles explore the state space along different dimensions in different orders, and resampling allows the particles to focus on most promising search regions. We prove that the obtained importance samples represent samples from the original target distribution. We evaluate the performance of the proposed algorithm on a problem of image jigsaw puzzles. As the experimental results demonstrate, it significantly outperforms the loopy belief propagation.

Image jigsaw puzzle problem is an instance of a constrained correspondence problem, which can also be viewed as a search problem on the correspondence graph. In this context, the proposed algorithm can be viewed as a randomized graph search algorithm, which keeps time complexity low by effectively exploiting the sparsity of the adjacency matrix (induced by the local nature of the neighborhood relation) on instances of NP-hard problems.

Acknowledgments The authors would like to thank Taeg Sang Cho for providing the code for the method in Cho et al. (2010). We would like to acknowledge the Waisman Core Grant P30 HD003352-45, the NSF under Grants IIS-1302164 and OIA-1027897.

[5] http://brainimaging.waisman.wisc.edu/~adluru/SMCSP.

References

Almohamad, H. A., & Duffuaa, S. O. (1993). A linear programming approach for the weighted graph matching problem. *IEEE TPAMI*, *15*(5), 522–525.

Altman, D., & Bland, J. (1983). Measurement in medicine: the analysis of method comparison studies. *Statistician*, *32*, 307–317.

Álvarez-Miranda, E., Ljubić, I., & Mutzel, P. (2013). The maximum weight connected subgraph problem. In: *Facets of Combinatorial Optimization*, pp. 245–270. Berlin: Springer.

Arora, S., Frieze, A., & Kaplan, H. (1996). A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In: *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 21–30.

Asahiro, Y., Hassin, R., & Iwama, K. (2002). Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, *121*, 15–26.

Bisiani, R. (1987). Beam search. In S. C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence* (pp. 56–58). New York: Wiley.

Bortz, A. B., Kalos, M. H., & Lebowitz, J. L. (1975). A new algorithm for monte carlo simulation of ising spin systems. *Journal of Computational Physics*, *17*, 10–18.

Burkard, R. E., Pardalos P. M., Cela, E., & Pitsoulis, L. (1998). The quadratic assignment problem. In: P. Pardalos, D.Z. Du (eds.) Handbook of combinatorial optimization, pp. 241–338. Philip Drive Norwell, MA: Kluwer Academic.

Caetano, T., Caelli, T., Schuurmans, D., & Barone, D. (2006). Graphical models and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1646–1663.

Carpenter, J., Clifford, P., & Fearnhead, P. (1999). *Building robust simulation-based filters for evolving data sets* (pp. 1–27). Department of Statistics, University of Oxford.

Chen, Z. (2003). *Bayesian filtering: From Kalman filters to particle filters, and beyond*. Technical report, McMaster University.

Chlebík, M., & Chlebíková, J. (2008). The steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, *406*(3), 207–214.

Cho, T. S., Avidan, S., & Freeman, W. T. (2010). A probabilistic image jigsaw puzzle solver. In: *CVPR*.

Cho, T. S., Butman, M., Avidan, S., & Freeman, W. T. (2008). The patch transform and its applications to image editing. In: *CVPR*.

Cho, M., Lee, J., & Lee, K. M. (2010). Reweighted random walks for graph matching. In: *Proceedings of the 11th European conference on Computer vision: Part V*, pp. 492–505.

Christofides, N., & Benavent, E. (1989). An exact algorithm for the quadratic assignment problem on a tree. *Operations Research*, *37*(5), 760–768.

Cour, T., Srinivasan, P., & Shi, J. (2007). Balanced graph matching. *Advances in Neural Information Processing Systems*, 19, 313–320.

Crisan, D., & Doucet, A. (2002). A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, *50*(3), 736–746.

Cross, A., & Hancock, E. (1998). Graph matching with a dual-step EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(11), 1236–1253.

Demaine, E. D., & Demaine, M. L. (2007). Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, *23*, 195–208.

Doucet, A., Freitas, N. D., & Gordon, N. (2001). *Sequential Monte Carlo methods in practice*. Berlin: Springer.

Dyer, M., Foulds, L., & Frieze, A. (1985). Analysis of heuristics for finding a maximum weight planar subgraph. *European Journal of Operational Research*, *20*(1), 102–114.

Fox, D., Thrun, S., Dellaert, F., & Burgard, W. (2000). Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, & N. Gordon (Eds.), *Sequential Monte Carlo Methods in practice*. New York: Springer.

Fred, G. (1989). Tabu search—part i. *ORSA Journal on Computing*, *1*(3), 190–206.

Freeman, H., & Garder, L. (1964). Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE TEC*, *13*, 118–127.

Furcy, D., & Koenig, S. (2005). Limited discrepancy beam search. In: *Proceedings of the 19th international joint conference on Artificial intelligence*, pp. 125–131.

Georgescu, B., & Meer, P. (2004). Point matching under large image deformations and illumination changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 674–688.

Gold, S., & Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *IEEE Transactions Pattern Analysis and Machine Intelligence*, *18*(4), 377–388.

Goldberg, D., Malon, C., & Bern, M. (2002). A global approach to automatic solution of jigsaw puzzles. In: *Symposium on Computational Geometry*.

Gordon, N., Salmond, D., & Smith, A. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings of Radar and Signal Processing*, *140*, 107–113.

Hamze, F., & de Freitas, N. (2005). Hot coupling: A particle approach to inference and normalization on pairwise undirected graphs of arbitrary topology. In *Advances in Neural Information Processing Systems*, Vol. 18, pp. 1–8.

Hamze, F., & de Freitas, N. (2007). Large-flip importance sampling. In *Uncertainty in Artificial Intelligence*, Vol. 1, pp. 167–174.

Hassin, R., & Rubinstein, S. (1994). Approximations for the maximum acyclic subgraph problem. *Information Processing Letters*, *51*(3), 133–140.

Horaud, R., & Skordas, T. (1989). Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*(11), 1168–1180.

Ioffe, S., & Forsyth, D. (2001). Probabilistic methods for finding people. *International Journal of Computer Vision*, *43*, 45–68.

Isard, M., & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In: *Proceedings of the 4th European Conference on Computer Vision, ECCV*, pp. 343–356.

Isard, M., & Blake, A. (1998). Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision*, *29*(1), 5–28.

Jiang, H., Drew, M., & Li, Z. (2007). Matching by linear programming and successive convexification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 959–975.

Khan, Z., Balch, T., & Dellaert, F. (2004). An mcmc-based particle filter for tracking multiple interacting targets. In *European Conference on Computer Vision*, Vol. 3024, pp. 279–290.

Kong, W., & Kimia, B. B. (2001). On solving 2d and 3d puzzles using curve matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 583–590.

Leordeanu, M., Hebert, M., & Sukthankar, R. (2009). An integer projected fixed point method for graph matching and map inference. In *Advances in Neural Information Processing Systems*, Vol. 1, pp. 1114–1122.

Leordeanu, M., & Hebert, M. (2005). A spectral technique for correspondence problems using pairwise constraints. *Proceedings of the Tenth IEEE International Conference on Computer Vision*, *2*, 1482–1489.

Liu, H., Latecki, L. J., & Yan, S. (2010). Robust clustering as ensembles of affinity relations. In *Advances in Neural Information Processing Systems*, Vol. 1, pp. 1414–1422.

Liu, J. S., Chen, R., & Logvinenko, T. (2001). A theoretical framework for sequential importance sampling with resampling. In A. Doucet, N. de Freitas, & J. Gordon (Eds.), *Sequential Monte Carlo methods in practice* (pp. 223–233). Berlin: Springer.

Liue, J. (2001). *Monte Carlo strategies in scientific computing*. Berlin: Springer.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Lu, C., Latecki, L. J., Adluru, N., Yang, X., & Ling, H. (2009). Shape guided contour grouping with particle filters. In *IEEE International Conference on Computer Vision*, Vol. 1, pp. 2288–2295.

Macambira, E. M. (2002). An application of tabu search heuristic for the maximum edge-weighted subgraph problem. *Annals of Operations Research*, *117*(1–4), 175–190.

Maciel, J., & Costeira, J. (2003). A global solution to sparse correspondence problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(2), 187–199.

Makridis, M., & Papamarkos, N. (2006). A new technique for solving a jigsaw puzzle. In *IEEE International Conference on Image Processing*, Vol. 1, pp. 2001–2004.

Misevicius, A. (2005). A tabu search algorithm for the quadratic assignment problem. *Computational Optimization and Applications*, *30*(1), 95–111.

Misevicius, A. (2012). An implementation of the iterated tabu search algorithm for the quadratic assignment problem. *OR Spectrum*, *34*(3), 665–690.

Montenegro, R., & Tetali, P. (2006). Mathematical aspects of mixing times in markov chains. *Foundations and Trends in Theoretical Computer Science*, *1*(3), 237–354.

Nielsen, T. R., Drewsen, P., & Hansen, K. (2008). Solving jigsaw puzzles using image features. *PRL*, *29*, 1924–1933.

Pavan, M., & Pelillo, M. (2007). Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 167–172.

Pomeranz, D., Shemesh, M., & Ben-Shahar, O. (2011). A fully automated greedy square jigsaw puzzle solver. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 9–16.

Radack, G. M., & Badler, N. I. (1982). Jigsaw puzzle matching using a boundary-centered polar encoding. *Computer Graphics and Image Processing*, 19(1), 1–17.

Rysz, M., Mirghorbani, M., Krokhmal, P., & Pasiliao, E. L. (2013). On risk-averse maximum weighted subgraph problems. *Journal of Combinatorial Optimization*, *43*, 1–19.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the Association of Computing Machinery*, *23*, 555–565.

Singh, R., Xu, J., & Berger, B. (2007). Research in computational biology. Pairwise global alignment of protein interaction networks by matching neighborhood topology, vol. 4453. Berlin: Springer.

Smith, K., Gatica-Perez, D., & Odobez, J. M. (2005). Using particles to track varying numbers of interacting people. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 962–969.

Sontag, D., Globerson, A., & Jaakkola, T. (2010). Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, & S. Wright (Eds.), *Optimization for Machine Learning*. Cambridge, MA: MIT Press.

Suh, Y., Cho, M., & Lee, K. M. (2012). Graph matching via sequential monte carlo. *ECCV*, *7574*, 624–637.

Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing*, *17*(4), 443–455.

Taillard, E. D. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science*, *3*(2), 87–105.

Thrun, S. (2002). Particle filters in robotics. In: *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. Cambridge: MIT Press.

Tillmann, C., & Ney, H. (2003). Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, *29*(1), 97–133.

Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE TPAMI*, *10*(5), 695–703.

Vassilevska, V., Williams, R., & Yuster, R. (2010). Finding heaviest h-subgraphs in real weighted graphs, with applications. *ACM Transactions on Algorithms (TALG)*, *6*(3), 44.

Williams, V. V., & Williams, R. (2013). Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing*, *42*(3), 831–854.

Wolfson, H., Schonberg, E., Kalvin, A., & Lamdam, Y. (1988). Solving jigsaw puzzles by computer. *Annals of Operations Research*, *12*, 51–64.

Yang, X., & Latecki, L. J. (2010). Weakly supervised shape based object detection with particle filter. In *European Conference on Computer Vision*, Vol. 1, pp. 757–770.

Yang, X., Adluru, N., & Latecki, L. J. (2011). Particle filter with state permutations for solving image jigsaw puzzles. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 2873–2880.

Yao, F. H., & Shao, G. F. (2003). A shape and image merging technique to solve jigsaw puzzles. *PRL*, *24*, 1819–1835.

Zaslavskiy, M., Bach, F., & Vert, J. (2009). A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(12), 2227–2242.

Zass, R., & Shashua, A. (2008). Probabilistic graph and hypergraph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 1–8.

Zhou, F., & De la Torre, F. (2012). Factorized graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 127–134.

Zhou, F., & De la Torre, F. (2013). Deformable graph matching.In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 2922–2929.