Soft Comput
DOI 10.1007/s00500-014-1310-0

METHODOLOGIES AND APPLICATION

Abductive inference in Bayesian networks using distributed overlapping swarm intelligence

Nathan Fortier · John Sheppard · Shane Strasser

Abstract In this paper we propose several approximation algorithms for the problems of full and partial abductive inference in Bayesian belief networks. Full abductive inference is the problem of finding the k most probable state assignments to all non-evidence variables in the network while partial abductive inference is the problem of finding the k most probable state assignments for a subset of the non-evidence variables in the network, called the explanation set. We developed several multi-swarm algorithms based on the overlapping swarm intelligence framework to find approximate solutions to these problems. For full abductive inference a swarm is associated with each node in the network. For partial abductive inference, a swarm is associated with each node in the explanation set and each node in the Markov blankets of the explanation set variables. Each swarm learns the value assignments for the variables in the Markov blanket associated with that swarm's node. Swarms learning state assignments for the same variable compete for inclusion in the final solution.

Keywords Abductive inference · Particle swarm optimization · Distributed computing

Communicated by A. Castiglione.

N. Fortier (✉) · J. Sheppard · S. Strasser
Department of Computer Science, Montana State University, Bozeman, MT 59717, USA
e-mail: nathan.fortier@gmail.com

J. Sheppard
e-mail: john.sheppard@cs.montana.edu

S. Strasser
e-mail: shane.strasser@cs.montana.edu

1 Introduction

Bayesian networks (BNs) have become popular in the artificial intelligence community and are commonly used in probabilistic expert systems. One reason for their popularity is that they are capable of providing compact representations of joint probability distributions by exploiting conditional independence properties of the distributions. BNs are directed acyclic graphs where the nodes represent random variables and edges specify conditional dependence and independence assumptions between the variables. Each node in the network has an associated conditional probability distribution that specifies a distribution over the values of a variable given the possible joint assignments of its parents.

There are several problems associated with performing inference on Bayesian networks. One such problem is that of abductive inference. Abductive inference is the problem of finding the maximum a posteriori (MAP) probability state of the variables of a network, given a set of evidence variables and their corresponding state. This problem is often referred to as the k-most probable explanation (k-MPE) problem. If we let $X_U = X \backslash X_O$, where X denotes the variables in the network, the problem of abductive inference is to find the most probable state assignment to the variables in X_U given the evidence $X_O = x_O$:

$$\text{MPE}(X_U, x_O) = \underset{x_u \in X_U}{\arg\max}\; P(x_u | x_O)$$

In Shimony (1994), it was shown that abductive inference for Bayesian networks is non-deterministic Polynomial-time hard (NP-hard). Because of this, much research has been done to explore the possibilities of obtaining partial or approximate solutions to the problem of both partial abductive inference and full abductive inference. However, in Dagum and Luby (1993) it was shown that even the prob-

Published online: 21 May 2014

lem of finding a constant factor approximation of the k-MPE is NP-hard.

Partial abductive inference (PAI) is the problem of finding the k most probable state assignments for a subset of the variables $X_E \subset X_U$, known as the explanation set:

$$\begin{aligned}\mathrm{PAI}(X_E, x_O) &= \arg\max_{x_E \in X_E} P(x_E|x_O)\\ &= \arg\max_{x_E \in X_E} \sum_{x_R \in X_R} P(x_E, x_R|x_O),\end{aligned}$$

where $X_R = X_U \backslash X_E$. This problem is more useful than general abductive inference in most practical applications since it allows for the selection of only the relevant variables as the explanation set. However, Gamez (1998) proved that partial abductive inference is an even more complex problem than full abductive inference.

In this paper, we present several swarm-based approximation algorithms to the partial and full abductive inference problems. Our approach to these problems is based on the overlapping swarm intelligence (OSI) framework, first introduced by Haberman and Sheppard (2012). In all of the algorithms presented here, a swarm is associated with a subset of the nodes in the network. Each swarm learns the value assignments for variables in the Markov blanket associated with that swarm's node. Swarms learning state assignments for the same variable are said to overlap. These swarms compete to determine which state assignments will be used in the final solution. Each swarm in our approach uses the discrete multi-valued PSO algorithm described in Veeramachaneni et al. (2007) to search for partial solutions; however, any swarm-based algorithm suitable for the task can be used.

For both problems, we developed both distributed and non-distributed versions of the algorithms. In the non-distributed algorithms, all swarms have access to the set of the k best global state assignments seen so far, while in the distributed algorithms, such global information is not maintained and consensus is reached through inter-swarm communication. We refer to the distributed versions of these algorithms as distributed overlapping swarm intelligence (DOSI).

We compare the results of our algorithms to several other local search algorithms. We hypothesized and will show that our algorithms outperform these competing algorithms in terms of the log likelihood of solutions found on a variety of complex networks.

Table 1 provides definitions for all abbreviations used in this work:

Table 1 Abbreviations used

Abbreviation	Description
PSO	Particle swarm optimization
PAI	Partial abductive inference
MPE	Most probable explanation
NP	Non-deterministic polynomial-time
DOSI	Distributed overlapping swarm intelligence
DOSI-Comp	DOSI (no competition mechanism)
DOSI-Comm	DOSI (no communication mechanism)
DMVPSO	Discrete particle swarm optimization
GA-Full	Genetic algorithm (full abductive inference)
GA-Part	Genetic algorithm (partial abductive inference)
MBE	Mini-bucket elimination
NGA	Nitching genetic algorithm
OSI	Overlapping swarm intelligence
SA	Simulated annealing
SLS	Stocastic local search

2 Background

2.1 Bayesian networks

A Bayesian network is a directed acyclic graph that encodes a joint probability distribution over a set of random variables, where each variable can assume one of an arbitrary number of mutually exclusive values (Koller and Friedman 2009). In a Bayesian network, each random variable is represented by a node, and edges between nodes in the network represent a probabilistic relationships between the random variables. Each root node contains a prior probability distribution while each non-root node contains a probability distribution conditioned on the node's parents. For any set of random variables in the network, the probability of any entry of the joint distribution can be computed using the chain rule.

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_{i+1}, ..., X_n)$$

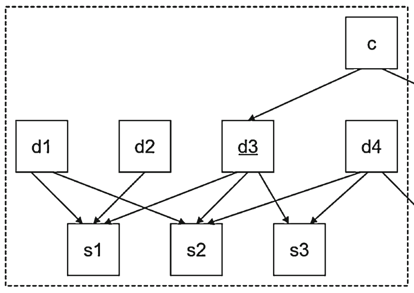
Using the local distributions specified by the BN, the joint distribution can be represented equivalently as

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\mathrm{Pa}(X_i)).$$

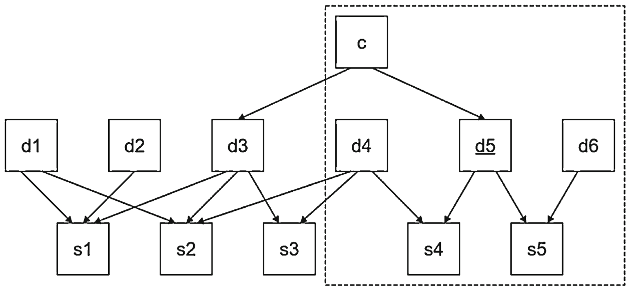
In a Bayesian network, the Markov blanket of a node consists of the node's parents, children, and children's parents. A variable X_i is conditionally independent of all other variables in the network given its Markov blanket.

$$\{X_i \perp (\mathbf{X}\backslash(\{X_i\} \cup MB(X_i)))\,|MB(X_i)\}$$

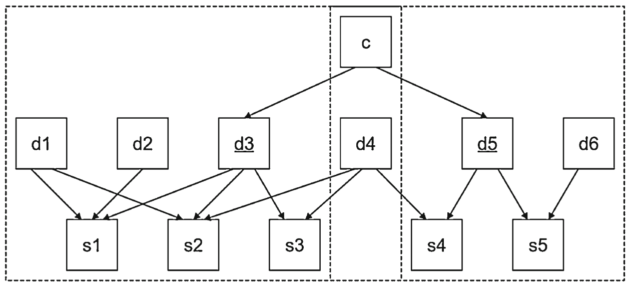
An example illustrating the concept of a Markov blanket is shown in Fig. 1. Figure 1a shows the Markov blanket of d_3, Fig. 1b shows the Markov blanket of d_5, and Fig. 1c shows the Markov blankets of both d_3 and d_5. In the example, nodes in the Markov blanket of d_3 and nodes in the Markov blanket d_5 are shown with a dashed rectangle. In Fig. 1c nodes that are in



(a) Markov blanket of d_3



(b) Markov blanket of d_5



(c) Overlap of Markov blankets for d_3 and d_5

Fig. 1 Markov blanket example

the Markov blankets of both d_3 and d_5 (namely c and d_4) are shown to be inside both dashed rectangles, thus indicating an overlap. We will exploit these overlaps later.

2.2 Particle swarm optimization

All algorithms proposed here are based on the particle swarm optimization (PSO) algorithm proposed by Eberhart and Kennedy (1995). PSO is a population-based search technique inspired by the behavior of fish schools and bird flocks. In PSO the population is initialized with a number of random solutions called particles. The search process updates the position vector of each particle based on that particle's corresponding velocity vector. These velocity vectors are updated at each iteration based on the fitness of the states visited by the particles. Eventually all particles move closer to an optimum in the search space. The pseudocode for the traditional PSO algorithm is presented in Algorithm 1.

Algorithm 1 Particle Swarm Optimization

```
repeat
  for each particle position $x_i \in \mathbf{P}$ do
    Evaluate position fitness $f(x_i)$
    if $f(x_i) > f(p_i)$ then
      $p_i = x_i$
    end if
    if $f(x_i) > f(p_g)$ then
      $p_g = x_i$
    end if
    $v_i = \omega v_i + \mathrm{U}(0, \phi_1) \otimes (p_i - x_i) + \mathrm{U}(0, \phi_2) \otimes (p_g - x_i)$
    $x_i = x_i + v_i$
  end for
until termination criterion is met
```

PSO begins by randomly initializing a swarm of particles over the search space. On each iteration of the algorithm, a particle's fitness is calculated using the fitness function. The personal best position is maintained in the vector p_i. The global best position found among all particles is maintained in the vector p_g. At the end of each iteration a particle's velocity, v_i, is updated based on p_i and p_g. The use of both personal best and global best positions in the velocity equation ensures diverse responses within the swarm and provides a balance between exploration and exploitation.

In Algorithm 1, $\mathbf{P}$ is the particle swarm, $U(0, \phi_i)$ is a vector of random numbers uniformly distributed in $[0, \phi_i]$, $\otimes$ is component-wise multiplication, v_i is the velocity of a particle, and x_i is the object parameters or position of a particle.

Three parameters need to be defined for the PSO algorithm:

– ϕ_1 determines the maximum force with which a particle is pulled toward p_i;
– ϕ_2 determines the maximum force with which a particle is pulled toward p_g;
– ω is the inertia weight.

The inertia weight ω is used to control the scope of the search and eliminate the need for a maximum velocity. Even so, it is customary to specify maximum velocity as well.

2.3 Discrete particle swarm optimization

Kennedy and Eberhart (1997) present a variation of the original PSO algorithm for problems with binary-valued solutions. In this algorithm, each particle's position is a vector from the d-dimensional binary solution space $x_i \in \{0, 1\}^d$ and each particle's velocity is a vector from the d-dimensional continuous space, $v_i \in [0, 1]^d$. Each velocity term denotes the probability of a particle's position term having a value of 0 or 1 in the next iteration. The velocity of a particle is updated as in traditional PSO (Eberhart and Kennedy 1995), while each particle's position is updated using the following equation:

$$p(x_i = 1) = \frac{1}{1 + \exp(-v_i)}$$

While this algorithm has been shown to be effective, it is limited to discrete problems with binary valued solution elements.

The binary state assumption was relaxed by Veeramachaneni et al. (2007) who propose a discrete multi-valued PSO (DMVPSO) algorithm. In this algorithm, each particle's position is a d-dimensional vector of discrete values in the range $[0, M - 1]$ where M is the cardinality of each state variable. The velocity of each particle is a d-dimensional vector of continuous values, as above. A particle's velocity v_i is transformed into a number S_i between $[0, M]$ using the following equation:

$$S_i = \frac{M}{1 + \exp(-v_i)}$$

Then each particle's position is updated by generating a random number according to the Gaussian distribution, $x_i \sim N(S_i, \sigma \times (M - 1))$ and rounding the result. To ensure the particle's position remains in the range $[0, M - 1]$ the following formula is applied:

$$x_i = \begin{cases} M - 1 & x_i > M - 1 \\ 0 & x_i < 0 \\ x_i & \text{otherwise} \end{cases}$$

2.4 Consensus

In a network of agents, a consensus is reached once the agents agree regarding certain quantities of interest that depend on the state of all agents (Olfati-Saber et al. 2007). A consensus algorithm is an interaction rule that specifies the information exchange between agents to ensure that consensus is reached. In such algorithms cooperation between agents is often required, where cooperation is defined as "giving consent to providing one's state and following a common protocol that serves the group objective." We have developed a cooperative consensus algorithm in the context of DOSI for full and partial abductive inference. In our distributed abductive inference algorithm we have removed the global set of state assignments used for fitness evaluation. Instead, each swarm maintains a set of personal state assignments that are updated through inter-swarm communication. During this communication, swarms compete and share state assignments to ensure consensus. This communication mechanism defines an interaction rule that ensures that the agents will reach consensus regarding the states of each node in the network.

3 Related work

3.1 Traditional approaches to the k-MPE problem

An exact algorithm to solve the k-MPE problem called bucket elimination was proposed by Dechter (1996). This algorithm uses a form of variable elimination in which the node with the fewest neighbors in eliminated at each iteration. Bucket elimination uses max-marginal-ization instead of sum-marginalization when eliminating a variable and stores the most probable state assignment for the variable. Like variable elimination, this algorithm has worst-case time complexity that is exponential in the tree width of the network.

An approximation algorithm for the MPE problem called mini bucket elimination (MBE) is described in Dechter (2003). This algorithm is a variation of bucket elimination and can address both partial and full abductive inference. The parameters of MBE can be modified so that the algorithm produces exact solutions to the abductive inference problem.

A divide and conquer algorithm that provides an exact solution to the k-MPE problem is described by Nilsson (1998). Nilsson's approach is based on the flow propagation algorithm proposed by Dawid (1992) for finding the k-MPE for junction trees. While the algorithm is faster than other exact abductive inference algorithms such as bucket elimination, it also has exponential time complexity and is impractical for large networks.

A simulated annealing algorithm (SA) for partial abductive inference was proposed by de Campos et al. (2001). This approach uses an evaluation function based on clique tree propagation. The algorithm begins with a single state assignment, and at each iteration a single variable is modified within that assignment. A hash table is maintained consisting of (assignment, probability) pairs and each new assignment is stored in this table. If an assignment is found that is not stored in the hash table, then the probability of the assignment is computed. Since SA only modifies a single state at each iteration, the algorithm can avoid recalculating all of the initial clique potentials when evaluating a new state assignment.

A stochastic local search (SLS) algorithm for solving the MPE problem was proposed by Kask and Dechter (1999). In this approach, stochastic local search was combined with Gibbs Sampling. The results of the author's experiments indicate that their approach outperforms other techniques such as stochastic simulation, simulated annealing, or hillclimbing alone.

The Elvira Consortium software environment uses a junction tree-based algorithm to approximate a solution to the k-MPE Problem (Elvira Consortium 2002). This algorithm is based on Nilsson's algorithm, but approximate probability trees are used in place of the true probability trees.

3.2 Soft approaches to the k-MPE problem

Several researchers have used soft computing techniques to find approximate solutions to the k-MPE problem. Gelsema (1995) describes a genetic algorithm for full abductive inference (GA-Full) in Bayesian networks. In this approach, the states of the variables in the Bayesian network are represented by a chromosome corresponding to a vector of Boolean values. Each value in the chromosome corresponds to a state assignment for a node in the network. Crossover and mutation are applied to the chromosomes to generate offspring from parent chromosomes. To evaluate chromosome fitness, the chain rule is applied.

A graph-based evolutionary algorithm for performing approximate abductive inference on Bayesian Networks was developed by Rojas-Guzman and Kramer (1993). In their method, the graphs specify a possible solution that is a complete description of a state assignment for a Bayesian network. Fitness of a chromosome is based on the absolute probability of the chromosome's corresponding assignment. However, this approach was designed to find a single most probable explanation rather than the k-MPE.

A genetic algorithm for partial abductive (GA-Part) inference was proposed by de Campos et al. (1999). In this approach, the state assignments for the subset of variables are represented as a chromosome consisting of integers. To evaluate the fitness of each chromosome, probabilistic propagation is used.

Sriwachirawat and Auwatanamongkol (2006) developed a niching genetic algorithm (NGA) to find k-MPE, designed to utilize the "multifractal characteristic and clustering property" of Bayesian networks. This algorithm makes use of the observation that there are regions within the joint probability distribution of the Bayesian network that are highly "self-similar." Because of this self-similarity, the authors chose to organize their GA using a probabilistic crowding method that biases the crossover operator toward self-similar individuals in the population. Chromosomes in this approach were encoded as described by Gelsema (1995).

Pillai and Sheppard (2012) describe a discrete multi-valued PSO (DMVPSO) approach for finding k-MPE (Veeramachaneni et al. 2007). In Pillai's algorithm, each particle's set of object parameters is represented by a string of integers corresponding to state assignments for each node in the network. The chain rule is used to calculate the fitness of each particle. The results of the authors' experiments indicated they were able to find competitive explanations much more efficiently than the approaches used by Gelsema (1995) and Sriwachirawat and Auwatanamongkol (2006).

3.3 Distributed optimization

Much work has been done in the area of distributed optimization. Rabbat and Nowak (2004) analyzed the convergence of distributed optimization algorithms in sensor networks. The authors proved that, for a large set of problems, these algorithms will converge to a solution within a certain distance of the global optimum.

Olfati-Saber et al. (2007) provided an analysis of consensus algorithms for multi-agent networked systems. The authors defined several types of consensus problems and described methods of convergence and performance analysis multi-agent distributed systems.

Patterson et al. (2010) provided an analysis of the convergence rate for the distributed average consensus algorithm. This work also included an analysis of the relationship between the convergence rate and the network topology.

Boyd et al. (2011) analyzed convex distributed optimization in the context of machine learning and statistics. The authors argued that the alternating direction method of multipliers (ADMM) can be applied to such distributed optimization algorithms. In ADMM a problem is divided into small local subproblems that are solved and used to find a solution to a large global problem. The authors showed that this approach can be applied to a wide variety of distributed optimization problems.

3.4 Distributed soft computing

Several distributed genetic algorithms (GA) have been proposed, which are commonly referred to as Island Models (Tanese et al. 1989; Whitley and Starkweather 1990; Belding 1995; Whitley et al. 1999). In these models, several subpopulations known as islands are maintained by the genetic algorithm, and members of the populations are exchanged through a process called migration. These methods have been shown to obtain better quality solutions than traditional GAs (Whitley and Starkweather 1990). Because the islands maintain some independence, each island can explore a different region of the search space while sharing information with other islands through migration. This improves genetic diversity and solution quality (Whitley et al. 1999).

van den Bergh and Engelbrecht (2000) developed several distributed PSO methods for the training of multi-layer feedforward neural networks. These methods include NSPLIT in which there is a single particle swarm for each neuron in the network and LSPLIT in which there is a swarm assigned to each layer of the network. The results obtained by van den Bergh and Engelbrecht indicate that the distributed algorithms outperform traditional PSO methods. Note, however, that these methods do not include any communication between the swarms and provide access to a global fitness function.

Recently, a new distributed approach to improve the performance of the PSO algorithm has been explored where multiple swarms are assigned to overlapping subproblems. This approach is called overlapping swarm intelligence (OSI) (Haberman and Sheppard 2012; Pillai and Sheppard 2011; Fortier et al. 2012). In OSI each swarm searches for a partial solution to the problem and solutions found by the different swarms are combined to form a complete solution once convergence has been reached. Where overlap occurs, communication and competition take place to determine the combined solution to the full problem.

Haberman and Sheppard (2012) first proposed OSI as a method to develop an energy-efficient routing protocol for sensor networks that ensures reliable path selection while minimizing the energy consumption during message transmission. This algorithm was shown to be able to extend the life of the sensor networks and to perform significantly better than current energy-aware routing protocols.

Pillai and Sheppard (2011) extended the OSI framework to learn the weights of deep artificial neural networks. In this algorithm, the structure of the network is separated into paths where each path begins at an input node and ends at an output node. Each of these paths is associated with a swarm that learns the weights for that path of the network. A common vector of weights is maintained across all swarms to describe a global view of the network. This vector is created by combining the weights of the best particles in each of the swarms. This method was shown to outperform the backpropagation algorithm, the traditional PSO algorithm, and both NSPLIT and LSPLIT on deep networks. A distributed version of this approach was developed subsequently by Fortier et al. (2012).

3.5 Swarm intelligence algorithms

Aside from PSO, a variety of swarm-based algorithms have been proposed by several authors. An in-depth discussion of these algorithms including a comparison against PSO is presented by Yang et al. (2013).

The Ant colony optimization (ACO) algorithm described by Dorigo et al. (2006) is based on the foraging behavior of ants. In ACO a population of ants build solutions to a given optimization problem and the ants exchange information on the quality of these solutions via a communication mechanism based on the pheromone producing behavior of ants in nature. Each artificial ant builds a solution by traversing a fully connected construction graph associated with the problem. As ants traverse the graph, they deposit a certain amount of pheromone on the edges of the graph. The amount of pheromone deposited depends on the quality of the solution found and ants have a higher likelihood of traversing edges with higher pheromone values.

The artificial bee colony (ABC) algorithm developed by Karaboga and Basturk (2008) is based on the behavior of honeybee swarms. In this algorithm, a colony of artificial bees is used to obtain solutions to a given optimization problem. The bee colony used in ABC contains three groups of bees: employed bees, onlookers, and scouts. These bees explore food sources that represent potential solutions within the search space. Employed bees explore the neighborhood of known food sources and communicate quality information to onlookers who then select one of the food sources, while scouts randomly search for new food sources.

A novel swarm-inspired algorithm called the Krill Herd (KH) algorithm was proposed by Gandomi and Alavi (2012). This algorithm is based on the herding behavior of krill swarms in response to biological and environmental processes. In KH, the fitness of each individual is a combination of the distance from the best solution found and from the highest density of the swarm. Three actions are considered to determine the position of an individual krill at each iteration: movement induced by other krill individuals, foraging activity, and random diffusion. Movement induced by other krill individuals causes each individual to be attracted to more fit individuals while being repelled by less fit individuals. This causes the krill to move toward better positions in the search space. After each iteration mutation and crossover are applied to the population to generate new individuals.

The Cuckoo search algorithm proposed by Gandomi et al. (2013) is based on the breeding behavior of some cuckoo species in combination with the Lévy flight behavior of birds and fruit flies. A Lévy flight is a random walk with steps defined in terms of step-lengths which are associated with a certain probability distribution. The directions of the steps in a Lévy flight are isotropic and random. In the Cuckoo search algorithm, eggs are used to represent potential solutions to an optimization problem and lower quality eggs are replaced by eggs representing higher quality solutions. To begin, an initial population of solution eggs is generated. At each iteration, a cuckoo generates a new solution by Lévy flights, evaluates its quality, and randomly chooses an existing solution to potentially replace. If the new solution is of a higher quality, then it replaces the old solution. Next, some percentage of the worst solutions are removed and replaced with new solutions.

The Firefly algorithm (FA) developed by Yang (2009) is a swarm-based algorithm based on firefly mating behavior. In this algorithm, a population of fireflies is initialized where the position of each firefly denotes a potential solution to an optimization problem. Each firefly has a brightness value based on the fitness of the solution encoded by its position. Fireflies move based on the brightness value of other fireflies, their distance to other fireflies, and a random coefficient. Unlike PSO, individuals in the firefly algorithm do not explore the search space based on the individual's personal best position combined with the global best position found by the algorithm as a whole. Instead, each firefly explores the solution space based on the fitness of its neighbors and its distance from those neighbors.

4 Full abductive inference

Here we describe our distributed and non-distributed approaches to solving the full abductive inference problem. We also discuss several experiments comparing these approaches to existing full abductive inference algorithms.

4.1 Full abductive inference via OSI

Previously, we developed an approach for full abductive inference based on the OSI methodology (Fortier et al. 2013). In our approach a swarm is associated with each node in the network. Each node's corresponding swarm learns the variable assignments associated with that node's Markov blanket using the DMVPSO algorithm. This representation provides an advantage since every node in the network is conditionally independent of all other nodes when conditioned on its Markov blanket. The pseudocode for our approach is shown in Algorithm 2.

The main loop of this algorithm consists of two phases. In the first phase, (lines 4–17) a single iteration of the traditional PSO algorithm is performed for each swarm. In the second phase, (lines 19–26) for each of the k state assignments, each swarm competes over the state of each node in the network. The way in which the swarms compete is based on the overlap of the Markov blankets for each node in the network.

This algorithm requires that a set of k global state assignments A be maintained across all swarms for inter-swarm communication. A is initialized by forward sampling $m \geq k$ samples and selecting the k most probable for inclusion in A. Each particle's position is defined by a d-dimensional vector of discrete values. Each position value corresponds to the state of a variable in the swarm's Markov blanket; thus each particle represents a state assignment for part of the network. We use log likelihood ℓ to determine the quality of a complete state assignment as follows:

Algorithm 2 Overlapping Swarm Intelligence

```
 1: Initialize $A$ using forward sampling
 2: Initialize particles in each swarm

 3: repeat
 4:   for each swarm $s$ do
 5:     for each particle, $p \in s$ do
 6:       Construct $B_p$
 7:       Add $B_p$ to $A$
 8:       Calculate particle fitness $f(p)$
 9:       if $f(p) >$ $p$'s personal best fitness then
10:         Update $p$'s personal best position and fitness
11:       end if
12:       if $f(p) >$ the global best fitness then
13:         Update global best position and fitness for $s$
14:       end if
15:       Update $p$'s velocity and position
16:     end for
17:   end for

18:   $A \leftarrow k$ most probable assignments in $A$

19:   for each state assignment $\alpha \in A$ do
20:     for each node $n$ in the network do
21:       Let $S$ be all swarms where $n \in mb_s$
22:       Let $v_n$ be the state of $n$ in $\alpha$
23:       $v_n \leftarrow Compete(S, n, \alpha)$
24:       $SeedSwarms(S, v_n)$
25:     end for
26:   end for
27:   Add $\alpha$ to $A$
28: until termination criterion is met

29: return $\alpha$
```

$$\ell(X) = \log\left(\prod_{i=1}^{n} P(x_i|\text{Pa}(x_i))\right) = \sum_{i=1}^{n} \log P(x_i|\text{Pa}(x_i)),$$

where $X = \{x_1, x_2 ... x_n\}$ is a complete state assignment and $\text{Pa}(x_i)$ corresponds to the assignments for the parents of x_i.

Here we describe the process used to evaluate the fitness of each individual particle p. This evaluation requires that a set of state assignments B_p be constructed using the state of the particle p and the set of global state assignments A. The sum of the log likelihoods for each state assignment in B_p will be used as the particle's fitness score. Given a partial state assignment x_p represented by some particle p in swarm s and the set of complete global state assignments $A = \{\alpha_1, \ldots, \alpha_k\}$ we can construct a new set of state assignments $B_p = \{\beta_1, \ldots, \beta_k\}$ by inserting x_p into each state assignment $\alpha_i \in A$ as follows:

$$\forall \beta_i \in B_p \quad \beta_i = \{x_p\} \cup \alpha_i \backslash mb_s,$$

where mb_s consists of the state assignments for the Markov blanket of swarm s within A_i. We use B_p to calculate the fitness f of each particle,

$$f(p) = \sum_{\beta_i \in B_p} \ell(\beta_i)$$

This function defines the fitness of particle p as the sum of the log likelihoods of the assignments in A when the value assignments encoded in p are substituted into A.

Algorithm 3 $SeedSwarms(S, v_n)$

```
1: for each swarm $s \in S$ do
2:     Let $p$ be the least fit particle in $s$
3:     Let $q_n$ be the state of $n$ in $p$
4:     $q_n \leftarrow v_n$
5: end for
```

Algorithm 4 $Compete(S, n, \alpha)$

```
1: $bestQ \leftarrow -\infty$
2: for each swarm $s \in S$ do
3:     Let $p_g$ be the most fit particle in $s$
4:     Let $v_n$ be the state of $n$ within $p_g$
5:     Insert $v_n$ into $\alpha$
6:     if $\ell(\alpha) > bestQ$ then
7:         $bestQ \leftarrow \ell(\alpha)$
8:         $bestV \leftarrow v_n$
9:     end if
10: end for
11: return $bestV$
```

When two or more swarms share a node in the network (such as c and d_4 in Fig. 1) these swarms are said to overlap. At the end of each iteration of the algorithm, overlapping swarms compete to determine which state is assigned to a given node in each assignment $\alpha_m \in A$. This process is shown in Algorithm 4. Algorithm 4 iterates over each competing swarm and evaluates the fitness of the state v_n encoded in the best fit particle of the swarm. This evaluation is performed by setting the state of node n in the assignment α to the value v_n and computing the log likelihood of this new assignment. The state of node n that results in the highest score is the state included in the state assignment α.

The competition described in Algorithm 4 is held between the state assignments found by the personal best particles in each swarm. The state resulting in the highest log-likelihood is the one selected for inclusion. Once a node's state has been selected through competition, each swarm associated with the node is seeded with this state. This is performed by the *SeedSwarms* function shown in Algorithm 3. This function iterates over each of the swarms S and replaces the least fit particle in the swarm with the state v_n for node n. This allows for the transfer of information between different swarms that are trying to optimize over the same values. In the example presented in Fig. 1, the swarms associated with d_3 and d_5 would compete to determine which state is assigned to the nodes c and d_4.

Whenever a state assignment is constructed, that assignment is stored in A. At each iteration, A is pruned so that it contains the k most probable assignments found so far. Once the algorithm has terminated, A is returned.

4.2 Full abductive inference via DOSI

We have modified the OSI approach proposed in Fortier et al. (2013) to eliminate the need for a set of global state assignments to be maintained across all swarms. In the distributed approach each swarm s maintains a set of personal state assignments denoted as A_s. The most probable assignments learned by each swarm are communicated to the other swarms and inserted into the their personal state assignments through a periodic communication mechanism. In this context, the most probable assignment learned by a swarm denotes the state assignment that has the highest probability with respect to the swarm's set of personal state assignments. Since this does not denote the probability with respect to any global network, no global set of state assignments is required to compute this probability. At each iteration of the algorithm the swarm associated with a given node will hold a competition between all swarms that share this node to determine the most probable state assignment. This state assignment will then be communicated to the other swarms. Because this approach does not require a global network to be shared between swarms, the learning process can be distributed. Initially a set A is obtained by forward sampling $m \geq k$ samples. Each A_s is initialized as the k most probable assignments in A.

Algorithm 5 shows the pseudocode for DOSI. Similar to OSI, the main loop of this algorithm consists of two phases. In the first phase (lines 8–22), a single iteration of the traditional PSO algorithm is performed for each swarm. In the second phase (lines 23–38), overlapping swarms compete over their set of personal state assignments and a communication mechanism is used to share information between the swarms.

The fitness evaluation for each particle requires that a set of state assignments $B_{s,p}$ be constructed using the state of the particle p and the set of personal state assignments A_s associated with the swarm s of particle p. The sum of the log likelihoods for each state assignment in $B_{s,p}$ will be used as the particle's fitness score. The fitness calculation and the calculation of $B_{s,p}$ for a given swarm s and particle p is similar to the calculation of B_p used in OSI. Given a partial state assignment x_p represented by some particle p in swarm s and the set of personal state assignments $A_s = \{\alpha_1, \ldots, \alpha_k\}$ we can construct a new set of state assignments $B_{s,p} = \{\beta_1, \ldots, \beta_k\}$ by inserting x_p into each state assignment $\alpha_i \in A_s$ as follows:

Algorithm 5 Distributed Overlapping Swarm Intelligence

```
1: Sample $A$ using forward sampling
2: Initialize particles in each swarm
3: for each swarm $s$ do
4:    Create an empty list of assignments $A_s$
5:    Add $k$ most probable assignments in $A$ to $A_s$
6: end for

7: repeat
8:    for each swarm $s$ do
9:       for each particle, $p \in s$ do
10:         Construct $B_p$
11:         Add $B_p$ to $A_s$
12:         Calculate particle fitness $f(p)$
13:         if $f(p) >$ $p$'s personal best fitness then
14:            Update $p$'s personal best position and fitness
15:         end if
16:         if $f(p) >$ the global best fitness then
17:            Update global best position and fitness for $s$
18:         end if
19:         Update $p$'s velocity and position
20:      end for
21:      $A_s \leftarrow k$ most probable assignments in $A_s$
22:   end for

23:   for each node $n$ in the network do
24:      Let $s$ be the swarm associated with $n$
25:      for each state assignment $\alpha \in A_s$ do
26:         Let $S$ be all swarms $x$ where $n \in mb_x$
27:         Let $v_n$ be the state of $n$ in $A$
28:         $v_n \leftarrow Compete(S, n, \alpha)$
29:         $SeedSwarm(s, v_n)$
30:      end for

31:      for $i = 0$ to $C$ do
32:         for each swarm $s_i$ do
33:            for each swarm $s_j \in mb_{s_i}$ do
34:               $ShareStates(s_i, s_j)$
35:            end for
36:         end for
37:      end for
38:   end for
39: until termination criterion is met

40: for each swarm $s$ do
41:    Add $A_s$ to $A$
42: end for
43: $A \leftarrow k$ most probable assignments in $A$
44: return $A$
```

$$\forall \beta_i \in B_{s,p}$$
$$\beta_i = \{x_p\} \cup \alpha_i \backslash mb_s,$$

where mb_s consists of the state assignments for the Markov blanket of swarm s within α_i. We use $B_{s,p}$ to calculate the fitness f of each particle,

$$f(p) = \sum_{\beta_i \in B_{s,p}} \ell(\beta_i)$$

This function defines the fitness of particle p in a swarm s as the sum of the log likelihoods of the assignments in α_s when the state assignments encoded in p are substituted into α_s.

Algorithm 6 $ShareStates(s_i, s_j)$

```
1: Let $A_i$ and $A_j$ be the set of personal state assignments
   of $s_i$ and $s_j$ respectively

2: for each node $n$ do
3:    if $s_i.\delta_n > s_j.\delta_n$ then
4:       $s_i.\delta_n \leftarrow s_j.\delta_n + 1$
5:       for $m = 0$ to $k$ do
6:          Let $\alpha_i$ be the $m^{th}$ assignment in $A_{s_i}$
7:          Let $\alpha_j$ be the $m^{th}$ assignment in $A_{s_j}$
8:          Insert $\alpha_j$'s value for $n$ into $\alpha_i$
9:       end for
10:   end if
11: end for
```

Unlike OSI, once a node's state has been selected by the competition function, only the swarm associated with the node is seeded with the new state. This is done to reduce communication overhead. To share state assignments, each swarm keeps track of a variable δ_n for each node n. These variables indicate the minimum distance between the swarm's node and n in the moralized graph of the Bayesian network. For example, a value of zero for δ_n indicates that n is the swarm's corresponding node while a value of one for δ_n indicates that the swarm's node is in the Markov blanket of n. Values for δ_n are initially set to infinity for all nodes other than the node assigned to that swarm. Values for δ_n are set to 0 for the node assigned to the swarm. At each iteration of the communication the tentative distances between each pair of nodes are updated. A swarm s_j will communicate its value for a node n to another swarm s_i if $s_i.\delta_n > s_j.\delta_n$. This is done because nodes that have a smaller value for δ_n are closer to n and, therefore, have a more current value for the state of n. This communication mechanism ensures that DOSI will reach consensus with respect to the values for δ_n and the assigned values for a given state.

The communication mechanism is described in Algorithm 6. This algorithm iterates over each node n in the network. If $s_i.\delta_n > s_j.\delta_n$, then $s_i.\delta_n$ is set to the incremented value of $s_j.\delta_n$ and the state assignment for n in the A_{s_j} is inserted into A_{s_i}.

An example of several iterations of the communication portion of the algorithm is shown in Fig. 2. Here the most probable state assignment for F in each personal state assignment is shown along with each swarm's value for δ_F at the beginning of each iteration. Values that are changed during a given iteration are shaded and the nodes corresponding to these changes are highlighted. In this example the swarm associated with node F has determined through competition that the best state assignment for F is 1. Prior to the first iteration of communication $\delta_F = 0$ for the swarm associated with F while $\delta_F = \infty$ in all other swarms. After the first iteration

$$D.\delta_F = F.\delta_F + 1 = 1$$

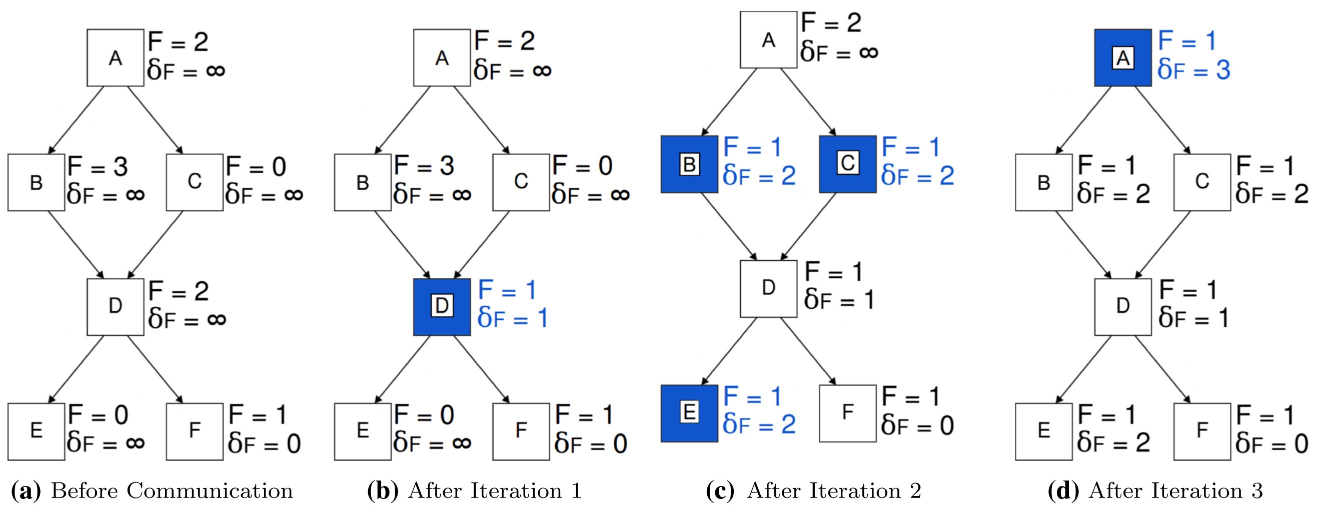


Fig. 2 Share states example

since $D \cdot \delta_F = \infty > 0 = F \cdot \delta_F$. When D updates its value for δ_F it receives the most probable state assignment, $F = 1$, from the swarm associated with F. During the second iteration of communication D will share states with nodes B, C, and E. Once the second iteration is complete, nodes B, C, and E have all set $F = 1$ in their personal state assignment and B, C, and E have all set

$$\delta_F = D.\delta_F + 1 = 2$$

During the third iteration node B will share states with node A. Since $A \cdot \delta_F = \infty > 2 = B \cdot \delta_F$ the swarm associated with A will update its value for δ_F such that

$$A.\delta_F = B.\delta_F + 1 = 3$$

and A will set $F = 1$ based on the state of F in the personal state assignment for node B. Once the third iteration is complete for any given node in the network δ_F will contain the minimum distance between that node and F in the moralized graph of the network and all nodes will agree upon a value for F. Once the personal state assignment for all swarms have identical state assignments for some variable we say that the network has reached consensus with respect to that variable. In this example after the third iteration of communication the swarms will reach consensus with respect to F. Eventually, all incorrect tentative distances will be updated in this way so that they reflect the correct distances between the nodes.

This algorithm is similar to Dijkstra's algorithm in that initially each node is assigned a tentative distance value for each δ_n (zero for our initial node and infinity for all other nodes) which is updated at each iteration through the communication mechanism. After each iteration, each node will update one of its δ's only if one of its neighbors has a lower δ value for the same node. Because each node starts with a δ value of 0 for itself and ∞ for all others, the only δ's that will be update are nodes that are within each other's MBs; therefore, after the first iteration, all δ's will either be 0, 1, or ∞. As this process is repeated, the δ will be increased by one for each node as the value is distributed throughout the network, which also corresponds to the distance from one node to another node in the moralized graph. Eventually, all of the nodes will have finite numbers for all δ's in a fully connected network. If the network is not fully connected, then there will still exist some delta's with values of ∞ for certain nodes.

4.3 Experimental design

For our first set of experiments, we compare our OSI and DOSI algorithms for full abductive inference to SLS, GA-Full, NGA, MBE, and DMVPSO. A cross-reference for identifying each of the tested algorithms is shown in Table 1.

For these comparisons we used the bipartite networks presented in Fig. 3 (Pillai and Sheppard 2012) along with four additional Bayesian networks obtained from the Bayesian network repository (Scutari 2012): Win95pts, Insurance, Hailfinder, and Hepar2. The properties for all networks are shown in Table 2. For all networks each leaf node in the network was set as evidence with a 50 % probability. The state of each evidence variable was chosen uniformly at random.

For each network, experiments were performed with different values of k: $k = 2, 4, 6, 8$. For all of the algorithms, initial populations were generated using forward sampling. In every experiment, the number of particles in each swarm was set to 20 and σ was set to 0.2. The value for σ was taken from Pillai and Sheppard (2012) to ensure consistency of results. For the genetic algorithms, the population size was set to 20. All algorithms were run until convergence. The sums of the log likelihoods for the k most fit solutions found in each run were averaged over ten runs of each algorithm.

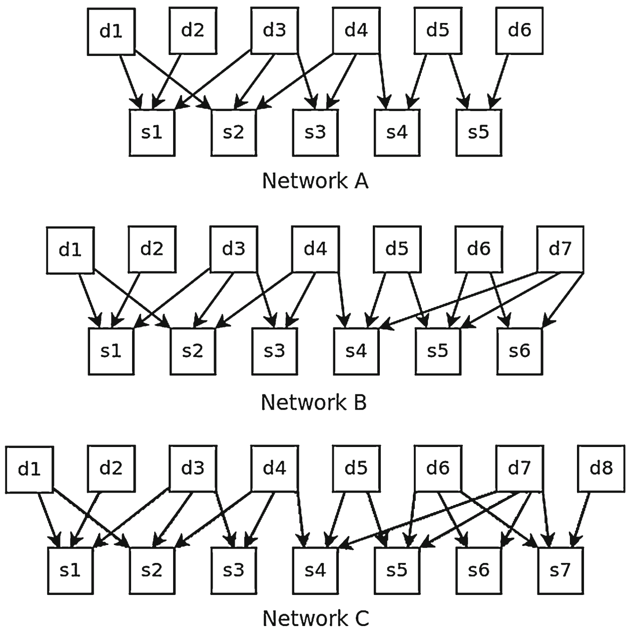


Fig. 3 Bipartite Bayesian networks used for experiments

Table 2 Properties of the test networks

Network	Nodes	Arcs	Parameters	Ave. MB size	Ave. states
Network A	11	12	261	4	3.00
Network B	13	16	399	4.53	3.00
Network C	15	12	483	4.60	3.00
Win95pts	76	112	574	5.92	2.00
Insurance	27	52	984	5.19	3.30
Hailfinder	70	66	2,656	3.54	3.98
Hepar2	56	1,236	1,453	3.51	2.31

We compared using a paired t test with a confidence interval of 95 % to evaluate significance. In addition to log likelihood, we also measured the number of fitness evaluations required by each algorithm for a comparison of computational complexity.

For networks A, B, and C the MBE algorithm is used to compute the exact solution for the full abductive inference problem. For all other networks MBE is used to find an approximate solution with parameters m and i set to 2 and 3, respectively, since the networks are too large for exact inference.

For our second set of experiments we performed a lesion study (Langley 1988) by implementing two alternative versions of DOSI. In the first implementation (DOSI-Comp), the competition mechanism has been disabled, while in the second implementation (DOSI-Comm), the communication mechanism has been disabled. We compare these alternative implementations to DOSI to validate our hypothesis that both the competition and communication of DOSI improve the algorithm's performance in terms of average log likelihood of solutions found. All other parameters and design decisions were the same as the first set of experiments.

4.4 Results

Tables 3, 4, 5 show the average sum of the log likelihoods for each algorithm and each value of k. Bold values indicate that the corresponding algorithm's performance is statistically significantly better than all other algorithms for the network given the corresponding value for k. Algorithms that tie statistically for best are bolded. Table 3 shows the average sum of the log likelihoods for the population-based algorithms while Table 4 shows the average sum of the log likelihoods for MBE and SLS. The results of the comparison between DOSI-Comm and DOSI-Comp are shown in Table 5.

In Fig. 4 we present the convergence plots for OSI, DOSI, and DMVPSO. We are limiting the convergence plot comparison to DMVPSO because the results in Tables 3 and 4 indicate that DMVPSO is the most competitive when compared to OSI and DOSI. In this figure, OSI-Best and DOSI-Best denotes the average score for the best particle in each swarm, while OSI-Avg and DOSI-Avg denotes the average score for all particles in each swarm. PSO-Avg denotes the average score for all particles in the swarm, while PSO-Best denotes the score of the best particle in the swarm. These plots were obtained by running each algorithm with k set to 2.

4.4.1 Comparison against existing algorithms

For all networks containing more than 15 nodes we observe that, based on the paired t tests on log likelihood, the OSI algorithm outperforms all other approximate algorithms and DOSI is never outperformed by any approximate algorithms other than OSI. For Network A all population-based algorithms tie statistically when k is set to 2, OSI, DOSI, and DMVPSO tie statistically when k is set to 2 and 6, and OSI outperforms all other approximate algorithms when k is set to 8. For Network B OSI, DOSI, and DMVPSO tie statistically when k is set equal to 4 while OSI and DOSI tie statistically for best when k is set equal to 2. OSI outperforms all other approximate algorithms when k is set to 6 and 8. For Network C all population-based algorithms tie statistically for best when k is set equal to 2, 4, and 6. OSI outperforms all other approximate algorithms when k is set to 8. For networks A, B, and C the average sum of the log likelihoods for OSI differs from the exact solution by at most 0.4 while, for DOSI, the average sum of the log likelihoods differs from the exact solution by at most 3.11. For Win95pts OSI and DOSI tie statistically for best when k is set equal to 2, 4, and 6. For Insurance OSI and DOSI tie statistically for best for

Table 3 Comparison against population-based approaches

Network	k	OSI	NGA	GA-Full	DMVPSO	DOSI
Network A	2	**−13.64 ± 0.02**	**−14.06 ± 0.51**	**−14.16 ± 0.87**	**−13.95 ± 0.73**	**−13.85 ± 0.58**
	4	**−27.31 ± 0.10**	−29.40 ± 1.59	−29.42 ± 1.69	**−28.33 ± 2.29**	**−27.81 ± 0.70**
	6	**−40.92 ± 0.03**	−46.70 ± 2.77	−47.07 ± 3.77	**−43.07 ± 2.67**	**−41.82 ± 2.51**
	8	**−54.56 ± 0.03**	−60.43 ± 2.12	−64.52 ± 4.18	−59.03 ± 3.68	−55.27 ± 0.45
Network B	2	**−17.12 ± 0.07**	−18.11 ± 0.48	−18.39 ± 0.56	−18.23 ± 0.40	**−17.20 ± 0.16**
	4	**−34.62 ± 0.14**	−36.90 ± 1.14	−37.32 ± 1.49	**−36.66 ± 0.57**	**−35.59 ± 1.45**
	6	**−51.87 ± 0.24**	−57.18 ± 2.63	−58.63 ± 2.16	−57.40 ± 1.90	−52.66 ± 0.64
	8	**−69.26 ± 0.32**	−77.25 ± 1.69	−77.74 ± 2.84	−76.05 ± 2.02	−71.97 ± 2.19
Network C	2	**−16.05 ± 0.02**	**−17.23 ± 1.53**	**−17.61 ± 1.47**	**−17.08 ± 1.26**	**−16.71 ± 0.96**
	4	**−32.09 ± 0.05**	**−37.69 ± 2.87**	**−36.82 ± 1.73**	**−35.05 ± 1.52**	**−34.58 ± 4.24**
	6	**−48.18 ± 0.11**	**−57.09 ± 2.56**	**−57.46 ± 1.69**	**−55.77 ± 2.45**	**−50.74 ± 2.15**
	8	**−64.28 ± 0.19**	−78.68 ± 5.01	−77.51 ± 3.21	−74.67 ± 2.77	−67.08 ± 2.59
Win95pts	2	**−32.27 ± 5.59**	−2,442.06 ± 545.94	−2,866.92 ± 755.73	−941.05 ± 1,236.95	**−40.60 ± 10.96**
	4	**−57.10 ± 3.85**	−4,561.73 ± 1602.60	−5,615.89 ± 2,551.70	−2,162.27 ± 2,810.62	**−125.07 ± 98.46**
	6	**−90.21 ± 13.53**	−9,580.33 ± 1,727.51	−9,314.64 ± 2,208.09	−3,858.00 ± 3,559.27	**−350.87 ± 368.13**
	8	**−124.78 ± 20.97**	−13,885.76 ± 1,843.64	−13,240.43 ± 4,066.10	−5,947.98 ± 5,681.22	−520.60 ± 517.70
Insurance	2	**−24.63 ± 2.29**	−36.57 ± 6.80	−34.55 ± 2.99	−31.85 ± 2.38	**−26.77 ± 4.14**
	4	**−48.85 ± 3.04**	−76.92 ± 19.59	−73.66 ± 7.87	−79.56 ± 8.43	**−54.05 ± 6.08**
	6	**−74.02 ± 9.14**	−120.14 ± 13.54	−133.65 ± 26.96	−132.08 ± 15.91	**−83.40 ± 9.83**
	8	**−100.25 ± 7.25**	−196.69 ± 35.05	−179.64 ± 25.23	−170.20 ± 24.44	**−102.37 ± 12.98**
Hailfinder	2	**−69.51 ± 2.19**	−176.20 ± 234.93	−102.08 ± 5.72	−247.79 ± 313.59	−86.74 ± 8.72
	4	**−142.31 ± 3.85**	−355.61 ± 314.29	−502.07 ± 931.43	−425.46 ± 500.03	−183.56 ± 16.64
	6	**−210.29 ± 6.16**	−1,210.99 ± 1,091.34	−1,069.50 ± 983.59	−1,795.57 ± 1,210.33	−260.12 ± 20.88
	8	**−278.88 ± 9.34**	−2,217.57 ± 1,809.46	−2,282.79 ± 1061.93	−3,986.79 ± 983.68	−357.30 ± 30.19
Hepar2	2	**−66.54 ± 0.00**	−79.40 ± 2.10	−80.79 ± 2.82	−72.92 ± 2.07	**−67.28 ± 1.44**
	4	**−134.19 ± 1.85**	−164.57 ± 3.87	−161.64 ± 4.02	−146.78 ± 7.59	**−136.34 ± 2.64**
	6	**−199.84 ± 0.67**	−249.92 ± 7.44	−253.03 ± 7.61	−217.70 ± 10.71	−205.37 ± 5.68
	8	**−405.12 ± 3.75**	−472.44 ± 5.72	−471.40 ± 7.42	−440.27 ± 11.95	**−409.87 ± 6.13**

all a values of k. For Hepar2 OSI and DOSI tie statistically for best when k is set equal to 2, 4, and 8.

Table 4 indicates that for Network A OSI and DOSI tie with exact inference when k is set to 2, 4, and 6, and OSI ties with exact inference when k is set to 8. For Network B both OSI and DOSI are outperformed by exact inference. For Network C OSI and DOSI tie with exact inference when k is set to 2, 4, and 6. For all other networks both OSI and DOSI outperform MBE and SLS.

4.4.2 *Comparison against modifications of DOSI*

For all networks DOSI performs either equivalently or better than DOSI-Comp and DOSI-Comm. For Network A all algorithms tie statistically when k is set to 2, 4, and 8. For Network B all algorithms tie statistically when k is set to 4 and 6, while DOSI and DOSI-Comp tie statistically when k is set to 2 and 8. For Network C all algorithms tie statistically for best when k is set equal to 2 and 4, while DOSI and DOSI-Comp tie statistically when k is set to 6 and 8. OSI outperforms all other approximate algorithms when k is set to 8. For Win95pts DOSI and DOSI-Comp tie statistically for best for all values of k. For Insurance DOSI and DOSI-Comp tie statistically for best when k is set to 2. For Hailfinder DOSI and DOSI-Comm tie statistically for best when k is set to 2 and 4, while DOSI and DOSI-Comp tie statistically for best when k is set to 6. For Hepar2 DOSI and DOSI-Comp tie statistically for best when k is set equal to 8.

4.5 Discussion

The paired t tests on the sum of the log likelihoods indicate that both OSI and DOSI performed better than the other approximate methods for nearly all values of k. These results show that our algorithms outperform the other methods for

Table 4 Comparison against MBE and SLS

Network	k	OSI	SLS	MBE	DOSI
Network A	2	**−13.64 ± 0.02**	−19.40 ± 2.58	**−13.64**	**−13.85 ± 0.58**
	4	**−27.31 ± 0.10**	−38.40 ± 4.12	**−27.27**	**−27.81 ± 0.70**
	6	**−40.92 ± 0.03**	−59.20 ± 3.69	**−40.91**	**−41.82 ± 2.51**
	8	**−54.56 ± 0.03**	−82.01 ± 8.47	**−54.55**	−55.27 ± 0.45
Network B	2	−17.12 ± 0.07	−24.65 ± 5.47	**−16.99**	−17.20 ± 0.16
	4	−34.62 ± 0.14	−48.90 ± 4.48	**−34.37**	−35.59 ± 1.45
	6	−51.87 ± 0.24	−72.73 ± 8.01	**−51.59**	−52.66 ± 0.64
	8	−69.26 ± 0.32	−93.71 ± 5.15	**−68.86**	−71.97 ± 2.19
Network C	2	**−16.05 ± 0.02**	−23.67 ± 4.68	**−16.03**	**−16.71 ± 0.96**
	4	**−32.09 ± 0.05**	−50.11 ± 5.89	**−32.07**	**−34.58 ± 4.24**
	6	**−48.18 ± 0.11**	−71.88 ± 6.51	**−48.10**	−50.74 ± 2.15
	8	−64.28 ± 0.19	−98.24 ± 8.31	**−64.14**	−67.08 ± 2.59
Win95pts	2	**−32.27 ± 5.59**	−3,529.16 ± 1,215.55	−2,992.69	**−40.60 ± 10.96**
	4	**−57.10 ± 3.85**	−8,469.61 ± 2,217.93	−5,991.64	**−125.07 ± 98.46**
	6	**−90.21 ± 13.53**	−11,412.51 ± 1,872.39	−8,996.11	**−350.87 ± 368.13**
	8	**−124.78 ± 20.97**	−14,708.08 ± 1,830.15	−12,006.28	−520.60 ± 517.70
Insurance	2	**−24.63 ± 2.29**	−1,004.74 ± 992.53	−39.76	**−26.77 ± 4.14**
	4	**−48.85 ± 3.04**	−2,097.83 ± 1,558.87	−83.85	**−54.05 ± 6.08**
	6	**−74.02 ± 9.14**	−3,778.13 ± 1,181.34	−125.14	**−83.40 ± 9.83**
	8	**−100.25 ± 7.25**	−4,797.66 ± 1,964.59	−170.11	**−102.37 ± 12.98**
Hailfinder	2	**−69.51 ± 2.19**	−2,121.23 ± 1,648.51	−90.44	−86.74 ± 8.72
	4	**−142.31 ± 3.85**	−6,175.14± 2,845.54	−186.99	−183.56 ± 16.64
	6	**−210.29 ± 6.16**	−9,255.36 ± 3,385.27	−274.01	−260.12 ± 20.88
	8	**−278.88 ± 9.34**	−10,032.40 ± 1,965.23	−368.84	−357.30 ± 30.19
Hepar2	2	**−66.54 ± 0.00**	−88.42 ± 9.07	−72.21	**−67.28 ± 1.44**
	4	**−134.19 ± 1.85**	−177.58 ± 9.46	−148.68	**−136.34 ± 2.64**
	6	**−199.84 ± 0.67**	−266.16 ± 9.42	−233.23	−205.37 ± 5.68
	8	**−405.12 ± 3.75**	−460.57 ± 25.72	−472.46	**−409.87 ± 6.13**

all networks containing more than 15 nodes. This indicates that both OSI and DOSI have an advantage when used to perform inference on more complex networks.

The results shown in Table 5 indicate that, for all networks, DOSI performs either equivalently or better than the modified versions of DOSI and for all networks containing more than 15 nodes DOSI outperformed either one or both of these alternative implementations. These results support the hypothesis that the increased performance obtained by OSI and DOSI is due to the representation of each swarm being based on the Markov blankets and the corresponding communication and competition that occurs between overlapping swarms. Recall that each variable X_i is conditionally independent of all other variables in the network given its Markov blanket. By defining each node's swarm to cover its Markov blanket, we ensure that the swarm learns the state assignments for all variables upon which that node may depend. Also, since multiple swarms learn the state assignments for a single variable, our approach ensures greater exploration of the search space. Through competition, we ensure that the best variable state assignments found by the swarms are used in the final k explanations.

The results in Table 3 and 4 also indicate that the OSI algorithm outperforms DOSI for several experiments. This result is reinforced by the convergence plots in Fig. 4, which show that OSI-Best often converges to a higher score than DOSI-Best. We believe that this is due to the communication delay that results from DOSI being truly distributed. Because the DOSI algorithm requires several iterations of the communication procedure for the swarms to reach consensus with respect to the k most probable state assignments for each variable, the fitness evaluations of a particle may be inaccurate prior to reaching consensus. If consensus has not been reached then some swarms may not have received updated state assignments from swarms outside of their Markov blankets, so the fitness evaluations performed in these swarms would be relying on outdated state assignments.

Table 5 Comparison against modifications of DOSI

Network	k	DOSI	DOSI-Comp	DOSI-Comm
Network A	2	**−13.15 ± 0.32**	**−13.25 ± 0.46**	**−13.14 ± 0.27**
	4	**−26.09 ± 0.39**	**−27.07 ± 2.35**	**−26.49 ± 0.51**
	6	**−39.29 ± 0.36**	−40.33 ± 1.44	−40.45 ± 1.40
	8	**−52.81 ± 1.30**	**−53.96 ± 1.67**	**−56.17 ± 4.35**
Network B	2	**−15.96 ± 0.65**	**−16.27 ± 0.67**	−17.41 ± 2.06
	4	**−33.60 ± 2.02**	**−33.58 ± 2.87**	**−33.18 ± 1.56**
	6	**−52.24 ± 2.26**	**−53.85 ± 5.13**	**−52.86 ± 4.00**
	8	**−68.62 ± 3.62**	**−69.36 ± 6.28**	−75.20 ± 3.56
Network C	2	**−17.33 ± 0.25**	**−17.23 ± 0.33**	**−17.83 ± 1.37**
	4	**−34.67 ± 0.62**	**−36.07 ± 3.76**	**−34.91 ± 0.85**
	6	**−52.71 ± 0.78**	**−53.62 ± 2.64**	−53.90 ± 1.37
	8	**69.88 ± 1.20**	**−71.20 ± 2.67**	−74.25 ± 2.53
Win95pts	2	**−65.64 ± 13.68**	**−211.43 ± 469.26**	−207.70 ± 471.30
	4	**−125.68 ± 13.49**	**−134.46 ± 17.50**	−4,067.57 ± 614.37
	6	**−564.42 ± 384.45**	**−1,027.76 ± 1440.54**	−6,289.36 ± 1197.70
	8	**−1,258.92 ± 367.16**	**−1,395.12 ± 1016.51**	−7,915.30 ± 1678.76
Insurance	2	**−36.20 ± 3.49**	**−37.13 ± 4.56**	−39.07 ± 3.75
	4	**−70.11 ± 5.83**	−82.21 ± 9.90	−91.77 ± 10.10
	6	**−105.18 ± 14.81**	−117.19 ± 14.62	−427.69 ± 371.47
	8	**−142.77 ± 15.25**	−163.98 ± 11.54	−722.26 ± 773.90
Hailfinder	2	**−83.50 ± 5.13**	−87.98 ± 6.71	**−85.27 ± 4.78**
	4	**−159.91 ± 8.63**	−242.77 ± 106.83	**−157.78 ± 9.44**
	6	**−274.13 ± 15.38**	**−277.02 ± 23.71**	−762.74 ± 662.30
	8	**−350.56 ± 28.82**	−379.61 ± 29.37	−1,111.45 ± 1,054.07
Hepar2	2	**−78.70 ± 0.32**	−79.19 ± 0.48	−80.94 ± 1.65
	4	**−157.60 ± 0.65**	−158.91 ± 1.64	−161.76 ± 1.94
	6	**−233.77 ± 1.67**	−236.04 ± 2.36	−252.35 ± 5.98
	8	**−232.18 ± 2.48**	**−234.58 ± 4.12**	−239.96 ± 5.49

The convergence plots presented in Fig. 4 show that, while OSI-Best often converges to a higher score than DOSI-Best, the average scores for the best particles are very close. This reinforces the results in Tables 3 and 4, illustrating that the difference in solution quality between the two algorithms is often insignificant. The convergence plots also show that the rates of convergence tend to be similar for both algorithms in terms of both average and best scores. This indicates that the communication mechanism used by DOSI does not greatly impact the number of iterations required for convergence.

The convergence plots presented in Fig. 4 also show that the average scores of the particles in the DMVPSO algorithm tend to be much lower than the average scores for particles in OSI and DOSI. This is consistent with the performance advantage that is seen when comparing OSI and DOSI to the other approximate methods in Tables 3 and 4.

While OSI and DOSI appear to outperform the other methods in terms of the log likelihoods of solutions found, these algorithms require many more fitness evaluations than the other approaches. Figure 5 indicates that, while the number of nodes in the network has little effect on the number of fitness evaluations required by the competing algorithms, the number of fitness evaluations required by the OSI and DOSI algorithms is higher for networks with a large number of nodes. This is because our algorithms create a separate swarm for each of the nodes in the network, causing the number of swarms and the number of fitness evaluations to increase with the number of nodes.

5 Partial abductive inference

In the following sections, we present our distributed and non-distributed approaches to solving the partial abductive inference problem. We also discuss several experiments comparing these approaches to existing partial abductive inference algorithms.

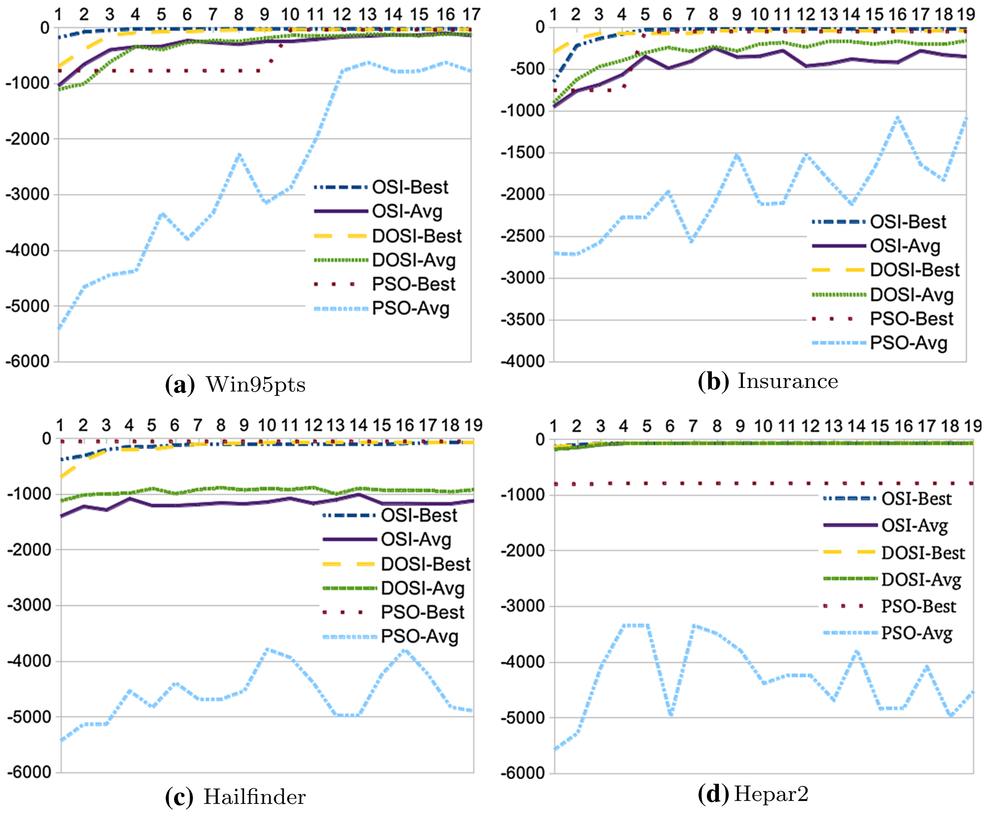


Fig. 4 Full abductive inference convergence plots

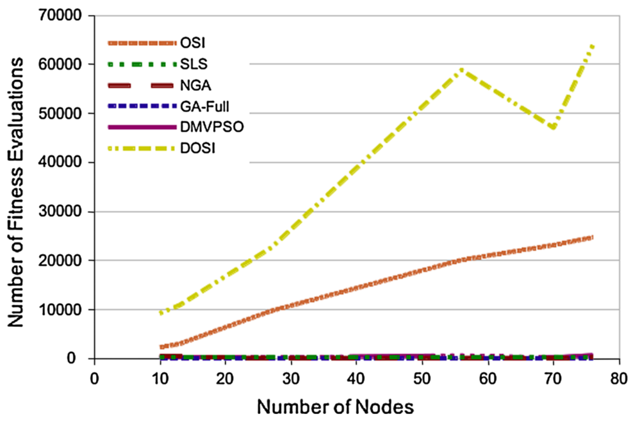


Fig. 5 Number of fitness evaluations

5.1 Partial abductive inference via OSI

Like the full abductive inference algorithm, our approach to partial abductive inference uses the DMVPSO algorithm (Veeramachaneni et al. 2007) and is based on the OSI methodologies described in Haberman and Sheppard (2012) and Pillai and Sheppard (2011). With this method, we assign a swarm to each node in the explanation set and a swarm to each node in the explanation node's Markov blanket.

Similar to full abductive inference, a global set of state assignments A is maintained across all swarms and is used for inter-swarm communication. These assignments are initially set to a state assignment obtained using a forward sampling process. Like full abductive inference, each particle's position is defined by a d-dimensional vector of discrete values and each value corresponds to the state of a variable in the swarm's Markov blanket. However, each particle's object parameters contain only state assignments for variables in the explanation set. Thus each particle represents a partial state assignment for the explanation set. The quality of each state assignment x_E for the explanation set is determined by the log likelihood ℓ of that assignment given the evidence x_O as shown below:

$$\ell(x_E) = \log(P(x_E|x_O)) = \log\left(\sum_{x_R} P(x_E, x_R|x_O)\right)$$

Since this calculation can be computationally expensive, we use the likelihood weighting algorithm to evaluate the quality of a state assignment (Neapolitan 2004).

Our algorithm begins by generating m samples using likelihood weighting given x_O as evidence. These samples are used to compute the approximate log likelihood (denoted ℓ_a):

$$\ell_a(x_E) = \log\left(\sum_{s\in M} \frac{W(x_E, s)}{w_s}\right)$$

where M is the set of samples generated by likelihood we of sample s, and $W(B_p, s)$ is defined as,

$$W(x_E, s) = \begin{cases} w_s & x_E \in s \\ 0 & \text{otherwise} \end{cases}$$

To evaluate the fitness of a given particle, a set of state assignments B_p must be constructed using the state of the particle p and the set of global state assignments A. The sum of the log likelihoods associated with the state assignments in B_p is used as the fitness for particle p. Given a partial state assignment x_p represented by some particle p in swarm s and the set of global state assignments for the explanation set $A = \{\alpha_1, \ldots, \alpha_k\}$ we can construct a new set of partial state assignments $B_p = \{\beta_1, \ldots, \beta_k\}$ by inserting x_p into each state assignment $\alpha_i \in A$ as follows:

$$\forall \beta_i \in B_p \quad \beta_i = \{x_p\} \cup \alpha_i \backslash mb_s,$$

where mb_s consists of the state assignments for explanation set variables in the Markov blanket of swarm s within α_i. We use B_p to calculate the fitness f of each particle,

$$f(p) = \sum_{\beta_i \in B_p} \ell_a(\beta_i)$$

Thus the fitness of particle p is the sum of the approximate log likelihoods of the assignments in A when the value assignments encoded in B_p are substituted into A.

After each iteration of the algorithm, swarms that share a node in the network compete to determine which state is assigned to the node in each assignment $\alpha_m \in A$. This competition is held between the partial state assignments found by the personal best particles in each swarm. The state that results in the highest approximate log likelihood is the one included in the assignment. This process is the same as for abductive inference and is shown in Algorithm 4.

Each time a state assignment is constructed, it is stored in A. At each iteration A is pruned so that the k most probable partial assignments are retained in A. Once the algorithm has terminated A is returned.

5.2 Partial abductive inference via DOSI

In this section we present our modification of the OSI algorithm for partial abductive inference that eliminates the need for a set of global state assignments to be maintained across all swarms. As with distributed full abductive inference, each swarm s maintains a set of the most probable personal state assignments found by that swarm, denoted as A_s. The most probable assignments learned by each swarm are communicated to the other swarms and inserted into the swarm's personal state assignment through a periodic communication mechanism. At each iteration of the algorithm the swarm associated with a given node n will hold a competition between all swarms that share n to determine the most probable state assignment for n.

Algorithm 7 Partial Abductive Inference via OSI

```
1: Generate $m$ samples using Likelihood weighting
2: Sample $A$ from the distributions
3: Initialize particles in each swarm

4: repeat
5:    for each swarm $s$ do
6:       for each particle, $p \in s$ do
7:          Construct $B_p$
8:          Add $B_p$ to $A$
9:          Calculate particle fitness $f(p)$ using samples
10:         if $f(p) >$ $p$'s personal best fitness then
11:            Update $p$'s personal best position and fitness
12:         end if
13:         if $f(p) >$ the global best fitness then
14:            Update global best position and fitness for $s$
15:         end if
16:         Update $p$'s velocity and position
17:      end for
18:   end for

19:   $A \leftarrow k$ most probable assignments in $A$

20:   for each state assignment $\alpha \in A$ do
21:      for each node $n$ in the explanation set do
22:         Let $S$ be all swarms where $n \in mb_s$
23:         Let $v_n$ be the state of $n$ in $\alpha$
24:         $v_n \leftarrow compete(S, n, \alpha)$
25:         Seed each swarm in $S$ with $v_n$
26:      end for
27:   end for
28: until termination criterion is met

29: return $A$
```

The fitness calculation and the calculation of $B_{s,p}$ for a given swarm s and particle p are similar to the calculation of B_p used in OSI. Given a partial state assignment x_p represented by some particle p in swarm s and the set of personal state assignments $A_s = \{\alpha_1, \ldots, \alpha_k\}$ we can construct a new set of state assignments $B_{s,p} = \{\beta_1, \ldots, \beta_k\}$ by inserting x_p into each state assignment $\alpha_i \in A_s$ as follows:

$$\forall \beta_i \in B_{s,p} \quad \beta_i = x_p \cup \alpha_i \backslash mb_s,$$

where mb_s consists of the state assignments for the Markov blanket of swarm s within α_i. We use $B_{s,p}$ to calculate the fitness f of each particle,

$$f(p) = \sum_{\beta_i \in B_{s,p}} \ell(\beta_i)$$

Thus the fitness of particle p in a swarm s is the sum of the approximate log likelihoods of the assignments in α_s when the state assignments encoded in p are substituted into α_s. Algorithm 8 shows the pseudocode for this algorithm.

Algorithm 8 Partial Abductive Inference via DOSI

```
1: Generate $m$ samples using Likelihood weighting
2: Create initial assignment $\alpha$ using forward sampling
3: Initialize particles in each swarm
4: for each swarm $s$ do
5:    Create an empty list of assignments $A_s$
6:    Add $\alpha$ to $A_s$
7: end for

8: repeat
9:    for each swarm $s$ do
10:       for each particle, $p \in s$ do
11:          Construct $B_p$
12:          Add $B_p$ to $A_s$
13:          Calculate particle fitness $f(p)$ using samples
14:          if $f(p) >$ $p$'s personal best fitness then
15:             Update $p$'s personal best position and fitness
16:          end if
17:          if $f(p) >$ the global best fitness then
18:             Update global best position and fitness for $s$
19:          end if
20:          Update $p$'s velocity and position
21:       end for
22:       $A_s \leftarrow k$ most probable assignments in $A_s$
23:    end for

24:    for each node $n \in X_E$ do
25:       Let $s$ be the swarm associated with $n$
26:       for each state assignment $\alpha \in A_s$ do
27:          Let $S$ be all swarms $x$ where $n \in mb_x$
28:          Let $v_n$ be the state of $n$ in $A$
29:          $v_n \leftarrow compete(S, n, \alpha)$
30:          Seed $s$ with $v_n$
31:       end for

32:       for $i = 0$ to $C$ do
33:          for each swarm $s_i$ do
34:             for each swarm $s_j \in mb_{s_i}$ do
35:                $ShareStates(s_i, s_j)$
36:             end for
37:          end for
38:       end for
39:    end for
40: until termination criterion is met

41: for each swarm $s$ do
42:    Add $A_s$ to $A$
43: end for
44: $A \leftarrow k$ most probable assignments in $A$
45: return $A$
```

The *ShareStates* method in this approach is identical to the *ShareStates* method described for full abductive inference via DOSI (Algorithm 6) except that only state assignments for nodes within the explanation set are shared. Similarly, the competition between nodes remains the same as the competition used for full abductive inference (Algorithm 4) except that swarms only compete over nodes within the explanation set. These modifications, combined with the use of likelihood weighting for fitness evaluation, are the primary differences between this approach and full abductive inference via DOSI.

5.3 Experimental design

We compare partial abductive inference via OSI and DOSI against four other partial abductive inference algorithms: MBE, DMVPSO, GA-Part, and SA. A cross-reference to the names of the algorithms tested is shown in Table 1. For these experiments, DMVPSO is a modification of the algorithm proposed in Pillai and Sheppard (2012) so that it can be used to perform partial abductive inference. As with OSI, our modified DMVPSO uses likelihood weighting to approximate the log likelihood for fitness evaluation. To compare these algorithms, we used the same networks as described in Fig. 3 and Table 2. Nodes were randomly selected for inclusion in the explanation set with a 25 % probability.

For our experiments we ran each algorithm on each of the networks. We evaluated all algorithms with k set to 2, 4, 6, and 8, respectively. For all of the algorithms, initial populations were generated using a forward sampling process. In every experiment, the number of particles in each swarm was set to 20. For the genetic algorithms, the population size was set to 20. We ran all algorithms until convergence. We averaged the log likelihoods of the solutions found by the various algorithms over 10 runs and compared the solutions using a paired t test to evaluate significance. For the t test the confidence interval was 95 %. We also measured the number of fitness evaluations performed by each of the algorithms to compare the computational complexity of each approach.

For networks A, B, and C the MBE algorithm is used to compute the exact solution for the partial abductive inference problem. For all other networks MBE is used to find an approximate solution with parameters m and i set to 2 and 3, respectively.

5.4 Results

In Table 6, we show the average sum of the log likelihoods for each algorithm and each value of k. Bold values indicate that the corresponding algorithm's performance is statistically significantly better than the other algorithms for the network given the corresponding value for k. Values for algorithms that tie statistically for best are also bolded.

For all networks other than Network A OSI and DOSI tie statistically with or outperform MBE. For Networks A, B, and C both OSI and DOSI find the optimal solution for all values of k with one exception. In Network A when k is set to 2 neither OSI or DOSI find the exact solution.

For Network A OSI, DOSI, and DMVPSO tie statistically when k is set to 2. Otherwise, OSI and DOSI have the best performance. For Network B OSI, DOSI, and DMVPSO tie statistically for best for all values of k. For Network C OSI, DMVPSO, and DOSI tie statistically when k is set to 4 while OSI and DOSI have the best performance otherwise.

Table 6 Comparison with other approaches

Network	k	OSI	GA-Part	DMVPSO	SA	MBE	DOSI
Network A	2	**−3.20 ± 0.00**	−7.70 ± 1.92	**−3.20 ± 0.00**	−7.70 ± 2.58	**−2.91**	**−3.20 ± 0.00**
	4	**−8.39 ± 0.00**	−14.59 ± 1.82	−8.92 ± 0.72	−11.86 ± 3.59	**−8.39**	**−8.39 ± 0.00**
	6	**−13.95 ± 0.00**	−21.89 ± 2.17	−14.62 ± 0.84	−21.24 ± 2.59	**−13.95**	**−13.95 ± 0.00**
	8	**−23.57 ± 0.00**	−24.95 ± 1.49	−23.57 ± 0.00	−25.01 ± 1.44	**−23.57**	**−23.57 ± 0.00**
Network B	2	**−2.87 ± 0.00**	−4.37 ± 0.65	**−2.87 ± 0.00**	−4.84 ± 0.68	**−2.87**	**−2.87 ± 0.00**
	4	**−7.55 ± 0.00**	−9.03 ± 0.94	**−7.55 ± 0.00**	−9.10 ± 1.30	**−7.55**	**−7.55 ± 0.00**
	6	**−12.40 ± 0.00**	−14.33± 0.54	**−12.40 ± 0.00**	−14.38 ± 0.76	**−12.40**	**−12.40 ± 0.00**
	8	**−18.22 ± 0.00**	−18.83 ± 0.60	**−18.22 ± 0.00**	−18.85 ± 0.43	**−18.22**	**−18.22 ± 0.00**
Network C	2	**−5.17 ± 0.00**	−7.26 ± 1.79	−5.17 ± 6.81	−8.01 ± 2.15	**−5.17**	**−5.17 ± 0.00**
	4	**−11.47 ± 0.00**	−15.77 ± 4.33	**−11.47 ± 0.00**	−18.51 ± 4.54	**−11.47**	**−11.47 ± 0.00**
	6	**−18.20 ± 0.00**	−24.40 ± 2.83	−18.41 ± 0.26	−37.80 ± 11.32	**−18.20**	**−18.20 ± 0.00**
	8	**−25.15 ± 0.00**	−33.29 ± 3.74	−26.32 ± 1.36	−58.56 ± 14.54	**−25.15**	**−25.15 ± 0.00**
Win95pts	2	**−5.29 ± 0.00**	−9.96 ± 2.98	−5.62 ± 0.45	−601.36 ± 310.82	−753.65	**−5.29 ± 0.00**
	4	**−12.45 ± 0.00**	−20.23 ± 2.47	−13.26 ± 0.97	−1,943.27 ± 381.27	−2,952.21	**−12.45 ± 0.00**
	6	**−20.48 ± 0.00**	−32.14 ± 3.82	−21.89 ± 1.58	−3,135.72 ± 468.31	−2,961.54	**−20.48 ± 0.00**
	8	**−28.69 ± 0.00**	−43.14 ± 5.16	−32.22 ± 3.32	−4,403.11 ± 886.28	−5,904.43	**−28.69 ± 0.00**
Insurance	2	**−6.47 ± 1.24**	−1,045.61 ± 380.91	−9.86 ± 2.19	−1,341.29 ± 310.79	−22.52	**−6.68 ± 1.22**
	4	**−13.05 ± 0.52**	−2,313.46 ± 811.42	−28.85 ± 17.83	−2,535.85 ± 380.35	−42.75	**−13.60 ± 1.16**
	6	**−20.29 ± 0.35**	−3,285.61 ± 864.35	−574.56 ± 760.88	−4,318.95 ± 311.18	−62.97	**−20.59 ± 0.92**
	8	**−27.88 ± 0.49**	−3,963.88 ± 855.26	−298.07 ± 346.66	−5,659.35 ± 516.60	−87.79	**−27.84 ± 0.53**
Hailfinder	2	**−14.49 ± 0.52**	−607.45 ± 574.35	**−14.65 ± 0.76**	−972.83 ± 353.91	−1,484.99	**−14.95 ± 1.03**
	4	**−31.32 ± 2.39**	−929.10 ± 821.42	**−31.97 ± 4.24**	−2,464.16 ± 352.25	−2,969.98	**−31.31 ± 1.70**
	6	**−49.79 ± 6.23**	−1456.29 ± 417.35	**−47.66 ± 4.22**	−4,101.11 ± 381.97	−4,454.96	**−49.80 ± 3.30**
	8	**−72.93 ± 4.33**	−1,694.31 ± 968.11	**−71.91 ± 3.14**	−5,441.21 ± 354.90	−5,939.95	**−72.03 ± 4.24**
Hepar2	2	**−8.12 ± 0.42**	−20.68 ± 1.57	−9.47 ± 0.93	−609.28 ± 308.44	−25.03	**−8.48 ± 0.58**
	4	**−17.44 ± 0.76**	−39.46 ± 4.09	−19.60 ± 2.01	−157.19 ± 11.44	−49.81	−19.80 ± 0.85
	6	**−27.49 ± 1.03**	−57.81 ± 5.53	−32.77 ± 2.07	−2,112.15 ± 1,336.86	−73.46	**−28.15 ± 1.41**
	8	**−36.87 ± 1.34**	−80.85 ± 7.33	−42.97 ± 2.18	−3,754.73 ± 1,094.14	−100.14	**−38.42 ± 1.49**

For Win95pts and Insurance both DOSI and OSI have the best performance for all values of k. For Hailfinder OSI, DMVPSO, and DOSI tie statistically for best for all values of k. For Hepar2 OSI and DOSI tie statistically when k is set to 2, 6, and 8. Otherwise, OSI has the best performance.

In Fig. 6 we compare the number of fitness evaluations required by each of the algorithms with respect to the number of nodes in the network. We find a very similar performance trend to Fig. 5 for similar reasons.

In Fig. 7 we present the convergence plots for OSI, DOSI, and DMVPSO. In this figure, OSI-Best and DOSI-Best denote the average score for the best particle in each swarm, while OSI-Avg and DOSI-Avg denote the average score for all particles in each swarm. PSO-Avg denotes the average score for all particles in the swarm, while PSO-Best denotes the score of the best particle in the swarm. These plots were obtained by running each algorithm with k set to 2 on the four largest networks.

5.5 Discussion

The paired t tests on the sum of the probabilities indicate that OSI and DOSI performed either equal to or better than the other methods for most values of k on nearly all networks. While the OSI approach does not appear to have an advantage over DMVPSO when used on Hailfinder and Network B, we can see that it outperformed the other methods on all other networks for most values of k. We believe that our algorithms were not able to outperform DMVPSO approaches when used on Hailfinder because this network has many more parameters than the other networks and the solution space for this network contains many local optima.

Consistent with our lesion studies on full abductive inference, we believe that the increased performance of OSI and DOSI is due to each swarm being associated with the Markov blanket of a single node and the result-

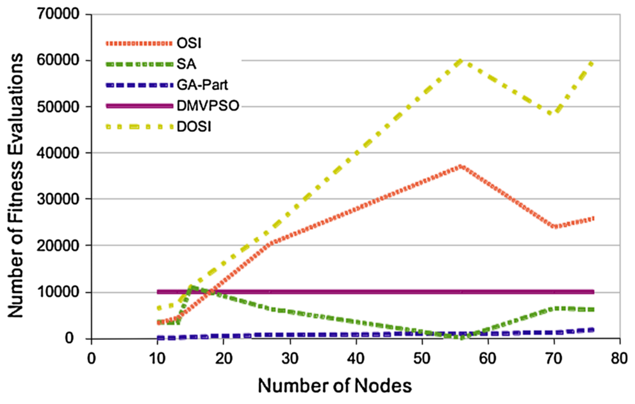


Fig. 6 Number of fitness evaluations

ing communication and competition between overlapping swarms. DOSI was found to outperform all approaches not based on OSI and DOSI tied statistically with OSI for most networks and values for k. This result indicates that DOSI provides an effective distributed alternative to traditional OSI for the problem of partial abductive inference.

Like the results for full abductive inference, the convergence plots presented in Fig. 7 show that, while OSI-Best often converges to a higher score than DOSI-Best, the average scores for the best particles are usually similar. In fact, the average scores of the best particles in DOSI are higher than those in OSI for the Hailfinder network. This reinforces the results in Table 6, illustrating that the difference between the two algorithms is often insignificant. However, higher average scores of the best particles in an algorithm may not necessarily indicate a higher log likelihood for the k best explanations found by the algorithm as a whole.

The convergence plots for partial abductive inference also show a much lower average score for particles in DMVPSO when compared to the average scores of particles in OSI and DOSI. This result, when combined with the results presented in Table 6, provides strong evidence for the performance advantage of OSI and DOSI when compared to other approximate methods.

While OSI and DOSI appear to outperform the other methods in terms of the log likelihood of solutions found, these methods require many more fitness evaluations than the other approaches. However, the correlation between the number of nodes and number of fitness evaluations is not as strong in the case of partial abductive inference since the number of required swarms is tied to the size of the explanation set rather than the number of nodes in the network.

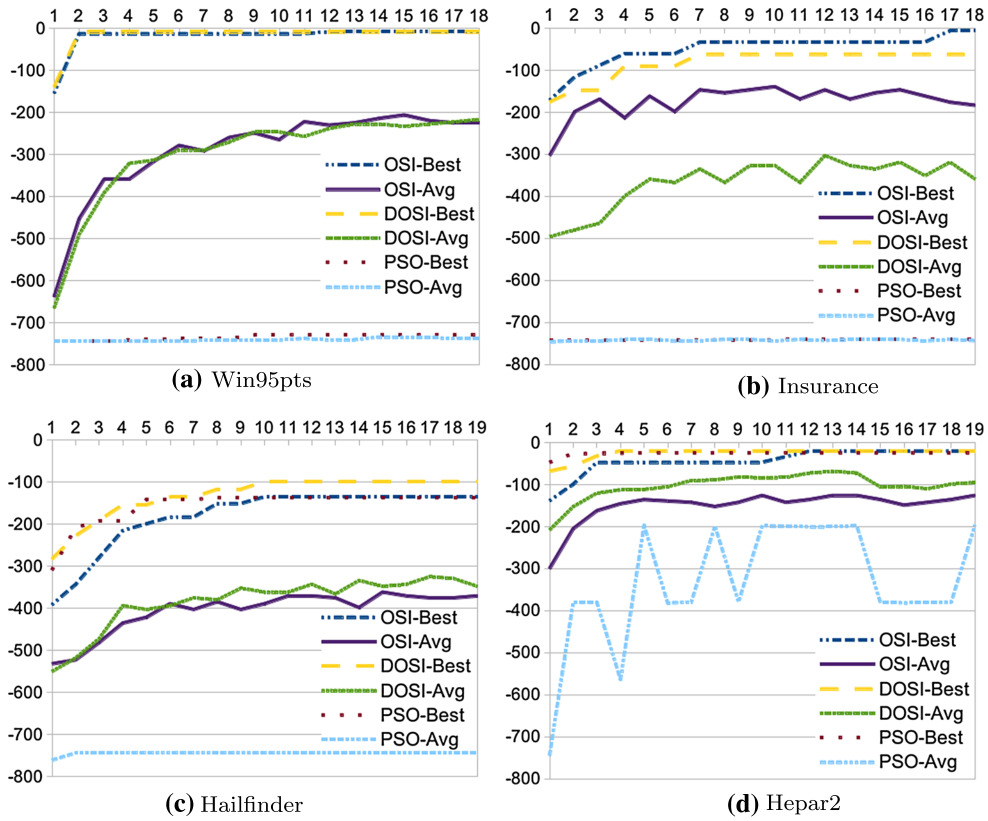


Fig. 7 Partial abductive inference convergence plots

6 Conclusions

We have presented several swarm-based algorithms for both full and partial abductive inference in Bayesian networks. In these algorithms a swarm is associated with each relevant node in the network and that swarm learns the state assignments for its node's Markov blanket. These algorithms are based on OSI and we have presented both distributed and non-distributed versions of each algorithm. We compared our algorithms to several other approaches to the partial and full abductive inference problems. Our results indicate that, while our approach is more computationally expensive than competing methods, both OSI and DOSI significantly outperform the other approaches on most networks studied in terms of the log likelihood of solutions found. Also, for nearly all of the experiments, DOSI outperformed all approaches that were not based on OSI and, in many cases, tied statistically with OSI.

For future work we plan to explore alternative representations and competition strategies to reduce the computational complexity of our algorithms. For example, we could vary the number of particles for each swarm based on the complexity of the swarm's Markov blankets. In this way we could reduce the computational complexity of our algorithm by assigning smaller swarms to nodes whose Markov blankets have fewer parameters. We are also developing the theory of the general OSI and DOSI methodologies by showing the convergence of OSI. To do so, we plan on drawing from the approaches of Patterson et al. (2010) and Boyd et al. (2011) on the distributed consensus problem to analyze the relationship between OSI convergence rate and the structure of the sub-swarms. Finally, we will also analyze the rate of consensus for DOSI when applied to problems with both discrete and continuous search spaces by drawing on the theory presented by Olfati-Saber et al. (2007).

Acknowledgments The authors would like to thank the members of the Numerical Intelligent Systems Laboratory at Montana State University for their comments and advise during the development of this work. We would also like to thank Dr. Brian Haberman at the Johns Hopkins University Applied Physics Laboratory and Karthik Ganesan Pillai in the Data Mining Laboratory at MSU for their ideas during the formative stages of this research.

References

Belding T (1995) The distributed genetic algorithm revisited. In: Proceedings of the International Conference on genetic algorithms, pp 114–121

van den Bergh F, Engelbrecht A (2000) Cooperative learning in neural networks using particle swarm optimizers. South African Computer J 26:90–94

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 3(1):1–122

de Campos L, Gamez J, Moral S (1999) Partial abductive inference in Bayesian belief networks using a genetic algorithm. Pattern Recogn Lett 20:1211–1217

de Campos L, Gamez J, Moral S (2001) Partial abductive inference in Bayesian belief networks by simulated annealing. Int J Approx Reason 27(3):263–283

Dagum P, Luby M (1993) Approximating probabilistic inference in Bayesian belief networks is NP-hard. Artif Intell 60(1):141–153

Dawid A (1992) Applications of a general propagation algorithm for probabilistic expert systems. Stat Comput 2(1):25–36

Dechter R (1996) Bucket elimination: a unifying framework for probabilistic inference. In: Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., pp 211–219

Dechter R, Irina R (2003) Mini-buckets: a general scheme for bounded inference. J ACM 50(2):107–153

Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. Comput Intell Magazine IEEE 1(4):28–39

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Micro machine and human science, 1995. MHS'95., Proceedings of the Sixth International Symposium on, IEEE, pp 39–43

Elvira Consortium (2002) Elvira: an environment for creating and using probabilistic graphical models. In: Proceedings of the first European workshop on probabilistic graphical models, pp 222–230

Fortier N, Sheppard JW, Pillai KG (2012) DOSI: training artificial neural networks using overlapping swarm intelligence with local credit assignment. In: Soft computing and intelligent systems (SCIS) and 13th International Symposium on advanced intelligent systems (ISIS), 2012 Joint 6th International Conference on, IEEE, pp 1420–1425

Fortier N, Sheppard JW, Pillai KG (2013) Bayesian abductive inference using overlapping swarm intelligence. In: 2013 IEEE Symposium on swarm intelligence (SIS 2013), pp 263–270

Gamez J (1998) Inferencia abductiva en redes causales. In: Thesis. Departamento de Ciencias de la Computacin e I.A. Escuela Tcnica Superior de Ingeniera Informtica

Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845

Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Computers 29(1):17–35

Gelsema E (1995) Abductive reasoning in Bayesian belief networks using a genetic algorithm. Pattern Recogn Lett 16:865–871

Haberman BK, Sheppard JW (2012) Overlapping particle swarms for energy-efficient routing in sensor networks. Wireless Netw 18(4):351–363

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (abc) algorithm. Appl Soft Comput 8(1):687–697

Kask K, Dechter R (1999) Stochastic local search for Bayesian networks. In: Workshop on AI and statistics. Morgan Kaufman Publishers, pp 113–122

Kennedy J, Eberhart R (1997) A discrete binary version of the particle swarm algorithm. In: Systems, man, and cybernetics, 1997. Computational cybernetics and simulation., 1997 IEEE International Conference on, vol 5, pp 4104–4108

Koller D, Friedman N (2009) Probabilistic graphical models—principles and techniques. MIT Press, New York

Langley P (1988) Machine learning as an experimental science. Mach Learn 3(1):5–8

Neapolitan RE (2004) Learning bayesian networks. Pearson Prentice Hall, Upper Saddle River

Nilsson D (1998) An efficient algorithm for finding the m most probable configurations in probabilistic expert systems. Stat Comput 8(2):159–173

Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multi-agent systems. Proc IEEE 95(1):215–233

Patterson S, Bamieh B, El Abbadi A (2010) Convergence rates of distributed average consensus with stochastic link failures. Autom Control IEEE Trans 55(4):880–892

Pillai KG, Sheppard JW (2011) Overlapping swarm intelligence for training artificial neural networks. In: Proceedings of the IEEE swarm intelligence symposium, pp 1–8

Pillai KG, Sheppard JW (2012) Abductive inference in Bayesian belief networks using swarm intelligence. In: Soft computing and intelligent systems (SCIS) and 13th International Symposium on advanced intelligent systems (ISIS), 2012 Joint 6th International Conference on, pp 375–380

Rabbat M, Nowak R (2004) Distributed optimization in sensor networks. In: Proceedings of the 3rd international symposium on Information processing in sensor networks, ACM, pp 20–27

Rojas-Guzman C, Kramer MA (1993) Galgo: a genetic algorithm decision support tool for complex uncertain systems modeled with bayesian belief networks. In: Proceedings of the Ninth international conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., pp 368–375

Scutari M (2012) Bayesian network repository. http://www.bnlearn.com/bnrepository/

Shimony S (1994) Finding MAPs for belief networks is NP-hard. Artif Intell 68:399–410

Sriwachirawat N, Auwatanamongkol S (2006) On approximating k-MPE of Bayesian networks using genetic algorithm. In: Cybernetics and intelligent systems, pp 1–6

Tanese R, Co-Chairman-Holland J, Co-Chairman-Stout Q (1989) Distributed genetic algorithms for function optimization. In: Proceedings of the International Conference on genetic algorithms, University of Michigan, pp 434–439

Veeramachaneni K, Osadciw L, Kamath G (2007) Probabilistically driven particle swarms for optimization of multi-valued discrete problems: design and analysis. In: Proceedings of the IEEE swarm intelligence symposium, pp 141–149

Whitley D, Starkweather T (1990) Genitor ii: a distributed genetic algorithm. J Exp Theor Artif Intell 2(3):189–214

Whitley D, Rana S, Heckendorn R (1999) The island model genetic algorithm: on separability, population size and convergence. J Comput Inf Technol 7:33–48

Yang XS, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (2013) Swarm intelligence and bio-inspired computation: theory and applications. Newnes, vol. 1, pp 13–20

Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications. Springer, pp 169–178