University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Michał Matuszak
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University

Bayesian Networks in Adaptation and Optimization of Behavioral Patterns

PhD dissertation

Supervisor
dr hab. Jacek Miękisz
Institute of Applied Mathematics and Mechanics
University of Warsaw

December 2012

Dedicated to the memory of

Prof. Tomasz Schreiber

Tomasz Schreiber (1975 – 2010) inspired my leap into the field of informatics and his constant enthusiasm and encouragement kept me going. He was a rising star of Polish mathematics. During my two and a half years of working with him, Professor Schreiber was always positive and optimistic about mathematics and life in general. I feel truly lucky to have had the opportunity to know Professor Schreiber and to work with him. I am very grateful for his concern and support. I could always count on his help and advice. He died on the first of December, 2010. I will always remember him smiling.

Abstract

In this thesis, we present several new methods and algorithmic results related to probabilistic graphical models. In the first part, we present a short introduction to graphical models in the context of the thesis results. Our results are summarized and possible further research are pointed out in the last chapter. Finally, we include published papers.

One of the most important result was developed for the strategy optimization in Bayesian influence diagrams. It is a well–known NP–complete problem. The proposed stochastic algorithm generates optimal decision strategies by an iterative self–annealing reinforced search procedure, gradually acquiring new information while driven by information already acquired. At the basis of the method lies the Chen–style stochastic optimization which was originally proposed for travelling salesman problems (TSP). The algorithm, after a substantial extension, is applied to the NP–hard problem of learning Bayesian network structure. Another application of the algorithm is in the NP–hard ramified optimal transport problem.

In Gaussian–network set up, we develop an algorithm for determining optimal transition paths between given configurations of systems consisting of many objects. The method is applied to a system controlling the motion and redeployment between unit's formations and to a realistic transformation between two sequences of character animations in a virtual environment.

Using the framework of polygonal Markov fields, we introduce an image segmentation algorithm. Our algorithm is based on the Markovian optimization dynamics combining the simulated annealing ideas with those of the Chen–style stochastic optimization – in which successive segmentation updates are carried out simultaneously with the adaptive optimization of the local activity functions.

Keywords: Bayesian Networks, Influence Diagrams, Polygonal Markov Fields, Gaussian Networks, Chen Adaptive Optimisation, Optimal Decision Strategies, Transition Path, Structure Learning, Optimal Transport Path, Image Segmentation.

ACM Computing Classification: I.2.6, I.4.6, G.3.

Streszczenie

W pracy przedstawiamy nowe metody dla probabilistycznych modeli graficznych. Pierwsza część pracy zawiera krótkie wprowadzenie do modeli graficznych w kontekście osiągniętych wyników. W ostatnim rozdziale zawarliśmy krótkie podsumowanie wraz z opisem dalszych kierunków badań. Nieodłącznym elementem rozprawy są załączone publikacje.

Jednym z kluczowych wyników zawartych w niniejszej rozprawie jest algorytm optymalizacji strategii decyzyjnych w bayesowskich diagramach wpływów. Jest to problem NP–zupełny. Zaproponowany stochastyczny algorytm generuje optymalne strategie decyzyjne wykorzystując metodę przeszukiwania wzmocnioną iteracyjnym samo-wyżarzaniem, która stopniowo pozyskuje nowe informacje. U podstaw stworzonego algorytmu leży zaproponowana przez Chena metoda stochastycznej optymalizacji dla problemu komiwojażera (TSP). Rozszerzona wersja algorytmu została z powodzeniem zastosowana do NP–trudnego problemu uczenia struktury sieci bayesowskiej, jak i do problemu ramifikacji transportu na płaszczyźnie, który jest również NP–trudny.

Wykorzystując sieci gaussowskie stworzyliśmy algorytm wyznaczający optymalne warunkowe trajektorie przejścia pomiędzy zadanymi konfiguracjami dla systemów wieloobiektowych. Opracowana metoda została wykorzystana do kontroli ruchu oraz przegrupowania formacji jednostek na płaszczyźnie. Kolejnym zastosowaniem opracowanego algorytmu jest symulacja realistycznych przejść pomiędzy animacjami postaci w wirtualnym środowisku.

W środowisku wielokątnych pól Markowa zaproponowalismy nowy algorytm segmentacji, który wykorzystuje optymalizację dynamiki markowowskiej łacząc ideę samowyżarzania ze stochastyczną optymalizacją Chenowską, w której następujące po sobie aktualizacje segmentacji są przeprowadzane równocześnie z adaptacyjną optymalizacją lokalnej funkcji aktywności.

Słowa kluczowe: sieci bayesowskie, diagramy wpływów, wielokątne pola Markowa, sieci gaussowskie, adaptacyjna optymalizacja Chena, optymalne strategie decyzyjne, ścieżki przejścia, uczenie struktury, optymalne ścieżki transportowe, segmentacja obrazów.

Klasyfikacja tematyczna ACM: I.2.6, I.4.6, G.3.

Acknowledgements

I want to thank my supervisor, dr hab. Jacek Miękisz, for his encouragement and support without which this thesis would not be possible. I appreciate his patience and support for some challenging tasks.

I would also thank my first supervisor, dr hab. Tomasz Schreiber (1975 – 2010), for the unparalleled support. He encouraged me to study Probabilistic Graphical Models and provided many valuable ideas.

I would like to thank my family for their love and support during all these years it took me to complete this thesis. I would also like to thank my friends who have helped make my graduate years so enjoyable.

I acknowledge, with gratitude, the generous financial support which helped me to write this thesis. First, I would like to thank the *National PhD Programme in Mathematical Sciences (Środowiskowe Studia Doktorancki z Nauk Matematycznych)* organized by leading Polish universities. The programme is supported by the European Social Fund as a part of the Sub–measure 4.1.1.

I appreciate the financial support from the National Science Centre (NCN) grant PRELUDIUM 2011/01/N/ST6/00573 (2011 – 2014) that funded parts of the research presented as well as the future one. This work was also partially supported by the Ministry of Science and Higher Education grant 3852/B/H03/2008/34 N N201 385234 (2008 – 2010).

I would like to acknowledge the support from the Nicolaus Copernicus University grant 1086–M (2012) and the grant 187–M (2012). I gratefully acknowledge the access to the PL–Grid and PLGrid Plus Project that is co–funded by the European Regional Development Fund as a part of the Innovative Economy program. Last, but no least, I acknowledge the support from the European Social Fund, Government of Poland and Kuyavian-Pomeranian Voivodeship as a part of Integrated Operational Program for Regional Development, Sub–measure 8.2.2 (*Step in the future program*, edition: 2009, 2010 and 2011).

Contents

Chapter 1

Introduction

1.1 Motivation

Many classical probabilistic systems such as the Kalman filter, the Ising model, hidden Markov models, and mixture models can be viewed as special cases of the probabilistic graphical model formalism. Due to this, specialized techniques developed in one field can be transferred to other fields, improved, and therefore used more widely.

In this dissertation, we have investigated the framework of probabilistic graphical models (PGM). The prior objective was to solve the miscellaneous and interesting NP–hard problems within that framework. We did not limit ourselves to one narrow subclass of the PGM, but we have tried to investigate a broad variety of different types of models. The second objective was to show possible applications and a strong performance of the Chen–style stochastic optimization, which is used in nearly all of our algorithms. All of the developed methods have been implemented and tested in computer simulations.

Influence diagrams (ID) are a powerful tool for reasoning under uncertainty. It is an intuitive framework in which incorporated knowledge of an expert can be combined with a computer generated knowledge. Medical diagnosis is an area where influence diagrams are of widespread use. They have been applied to prenatal testing [60] or plans of radiation therapy for prostate cancer [56]. Solving ID has been found useful in Vista system at NASA Mission Control Centre [31], Pathfinder system [82], and as an effective tool to value real options [22]. They found also application in computer vision [7], multiagent systems, and game theory [42]. Motivated by a wide range of possible applications, we focused our attention on developing stochastic algorithms for solving ID. As we will later see, solutions of the ID can be also applied to the Bayesian network structure learning (see Sec. 2.3.5) or to an optimal transport problem [1, 88].

Solutions of the problem of determining optimal transition paths between given configurations of systems consisting of many objects can be applied to two general problems: (1) controlling motion and redeployment between unit's formations [4, 5], (2) realistic transformation between two sequences of character animations in a virtual environment [8, 70]. The first problem was induced by observation of coordination in biological systems [66]. Movement in formation allows for a more effective use of limited resources, like radars or visual perception. Fast redeployment has a wide military application, such as reduction of the number of casualties in a hostile fire, less vulnerable targets or evacuation from an exposed area [85]. It can be also applied to spacecrafts in a deep space or in the Earth orbit [71, 72]. In civil environment, it can be used to control parking systems to smoothly relocate cars in a parking lot or in the entertainment industry in real-time computer games. The second one (2) plays an important role in computer graphics and allows for the production of enjoyable computer games, credible CG movies or medical visualizations.

One of the fundamental problem in computer vision is *image segmentation*, the partitioning of an image into distinct (non–overlapping) segments which are homogeneous with respect to some characteristic [68]. For example, in an image of a street we might like to distinguish people, cars, a road, and other objects. Image segmentation can be applied to medical imaging [65], locating pathologies [89] or to determine shapes and sizes of organs [48, 86]. Neuroscientists can apply these techniques to check whether synapses exist at points of contacts, which helps to determine connections between neurons [34]. Hundreds of image segmentation algorithms have been developed, we refer the reader to [61] for a detailed discussion on such techniques. Still there is no single method which achieves satisfactory results for all images. We will mention only the most relevant classes: the morphological watershed [6], Markov random fields [87], and intermediate level methods which focus on the partition of the image that is the outcome of a segmentation [25, 57, 59]. The latter will be the main competitor of our algorithm.

1.2 Overview

Chapter 2 provides a brief introduction to the tools being used. In Section 2.1, we present the Chen's self–annealing algorithm whose extended versions are widely used in the dissertation. Next, some background information about: Bayesian networks, Influence diagrams, Gaussian Bayesian networks, and Polygonal Markov fields is provided. In Chapter 3, we conclude the thesis and discuss directions of a future research.

This PhD thesis is based on five publications which are included in appendices:

- Paper **A**: Introduces a stochastic algorithm for solving general Bayesian influence diagrams [53].
- Paper **B**: Proposes an algorithm for determining optimal transition paths

[51]. The successful application of the method to two problems has been also presented.

- Paper **C**: Presents a new image segmentation algorithm [55] within the framework of polygonal Markov fields.
- Paper **D**: Provides an extension of [53] for solving ramified optimal transport problem [50].
- Paper **E**: Introduces a stochastic algorithm that translates the structure learning problem into the strategy optimization problem [52], for which an extension of [53] is applied.

Chapter 2

Preliminaries

This chapter provides the background material about Bayesian networks (BN) [63], Influence diagrams (ID), Gaussian networks, and finally Markov networks. First of all, we present the Chen's self–annealing stochastic optimization algorithm, highly effective and yet not very popular in literature.

2.1 Chen's Algorithm

The travelling salesman problem (TSP) is a classical example of an NP–hard[1] optimization problem [18]. For a given set of cities and a cost function which assigns a cost to every pair of them, the task is to find the shortest closed route that visits each city exactly once. Due to the simplicity of the statement of the problem, the TSP is one of the most intensively studied problems in optimization; however, yet no effective solution for the general case has been presented.

The optimization by a *simulated annealing* (SA) [38] can by successfully applied to the TSP problem. The state (configuration) can by defined as a permutation of cities to be visited. The optimization is based on the evolution of the configuration, which is carried out by the Monte–Carlo simulations. In each step, a new configuration is generated from the previous one and is accepted with the probability depending on the difference between total route lengths and on a parameter T (temperature). During the optimization process, the temperature T is decreased according to an annealing schedule. The performance of the optimization highly depends on the annealing schedule and the initial configuration.

Another optimization technique, which can be used for TSP problem, is the *genetic algorithm* (GA) [30]. The algorithm is based on the natural process of

[1] **NP** (non–deterministic polynomial time) is a set of decision problems where the answer *yes* can be given in polynomial time by a non–deterministic Turing machine. Equivalently, it is the class of decision problems, where we can test the correctness of a solution in a polynomial time. **NP–complete** means being in NP and every other problem in NP is reducible to it in a polynomial time. **NP–hard** (non-deterministic polynomial-time hard) – a problem L is NP–hard iff there is an NP–complete problem that is polynomial time Turing-reducible to L. Note that NP–hard problems do not have to be in NP.

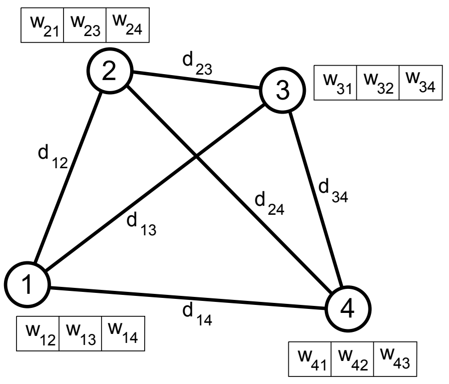


Figure 2.1: Initialization of the Chen's algorithm.

evolution and uses a 'survival of the fittest' technique, where the best specimens (solutions) survive and are varied until we get a good result. This shows the important role of keeping a large population of species in the optimization procedure. The GA algorithm has proved to be effective for a small–size TSP [75].

In [11], Chen proposed a simple stochastic optimization algorithm for the TSP problem, which unites advantages of the simulated annealing and genetic algorithms. In the Chen's algorithm, the cities are treated as *neurons* and the paths between them as *synapses*. The algorithm has been also successfully applied to the spin glass problem [10]. Further discussion of the Chen's algorithm can be found in [64].

Initially, each city is assigned a table of synaptic weights for connections to all remaining cities. We associate the synaptic weight $w_{i,j}$ with the path $i \to j$ that connects city $i \in \{1, \dots, N\}$ with city $j \in \{1, \dots, N\}$. Let us denote the distance from city i to j as $d(i,j)$. The initial values of weights $w_{i,j}$ depend on distances $d(i,j)$ according to the following equation:

$$w_{i,j} = e^{-d(i,j)/T}, \tag{2.1}$$

where T can be identified as the initial strength of the synapses.

These weights help to generate probabilistically new configurations. Each time, the choice of the next city to visit is made by a sample among cities not yet visited, with probabilities proportional to the corresponding connection weights i.e. the probability of selecting a route to city j from city i is given by

$$P(i \to j) = \frac{w_{i,j}}{\sum_{k \in \{1,\dots,N\} \setminus [V \cup \{i\}]} w_{i,k}}, \tag{2.2}$$

where V denotes a set of already visited cities. This way, all cities get visited and eventually we get back to the starting point.

Next, the generated tour is compared with the one obtained in the previous iteration. Let us denote the current tour as $\{i_1, i_2, \dots, i_N\}$ and the previous one as $\{j_1, j_2, \ldots, j_N\}$. The total tour lengths are labelled as d and d', respectively. Depending on whether the new cycle is longer or shorter than the previous one, the connection weights between neighbouring cities in both tours are correspondingly reinforced or faded

$$w^{new}_{i_k,i_{k+1}} = w^{old}_{i_k,i_{k+1}} e^{-(d-d')/T}, k = 1, \ldots, N \tag{2.3}$$
$$w^{new}_{j_k,j_{k+1}} = w^{old}_{j_k,j_{k+1}} e^{-(d'-d)/T}, k = 1, \ldots, N, \tag{2.4}$$

where $i_{N+1} = i_1$ and similarly $j_{N+1} = j_1$.

The connection weights of the cities from the intersection of the current and previous tour do not change during the course of the current iteration, because the weights are reinforced and faded at once. We modify only the weights of the cities from the symmetric difference (the union without the intersection) of the current and previous tour. Supposing, the current total tour length is smaller than the previous one, then the weights of the cities from the current tour, not occurring on the previous tour, are increased and the weights of the cities from the previous tour, not occurring on the current tour, are decreased. A similar situation takes place, if the previous total tour length is smaller than the current one.

The learning algorithm maintains probability weights that are associated with each city in the configuration. This allows many configurations to be kept implicitly, yet learning improves over time. The algorithm stops when the weights converge to the optimal tour under suitable reinforcement/fading protocols. It occurs because during the learning process some weights decrease, while others increase.

2.2 Graphical Models

Probabilistic graphical models received their name from the fact that their joint probability distributions can be easily described in graphical terms, where the vertices (nodes) of a graph represent variables over which a joint probability distribution is defined and the existence or absence of links represents dependence or independence between the variables. Their important advantage is a mechanism for describing a complex distribution in a compact way. It is possible due to the conditional independence properties of the model which can be determined by an inspection of the graph. The graphical structure of the model provides a simple way to visualize the structure and can be evaluated by a human expert. Complex computations, like inference in graphical models, can be conducted much faster than a direct manipulation of the joint distribution [41].

Graphical models can be divided into three general classes: those based on the Directed Acyclic Graphs (DAG), those based on undirected graphs, and mixed ones which use both DAG and undirected graphs. We will focus our attention on the first two cases; thus, we will not deal with the mixed ones. In the Figure 2.2, a DAG (on the left) and an undirected graph (on the right) are presented.

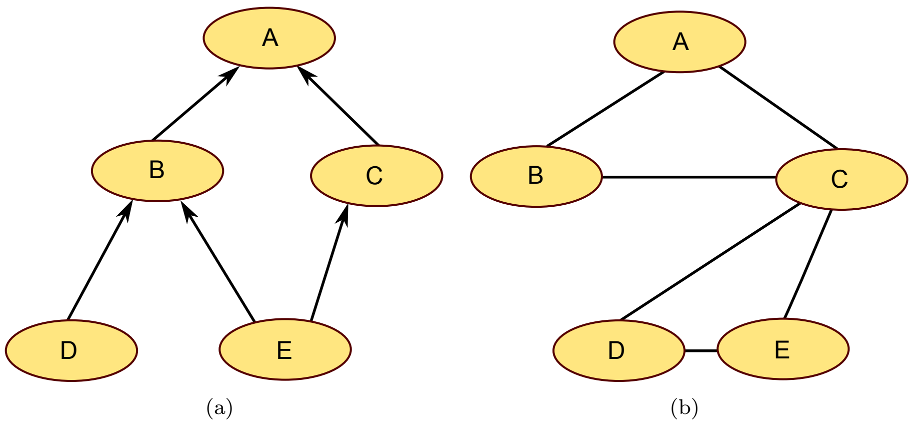


Figure 2.2: (a) An example of a directed acyclic graph (DAG); (b) An example of an undirected graph.

In this chapter, the notation of nodes and variables will be used interchangeably for models which consist of the chance variables only (like Bayesian, Gaussian or Markov networks). In influence diagrams (ID), which are introduced in Section 2.4, it is convenient to distinguish between variables and nodes because a node does not need to possess a variable. For example, decision and utility nodes in ID do not represent variables (see Section 2.4).

In Sections 2.3, 2.4, 2.5, probabilistic graphical models built on DAG are described. Finally, Section 2.6 describes models on undirected graphs.

2.3 Bayesian Networks

2.3.1 Introduction

Bayesian networks (BN), also known as belief networks, were introduced by Pearl [62] as probabilistic graphical models to represent an uncertain knowledge and to make decisions on its basis. BN provide a systematic way to represent (conditional) dependence relationships among random variables. They can be defined by an expert, encoded graphically, reasoned about, and can be used to our computational advantage. BN provide an effective representation and computation of the joint probability distribution over a set of random variables.

Definition 2.3.1 (Bayesian network structure) *A* ***Bayesian network structure*** *is a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$ whose nodes are associated with random variables i.e. for each node $v \in V$, a random variable X_v is linked which can take values from a given set of attributes $\mathcal{A}_v$, and a set of edges E describes*

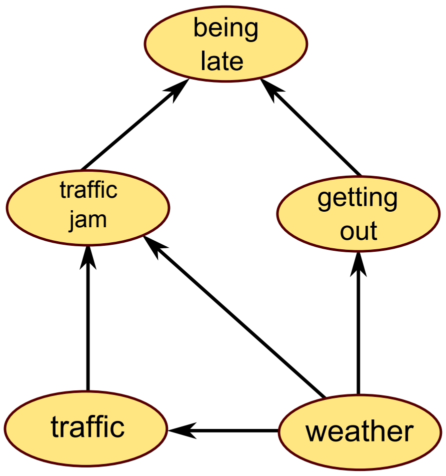


Figure 2.3: Example of a Bayesian network.

a direct causal relationship between the vertices. Let $\mathcal{X} = \bigotimes_{v\in V} \mathcal{A}_v$*, then* $x \in \mathcal{X}$ *is the configuration* $x = (x_v)_{v\in V}$*,* $x_v \in \mathcal{A}_v$*.*

Definition 2.3.2 (Bayesian network) *A* ***Bayesian network*** *is a pair* $\mathcal{B} = (\mathcal{G}, P)$*, where* $\mathcal{G}$ *is a Bayesian network structure and* P *is a distribution over vectors from* $\mathcal{X}$ *which may be factorized as follows:*

$$P(X = x) = P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid X_{pa(i)} = x_{pa(i)}), \qquad (2.5)$$

with each node $v \in V$*, a conditional probability table (CPT) describing the probability distribution of a given node is associated, which depends on the configuration of its parents. The probability distribution of nodes which do not have parents are defined a priori.*

(a)

		being late		
traffic jam	getting out	no	little	seriously
no	early	85%	10%	5%
no	late	15%	25%	60%
yes	early	30%	40%	30%
yes	late	8%	15%	77%

(b)

	weather		
	sunny	rainy	snowy
-	70%	20%	10%

Table 2.1: Part of the Conditional Probability Tables (CPT) (a) for node *being late* with 3 possible states: 'no', 'little', and 'seriously'. (b) For node *weather* the CPT represents a priori knowledge.

In the Figure 2.3, a simple example of a Bayesian network is shown. The utility node *being late* describes if (and if 'yes', then how much) we are late.

As we can see from Table 2.1, the probability of being late increases if *traffic jam* occurs and is smaller if we *get out* earlier. The probability of *getting out* early is the highest when the *weather* is sunny, is smaller during rain (we need to take more clothes like a coat and an umbrella) and is the smallest during snow. Probability distribution for node *weather* is given *a priori*. Node *traffic* is described by a binary variable which can take values 'high' or 'low'. Rainy weather will increase the probability of higher *traffic* and snow can decrease the traffic intensity. Bad weather conditions have a negative impact on the traffic flow and can cause *traffic jams*. No connection between node *traffic* and node *getting late* indicates no direct dependence between them.

Overall, the probability space, corresponding to this five variables, has $3 \times 2 \times 2 \times 2 \times 3 = 72$ possible values. This parametrization requires 71 non–redundant parameters (the last parameter can be easily determined, because all values of the distributions sum to 1). Using the definition of a Bayesian network, the probability distribution factorizes in the following form:

$$\begin{aligned}
&P(\textit{being late},\ \textit{getting out},\ \textit{traffic jam},\ \textit{traffic},\ \textit{weather})\\
&\quad = P(\textit{being late} \mid \textit{traffic jam},\ \textit{getting out})\\
&\qquad P(\textit{getting out} \mid \textit{weather})\\
&\qquad P(\textit{traffic jam} \mid \textit{traffic},\ \textit{weather})\\
&\qquad P(\textit{traffic} \mid \textit{weather})\\
&\qquad P(\textit{weather})
\end{aligned}$$

To encode *a priori* node *weather* we need 2 parameters. For node *being late* 8 parameters is required and finally, we need $8 + 4 + 3 + 6 + 2 = 23$ non–redundant parameters as opposed to 71 in the original joint distribution.

2.3.2 D–separation

The d–separation connectivity criterion [63], the "d" denoting "directional–dependent", identifies conditional interdependencies in Bayesian networks. The criterion permits us to determine relationship among three variables (or three sets of variables) denoted as X_A, X_B, and X_C i.e. which variable (set of variables) is independent of the other one given a third one; for example $X_A \perp X_C \mid X_B$.

Definition 2.3.3 (Instantiation) *A variable X_v, $v \in V$, is instantiated if the state of X_v is known i.e. $X_v = x_v$. Notation of instantiation is presented in Fig. 2.4.*

In a Bayesian network, there are three possible ways of connection in which two variables are connected by a third one and no direct connection between them occurs. We named them a **chain** (also known as a **causal/evidential trail** or

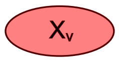


Figure 2.4: Node instantiation.

serial connection), a **common cause** (**fork** or **diverging**), and a **common effect** (**collider**) connection. The *common effect* connection is also called a **v–structure**. Now we will discuss them briefly.

Let us consider a DAG with three variables (sets of variables) named X_A, X_B, and X_C. The *chain* connection is presented in Fig. 2.5. This connection describes a situation where the variables X_A and X_C are not directly connected, but they form a trail with the node X_B between them. Node X_B is named a chain node.

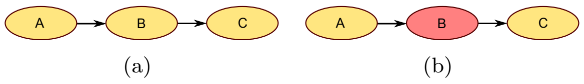


Figure 2.5: Causal chain connection.

The probability distribution over (X_A, X_B, X_C) that factorizes along the graph in Fig. 2.5 may be written as:

$$P(X_A, X_B, X_C) = P(X_A)P(X_B \mid X_A)P(X_C \mid X_B) \tag{2.6}$$

Clearly, $X_C \not\perp X_A$ in general and if an evidence in the middle variable $X_B = b$ is presented, X_B is instantiated, then $X_C \perp X_A \mid X_B$.

The *common cause* connection is presented in Fig. 2.6. Node X_B is called a common cause node.

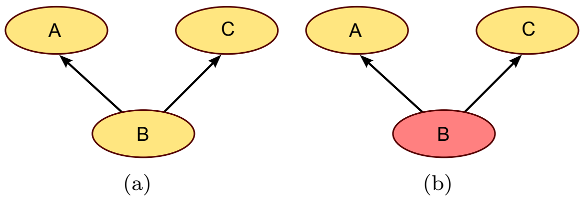


Figure 2.6: Common cause connection.

The probability distribution over (X_A, X_B, X_C) that factorizes along the graph in Fig. 2.6 may be expressed as:

$$P(X_A, X_B, X_C) = P(X_B)P(X_A \mid X_B)P(X_C \mid X_B) \tag{2.7}$$

In general $X_A \not\perp X_C$, but if the middle variable $X_B = b$ is instantiated, then $X_A \perp X_C \mid X_B$.

The *common effect* connection is presented in Fig. 2.7. Nodes X_A and X_C have a common child and are not directly connected. Node X_B is called a common effect node.

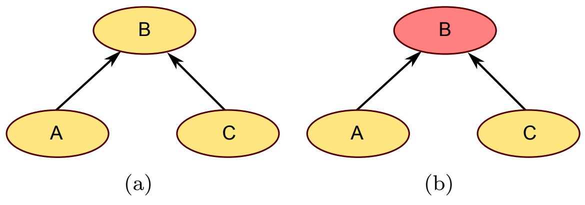


Figure 2.7: Common effect connection.

The probability distribution over (X_A, X_B, X_C) that factorizes along the graph in Fig. 2.7 may be expressed as:

$$P(X_A, X_B, X_C) = P(X_A)P(X_C)P(X_B \mid X_A, X_C) \tag{2.8}$$

In general, we have $X_A \not\perp X_C \mid X_B$, also X_A and X_C are unrelated: $X_A \perp X_C$.

Definition 2.3.4 (Active trail) *For a DAG $\mathcal{G} = (V, E)$, let S be a subset of V and $A, B \in V \setminus S$, suppose that nodes in S are instantiated. A trail (see Def. A.0.4) between A and B, given S, is* ***active*** *if*

- *Every node with two incoming arcs (v–structure) on the trail is in S or has descendants in S.*
- *Every other node along the trail is not in S.*

We can distinguish a special case, that is, an active trail can be a single node.

Definition 2.3.5 (Blocked trail) *A trail between A and B, given S is said to be* ***blocked****, if it is not active given S.*

Definition 2.3.6 (d-separation) *For a DAG $\mathcal{G} = (V, E)$, suppose S be a subset of V and that only nodes in S are instantiated. Let $A, B \in V \setminus S$, if all trails between A and B are blocked by S then we can say that A and B are* ***d–separated*** *by S. We denote this as*

$$A \perp B \parallel_{\mathcal{G}} S$$

Suppose that C and D are subsets of $V \setminus S$. If all trails from any variable in C to any variable in D is blocked by S, then the sets C and D are d–separated by S.

The d–separation property between given nodes could be easily determined with the Bayes–ball algorithm [78].

Definition 2.3.7 (d-connection) *If two variables are not d–separated, then they are* ***d–connected****.*

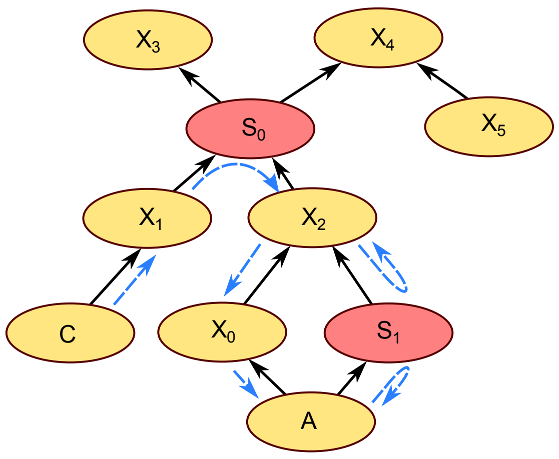


Figure 2.8: Example of a BN. Nodes S_0 and S_1 are instantiated.

Let us consider the BN in Fig. 2.8 with the instantiated nodes S_0 and S_1. The trail $X_2 \leftarrow X_0 \leftarrow A$ is active because no node along the trail is instantiated. The trail $X_2 \leftarrow S_1 \leftarrow A$ is blocked because node S_1 is instantiated. Nodes A and X_2 are d–connected because there exist active trail $X_2 \leftarrow X_0 \leftarrow A$ between them. The trail $A \rightarrow X_0 \rightarrow X_2 \rightarrow S_0 \leftarrow X_1 \leftarrow C$ is active because no node along the trail is instantiated; except S_0 which is a common effect node of the v–structure $X_2 \rightarrow S_0 \leftarrow X_1$. The trail $X_3 \leftarrow S_0 \rightarrow X_4 \rightarrow X_5$ is blocked because node S_0 (common cause node) is instantiated. No other trail between X_3 and X_5 exists, thus X_3 and X_5 are d–separated, given S_0.

Theorem 2.3.8 *Let $\mathcal{G} = (V, E)$, A, B and S be disjoint subsets of V, then a d–separation of A and B by S implies conditional independence between A and B given S:*

$$A \perp B \parallel_{\mathcal{G}} S \Rightarrow A \perp B \mid S \tag{2.9}$$

Proof can by found in [44] (Section 2.8) or [41] (Section 4.5.1).

Definition 2.3.9 (Faithful graph) *DAG $\mathcal{G} = (V, E)$ is* ***faithful*** *to probability distribution P (see Def. 2.3.2), if whenever $A \perp B \mid S$ implies $A \perp B \parallel_{\mathcal{G}} S$.*

Thus, for a faithful DAG $\mathcal{G}$, the conditional independence and d–separation are equivalent. Many structure learning algorithms assume that there exists a faithful graph for a given data.

2.3.3 Essential graph

The following definition of the Markov equivalence class will simplify finding graph structures. Instead of searching the space of all available DAG's, we will only examine different equivalence classes [44].

Definition 2.3.10 (immorality) *For a graph $\mathcal{G}$, an immorality is a v–structure i.e. it is a triple (X, Y, Z) with the following connection $X \rightarrow Y \leftarrow Z$, where edges $X \rightarrow Z$, $Z \rightarrow X$ and $X - Z$ are forbidden.*

Definition 2.3.11 (skeleton) *The **skeleton** of a DAG $\mathcal{G}$ is a graph obtained by making every edge undirected.*

Definition 2.3.12 (Markov equivalence) *Two DAG $\mathcal{G}_1$ and $\mathcal{G}_2$ over the same variables are **Markov equivalent** if and only if any pair of variables d–separated by a set in $\mathcal{G}_1$ is also d–separated by the same set in $\mathcal{G}_2$*

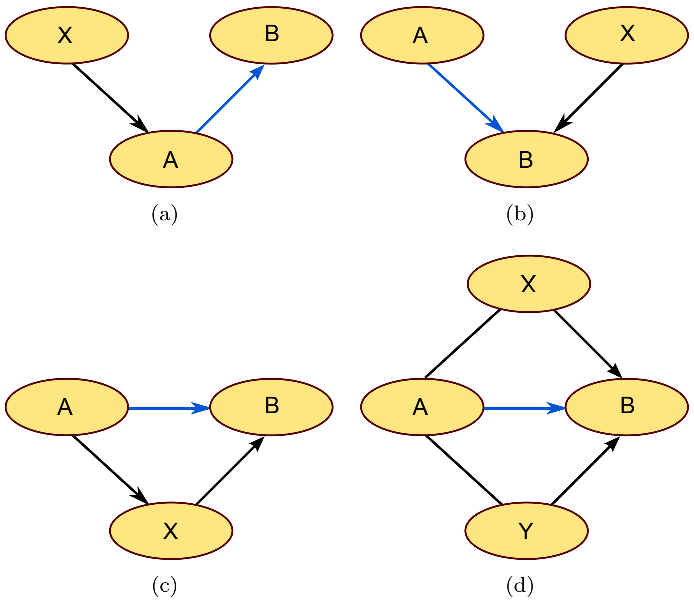


Figure 2.9: Possible compelled edges [44]. Compelled edges are marked with blue.

Definition 2.3.13 (Essential graph, compelled edge) *For a DAG $\mathcal{G}$, an **essential graph** $\mathcal{G}^*$ (it may contain either directed or undirected edges) is defined as a graph, build on the same skeleton as $\mathcal{G}$, where an undirected edge is replaced with a directed one if and only if the direct edge occurs in every graph that is a Markov equivalent to $\mathcal{G}$. That edge is named **compelled edge** in $\mathcal{G}^*$.*

2.3.4 Inference

The most common action in a Bayesian network is to compute beliefs of alternative hypotheses in the context of presented knowledge. That operation is called a **probabilistic inference** or a **belief updating**. Considering the Bayesian network presented in Fig. 2.3, an example of a probabilistic inference could be to compute the probability that *being late* = *'no'*, given the knowledge that *weather* = *'snowy'* or the probability distribution in node *traffic jam*, given that *weather* = *'sunny'* and *being late* = *'seriously'*. In general, probabilistic inference on a BN is the process of computing $P(X_Y \mid X_S = x_S)$, where nodes in a set Y are called query nodes and nodes in a set S are observed (instantiated) nodes.

Definition 2.3.14 (conditional probability query) *For BN $\mathcal{B} = (\mathcal{G}, P)$, where $\mathcal{G}$ is a DAG $\mathcal{G} = (V, E)$, let Y and S be subsets of V, suppose that nodes in S are instantiated ($X_S = x_S$). The conditional probability query is in the following form:*

$$P(X_Y \mid X_S = x_S). \tag{2.10}$$

There exist two major classes of inference algorithms – exact and approximate ones. The exact inference (answering probabilistic queries) in an arbitrary BN has been shown to be NP–hard, by reducing 3SAT problem[2] to exact inference [16], i.e. an exponential relationship between computation complexity and the number of variables (network complexity) is often presented. Thus this limits the number of networks where exact inference methods are computationally feasible. In [19], there has been shown that even an approximate inference in BN is NP–hard. Only in restricted classes of networks, the probabilistic inference can be polynomial or even linear. The linear complexity [63] is available in tree structured networks (every node, excluding the root, has exactly one incoming edge).

However, tree structured networks cover only a small part of interesting real–world problems. Some other have been successfully applied to handling general networks. First, the problem has been solved, with a polynomial complexity [63, 36], when the Bayesian network structure is represented by a polytree – a node may have multiple parents, but the network is *single connected*, i.e. no more than one path exists between any two nodes. That algorithm, called the **message passing algorithm**, is the basis for the most popular exact BN inference algorithms in general BN – the **clique-tree propagation** algorithm (also known as the **junction tree** algorithm) [49]. First, the algorithm moralizes (see Def. A.0.11) the graph, then the underlying undirected graph is triangulated (check Def. A.0.12). Now, from disjoint cliques, a clique tree (junction tree) can be formed, and the message passing algorithm can be applied. The efficiency of the algorithm depends on the density of the network (the complexity is exponential in the size of the largest clique).

For most of small enough networks (up 40 nodes [43]), the exact algorithms work good enough. For densely connected and bigger networks, the junction tree algorithm requires very large tables for the cliques in the triangulated graph. Sizes of cliques may exceed the available hardware limitations. In such a case,

[2]For the formula with Boolean variables which is expressed as an AND of ORs (conjunctive normal form), where each clause has at most 3 variables in it, the 3–Satisfiability (3SAT) problem involves determining whether there is an assignment to the variables that satisfies the formula if one exist. It is the first example of a NP–complete problem [15] and it can be used for proving that other problems are NP–hard.

the **recursive conditioning** can be applied, which reduces the required storage, but greatly increases the computational time [35]. If computation is not feasible by exact algorithms, then the approximate ones should be applied.

The simplest approximate algorithm is the **logic sampling (LS)** [28]. The algorithm starts at leafs, follows the influence arrows (hence also the name forward sampling) and generates cases by randomly selecting a value at each node, weighted by the probability of that value occurring. Instances of the network that are inconsistent with the evidence $X_S = x_S$ are discarded. The probability of a query is estimated by dividing the number of cases, where both the query and the evidence are meet, the number of cases where the evidence is meet. The algorithm works very well when no evidence is provided and all network instances are proceeded. When several evidence nodes are introduced, then most of the instances are discarded and the algorithm works poorly.

The main drawback of the logic sampling algorithm can be easily fixed. The resulted algorithm, named the **likelihood weighting (LW)** [23, 77], does not discard inconsistent instances, but instead, when an evidence node is proceed, weights the sample by the probability of the evidence given parents. The algorithm converges faster than the logic sampling, but both perform better when the evidence is located near root nodes rather than leafs [43]. In [20], a modified version of LWS is presented. The **bounded–variance** algorithm achieves faster convergence by reducing the variance of sample estimates. In addition, a stopping rule is added; thus, the number of samples can be estimated to achieve a desired accuracy.

The **Gibbs sampling** algorithm, differs from the previous ones because it does not generate a new instance of the network from a scratch. Instead, it generates a sample by making a random change in a previous instance. It works by sampling a value of a non–evidence node conditioned on its Markov blanket [35, 44]. Instances are constantly consistent with the knowledge. The algorithm is easy in implementation, but it is difficult to say when it converges; furthermore, it experiences a slow performance for bigger Markov blankets [35].

2.3.5 Learning

Bayesian networks can be constructed by a human expert or by an automatic learning method. There are many algorithms producing graphical models from a given data. They are useful because automatically produced models show usually better performance than those hand constructed. There are two basic learning problems in Bayesian networks: learning the structure of a graph and learning the conditional probability potentials.

Parameter estimation

The task of parameter estimation in a Bayesian network with a given DAG is widely described in [35, 44, 58]. In this problem, we assume that network structure $\mathcal{G} = (V, E)$, where $V = \{X_1, X_2, \ldots, X_n\}$ and a data set $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ of m i.i.d. examples are given. Each element M_i from the data set contains observations of all variables in V, i.e. $M_i = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})^T$ is a vector of instances of variables $X_1, X_2, \ldots, X_n$.

The objective is to estimate an unknown vector of parameters θ, which defines the set of parameters for all variables, i.e. for a given node X_i, let k_i be a number of possible states of X_i, $x_{i,k}$ be a k^{th} possible state of X_i and $q_{i,l}$ be an l^{th} configuration of parents of the node X_i, parameter θ is defined as $\theta_{X_i = x_{i,k} \mid q_{i,l}, \mathcal{M}, \mathcal{G}} = P(X_i = x_{i,k} \mid q_{i,l})$.

The likelihood function, for a given network structure $\mathcal{G}$ and parameters θ, is defined as:

$$L(\theta \mid \mathcal{M}, \mathcal{G}) = \prod_{i=1}^{m} P(M_i \mid \theta, \mathcal{G}) \tag{2.11}$$

Definition 2.3.15 *The* ***Maximum Likelihood Estimate*** *is defined as a value of $\hat{\theta}$ that maximizes the likelihood function $L(\theta \mid \mathcal{M}, \mathcal{G})$.*

It has been shown in [41] that the likelihood can be decomposed (*parameters independence*) into independent terms and each term can be maximized locally.

$$L(\theta \mid \mathcal{M}, \mathcal{G}) = \prod_{i=1}^{n} L_i(\theta_{X_i \mid pa(X_i)} \mid \mathcal{M}, \mathcal{G}), \tag{2.12}$$

where local likelihood is defined as:

$$L_i(\theta_{X_i \mid pa(X_i)} \mid \mathcal{M}, \mathcal{G}) = \prod_{j=1}^{m} P(x_i^{(j)} \mid pa(x_i^{(j)}), \theta, \mathcal{G}), \tag{2.13}$$

where $x_i^{(j)}$ is a j^{th} instance of X_i in data set $\mathcal{M}$ and $pa(x_i^{(j)})$ is a j^{th} instance of node's X_i parents. Then the maximum likelihood estimate of parameter $\theta_{X_i = x_{i,k} \mid q_{i,l}}$ is:

$$\hat{\theta}_{X_i = x_{i,k} \mid q_{i,l}} = \frac{n_{\mathcal{M}}(X_i = x_{i,k} \mid q_{i,l})}{\sum_{j=1}^{k_i} n_{\mathcal{M}}(X_i = x_{i,j} \mid q_{i,l})}, \tag{2.14}$$

where function $n_{\mathcal{M}}(X = x)$ counts number of entries in $\mathcal{M}$ that have $X = x$.

Estimating parameters from a data in the MLE approach shows a weak performance when learning large, dense networks. The problem occurs because when the number of node's parents grows, then the number of parameter assignments grows exponentially. The data have to be partitioned into small subsets (*data fragmentation*) and we have a small number of data to estimate a parameter; thus, the estimate can be noisy or even we can end up with a large number of zeros. For such situations, the **Bayesian approach** with the mean posterior estimate **(MPE)** [27, 41, 44] is a better choice.

Learning Bayesian network structure

Now we will focus our attention on learning the structure of a Bayesian network from a given, fully–observed dataset. The task has been proven to be an NP–hard problem, even for networks with two parents [13]. Some special structures, like trees, can be learned in a polynomial time with a popular Chow–Liu algorithm [14], but it has been proven in [21] that even learning 2–polytrees is an NP–hard problem.

In recent years, many learning algorithms (see [35, 41, 44, 58]) for building a structure of Bayesian networks have been developed. Generally, they can be divided into three main groups: **constraint based** algorithms, **search and score** techniques, and **hybrid** methods that combine the two previous ones.

The problem of learning a Bayesian network structure is given as follows: for a set of random variables $C = \{X_1, \dots, X_n\}$ and a database of m cases $M = \{M_1, \dots, M_m\}$, where each case contains complete observations of all variables in C, i.e. $M_i = (x_1, \dots, x_n)^T$ is a vector of instances of variables $X_1, \dots, X_n$, find a DAG (that is a set of directed edges) which best matches M.

The structure of a Bayesian network encodes dependencies of the model, whereas the **constraint based** algorithms perform a study of the dependence and independence relationships among variables of the Bayesian network and try to use them to infer the graph structure. The relationship between nodes can be determined by conditional independence tests. For datasets large enough, χ^2 or G^2 tests can be performed and for the smaller ones exact tests could be done as well.

The G^2-test is currently recommended by many researchers over the χ^2. Values of the G^2 statistics are equivalent to the **mutual information** up to a constant [47]. To test whether $X \perp Y \mid Z$, the G^2 statistics should be computed, which is defined as follows:

$$G^2(X,Y,Z) = 2\sum_x \sum_y \sum_z N_{xyz} log \frac{N_{xyz}N_z}{N_{xz}N_{yz}}, \tag{2.15}$$

where N_{xyz} denote the number of times $(X,Y,Z) = (x,y,z)$ appears in the data. Similarly, N_{xz} and N_{yz} are defined. The degree of freedom, denoted by k, is given by

$$k = (\|X\| - 1)(\|Y\| - 1)\|Z\|, \tag{2.16}$$

where $\|X\|$ is the number of values that variable X can take. The distribution of G^2 is asymptotically a χ^2 one on k degrees of freedom.

Constrained based algorithms assume that there exists a DAG that is faithful to the probability distribution (each conditional independence is represented by a d–separation statement). A detailed discussion of the algorithms can be found in [12, 45]. We will discuss briefly only the classical ones. Most of the algorithms, instead of learning a full DAG, return an **essential graph** (see Def. 2.3.13). The

SGS algorithm (named after the authors: Spirtes, Glymour, and Scheines) [81] starts with a *skeleton identification* stage:

1. Start with a fully connected, undirected graph $\mathcal{G} = (V, U)$.
2. Arcs removal – for each pair of vertices A and B, if there exists a set that d–separates them, then the edge $A - B$ is removed.

That is followed by an *orientation* stage:

1. v–structures identification – for each triple of vertices connected through a middle one, i.e. $A - B - C$ and $A \nsim C$ if and only if there is no subset of $\{B\} \cup V \setminus \{A, C\}$ d–separates A and C, then orient $A - B - C$ as $A \rightarrow B \leftarrow C$.
2. Add compelled edges using rules from Def. 2.3.13.

The SGS algorithm is very computationally inefficient because a number of conditional tests during skeleton–identification stage grows exponentially in a number of graph vertices. Also, testing the higher order conditional independence relations is less reliable than the lower order relations and requires more data to be proceeded.

The **PC algorithm** [80, 81] (PC is an acronym of the first letters of the first names of the scientists that proposed the algorithm: Peter Spirtes & Clark Glymour [79]) is very similar to SGS algorithms, but enhances the arcs removal step:

- First, edges with the zero order conditional independence (conditioned on the empty set) are removed, then the first order conditional independence (conditioned on single variable) is checked, etc.
- The set of conditioned variables is a subset of a set of variables adjacent to conditioned vertices.

For the PC algorithm, the number of independence tests grows exponentially with the maximum degree of any vertex in $\mathcal{G}$ [81].

The main drawbacks of the algorithms mentioned above are: the computational complexity of the independence tests, requirements for large data sets, unreliable results of the independence tests, and also the fact that the most widely used algorithms require an existence of a faithful graph [44].

The **search and score** techniques attempt to find a graph structure that maximizes the value of a given scoring function.

The Chow–Liu algorithm [14] tries to find the best fitting tree. The distance is measured by the Kullback–Leibler divergence and the objective is to minimize that measure.

Definition 2.3.16 (Kullback–Leibler divergence) *For two probability distributions P and Q over the same finite state space, the **Kullback–Leibler divergence** is defined as*

$$D_{KL}(P \parallel Q) = \sum_x P(x) log \frac{P(x)}{Q(x)}. \tag{2.17}$$

Definition 2.3.17 (Mutual information) *For two discrete random variables X_i and X_j, the **mutual information** can be defined as*

$$\mathcal{I}(X_i, X_j) = \sum_{x_i} \sum_{x_j} P(x_i, x_j) log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}. \tag{2.18}$$

The mutual information can be seen as a Kullback–Leibler divergence between $P(X_i, X_j)$ and $P(X_i)P(X_j)$:

$$\mathcal{I}(X_i, X_j) = D_{KL}(P(X_i, X_j) \parallel P(X_i)P(X_j)) \tag{2.19}$$

First, the algorithm computes the **mutual information** between any two variables and associates these values with weights on the corresponding edges. Then, in order to minimize the KL divergence, we construct a tree that maximizes total weights, i.e. a tree with a maximal value of the sum of weights on tree's edges. Such problem is well known as a Maximum Spanning Tree problem and can be solved in quadratic time [46].

Different **scoring functions** have been proposed to evaluate a graph structure [9, 45]. They can be divided into two main classes:

- `Bayesian scoring` functions:
  - K2,
  - Bayesian Dirichlet test and its variants (BD, BDe, BDeu).
- `Information-theoretic` scoring functions:
  - Log Likelihood (LL),
  - Bayesian Information Criterion (BIC),
  - Akaike Information Criterion (AIC).

One of the most popular scoring function is a modification of the Cooper–Herskovits likelihood (belonging to the Bayesian scoring class) [17], for a DAG G and a dataset M with $P(G)$ as a prior probability of G. It has the following form:

$$P(G, M) = P(G) \times \prod_{k=1}^{n} \prod_{j=1}^{|\phi_k|} \frac{(s_k - 1)!}{(s_{kj} + s_k - 1)!} \prod_{l=1}^{s_k} \alpha_{kjl}! \tag{2.20}$$

where s_k is a number of states of the variable X_k, ϕ_k is a variable describing joint configurations of variables in $pa(X_k)$, $|\phi_k|$ is a number of states of ϕ_k and α_{kjl} is a number of cases in M in which X_k is at the l^{th} state and ϕ_k is at the j^{th} state. Also $s_{kj} = \sum_{l=1}^{s_k} \alpha_{kjl}$. A broad spectrum of scoring functions has been evaluated in [9] with the conclusion that there are only small differences between various scoring functions and all of them behave in a similar way (only the BIC score was clearly the worst).

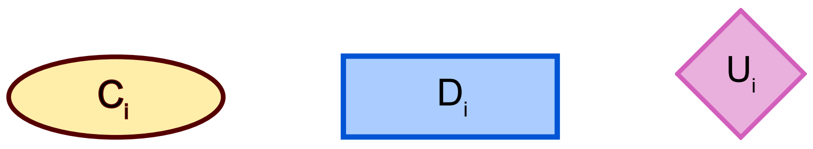
A search space of possible DAG's grows super–exponentially [67], so testing all possible DAG patterns is computationally unfeasible. There have been developed many algorithms for searching the graph space (see [9, 45]). A classical algorithm of a **greedy search** has been proposed in [17], called the **K2** algorithm. It requires that a node ordering is given and the graph has no edges. The algorithm proceeds from the root node (nodes are ordered) and adds incrementally parents that increases mostly a value of a scoring function (K2). If adding an additional parent does not increase the scoring function, then we stop adding parents to that node and proceed with the next one. The algorithm stops when no additional parents can be added to the last vertex.

2.4 Influence Diagrams

Influence diagrams (ID) [32, 63] are widely acknowledged as compact representations of decision problems. They can be viewed as a decision tool extending the capabilities of Bayesian networks. The ID can be considered as generalizations of (symmetric) decision trees, see [35].

An **influence diagram** is built on a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$ whose nodes and edges admit standard interpretations and extend those used for Bayesian networks. Three principal types of nodes in $V = C \cup D \cup U$ are considered (see Fig. 2.10):

- C – **chance nodes** representing random variables (represented by ovals, see Fig. 2.10(a))
- D – **decision nodes** corresponding to available decisions (represented by rectangles, see Fig. 2.10(b)),
- U – **utility nodes** specifying the utilities to be maximized by suitable choices of decision policies (represented by rhombuses, see Fig. 2.10(c)).



(a) chance node (b) decision node (c) utility node

Figure 2.10: Types of nodes in an influence diagram.

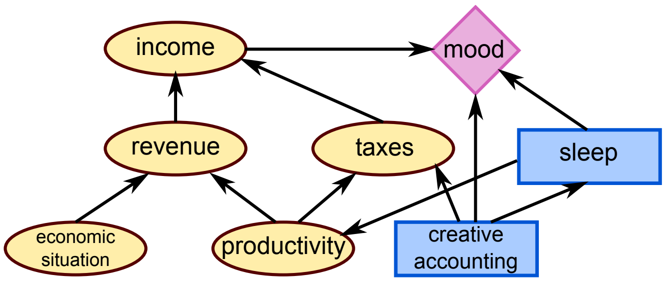


Figure 2.11: Illustration of a simple influence diagram.

The chance nodes C inherit their own properties from chance nodes in BNs, i.e. the nodes are associated with random variables. The utility nodes $U = \{U_1, U_2, \dots, U_k\}$ have no children and they do not have states. With each utility node U_i a real–valued function is associated. The decision nodes $D = \{D_1, D_2, \dots, D_m\}$ have finite sets of states, i.e. each decision node $D_i \in D$ can take values from a given (finite) set of states $\mathcal{A}_{D_i}$. We also suppose that there exists a directed path over V that contains all decision nodes D. This assumption justifies that there exists a sequence of decisions.

The edges in E can indicate two types of influences:

- **conditioning influence** – represented by arrows leading to chance nodes indicating a direct causal relationship,
- **informational influence** – represented by arrows leading to decision nodes and specifying the information available at the moment of decision making.

An example of an influence diagram can be seen in Fig. 2.11. The network consists of five chance nodes, two decision nodes, and one utility node. A chance node *income* represents a probabilistic distribution of our profits that are directly affected by nodes *revenue* and *taxes*. The level of taxation depends on the *productivity*. *Productivity* also has a direct impact on the *revenue* that is additionally influenced by the current *economic situation*. The decision node *creative accounting* represents the choice whether or not to misrepresent our true income. A creative accounting does decrease the taxes, yet it also causes negative effects, such as the fear of being caught. Finally, more *sleep* decreases the chance of a higher productivity but it also positively affects our mood. All these effects are jointly taken into account in the utility node *mood*.

Definition 2.4.1 (outcome) *Let $G = (V, E)$ be an ID. An **outcome** is an instantiation of all variables in V. We denote this as Ψ.*

Then, the $P(\Psi)$ denotes the probability of a fully–observed network and $U_i(\Psi)$ denotes the value of utility function in node U_i. We should emphasize that the value of an utility function in node U_i depends only on the parents of U_i.

Definition 2.4.2 (policy) *A* ***policy*** δ_{D_i} *for a decision node* $D_i \in D$ *is an action for each possible configuration of* D_i *parents. It is a mapping*

$$\delta_{D_i} : \bigotimes_{Z \in pa(D_i)} \mathcal{A}_Z \to \mathcal{A}_{D_i}. \tag{2.21}$$

The adopted definition of the policy, given in [35], considers it as a deterministic action. A different approach is presented in [41], where policies are represented by probability distributions.

Definition 2.4.3 (strategy) *A set of policies, one policy for each decision node* $D_i \in D$*, is called a* ***strategy****.*

For a fixed strategy $\Delta = (\delta_1, \delta_2, \ldots, \delta_m)$, a decision node $D_i \in D$ is equivalent to a chance node and it has the following conditional probability distribution:

$$P(D_i = d \,|\, pa(D_i)) = \begin{cases} 1 & \text{if } \delta_i(pa(D_i)) = d, \\ 0 & \text{otherwise.} \end{cases} \tag{2.22}$$

Definition 2.4.4 (expected utility) *For an influence diagram build on a DAG* $\mathcal{G} = (C \cup D \cup U, E)$ *and a fixed strategy* Δ*, the* ***expected utility*** *of an influence diagram is defined as a sum over all possible outcomes* Ψ*:*

$$EU(\Delta) = \sum_{\Psi} P(\Psi) U(\Psi), \tag{2.23}$$

where value of the utility U *is the sum of individual nodes in* U*:*

$$U(\Psi) = \sum_{i: U_i \in U} U_i(\Psi). \tag{2.24}$$

Definition 2.4.5 (optimal strategy, maximum expected utility) *If a strategy* Δ *maximizes the expected utility function, then the resulting strategy is called* ***optimal strategy****, the value of the function* $EU(\Delta)$ *is named* ***maximum expected utility*** *and policy* $\delta \in \Delta$ *is called an* ***optimal policy****.*

In general, finding an optimal decision strategy for an influence diagram is an NP–hard problem. One can show this easily by reducing the travelling salesman problem (that is NP–complete) to our task [35].

There has been proposed a number of algorithms for solving (finding an optimal decision strategy) an influence diagram. A detailed discussion can be found in the following positions: Chapter 10 in [35], Subsection 5.2.2 in [58], and Chapter 23 in [41].

2.5 Gaussian Bayesian Networks

Gaussian Bayesian networks have been introduced in [76] and describe an extension of Bayesian networks to variables that are continuous. They are a popular

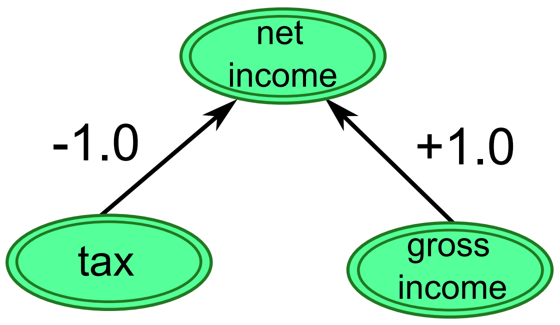


Figure 2.12: Example of a simple Gaussian Bayesian network.

tool for decision making and inference. For example, in molecular biology they are used to describe protein dynamics [26]. Gaussian networks offer polynomial time of manipulation and their local nature of computation can be helpful in parallel applications [76].

Definition 2.5.1 (linear Gaussian model) *Let Y be a continuous random variable associated with a given node with only continuous parents $pa(Y) = \{X_1, \dots, X_k\}$. Y is a **linear Gaussian model** if it can be described using parameters $\beta_0, \dots, \beta_k$ and σ^2 such that the probability density of the random variable Y is given by:*

$$P(Y \mid x_1, \dots, x_k) = \mathcal{N}(\beta_0 + \beta_1 x_1 + \dots + \beta_k x_k, \sigma^2) \tag{2.25}$$

or using a vector notation we can simply write:

$$P(Y \mid \mathbf{x}) = \mathcal{N}(\beta_0 + \beta^T \mathbf{x}, \sigma^2) \tag{2.26}$$

We should notice that in a linear Gaussian model, the variance does not depend on parents values.

In Fig. 2.12, an example of a Gaussian Bayesian network is presented. The variable *net income* depends on variables *tax* and *gross income* with parameters β equal to -1 and $+1$ respectively.

Definition 2.5.2 (linear Gaussian Bayesian network) *A **linear Gaussian Bayesian network** $\mathcal{G} = (V, E)$ is a Bayesian network, where all variables are continuous and all conditional probability distributions are linear Gaussian ones.*

That is, the continuous variable Y is normally distributed around a mean that depends linearly only on the values of its parents. The model captures many interesting dependencies. Next advantage of that model is its simple representation. The drawback of Gaussian networks is that their representation is limited to modelling linear dependencies between variables and fixed variance (it does not depend on parents values).

The Linear Gaussian Bayesian network defines a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ [41, 69].

Let us investigate the connection between the graph $\mathcal{G}$ and the parameters of the normal distribution. Similarly, we will be interested in the dependence properties. The mean parameter μ does not have any influence on the dependencies and all the information is stored in the covariance matrix Σ. The covariance matrix encodes the marginal independencies i.e.:

$$\Sigma_{ij} = 0 \Leftrightarrow X_i \perp X_j \tag{2.27}$$

The result is not very useful, because the knowledge of conditional dependencies is preferred. Thus, it is better to consider the inverse covariance matrix, the **precision matrix** $Q = \Sigma^{-1}$. The precision matrix Q directly encodes the conditional independences:

$$Q_{ij} = 0 \Leftrightarrow X_i \perp X_j \mid X_{-ij}, \tag{2.28}$$

where $i \neq j$ and X_{-a} denotes a set of all variables except for X_a. The nonzero values in Q determine connections in $\mathcal{G}$, so it is possible to determine whether X_i and X_j are conditionally independent.

The **canonical representation** is defined with **canonical parameters** $b = Q\mu$ and Q is denoted as $N_C(b, Q)$. The relation to the normal distribution is as follows: $N_C(Q\mu, Q) = N(\mu, Q^{-1})$

Sampling from canonical representation is similar to sampling from a multivariate normal distribution. The algorithm for sampling from $N_C(b, Q)$ works in the following manner [69]:

1. Compute Cholesky factorization, $Q = LL^T$
2. Compute μ by solving $Lw = b$ and then $L^T\mu = w$
3. Sample z from $N(\mathbf{0}, \mathbf{I})$
4. Compute and return $x = \mu + v$, where v is a solution of $L^T v = z$

Learning

To characterize graph $\mathcal{G}$, we can provide the mean μ and precision matrix Q. The mean vector $\hat{\mu}$ can be simply estimated using means. The $\hat{Q}$ could by estimated by first computing the $\hat{\Sigma}$ (using for example maximum likelihood estimator) and then inverting the matrix. It is a highly unstable and not feasible approach. The simple and effective method for estimating $\hat{Q}$ is the use of regression methods.

2.6 Polygonal Markov Fields

In this chapter we will focus on another important class of probabilistic graphical models, build on the basis of undirected graphs, called Markov Network (also known as Markov Random Field (MRF)). This model has received a great deal of attention, due to its wide applications to computer vision [41], such as: edge detection, image restoration, stereovision, image classification, and image segmentation. This type of modelling in vision was introduced in [24]. In most of that vision algorithms use a subclass of MRF structured in the form of a grid, where variables correspond to pixels of the image and the edges correspond to interactions between adjacent pixels. For some tasks, like image segmentation, it is more natural to process regions of pixels with edges that define interactions between adjacent regions. That concept was introduced in Polygonal Markov fields (PMF), originally constructed by Arak, Clifford [2, 3], and Surgailis [83]. PMF are random ensembles of non–intersecting contours in the plane, arising in a Gibbsian set-up and sharing a number of important features with the two-dimensional Ising model. Due to their purely continuum nature, polygonal fields admit natural applications for instance in a digital image processing and segmentation, where they can be used for the majority of tasks traditionally reserved for lattice–based Markov fields, while being completely free of directional lattice artefacts [39, 40].

Now we will recall the formal construction of the consistent multicoloured polygonal Markov fields [2], adapted from [40] to better fit our needs.

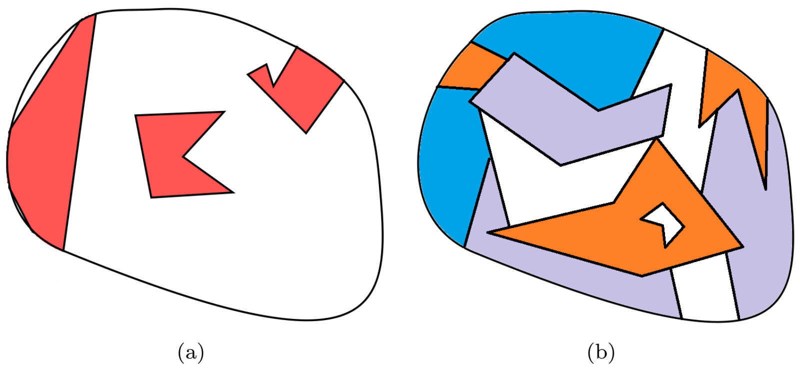


Figure 2.13: Realization of a polygonal Markov field: (a) with two ($k = 2$) available colours; (b) with four ($k = 4$) available colours.

Let $D \subseteq \mathbb{R}^2$ be a convex bounded domain and $\mu(dl)$ be a finite and non–atomic measure on set $\mathcal{L}_D$ of all lines l in $\mathbb{R}^2$ which intersect D, and let $J = \{1, \dots, k\}$ be a set of available colours, $k \geq 2$. For any collection $(l)_n = (l_1 \dots, l_n)$ of lines, where $l_i \in \mathcal{L}_D$, the set $\Gamma_D(l)_n$ consists of all functions $\omega : D \to J$ such that

1. $\partial\omega \subset \bigcup_{n=0}^{\infty} l_i \cap D$, where $\partial\omega$ is the set of discontinuity points od ω.

2. For any $i = 1, \dots, n$, the intersection $l_i \cap \partial\omega$ consists of a single segment with a positive length and possibly some isolated points.

Define the set of all realizations on D by

$$\Gamma_D = \bigcup_{n=0}^{\infty} \bigcup_{(l_n)} \Gamma_D(l)n. \tag{2.29}$$

$\hat{\Gamma}_D$ is the set of all planar graphs γ in $D \cup \partial D$ with faces coloured by labels in J such that the following conditions are satisfied

- all edges of γ lie on the lines of $\mathcal{L}_D$,
- all vertices of $\gamma \in D$ are of degree 2, 3, or 4,
- all vertices of γ on ∂D, are of degree 1,
- no adjacent regions share the same colour.

In other words, γ consists of a finite number of disjoint polygons, possibly nested and chopped off by the boundary – see Fig. 2.13 for a typical realization of that process.

The polygonal Markov field $\mathcal{A}_D$ on D with the Hamiltonian given by total edge length is described as

$$P(\mathcal{A}_D \in G) = \frac{\mathbb{E}\sum_{\gamma\in\Gamma_D(\Lambda_D)\cap G} exp(-2length(\gamma))}{\mathbb{E}\sum_{\gamma\in\Gamma_D(\Lambda_D)} exp(-2length(\gamma))}, \tag{2.30}$$

for all $G \subseteq \Gamma_D$ Borel measurable and Λ_D is the restriction of the homogeneous Poisson line process Λ to D.

An efficient simulation technique for polygonal Markov fields has been presented in [73].

Chapter 3

Short description of the results

In this dissertation, we have studied problems related to probabilistic graphical models. Hereby, the Chen–style stochastic optimization has been extensively used. In the framework of graphical models, we have developed several new algorithms and methods that are used in various fields. All the results have been confirmed by computer simulations.

The new stochastic algorithm for solving influence diagrams has been presented in Paper **A**. The proposed algorithm generates optimal decision strategies, by attaching randomized policies to each decision node. These randomized policies evolve in the course of the optimization process and eventually become (sub)optimal deterministic policies collectively determining the utility maximizing strategy for the influence diagram being considered. At the basis of the evolution of policies lies the Chen–style stochastic optimization. The algorithm generates optimal decision policies for a given influence diagram. Its capabilities have been tested not only in ID but also in real problems in Paper **D** and **E**. The idea of the algorithm was introduced by T. Schreiber and was further investigated in cooperation with M. Matuszak. M.M. implemented the algorithm and wrote the paper jointly with T.S.

An algorithm for solving the ramified optimal transport problem has been developed in Paper **D**. The objective of the ramified optimal transport is to find an optimal path from possible multiple sources to destination locations. First, we present a transformation of the optimal transport problem into an influence diagram framework. Resulting ID could consist of hundreds of decision nodes and is a good opportunity to show a significant performance of our stochastic algorithm in solving influence diagrams. Finally, we translate the optimized decision policy into an optimal transport path. The algorithm introduces an innovative application of Bayesian influence diagrams. The concept of using the ID in the ramified optimal transport problem was introduced by T.S. The research was conducted and the algorithm was implemented by M.M. The paper was written by M.M. with the help of J. Miękisz.

The extended version of the algorithm from Paper **A** has been applied to learning the Bayesian network structure. The proposed method (see Paper **E** for more details) translates, within the framework of influence diagrams, structure learning task into the strategy optimization problem, which can be solved with a significantly extended version of our stochastic algorithm for solving influence diagrams. Experimental evaluations prove the competitiveness of our method for some classes of graphs. The idea of the research was introduced by M.M. and was investigated by M.M. and J.M. The algorithm was implemented by M.M., and the paper was written by M.M..

In Paper **B**, we present an algorithm for determining optimal transition paths between given configurations of systems consisting of many objects. We propose a Lagrange function for our system and then minimize the action functional on a given time interval. We apply that algorithm to:

- Controlling motion and redeployment between unit's formations. The resulting group dynamics is highly complex and a great care must be taken when such an environment is simulated.

- Transformation between two sequences of character animations in a virtual environment. The character's motion has been encoded in terms of Gaussian networks. The resulting animations were satisfying for the viewer.

Most of the time, consuming simulations have been performed on a graphic card (GPU). The concept of the research was introduced by T.S. However, the research itself was carried out by J.M., T.S., and M.M. The algorithm was implemented by M.M. The paper was written by M.M.

For a class of polygonal Markov fields with local activity functions, we have developed an optimization dynamics for the image segmentation, under which the polygonal configuration evolves according to a simulated annealing scheme and the local activity function, initially chosen to reflect the image gradient information, evolves according to the Chen algorithm. Such mechanism has been applied to the image segmentation [Paper **C**]. The algorithm works on the intuitive level, i.e., the real world is not a collection of pixels, and as a consequence, if we do not know what is in the image, we cannot model the objects. We should also emphasize that our implementation is fast and easy. The idea of the research was introduced by T.S., who had various significant results in the field of stochastic geometry. It was the first joint work of M.M and T.S. M.M proposed improvements to the algorithm and performed the implementation. The theoretical part of the paper was written by T.S.; whereas the *results and discussion* were described by M.M.

3.1 Future Work

The results of this dissertation point to several interesting directions for future work.

First of all, in the influence diagrams framework, we are going to develop an agent capable to perform like a human player in computer strategic games. Agent should be able to adapt to changes of opponent's tactics or skills and utilizes only information available to the human player (which might be incomplete).

The solution for the smooth conditional transition paths problem opens the door to a wide range of possible enhancements. Currently, the transition time has to be given before the algorithm starts. We are going to optimize not only the trajectories, but also the transition time. In complex situations with many constraints, where the solutions cannot be obtained analytically, the geometric minimum action method [29] will be applied. The optimal transition paths can be also applied in computer graphics. We are going to encode the vertices of a 2D or 3D model and thanks to the presented method, a smooth Mesh Morphing will be achieved.

The described image segmentation algorithm [55] is a great starting point for further developments, like a multi–coloured image segmentation with multi–coloured polygonal Markov fields [40]. Moreover, feasible measures for an image segmentation quality, like *Rand error* [84] or *warping error* [33] instead of *pixel error*, should be implemented. Very interesting results could be also achieved when we combine the presented algorithm with [40, 74].

Finally, we are going to develop a concurrent structure learning algorithm of Bayesian networks with the use of the capabilities of modern graphic cards [37, 54].

Bibliography

[1] Ambrosio, A. Optimal transport maps in Monge–Kantorovich problem, *Proceedings of the ICM, Beijing* 3: 131–140 (2002). 1

[2] Arak, T., Surgailis, D. Markov Fields with Polygonal Realizations, *Probab. Th. Rel. Fields* **80**, 543-579 (1989). 26

[3] Arak, T., Surgailis, D. Consistent polygonal fields, *Probab. Th. Rel. Fields* **89**, 319-346 (1991). 26

[4] Balch, T., Arkin, R.C. Behavior–based formation control for multirobot teams, *IEEE Transactions on Robotics and Automation*, 14(6), pp. 926–939 (1998). 2

[5] Balch, T., Hybinette, M. Behavior–Based Coordination of Large-Scale Robot Formations, *Multi-Agent Systems*, International Conference on Multiagent Systems – ICMAS, pp. 363–364, (2000). 2

[6] Beucher, S., Lantujoul, C. Use of watersheds in contour detection, In International workshop on image processing, real–time edge and motion detection (1979). 2

[7] Binford, T. O., Levitt, T. S. Evidential Reasoning for Object Recognition, IEEE Trans. Pattern Anal. Mach. Intell. 25(7). pp. 837–851 (2003). 1

[8] Burtnyk, N. and Wein, M. Interactive skeleton techniques for enhancing motion dynamics in key frame animation, *Commun. ACM 19*, pp. 564–569 (Oct. 1976). 2

[9] de Campos, L. M. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests, Journal of Machine Learning Research 7, pp. 2149–2187 (2006). 20, 21

[10] Chen, K. A general learning algorithm for solving optimization problems and its application to the spin glass problem, *Europhys. Lett.* **43** 6, 635–640 (1998). 6

[11] Chen, K. Simple learning algorithm for the traveling salesman problem, *Phys. Rev. E* **55**, 7809–7812 (1997). 6

[12] Cheng, J., Bell, D., Liu, W. Learning Bayesian Networks from Data: An Efficient Approach Based on Information Theory, Technical report, University of Alberta, Canada (1998) 18

[13] Chickering, D. M. Learning Bayesian networks is NP–complete, Learning from Data: Artificial Intelligence and Statistics V, pp. 121–130. Springer–Verlag (1996). 18

[14] Chow, C. K., Liu, C. N. Approximating discrete probability distributions with dependence trees. IEEE Trans. Info. Theory, 14(3), pp. 462–467, (1968). 18, 19

[15] Cook, S The complexity of theorem proving procedures, Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151–158 (1971). 15

[16] Cooper, G.F. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Network, Artificial Intelligence, 42: 393–347 (1990). 15

[17] Cooper, G.F., Herskovits, E. A Bayesian method for the induction of probabilistic networks, Data Machine Learning vol. 9, pp. 309 – 347 (1992). 20, 21

[18] Cormen, T.H., Leiserson, C.E., Rivest, R. L., Stein, C. Introduction to Algorithms (2nd ed.), MIT Press and McGraw-Hill (2001). 5

[19] Dagum, P., Luby, M. Approximating probabilistic inference in Bayesian belief networks is NP–hard, Artificial Intelligence, 60(1):141–153 (1993). 15

[20] Dagum, P., Luby, M. An optimal approximation algorithm for bayesian inference, Artificial Intelligence, 93(1–2):1–27 (1997). 16

[21] Dasgupta, S. Learning polytrees, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, pages 131–141, Morgan Kaufmann (1999). 18

[22] Demirer, R., Charnes, J. M., Kellogg, D. Influence Diagrams for Real Options Valuation, Journal of Finance Case Research 9, No. 1, pp. 43–70 (2007). 1

[23] Fung, R., and Chang, K.-C. Weighting and integrating evidence for stochastic simulation in Bayesian networks, Uncertainty in Artificial Intelligence 5, pp. 209–219 (1990). 16

[24] GEMAN, S., GEMAN, D. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, IEEE Trans. Pattern Analysis and Machine Intelligence 6; pp. 721–741 (1984). 26

[25] GREEN, P. J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, Biometrika 82, 711–732 (1995). 2

[26] HALILOGLU, T., BAHAR, I., ERMAN, B. Gaussian dynamics of folded proteins, *Physical Review Letters 79*, pp. 3090–3093, (1997). 24

[27] HECKERMAN, D. A Tutorial on Learning with Bayesian Networks, In Learning in Graphical Models, (1999). 17

[28] HENRION, M. Propagating uncertainty in Bayesian networks by probabilistic logic sampling, In Uncertainty in Artificial Intelligence 2, pp. 149–163 (1988). 16

[29] HEYMANN, M., VANDEN-EIJNDEN, E. The Geometric Minimum Action Method: A Least Action Principle on the Space of Curves, *Comm. Pure Appl. Math.***61.8**, 1052–1117 (2008). 31

[30] HOLLAND, J.H. Adaptation in Natural and Artificial Systems, University of Michigan Press (1975). 5

[31] HORVITZ, E., BARRY, M. Display of Information for Time-Critical Decision Making, Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence, Montreal, pp. 296–305 (1995). 1

[32] HOWARD, R.A., MATHESON, J.E. Influence diagrams, Readings on the Principles and Applications of Decision Analysis II, pp. 719–762; Reprinted: Decision Anal. 2(3), pp. 127–143 (1981(1984)/2005). 21

[33] JAIN, V., BOLLMANN, B., RICHARDSON, M., BERGER, D., ET AL. Boundary learning by optimization with topological constraints In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2010). 31

[34] JAIN, V., SEUNG, H.S., TURAGA, S.C. Machines that learn to segment images: a crucial technology for connectomics, Current Opinion in Neurobiology, Vol. 20, Issue 5, pp. 653666 (2010). 2

[35] JENSEN, F.V., NIELSEN, T.D. Bayesian Networks and Decision Graphs, 2nd Ed., *Springer* (2007). 16, 17, 18, 21, 23

[36] KIM, J. H., PEARL, J. A computational model for causal and diagnostic reasoning in inference systems, In Proceedings of the IJCAI-83, 1983. 15

[37] KIRK, D.B., HWU, W.-M.W. Programming Massively Parallel Processors: A Hands–on Approach, *Morgan Kaufmann*, 1 edition (2010). 31

[38] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. Optimization by Simulated Annealing, *Science 220*, pp. 671-680 (1983). 5

[39] Kluszczyński, R., Lieshout, M.N.M. van, Schreiber, T. An algorithm for binary image segmentation using polygonal Markov fields. In: F. Roli and S. Vitulano (Eds.), Image Analysis and Processing, Proceedings of the 13th International Conference on Image Analysis and Processing. *Lecture Notes in Comput. Sci.* **3615**, 383-390 (2005). 26

[40] Kluszczyński, R., Lieshout, M.N.M. van, Schreiber, T. Image segmentation by polygonal Markov fields. *Ann. Inst. Statist. Math.*, **59**, 465-486 (2007). 26, 31

[41] Koller, D., Friedman, N. Probabilistic Graphical Models: Principles and Techniques The MIT Press; 1 edition (2009). 7, 13, 17, 18, 23, 24, 26

[42] Koller, D., Milch, B. Multi–Agent Influence Diagrams for Representing and Solving Games, Games and Economic Behavior, 45(1), pp. 181–221 (2003). 1

[43] Korb, K. B., Nicholson, A. E. Bayesian Artificial Intelligence, Computer Science and Data Analysis, Chapmanand Hall/CRC (2004). 15, 16

[44] Koski, T., Noble, J. Bayesian Networks: An Introduction, *John Wiley & Sons, Ltd* (2009). 13, 14, 16, 17, 18, 19

[45] Koski, T.J.T., Noble, J.M. A Review of Bayesian Networks and Structure Learning, Mathematica Applicanda, Vol. 40(1) pp. 53–103 (2012). 18, 20, 21

[46] Kruskal, J. B. On the shortest spanning subtree of a graph and the Traveling Salesman Problem, Proceedings of the American Mathematical Society, 7, pp. 48–50 (1956). 20

[47] Kullback, S. Information Theory and Statistics, John Wiley & Sons (1959). 18

[48] Larie, S.M., Abukmeil, S.S. Brain abnormality in schizophrenia: a systematic and quantitative review of volumetric magnetic resonance imaging studies, J. Psych., 172:110–120 (1998). 2

[49] Lauritzen, S. L., Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems, Journal of the Royal Statistical Society, Series B, 50(2):157–224 1988. 15

[50] Matuszak, M., Międkisz, J., Schreiber, T. Solving Ramified Optimal Transport Problem in the Bayesian Influence Diagram Framework, ICAISC 2012, Part II, LNCS 7268, pp. 582–590 (2012). 3

[51] Matuszak, M., Miękisz, J., Schreiber, T. Smooth Conditional Transition Paths in Dynamical Gaussian Networks, KI 2011: Advances in Artificial Intelligence, LNAI 7006, pp. 204–215 (2011). 3

[52] Matuszak, M., Miękisz, J. Stochastic Techniques in Influence Diagrams for Learning Bayesian Network Structure, ICANN 2012, Part I, LNCS 7552, pp. 33–40 (2012). 3

[53] Matuszak, M., Schreiber, T. A new stochastic algorithm for strategy optimisation in Bayesian influence diagrams, LNAI 6114, pp. 574–581 (2010). 2, 3

[54] Matuszak, M., Schreiber, T. GPU Accelerated Smooth Formation Redeployment in Multiagent Environment, Mathematical Methods in Modelling and Analysis of Concurrent Systems, (2011). 31

[55] Matuszak, M., Schreiber, T. Locally specified polygonal Markov fields for image segmentation, Mathematical Methods for Signal and Image Analysis and Representation, Series: Computational Imaging and Vision, Vol. 41 (2012). 3, 31

[56] Meyer, J., Phillips, M. H., Cho, P. S , Kalet, I., Doctor, J. N. Application of influence diagrams to prostate intensity-modulated radiation therapy plan selection, Phys Med Biol 49: 9., pp. 1637–1653 (2004). 1

[57] Moller, J., Skare, O. Bayesian image analysis with coloured Voronoi tessellations and a view to applications in reservoir modelling, Stat. Modelling 1, 213–232, (2001). 2

[58] Neapolitan, R. E. Learning Bayesian Networks, *Prentice Hall Series in Artificial Intelligence*, *Pearson Prentice Hall* (2004). 17, 18, 23

[59] Nicholls, G.K. Bayesian image analysis with Markov chain Monte Carlo and coloured continuum triangulation models. J. Roy. Statist. Soc. Ser. B Statist. Methodol. 60, 643–659, (1998). 2

[60] Norman, J., Shahar, Y., Gold B. Decision-theoretic analysis of prenatal testing strategies, Technical Report SMI-98-0711, Stanford University, Section on Medical Informatics (1998). 1

[61] Pal, N.R., Pal, S.K. A review on image segmentation techniques, Pattern Recognition, Vol. 26, Issue 9, pp. 1277–1294 (1993). 2

[62] Pearl, J. Fusion, propagation, and structuring in belief networks, Artificial Intelligence, Vol. 29, Issue 3, pp. 241–288, (1986). 8

[63] Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, *Morgan Kaufmann Publishers Inc.* (1988). 5, 10, 15, 21

[64] Peretto, P. An Introduction to the Modeling of Neural Networks, *Collection Aléa-Saclay, Cambridge University Press* (1992). 6

[65] Pham, D.L., Xu, C., Prince, J.L. A Survey of Current Methods in Medical Image Segmentation, Annual Review of Biomedical Engineering, Volume 2, pp.315–337 (2000). 2

[66] Reynolds, C. W. Flocks, Herds, and Schools: A Distributed Behavioral Model, Computer Graphics, 21(4) (SIGGRAPH '87), pp. 25–34 (1987). 2

[67] Robinson, R.W. Counting Unlabelled Acyclic Digraphs, Springer Lecture Notes in Mathematics, Combinatorial Mathematics V, pp. 28 – 43 (1977). 21

[68] Rosenfeld, A., Kak, A. C. Digital picture processing, 2nd edn, Vol. 2. Orlando: Academic Press (1982). 2

[69] Rue, H., Held, L. Gaussian Markov Random Fields: Theory and Applications, *Monographs on Statistics & Applied Probability 104*, (2005). 24, 25

[70] Safonova, A., Hodgins, J.K. Analyzing the physical correctness of interpolated human motion, ACM Siggraph/Eurographics Symposium on Computer Animation (SCA '05), 171–180 (2005). 2

[71] Scharf, D.P., Hadaegh, F.Y. and Ploen, S.R. A Survey of Spacecraft Formation Flying Guidance and Control (Part I): Guidance, *American Control Conference, Vol. 2*, pp. 1733 – 1739, (June 2003). 2

[72] Scharf, D.P., Hadaegh, F.Y. and Ploen, S.R. A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control, *American Control Conference, Vol. 4*, pp. 2976–2985, (30 June – 2 July 2004). 2

[73] Schreiber, T. Random dynamics and thermodynamic limits for polygonal Markov fields in the plane, *Advances in Applied Probability* **37**, 884-907 (2005). 27

[74] Schreiber, T., Lieshout, M.N.M. van Disagreement loop and path creation/annihilation algorithms for binary planar Markov fields with applications to image segmentation, *Scand. J. Stat., Vol. 37 no. 2*, 264-285 (2010). 31

[75] Sengoku, H., Yoshihara I. A fast TSP solver using GA on JAVA, AROB III 98 Japan, pp. 283-288 (1998). 6

[76] Shachter, R.D., Kenley, C.R. Gaussian influence diagrams *Management Science*, 35(5), pp. 527–550, (1989). 23, 24

[77] Shachter, R. D., and Peot, M. A. Simulation approaches to general probabilistic inference on belief networks, Uncertainty in Artificial Intelligence 5, pp. 221–231 (1990). 16

[78] Shachter, R.D. Bayes Ball: The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams), Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence, pp. 480–487 (1998). 12

[79] Shalizi, C.R. Advanced Data Analysis from an Elementary Point of View, Carnegie Mellon (2012). 19

[80] Spirtes, P., Glymour, C., and Scheines, R. An algorithm for fast recovery of sparse causal graphs, Social Science Computer Review, vol. 9, pp. 62–72, (1991). 19

[81] Spirtes, P., Glymour, C., Scheines, R. Causation, Prediction, and Search, 1st edn., Berlin: Springer-Verlag (1993). 19

[82] Suermondt, H., Cooper, G., Heckerman, D. A combination of cutset conditioning with clique-tree propagation in the Pathfinder system, Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, Boston, pp. 273–279 (1990). 1

[83] Surgailis, D. Thermodynamic limit of polygonal models, *Acta applicandae mathematicae*, **22**, 77-102 (1991). 26

[84] Unnikrishnan, R., Pantofaru, C., Hebert, M. Toward objective evaluation of image segmentation algorithms, IEEE Trans Pattern Anal Mach Intell, Vol. 29, pp. 929–944 (2007). 31

[85] U.S. Marine Corps Marine Rifle Squad, *Marine Corps Warfighting Publication (MCWP)* 3–11.2, (1997). 2

[86] Worth, A.J., Makris, N., Caviness, V.S., Kennedy, D.N. Neuroanatomical segmentation in MRI: technological objectives, Int. J. Patt. Rec. Art. Intel., 11:1161–1187 (1997). 2

[87] Yang, F., Jiang, T. Pixon-Based Image Segmentation With Markov Random Fields, IEEE Transactions on Image Processing, Vol. 12, No. 12 (2003). 2

[88] Xia, Q. Ramified optimal transportation in geodesic metric spaces, *Adv. Calc. Var. Volume 4, Issue 3*, pp. 277–307 (2011). 1

[89] Zijdenbos, A.P., Dawant, B.M. Brain segmentation and white matter lesion detection in MR images, Critical Reviews in Biomedical Engineering, 22:401–465 (1994). 2

Appendix A

Graph Theory

This section provides a short review of basic definitions in the theory of graphs.

Definition A.0.1 (Graph) *A graph is an ordered pair $G = (V, E)$, where V is a finite set of distinct vertices and E is a set of edges i.e. $E \subseteq V \times V$.*

For distinct nodes $u, v \in V$, an ordered pair $(u, v) \in E$ defines a direct edge from node u to node v. We denote this as $u \to v$. If $(u, v) \in E$ and $(v, u) \in E$, therefore, the edge between node u and v is undirected and we denote it as $\langle u, v \rangle \in E$ or $u - v$.

In this text, we will consider only *simple graphs*, i.e. graphs in which each edge is a distinct pair of vertices (loops are forbidden) and there is at most one edge between any two vertices. We will count an undirected edge as a single edge.

Definition A.0.2 (Directed (Undirected) Graph) *Graph $G = (V, E)$ is a* directed (undirected) graph *if E does not contain undirected (directed) edges.*

Definition A.0.3 (Parent, Children, Neighbour) *For a directed graph $G = (V, E)$ and $v \in V$ we define the following functions:*

- *a set of parents as*
$$pa(v) = \{w \in V : w \to v\}, \tag{A.1}$$
- *a set of children as*
$$ch(v) = \{w \in V : v \to w\}, \tag{A.2}$$
- *a set of neighbours as*
$$ng(v) = pa(v) \cup ch(v). \tag{A.3}$$

For undirected graph $G = (V, E)$ and $v \in V$ we can define a set of neighbours *as*

$$ng(v) = \{w \in V : v - w\}. \tag{A.4}$$

Definition A.0.4 (Trail) *A **trail** between nodes $u, v \in V$ is a sequence of distinct nodes $(v_1, \dots, v_n)$, where $v_i \in V$ for $i = 1, \dots, n$, $v_1 = u$, $v_n = v$ and $(v_k, v_{k+1}) \in E$ or $(v_{k+1}, v_k) \in E$ for $k = 1, \dots, n-1$.*

Definition A.0.5 (Connected Graph) *Graph $G = (V, E)$ is **connected** if for every $u, v \in V$ there is a trail between u and v.*

Definition A.0.6 (Path) *A **path** between nodes $u, v \in V$ is a sequence of distinct nodes $(v_1, \dots, v_n)$, where $v_i \in V$ for $i = 1, \dots, n$, $v_1 = u$, $v_n = v$ and $(v_k, v_{k+1}) \in E$ or $\langle v_k, v_{k+1} \rangle \in E$ for $k = 1, \dots, n-1$.*

Definition A.0.7 (Directed Path) *A **directed path** between nodes $u, v \in V$ is a path that consists only of directed edges.*

Definition A.0.8 (Cycle) *A **cycle** in $G = (V, E)$ is a directed path $(v_1, \dots, v_n)$, where $v_i \in V$ for $i = 1, \dots, n$ and $v_1 = v_n$.*

Definition A.0.9 (Acyclic Graph) *Graph $G = (V, E)$ is **acyclic** if it contains no cycles.*

Definition A.0.10 (Directed Acyclic Graph (DAG)) *If graph $G = (V, E)$ is a* directed graph *and if G satisfy acyclic property, then G is called* Directed Acyclic Graph *or just* DAG.

Definition A.0.11 (Moral graph) *Let $G = (V, E)$ be a DAG. The **moral graph** $G^M = (V, U)$ is an undirected graph with the same set of vertices as G and with a set of undirected edges U that contains edges $X - Y$, for $X, Y \in V$, if either $X \to Y$, $Y \to X$ in G or if X and Y are parents of the same node.*

Definition A.0.12 (Triangulated graph, chord) *An undirected graph $G = (V, U)$ is **triangulated** if for every cycle of a length greater than 3 exist an edge (called a **chord**) between two non–consecutive vertices from the cycle.*

Papers included in the PhD thesis

Paper A

Matuszak, M., Schreiber, T.
A new stochastic algorithm for strategy optimisation in Bayesian influence diagrams

A New Stochastic Algorithm for Strategy Optimisation in Bayesian Influence Diagrams

Michal Matuszak & Tomasz Schreiber

Faculty of Mathematics and Computer Science,
Nicolaus Copernicus University,
Toruń, Poland
{gruby, tomeks}@mat.umk.pl

Abstract. The problem of solving general Bayesian influence diagrams is well known to be NP-complete, whence looking for efficient approximate stochastic techniques yielding suboptimal solutions in reasonable time is well justified. The purpose of this paper is to propose a new stochastic algorithm for strategy optimisation in Bayesian influence diagrams. The underlying idea is an extension of that presented in [2] by Chen who developed a self-annealing algorithm for optimal tour generation in traveling salesman problems (TSP). Our algorithm generates optimal decision strategies by iterative self-annealing reinforced search procedure, gradually acquiring new information while driven by information already acquired. The effectiveness of our method has been tested on computer-generated examples.

1 Introduction

Influence diagrams[4–6] are widely acknowledged as an important probabilistically oriented graphical representation paradigm for decision problems. An influence diagram is built on a directed acyclic graph (DAG) whose nodes and arcs admit standard interpretations steming from and extending those used for Bayesian (belief) networks. Three principal types of nodes are considered: chance nodes standing for random variables (represented as ovals in our figures below), decision nodes corresponding to available decisions (rectangles in our figures) and utility nodes (rhombi) specifying the utilities to be maximized by suitable choices of decision policies. The arcs leading to chance nodes indicate direct causal relationships (at least if the network is well designed) not necessarily corresponding to any temporal ordering. On the other hand, the arcs leading to decision nodes specify the information available at the moment of decision making, thus feeding input to decision policies. Some arcs between decision nodes may also be of informative nature as determining the order of decision making. The influence diagrams can be considered as a generalization of (symmetric) decision trees, see [4].

In Fig. 1, generated by Hugin Lite package [3], a simple example of an influence diagram is shown. The decision node *treatment* represents the choice whether or not to visit a doctor. A visit to a doctor does increase the chance of no cough, yet it also causes negative effects, such as the need to pay the fee for visit. Further, wearing a *scarf* decreases the chance of getting sore throat but also negatively affects our appearence. All these effects are jointly taken into account in the utility node *happiness*.

A number of algorithms for solving influence diagrams have been developed, falling beyond the scope of the present article. We refer the reader to Chapter 10 in [4] for a detailed discussion, see also the references in Subsection 5.2.2 of [5].

In general, finding an optimal decision strategy for an influence diagram is an NP-hard task. This is easily shown by reducing an NP-complete problem to the considered task. A natural choice is the traveling salesman problem (TSP) known as a classical NP-complete task. To each city a decision node is ascribed with the remaining cities as admissible states. The decision taken coincides with the next city to visit. Further, a utility node is created with incoming arcs from all decision nodes. The utility function is defined by summing up the negative distances between cities and their successors in case the decision sequence yields a valid Hamilton tour, and is set to $-\infty$ otherwise. Clearly, solving the TSP problem is, in this set-up, equivalent to finding the optimal decision strategy for the so-constructed influence diagram.

In his work [2] (see also the discussion in [7]) Chen proposed an appealing and simple stochastic optimisation algorithm for the TSP problem, quite original in its design, highly effective and yet apparently somewhat underestimated in the literature. In the course of an iterative procedure subsequent TSP tours are randomly generated: each city is assigned a table of weights for connections to all remaining cities and each time the choice of the next city to visit is made by random among cities not yet visited, with probabilities proportional to the corresponding connection weights. This way all cities get visited and eventually we get back to the starting point. Next, the so generated tour is compared with the one obtained in the previous iteration. Depending on whether the new cycle is longer or shorter than the previous one, the connection weights between cities neighbouring in both tours are correspondingly reinforced or faded. The algorithm stops when the re-normalised weights are close enough to zeros and ones, which corresponds to a deterministic tour choice, converging to the optimal one under suitable reinforcement/fading protocols.

With the TSP problem regarded as a particular case of decision strategy optimisation, the purpose of the present paper is to is to extend Chen's approach to general influence diagrams. As we will see, this can be done neatly and effectively, although not without substantial extensions of Chen's idea.

2 The algorithm

To give a formal description of the proposed decision strategy optimisation algorithm, assume an influence diagram $(\mathcal{S}, \mathcal{P}, \mathcal{U})$ is given, built on a connected DAG $\mathcal{S}$, with CPTs $\mathcal{P}$ and utility functions $\mathcal{U}$. The set of nodes in $\mathcal{S}$ splits into chance nodes $C_{\mathcal{S}}$, decision nodes $D_{\mathcal{S}}$ and utility nodes $U_{\mathcal{S}}$. All these objects are

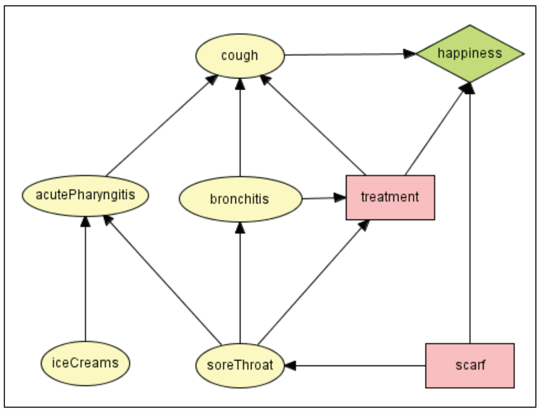


Fig. 1. Sample influence diagram.

assumed fixed in the course of strategy optimisation. In addition, for optmisation purposes we attach to each decision node $D \in D_{\mathcal{S}}$ the *randomised policy* δ_D which assigns to each configuration $\bar{w}$ of $\mathrm{pa}(D)$ a probability distribution on possible decisions to be taken, that is to say $\delta_D(d|\bar{w})$ stands for the probability of choosing decision d given that $\mathrm{pa}(D) = \bar{w}$. These randomised policies will evolve in the course of the optimisation process, eventually to become (sub)optimal deterministic policies collectively determining the utility maximizing strategy for the influence diagram considered. The initial choice of $\delta_D, \ D \in D_{\mathcal{C}}$ can be either *uniform*, with all decisions equiprobable, or *heuristic* provided some additional knowledge is available on our system allowing us to make a good *first guess* about the optimal strategy.

Our algorithm relies on an iterative procedure carried out in *epochs* of fixed length N. At the beginning of each epoch $t, \ t = 1, 2, \ldots$ we divide the collection $D_{\mathcal{C}}$ of decision nodes into the set of *trainable decision nodes* $TD^t_{\mathcal{C}}$ whose randomised decision policies are to undergo updates, and *frozen decision nodes* $FD^t_{\mathcal{C}}$ whose status is to remain unaltered throughout the epoch. Roughly speaking, the purpose of this splitting is to ensure that only a modest fraction of system parameters are updated at a time, which is indispensible for the stability of algorithm, see below for a more detailed discussion. In addition, we impose the following requirement, whose relevance is discussed in the sequel.

[**Forbidden path condition**] No two trainable nodes are connected by a directed path in $\mathcal{S}$.

With $\mathcal{D} \subseteq D_{\mathcal{S}}$ write $A[\mathcal{D}]$ for the ancestry of $\mathcal{D}$ in $\mathcal{S}$ that is to say $A[\mathcal{D}]$ is the collection of nodes in $\mathcal{S}$ from which a decision node from $\mathcal{D}$ can be reached along a directed path in $\mathcal{S}$. Clearly, the forbidden path condition is equivalent to requiring that $A[TD^t_{\mathcal{S}}] \cap TD^t_{\mathcal{S}} = \emptyset$ during tth epoch. Moreover, since the utility nodes have no progeny, we readily conclude that $A[TD^t_{\mathcal{S}}] \subseteq C_{\mathcal{S}} \cup FD^t_{\mathcal{S}}$.

A standard way of selecting the trainable collection, as implemented in our software, goes as follows.

– Whenever passing to a new epoch, do sequentially for all decision nodes $D \in D_{\mathcal{C}}$
  - If D is a trainable node then freeze it with some probability p_F,
  - If D is a frozen node, make it trainable with some probability $p_T < p_F$ unless doing so violates the forbidden path condition and unless the fraction of time during which D was trainable exceeds maximal admissible value (specified as algorithm parameter).

Note that the *quota* imposed on the fraction of time a given node is trainable is aimed at preventing the situation where a decision node with numerous progeny in $D_{\mathcal{C}}$ receives only a very poor training time fraction as predominantly blocked by its progeny.

The following iterative procedure, repeated a fixed number N of times during each optimisation epoch, say tth epoch, and directly motivated by the ideas developed in [2], lies at the very heart of our algorithm.

1. Set the iteration counter $i := 0$.
2. Generate an instance $\bar{w}_A^{(i)}$ of the *ancestral variable configuration* for $A[TD^t_{\mathcal{S}}] \subseteq C_{\mathcal{S}} \cup FD^t_{\mathcal{S}}$ according to the CPTs of chance nodes and using the randomised decision policies of frozen decision nodes in $FD^t_{\mathcal{S}} \cap A[TD^t_{\mathcal{S}}]$ as CPTs. This is carried out in the standard way with nodes handled recursively proceeding from causes to effects in the policy subnetwork $A[TD^t_{\mathcal{S}}]$. This is where the forbidden path condition is of use as ensuring that no trainable node falls into $A[TD^t_{\mathcal{S}}]$.
3. Repeat a fixed number M of times
   (a) Sample the decisions taken, $d_1, \ldots, d_k$, independently for all trainable decision nodes $D_1, \ldots, D_k \in TD^t_{\mathcal{S}}$ according to their respective current randomised policies $\delta_{D_j},\ j = 1, \ldots, k$ given $\bar{w}_A^{(i)}$. Note that $\bigcup_{j=1}^k \mathrm{pa}(D_j) \subseteq A[TD^t_{\mathcal{S}}]$ and thus the knowledge of $\bar{w}_A^{(i)}$ is sufficient for this sampling.
   (b) Evaluate the expected total utility
$$u^{(i)} := \mathbb{E}[U_{\text{total}} | \bar{w}_A^{(i)}; d_1, \ldots, d_k]$$
   given $\bar{w}_A^{(i)}$ and $d_1, \ldots, d_k$, under the randomised policies of frozen nodes used as respective CPTs. This is easily done by standard Monte-Carlo network instance generating scheme, recursively proceeding from causes to effects. This is possible because the non-instantiated part of the network $\mathcal{S} \setminus [A[TD^t_{\mathcal{S}}] \cup TD^t_{\mathcal{S}}]$ contains no trainable decision nodes and it has the upward cone property – whenever it contains a node X it also contains all its children and, inductively, its whole progeny.
   (c) If $i \geq 1$, set
$$\varDelta := u^{(i)} - u^{(i-1)}$$

and update the policies $\delta_{D_j},\ j = 1, \ldots, k$ for trainable nodes by putting

$$\delta_{D_j}(d_j | \bar{w}_A^{(i)} \cap \mathrm{pa}(D_j)) := \exp(\beta \varDelta) \delta_{D_j}(d_j | \bar{w}_A^{(i)} \cap \mathrm{pa}(D_j))$$

and, thereupon, re-normalising $\delta_{D_j}(\cdot | \bar{w}_A^{(i)} \cap \mathrm{pa}(D_j))$ so that it remain a probability distribution. The positive constant β here is a parameter of the algorithm, the larger it is the faster the system learns but the less stable the optimisation is.

4. Set $i := i + 1$ and, if $i < N$, return to 1. Otherwise terminate the current epoch.

The intuitive meaning of the above procedure is that the network is presented with a configuration sampled according to the CPTs and current randomised policies of the frozen nodes, whereupon the randomised policies of the trainable nodes are used for decision sampling, with succesful choices (positive $\varDelta$) leading to reinforcement of the corresponding probability entries and, on the other hand, with choices deteriorating the performance resulting in fading of the corresponding probability entries (negative $\varDelta$). The reinforcement/fading strength depends on the value of the utility gain/loss compared to the previous run. In analogy to Chen's work [2], also here after a large enough number of epochs we eventually end up with the situation where all randomised policies become nearly deterministic in that, for each $D \in D_{\mathcal{C}}$ and parent configuration $\bar{w}$ for $\mathrm{pa}(D)$ the value of $\delta_D(d|\bar{w})$ is close to one for a unique d and close to zero otherwise. This determinism can be easily quantified by looking at the maximal value of $\min(\delta_D(d|\bar{w}), 1 - \delta_D(d|\bar{w}))$ and declaring a policy *nearly deterministic* when this falls below, say, 0.01. To sum up, our algorithm carries out subsequent optimisation epochs until all policies become nearly deterministic. Note in this context that is is crucial to ensure that each node is trainable during a sufficiently large fraction of time, for otherwise it might long remain untrained slowing down the entire process. As already mentioned, this is handled by our training selection scheme discussed above.

3 Implementation and examples

The programme has been implemented in language `D` [8], currently gaining popularity as a natural successor to `C++`, and uses the Tango library [1]. The implementation, aimed so far mainly at algorithm evaluation purposes, can be described as careful but not fully performance-optimised, with the total utility evaluation under frozen decision nodes in 3.(b) performed using the standard Monte-Carlo rather than a more refined and effective scheme. The graph of the diagram was represented using neighbourhood lists. The utility functions were always fit to $[0, 1]$. All test runs were executed on a machine with Intel Core 2 Q9300 2.50 GHz CPU and 2GB RAM.

A sequential version of our algorithm run for a randomly generated influence diagram with 10 chance nodes, 10 decision nodes, 10 utility nodes and 40 arcs, see Fig. 3 generated from Hugin Lite [3], required 1.1ms time per epoch. The algorithm parameters were set as follows: $p_T = 0.01$, $p_F = 0.04$ and $\beta = 0.01$. After 10000 epochs (11 seconds) the strategy output by our algorithm achieved utility only 4% inferior to the optimal one (as determined using the Hugin package). For a different randomly generated network (doubled number of nodes) execution time per one epoch was 1.8ms, with 10000 epochs. Again, the utility of the output strategy was only 4% inferior to the optimal one.

Getting back to the network depicted in Fig. 1, we performed 10000 iterations of our algorithm, with parameters $p_T = 0.01$, $p_F = 0.04$ and $\beta = 0.03$. The convergence of decision policies for node *treatment* in the course of our algorithm is shown in Fig. 3. The table 3 represents the policy obtained upon convergence of the algorithm. It can be concluded that the sore throat is of crucial importance for our decision whether or not to visit a doctor. On the other hand, the bronchitis appears to be ignored. This does coincide with the optimal strategy as determined by the Hugin Lite package.

Table 1. Decision policy for node *treatment*.

Y	N	
1.00	0.00	sore throat = Y, bronchitis = Y
0.99	0.01	sore throat = Y, bronchitis = N
0.03	0.97	sore throat = N, bronchitis = Y
0.01	0.99	sore throat = N, bronchitis = N

For the diagram given in Fig. 3 (12 chance nodes, 6 decision nodes, one utility node and 42 arcs) the mean execution time per epoch was 0.8ms. The decision strategy after 10000 iterations with parameters $p_T = 0.01$, $p_F = 0.04$ and $\beta = 0.01$ was inferior by 5 % to the optimal strategy.

References

1. Bell, K., Igesund, L.I., Kelly, S., Parker, M. Learn to Tango with D, *Apress*, 2008.
2. Chen, K. Simple learning algorithm for the traveling salesman problem, *Phys. Rev. E* **55**, 7809-7812 (1997).
3. `http://www.hugin.com`
4. Jensen, F.V., Nielsen, T.D. Bayesian Networks and Decision Graphs, 2nd Ed., *Springer*, 2007.
5. Neapolitan, R. E. Learning Bayesian Networks, *Prentice Hall Series in Artificial Intelligence*, *Pearson Prentice Hall*, 2004.
6. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, *Morgan Kaufmann Publishers Inc.*, 1988.
7. Peretto, P. An Introduction to the Modeling of Neural Networks, *Collection Aléa-Saclay*, *Cambridge University Press*, 1992.
8. `http://www.digitalmars.com/d/`

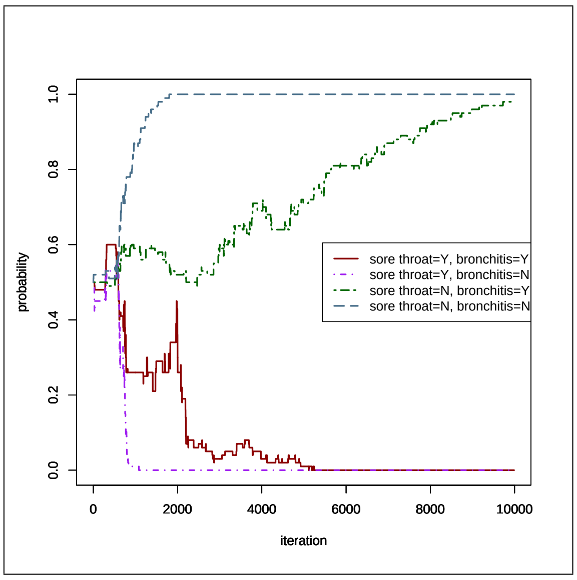


Fig. 2. Convergence of decision policies.

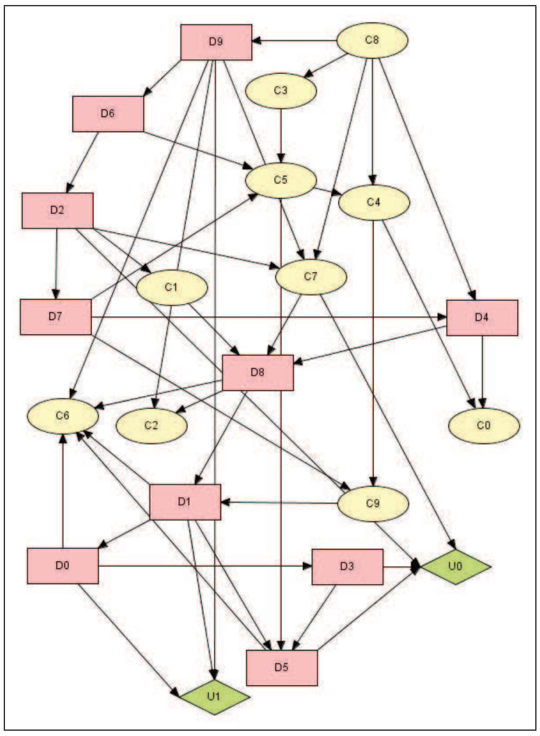


Fig. 3. Randomly generated influence diagram.

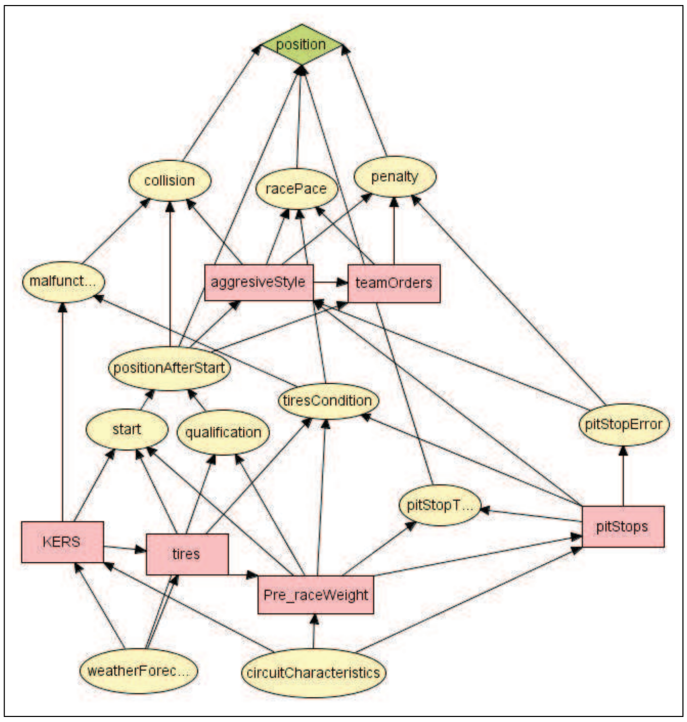


Fig. 4. Race results.

Paper B

Matuszak, M., Miękisz, J., Schreiber, T.

Smooth Conditional Transition Paths in Dynamical Gaussian Networks

Smooth Conditional Transition Paths in Dynamical Gaussian Networks

Micha Matuszak[1], Jacek Miękisz[2], and Tomasz Schreiber[1,⋆]

[1] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopina 12/18, 87–100 Torun, Poland
{gruby,tomeks}@mat.umk.pl

[2] Institute of Applied Mathematics and Mechanics, University of Warsaw, Banacha 2, 02–097 Warsaw, Poland
miekisz@mimuw.edu.pl

Abstract. We propose an algorithm for determining optimal transition paths between given configurations of systems consisting of many objects. It is based on the Principle of Least Action and variational equations for Freidlin–Wentzell action functionals in Gaussian networks set-up. We use our method to construct a system controlling motion and redeployment between unit's formations. Another application of the algorithm allows a realistic transformation between two sequences of character animations in a virtual environment. The efficiency of the algorithm has been evaluated in a simple sandbox environment implemented with the use of the NVIDIA CUDA technology.

Keywords: Formation Redeployment, Animation Blending, Transition Path, Reconfiguration, CUDA.

1 Introduction

Simulations of moving groups of agents that preserve their motoric characterizations play an important role across a broad spectrum of applications. Observation of biological systems initiated works on coordination among multiple agents. In a pioneering work [15], a computer model was constructed for synchronized animal motion observed for example in bird flocks or fish schools. It is important to emphasize that motion of individual units was calculated only on the basis of their local environment. In military applications [1], formations allow for a more effective use of limited resources, such as sensors, by division of the environment into portions so each formation's member can focus attention on an assigned segment while the rest is covered by the partners. This mechanism is used for example by groups of fighter pilots to optimize the usage of their radars and visual perception. Such an approach can also be applied to spacecrafts in a deep space or in the Earth orbit, see survey papers [20,21] for a comprehensive description. We focus our attention on the problem of a reconfiguration of formations.

⋆ Deceased author (1975 – 2010).

Our method can be used to reduce casualties from a hostile fire, to present less vulnerable targets or to evacuate from an exposed area [22]. Other applications involve parking systems that smoothly drive cars in and out from a parking lot. Finally, the entertainment industry may use our algorithm in real-time computer strategies.

The character animation is undoubtedly an element of the computer graphics, which is of a great importance for enjoyable computer games, credible CG movies or medical visualizations. One of the most popular techniques for generating realistic motion in virtual environments is a skeletal animation technique [4]. Skeletal systems are hierarchical in nature and provide the artist with a control of the character in an efficient manner. An animator can focus his attention on the motion of a simplified structure (the skeleton) rather than manually alter the geometry (character's meshes) itself. For smooth animations it is crucial to generate smooth transitions between system's configurations. Such transitions can be generated by mixing existing ones. One of the available methods is an interpolation of motion data, which has been shown to be a powerful technique if changes between interpolated classes meet predefined/given constraints [18]. The algorithm presented below attempts to address that drawback and provides methodology to produce optimal transition paths between arbitrary configurations.

One of the approaches to describe motion of systems of interacting particles is based on a variational principle. It is assumed in classical mechanics that the trajectory of a system between two points in the space minimizes the action functional. Then by a variational calculus one obtains Euler-Lagrange equations of motion. Reformulation of such an approach to the case of the space of curves in a set-up well suited for our needs was presented in [8].

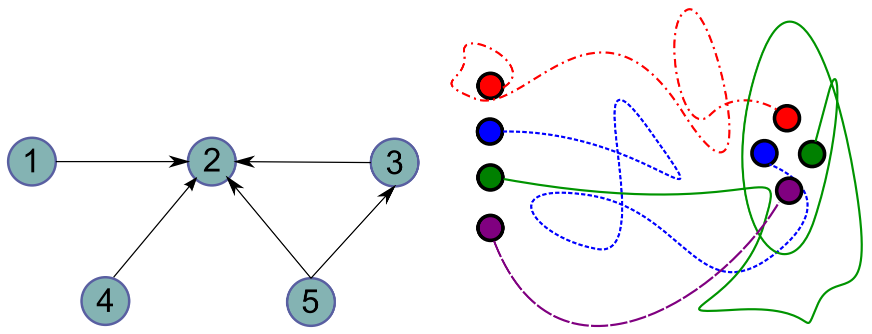
Here we use Gaussian networks. The term Gaussian network was introduced in [19] and describes an extension of a Bayesian network [12] to continuous variables. Gaussian networks are widely used for decision making and inference. In molecular biology they are used to describe protein dynamics [7]. Gaussian networks better characterize classes of behavior and provide better understanding than the standard representations [19]. To describe time evolution of Gaussian networks we use stochastic diffusion processes whose behavior is effectively characterized by the large deviation theorem due to Freidlin and Wentzell [6]. The theory is based on the property that very unlikely events, when they occur, do so with a high probability by following the pathway that is the most probable. Thus the rare events become in a sense predictable. The crucial role in the theory plays an action functional whose minimization produces an approximation of the probability of rare events and enables the computation of the maximum likelihood trajectory by which such an event occurs [8].

In Section 2, we outline the Principle of Least Action adapted to our needs and present our algorithm to simulate smooth optimal transition paths. Applications, implementation, and a discussion are contained in following sections.

2 Gaussian Network

Gaussian networks [17] are systems which consist of a finite number of nodes whose states are described by continuous variables (these might be positions of certain objects as in our examples). A state of each node is subject to a stochastic dynamics which can be decomposed into a deterministic drift and a stochastic part of the diffusion type. Both the drift and the stochastic part depend on the states of other variables (perhaps just neighboring ones in the spatial networks). We may think about such a dynamics as a continuous limit of a collection of random walks biased by states of other walkers.

In Fig. 1(a), a simple Gaussian network is presented. Arrows describe influence of neighboring nodes on a stochastic dynamics of a given node. More formally, interactions between nodes are contained in the function μ and the matrix Σ in Eq. (1) below. In Fig.1(b) we can observe transitions between stable space configurations of a Gaussian network.



(a) Gaussian Network (b) Transition

Fig. 1. (a) A schematic representation of nodes in a Gaussian network. (b) Without noise the system would stay in a stable configuration. With noise the system may leave the domain of attraction and can experience rare transitions between stable configurations.

Let $\psi(t)$ is the configuration of the system at time t. For a Gaussian network with n nodes it is a column vector in R^n which evolves according to

$$\psi(t + \Delta t) = \psi(t) + \mu(\psi(t))\Delta t + \sqrt{\epsilon}\Sigma\Delta W(t), \tag{1}$$

where:

- μ is called an instantaneous drift. We will assume that $\mu(\psi) = B\psi$, for a given $n \times n$ matrix B.
- $\Delta W(t)$ are independent normal random variables with the zero mean and the variance equal to Δt. and Σ is a $n \times n$ matrix which can introduce correlations between stochastic parts of time evolution of different nodes in Eq. (1). We can think about $\Sigma\Delta W(t)$ as the source of a noise.
- ϵ is called the instantaneous variance.

In continuous time, Eq.(1) can be written as the Ito stochastic differential equation

$$d\psi(t) = B\psi(t)dt + \sqrt{\epsilon}\Sigma dW(t) \tag{2}$$

We introduce a local steering contribution $\dot{w}$ as

$$\dot{w} = \dot{\psi} - B\psi$$

where by a dot we denote a derivative with respect to time t.

In the discrete case, $\Delta w = \sqrt{\epsilon}\Sigma\Delta W$. Hence $w(t)$ is called the full "error" or the fluctuation (deviation) of our system. During the minimization process, the value of $\sqrt{\epsilon}$ can be omitted (set to 1).

We propose the following Lagrange function which defines our system:

$$L(\psi, \dot{\psi}) := \frac{1}{2}\left(\dot{\psi} - B\psi\right)' A^{-1}\left(\dot{\psi} - B\psi\right),$$

where by an apostrophe we denote a transpose of a vector or a matrix and $A := \Sigma\Sigma'$.

The Principle of Least Action – of fundamental use for our applications – indicates that the system moves along the path which minimizes the action functional on the time interval $[0, T]$:

$$S(\psi) := \int_0^T L(\psi(t), \dot{\psi}(t))dt \tag{3}$$

Our construction is based on the Freidlin-Wentzell theorem [6] on large deviations in stochastic processes which roughly states that the probability of the trajectory $\bar{\psi}$ which deviates from the optimal one is proportional to $exp(-\frac{S(\bar{\psi})}{\epsilon})$.

To minimize the action functional we use the Euler-Lagrange differential equation (for the Lagrange function of a system of interacting particles we obtain in this way the Newton equations of motion),

$$\frac{\delta L}{\delta\psi} - \frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\psi}}\right) = 0. \tag{4}$$

For a symmetric matrix B one obtains

$$\frac{\delta L}{\delta\psi} = -\left(BA^{-1}\dot{w}(t)\right)$$
$$\frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\psi}}\right) = \left(A^{-1}\ddot{w}(t)\right)$$

and hence we get

$$\ddot{w}(t) = -ABA^{-1}\dot{w}(t) \tag{5}$$

We solve Eq. (2) as a system of linear ordinary differential equations along a fixed stochastic trajectory and get

$$\psi(T) = exp(TB)\psi(0) + \left[\int_0^T exp((T-s)B)\dot{w}(s)ds\right]$$

We know that $\dot{w}(s) = exp((-ABA^{-1})s)\dot{w}(0)$, so

$$\psi(T) = exp(TB)\psi(0) + \left[\int_0^T exp((T-s)B)exp(-sABA^{-1})ds\right]\dot{w}(0)$$

For the uncorrelated noise ($\Sigma = \mathbf{1}$) we can write:

$$\begin{aligned}\psi(T) &= exp(TB)\psi(0) + \left[\int_0^T exp((T-2s)B))ds\right]\dot{w}(0)\\ &= exp(TB)\psi(0) + exp(TB)\left[\int_0^T exp(-2sB)ds\right]\dot{w}(0)\\ &= exp(TB)\psi(0) + \frac{1}{2}exp(TB)B^{-1}\left[\mathbf{1} - exp(-2TB)\right]\dot{w}(0)\end{aligned}$$

From the above we get the initial steering configuration

$$\dot{w}(0) = 2B\left[\mathbf{1} - exp(-2TB)\right]^{-1}\left[exp(-TB)\psi(T) - \psi(0)\right] \tag{6}$$

The following procedure lies at the heart of the algorithm. The starting configuration $\psi(0)$ and matrix B must be given.

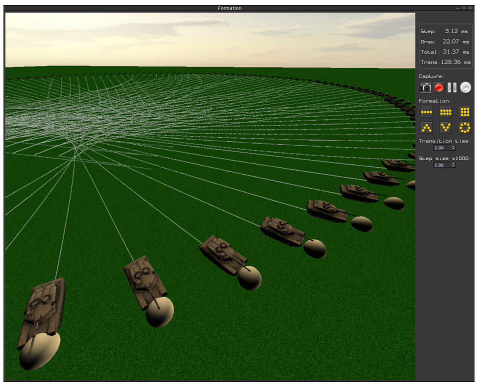
1. Set the timer $t := 0$.
2. If we do not initialize the 'force' transition to a new configuration, then
   (a) Use the rules given by Eq. 1 with $\epsilon = 1$ and $\Sigma = \mathbf{1}$
   (b) Set $t := t + \Delta t$.
3. If we initialize the 'force' transition to a new configuration and provide matrix B_1, then
   (a) Compute initial steering configuration for a given transition time T, given by Eq. 6 and denote it as $\dot{w}(t)$.
   (b) Set local timer $t1 := 0$.
   (c) Compute the new configuration
   $$\psi(t + \Delta t) = \psi(t) + (B\psi(t) + \dot{w}(t))\Delta t$$
   (d) Update the local steering contribution
   $$\dot{w}(t + \Delta t) = \dot{w}(t) - (B\dot{w}(t))\Delta t$$
   (e) Set $t1 := t1 + \Delta t$
   (f) Update global timer $t := t + \Delta t$ and, if $t1 < T$, return to 3c else set $B := B_1$ and terminate the transition stage.
4. Return to 2.

3 Formation Redeployment

Similarly to [20] we define a *formation* as a set of more than one unit, whose dynamic states are coupled through a common control law. That is, the members of the set must

- Track a desired state relative to a non–empty subset of other members
- The tracking control law must depend at least upon the state of this subset at the minimum.

The second point ensures that the motion of a unit is controlled not only with use of its individual state (position, velocity, etc.), but also is affected by the state of other units. Orbit correction algorithm of the GPS satellites only require position and velocity of an individual satellite thus they do not satisfy the second requirement. Several formations for a set of units are considered:


(a) square $\mapsto$ circle

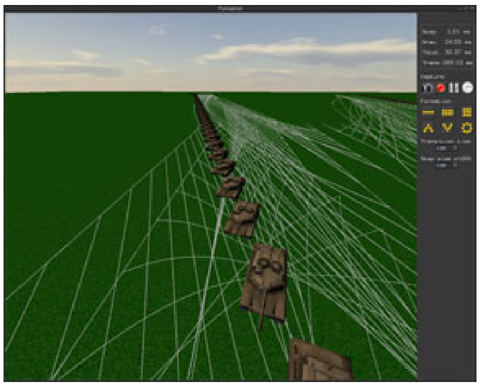

(b) wedge $\mapsto$ line

Fig. 2. Screenshots from the sample application. In (a) 225 units transit from the square formation to the circle formation. The current position of an agent is represented by a tank and the distance covered is denoted with a white trace. Similarly in (b) a transition occurs from wedge to line configuration.

- *line* - where the units move in a row
- *double line* - where the units move in a two parallel rows
- *square* - where the units are regularly distributed inside a square
- *circle* - where the units move on the edge of a circle
- *V* - where the units move in a "V" shape
- *wedge* - where the units move in a reverse "V" shape

Fig. 3 shows a schematic description of the formations i.e. every unit is represented by a dot and is connected to its spatial neighbours by edges which illustrate neighbour influence on the node state. The configuration is described as a position of each unit on the plane.

More formally the formation is represented by a Gaussian network with units as nodes and presented connections as arcs. For each pair of connected nodes $\mathfrak{x}$ and $\mathfrak{y}$ a pair of vectors is given:

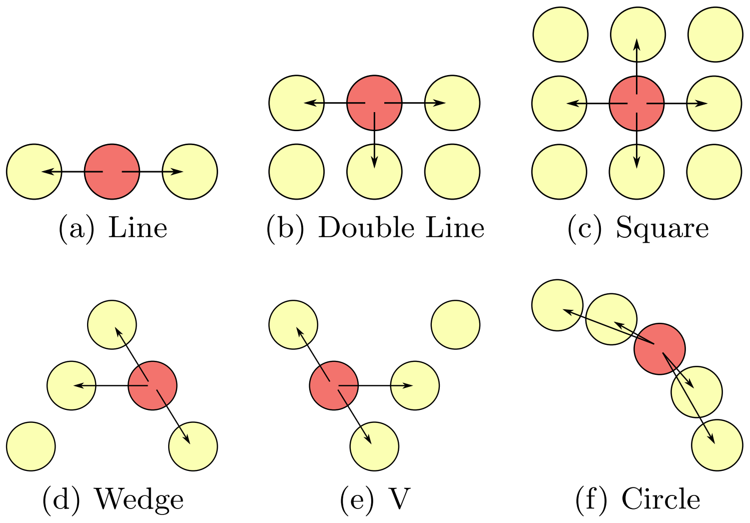


Fig. 3. Relationship between nodes in defined formations. Simple formations, such as *line* or *double line*, require only a connection between the nearest nodes. In more complex structures we have to extend the number of arcs. In *circle* formation each node is connected to the adjacent nodes and to their neighbors. Formations *V* and *wedge* require an additional connection to the nodes on the opposite branch.

- $\boldsymbol{v}_{x\to y}$
- $\boldsymbol{v}_{y\to x} = -\boldsymbol{v}_{x\to y}$

For movement of the group of agents let $N(x)$ denote the number of neighbors of node x, $\boldsymbol{v}$ the velocity of the entire formation and $\alpha \in [0,1]$ represents the impact of the neighbors on node position then state of x in time $t+\Delta t$ is defined as follow

$$\psi^{(x)}(t+\Delta t) = \psi^{(x)}(t)\,(1-\alpha\Delta t) + \vec{v}\Delta t + \alpha\Delta t \left[\frac{1}{N(x)} \sum_{y\sim x} \left(\psi^{(y)}(t) + \vec{v}_{y\to x}\right)\right] \tag{7}$$

More precisely, the matrix B can be constructed in the following way:

1. Add an artificial node $e \equiv 1$ to the Gaussian network.
2. Set the coefficients as follow
   - $B_{xx} = -\alpha$
   - $B_{xy} = \begin{cases} \frac{\alpha}{N(x)} & \text{if } y \sim x \\ 0 & \text{if } y \nsim x \end{cases}$
   - $B_{xe} = \boldsymbol{v} + \frac{\alpha}{N(x)} \sum_{y\sim x} \boldsymbol{v}_{x\to y}$,

The construction of matrix B was a crucial point in this paragraph and allows for straightforward use of Eq. (1) and Eq. (6) for the simulation. In Fig. 4 the traveled paths are presented by agents during transitions between given formations. As we can see, optimal paths are not the shortest ones. It is expected behavior because the algorithm does not minimize the length of the paths, but rather looks for most probable ones (see Fig. 2). Using shortest paths is highly

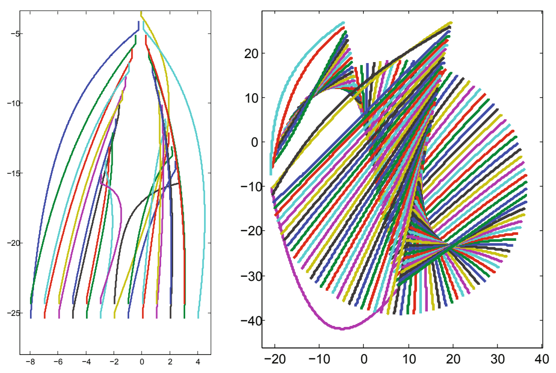


Fig. 4. On the left we see a transition of 25 units from *V* formation to *double line*. On the right 121 tanks redeploy from *circle* to *wedge* configuration.

dissuaded from use due to observed phenomenon in military, where all movements incidents during changes of formation were mainly caused by obtaining the shortest practical route [22].

4 Transitions between Animations

The main purpose of this section is to apply the technique developed in Paragraph 2 to solving the problem of finding optimal transition path between animations. First we have to translate the motion capture date into terms of a Gaussian network. The resulted network should reproduce smoother version of the given motion. Coefficients of the matrix B will be learnt with the use of a regression method. It turn out that the task is non–trivial and first we have to define hierarchical dependencies in the motion data.

The motion capture data[1] grants us with a structure of bones S and set of motions M. S is the skeleton of an animated character that is typically defined as a hierarchy of segments. The skeleton consists of a *root* segment, that is on the top in the hierarchy and does not have any ancestors. Each other segment poses a parent segment and may have one or more children. Segments represent bones and include their properties such as their length, direction, degrees of freedom, local coordinate system, etc. Character motion M_i, with respect to the skeleton S, is defined as a doublet $M_i := (R_i, A_i)$, where R_i is the position of the skeleton's *root* segment for each frame, A_i is a sequence of orientation of each segment that is $A_i = \{a^t_{i,k}; t = 0, \dots, T; k = 0, \dots, \|S\|\}$ so particularly $a^t_{i,k}$ represents an orientation of kth bone during frame t for an ith animation. Current character configuration is defined as the orientation of each bone in the local coordinate system. Using the technique of skeletal animation [4], we can simulate defined motion.

Coefficients of the Gaussian network can be learnt with the use of a linear regression method [5]. For each segment k the regression model can be described

[1] The data used in this project was obtained from `http://www.mocap.cs.cmu.edu`. The database was created with funding from NSF EIA-0196217.

by the dependent vector $a^{t+1}_{i,k}$ and a set of regressors $\{a^t_{i,1}, \dots, a^t_{i,\|S\|}\}$. Results of that straightforward method produced very unstable animations i.e. motion of a character becomes unreliable. Another approach of using *multiple linear regression* [5] with regressors defined by hierarchical structure of the skeleton suffered similar drawbacks. We were also unable to manually define correct relations in given skeletons. It appears that intuitive dependencies between human bones are misleading. For example, the position of *head* was the best described by configuration of: *right femur*, *right tibia*, *right foot* and *left wrist*. Resulting motion did not reproduce correctly given motion and was highly unstable.

To find proper relations in a given motion we decided to use one of the optimization techniques. The number of possible DAGs (Direct Acyclic Graphs) as a function of the number of nodes, G(n), is given by the following recursive function [16]

$$G(1) = 1, G(n) = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} G(n-k) \tag{8}$$

Let us enumerate some values of function G $G(2) = 3$, $G(4) = 543$, $G(5) = 29281$, $G(10) = 4.2 \times 10^{18}$, $G(100) = 1.1 \times 10^{1631}$. Testing all possible DAG patterns is computationally unfeasible because the number of DAGs grows super-exponentially [12]. In our implementation we have used simulated annealing method [11] to search in possible DAG space. For a given skeleton S and animation that consists of F frames the energy function has the following form:

$$\mathcal{E}(M_i) = \frac{1}{F\|S\|} \sum_{f=1}^{F} \sum_{b=1}^{\|S\|} \|c^f_{i,b} - s^f_{i,b}\|^2, \tag{9}$$

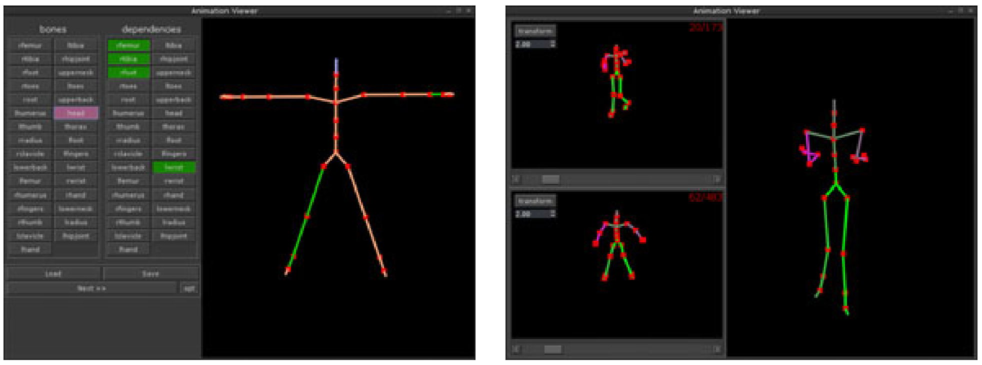
where $c^f_{i,b}$ is a configuration of bone b on the ith animation at frame t obtained from motion capture data, $s^f_{i,b}$ has similar meaning, but is computed by simulation of the Gaussian network. In other words we compute *mean square error* between positions of bones given by motion capture data in each frame and their configuration computed with presented method.

In addition, we impose the following requirements:

– Coefficients in matrix B are bound by constant b_r.
– The mean square error computed for the entire skeleton should not exceed some (large) constant s_b.

The purpose of these conditions is to ensure the numerical stability of the algorithm. During simulations, we set $b_r = 10^3$ and $s_b = 10^{12}$. The applied simulated annealing algorithm would eventually converge to the target bones hierarchy (see Image 5(a)).

After applying described scheme to the motion we obtain matrix B. The animation obtained from the simulation of the Gaussian network looks very similar to the original motion. The main difference is that the new animation is more smooth. To maintain the motion in large time horizon we add small noise to



(a) bones hierarchy (b) transitions between animations

Fig. 5. Screenshots from the sandbox. In (a) we can see that orientation of the head is best described by orientation of: right femur, right tibia, right foot and left wrist, (b) presents transition from walking to jumping animation.

the simulation [2], without that, the motion gradually fades away (it converges to the steady state). At any moment, the animation of a motion can be interrupted and by use of the method from Par. 2, even for a nonsymmetric matrix B, smoothly transformed to a selected frame of another motion (Fig. 5(b)).

5 Implementation

The programme has been implemented in language `D` [3]. Some matrix operations incorporate LAPACK, BLAS and CUBLAS [14] subroutines. All test runs were executed on a machine with Intel Core 2 Q9300 2.50 GHz CPU, 4GB RAM and NVIDIA GTX 480. The application is single threaded, so it is applicable to the core of only one processor. All computations were performed with double precision arithmetic.

The mean square error (MSE) between the target configuration and a simulated one can be controlled by adjusting the step size. For the *transition time* set on 2.0 and *step size* on 0.002 the observed value of MSE was lower than 0.0001.

An efficient computation of a matrix exponential plays a crucial role in the presented method. Numerous methods for computing e^A were developed. The straightforward one, which uses Euler series, is inefficient even in the scalar case. The survey [13] presented a wide variety of methods and pointed the most powerful ones. Today, due to evolution in computer hardware and highly optimized BLAS subroutines the most cogent technique is the scaling and squaring method combined with Padé approximants [9]. The sequential version of that algorithm has been implemented. To parallelize the computation of the matrix exponential we use NVIDIA CUBLAS library [10,14] and substitute serial matrix operations with parallel ones.

In Table 1 we can see dependencies between the number of simulated objects (tanks) and the time required for a single step and a transition. As we can see,

Table 1. Step and transition times

quantity	step (ms)	CPU (ms)	GPU (ms)	speedup
25	0.04	1.7	40	0.07
100	0.6	24	109	0.22
169	1.6	75	193	0.38
225	3.1	172	275	0.63
400	9	680	637	1.07
625	22	2200	1470	1.50
900	49	12000	3480	3.45
1225	89	54000	6890	7.84
1600	155	253000	12400	17.57
2025	240	785000	26200	**29.96**

a single step can be computed very fast. The most consuming part during w_0 calculation is matrix exponential. GPU accelerated version is up to 30 times faster than the sequential one. An analysis of parallel profiler output shows that time is mainly (up to 70%) spent on matrix multiplications (*dgemm*). Transfers of the data from host to device (CPU $\rightarrow$ GPU) and vice versa take up to 30% of the time. The rest of the time is consumed on memory allocations and other matrix operations. We should emphasize that computations are performed for x, y and z coordinates independently.

6 Conclusions

The algorithm for determining optimal transition paths was presented. For the evaluation purpose two applications have been successfully implemented. The problem of the formation redeployment was solved by our algorithm and produced enjoyable results. The resulting group dynamics is highly complex and a great care must be taken when such an environment is simulated. The other tested application was devoted to transitions between animations. The problem was more complicated due to the need of representing the motion in terms of Gaussian networks. The obtained animations were satisfying for the viewer. Simulations on the GPU were also performed and provided a significant gain in the runtime, but only in the large enough networks. Future enhancements may include hierarchical grouping of Gaussian nodes before transitions which would significantly improve a computation time. Another option is to optimize the transition time rather than to use the given one.

Acknowledgements. M. Matuszak and T. Schreiber gratefully acknowledge the support from the Polish Ministry of Scientific Research and Higher Education grant N N201 385234 (2008-2010). The work of M. Matuszak has also been supported by the European Social Fund, Government of Poland and Kuyavian-Pomeranian Voivodeship as a part of Integrated Operational Program for Regional Development, Activity 8.2.2.

References

1. Balch, T., Arkin, R.C.: Behavior–based formation control for multirobot teams. IEEE Transactions on Robotics and Automation 14(6), 926–939 (1998)
2. Balch, T., Hybinette, M.: Behavior–Based Coordination of Large-Scale Robot Formations. In: International Conference on Multiagent Systems – ICMAS, pp. 363–364 (2000)
3. Bell, K., Igesund, L.I., Kelly, S., Parker, M.: Learn to Tango with D. Apress (2008)
4. Burtnyk, N., Wein, M.: Interactive skeleton techniques for enhancing motion dynamics in key frame animation. Commun. ACM 19, 564–569 (1976)
5. Draper, N.R., Smith, H.: Applied regression analysis, 3rd edn. Wiley, New York (1998)
6. Freidlin, M.I., Wentzell, A.D.: Random perturbations of dynamical systems, 2nd edn. Grundlehren der Mathematischen Wissenschaften, vol. 260. Springer, New York (1998)
7. Haliloglu, T., Bahar, I., Erman, B.: Gaussian dynamics of folded proteins. Physical Review Letters 79, 3090–3093 (1997)
8. Heymann, M., Vanden-Eijnden, E.: The Geometric Minimum Action Method: A Least Action Principle on the Space of Curves. Comm. Pure Appl. Math. 61(8), 1052–1117 (2008)
9. Higham, N.J.: The Scaling and Squaring Method for the Matrix Exponential Revisited. SIAM J. Matrix Anal. Appl. 26(4), 1179–1193 (2005)
10. Kirk, D.B., Hwu, W.-M.W.: Programming Massively Parallel Processors: A Hands–on Approach, 1st edn. Morgan Kaufmann, San Francisco (2010)
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220, 671–680 (1983)
12. Koski, T., Noble, J.: Bayesian Networks: An Introduction. John Wiley & Sons, Ltd., Chichester (2009)
13. Moler, C., Van Loan, C.: Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty–Five Years Later. SIAM Rev. 45(3) (2003)
14. NVIDIA CUBLAS Library, NVIDIA Corporation (2009)
15. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics. In: SIGGRAPH 1987 vol. 21(4), pp. 25–34 (1987)
16. Robinson, R.W.: Counting Unlabelled Acyclic Digraphs. In: Combinatorial Mathematics V. Springer Lecture Notes in Mathematics, pp. 28–43 (1977)
17. Rue, H., Held, L.: Gaussian Markov Random Fields: Theory and Applications. Monographs on Statistics & Applied Probability 104 (2005)
18. Safonova, A., Hodgins, J.K.: Analyzing the physical correctness of interpolated human motion. In: ACM Siggraph/Eurographics Symposium on Computer Animation (SCA 2005), pp. 171–180 (2005)
19. Shachter, R.D., Kenley, C.R.: Gaussian influence diagrams. Management Science 35(5), 527–550 (1989)
20. Scharf, D.P., Hadaegh, F.Y., Ploen, S.R.: A Survey of Spacecraft Formation Flying Guidance and Control (Part I): Guidance. In: American Control Conference, vol. 2, pp. 1733–1739 (June 2003)
21. Scharf, D.P., Hadaegh, F.Y., Ploen, S.R.: A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control. In: American Control Conference, vol. 4, pp. 2976–2985 (June 30 - July 2, 2004)
22. U.S. Marine Corps Marine Rifle Squad. Marine Corps Warfighting Publication (MCWP) 3–11.2 (1997)

Paper C

Matuszak, M., Schreiber, T.
Locally specified polygonal Markov fields for image segmentation

Chapter 15
Locally Specified Polygonal Markov Fields for Image Segmentation

Michal Matuszak and Tomasz Schreiber

Abstract We introduce a class of polygonal Markov fields driven by local activity functions. Whereas the local rather than global nature of the field specification ensures substantial additional flexibility for statistical applications in comparison to classical polygonal fields, we show that a number of simulation algorithms and graphical constructions, as developed in our previous joint work with M.N.M. van Lieshout and R. Kluszczynski, carry over to this more general framework. Moreover, we provide explicit formulae for the partition function of the model, which directly implies the availability of closed form expressions for the corresponding likelihood functions. Within the framework of this theory we develop an image segmentation algorithm based on Markovian optimization dynamics combining the simulated annealing ideas with those of Chen-style stochastic optimization, in which successive segmentation updates are carried out simultaneously with adaptive optimization of the local activity functions.

15.1 Introduction

The polygonal Markov fields, originally introduced by Arak and Surgailis [10–12] and then studied by a number of authors [13, 319, 377–379, 395], arise as continuum ensembles of non-intersecting polygonal contours in the plane. One of the sources of theoretical interest in these processes lies in that they share a number of salient features with the two-dimensional Ising model, including the geometry of phase transitions and phase separation phenomenon [319, 377, 378] as well as the availability of explicit formulae for important numerical characteristics [11, 12, 379] yielding in particular closed form expressions for the likelihood functions. The idea that the polygonal Markov fields can carry out image processing tasks traditionally reserved for lattice-indexed Markov fields (see [448] for a comprehensive survey, cf.

M. Matuszak (✉) · T. Schreiber
Faculty of Mathematics & Computer Science, Nicolaus Copernicus University, ul. Chopina 12/18, 87-100 Toruń, Poland
e-mail: gruby@mat.uni.torun.pl

T. Schreiber
e-mail: tomeks@mat.uni.torun.pl

also Chap. 14 and the references therein for new developments) has emerged quite early and originates from Clifford, Middleton and Nicholls [79] who formulated it in a Bayesian setting. The obvious crucial advantage of polygonal fields in this context is their continuum nature which makes them completely free of lattice artifacts in image processing applications. The significant problem which slowed down the progress of this early work was the lack of efficient samplers and simulation algorithms for polygonal fields. These were introduced a decade later in a series of our joint papers with M.N.M. van Lieshout and R. Kluszczynski [256, 257, 377, 380, 424] where a polygonal field optimization approach for image segmentation was advocated. Although these methods were quite succesful in global shape recognition, the problem we faced in that work was related to the lack of local parametrization tools designed to deal with intermediate scale image characteristics—even though the applied simulated annealing algorithm would eventually converge to the target polygonal segmentation, we were looking for a more efficient explicit mechanism to drive the local search. Introducing such mechanisms and applying them to image segmentation is the principal purpose of the present paper. We construct a class of polygonal Markov fields with local activity functions (Sect. 15.2) and discuss their properties and graphical representations (Sect. 15.3). Next, in Sects. 15.4 and 15.5 we develop a Markovian optimization dynamics for image segmentation, under which both the polygonal configuration and the underlying local activity function are subject to optimization—whereas the polygonal configuration evolves according to a simulated annealing scheme in the spirit of [256, 257], the local activity function is initially chosen to reflect the image gradient information, whereupon it undergoes adaptive updates in the spirit of the celebrated Chen algorithm, see [73] and 10.2.4.c. in [333], with the activity profile reinforced along polygonal paths contributing to the improvement of the overall segmentation quality and faded along paths which deteriorate the segmentation quality. The sample results of our software are presented in the final Sect. 15.6.

15.2 Locally Specified Polygonal Markov Fields

Fix an open bounded convex set D in the plane $\mathbb{R}^2$, referred to as the field domain in the sequel, and define the family Γ_D of admissible polygonal configurations in D, by taking all the finite planar graphs γ in $D \cup \partial D$, with straight-line segments as edges, such that

- The edges of γ do not intersect,
- All the interior vertices of γ (lying in D) are of degree 2,
- All the boundary vertices of γ (lying in ∂D) are of degree 1,
- No two edges of γ are colinear.

In other words, γ consists of a finite number of disjoint polygons, possibly nested and chopped off by the boundary. We shall write $\Gamma_D[k] \subset \Gamma_D$ for the set of all admissible polygonal configurations in D with precisely k edges.

For a Borel subset of $A \subseteq \mathbb{R}^2$ by $[\![A]\!]$ we shall denote the family of all straight lines hitting A so that in particular $[\![\mathbb{R}^2]\!]$ stands for the collection of all straight lines in $\mathbb{R}^2$. Further, we let μ be the standard isometry-invariant Haar-Lebesgue measure on the space $[\![\mathbb{R}^2]\!]$ of straight lines in $\mathbb{R}^2$. Recall that one possible construction of μ goes by identifying a straight line l with the pair $(\phi, \rho) \in [0, \pi) \times \mathbb{R}$, where $(\rho \sin(\phi), \rho \cos(\phi))$ is the vector orthogonal to l, and joining it to the origin, and then by endowing the parameter space $[0, \pi) \times \mathbb{R}$ with the usual Lebesgue measure. Note that the above parametrisation of $[\![\mathbb{R}^2]\!]$ with $[0, \pi) \times \mathbb{R}$ endows $[\![\mathbb{R}^2]\!]$ with a natural metric, topology and Borel σ-field which will be used in this paper.

On $[\![D]\!] \times D$ we consider a non-negative bounded *local activity function* $\mathscr{M}(\cdot\,;\cdot)$ which will determine the local activity structure of the polygonal field. Define the formal Hamiltonian $L^{\mathscr{M}} : \Gamma_D \to \mathbb{R}_+$ given by

$$L^{\mathscr{M}}(\gamma) := \sum_{e \in \mathrm{Edges}(\gamma)} \int_{l \in [\![e]\!]} \mathscr{M}(l; l \cap e)\mu(dl), \quad \gamma \in \Gamma_D. \tag{15.1}$$

We note that the energy function $L^{\mathscr{M}}$ should be regarded as an anisotropic environment-specific version of the length functional. Indeed, for a line l hitting a graph edge $e \in \mathrm{Edges}(\gamma)$ at their intersection point $x = l \cap e$, the local activity $\mathscr{M}(l; l \cap e)$ shall be interpreted as the likelihood of a new edge being created along l intersecting and hence fracturing at x the edge e in γ. Under this interpretation we see that, roughly speaking, the value of $\int_{l \in [\![e]\!]} \mathscr{M}(l; l \cap e)\mu(dl)$ determines how likely the edge e is to be fractured by another edge present in the environment. In other words, $L^{\mathscr{M}}(\gamma)$ determines *how difficult it is to maintain* the whole graph $\gamma \in \Gamma_D$ without fractures in the environment whose local activity profile is characterised by $\mathscr{M}(\cdot\,;\cdot)$—note that due to the anisotropy of the environment there may be graphs of a higher (lower) total edge length than γ and yet of lower (higher) energy and thus easier (more difficult) to maintain and to keep unfractured due to the lack (presence) of high local activity lines likely to fracture their edges. In the particular case where $\mathscr{M}$ is constant, $L^{\mathscr{M}}$ is readily verified to be a multiple of the usual length functional, see e.g. p. 554 in [11].

We assume that a measurable *anchor mapping* $\mathbb{A} : [\![D]\!] \to D$ is given on the set of lines crossing D, assigning to each of them its *anchor point*, also interpreted as the *initial point* of the line. This allows us to define for each bounded linear segment/graph edge e in D its initial point $\iota[e]$ which is the point of e closest to the anchor $\mathbb{A}(l[e])$, where $l[e]$ is the straight line extending e. In particular, if $\mathbb{A}(l[e]) \in e$ then $\iota[e] = \mathbb{A}(l[e])$, otherwise $\iota[e]$ is the endpoint of e closest to $\mathbb{A}(l[e])$.

The polygonal Markov field $\mathscr{A}_D^{\mathscr{M}}$ with local activity function $\mathscr{M}$ in D is defined by

$$\begin{aligned}&\mathbb{P}(\mathscr{A}_D^{\mathscr{M}} \in d\gamma)\\ &\quad \propto \exp(-L^{\mathscr{M}}(\gamma)) \prod_{e \in \mathrm{Edges}(\gamma)} [\mathscr{M}(l[e]; \iota[e])\mu(dl[e])], \quad \gamma \in \Gamma_D.\end{aligned} \tag{15.2}$$

In other words, the probability of having $\mathscr{A}_D^{\mathscr{M}} \in d\gamma$ is proportional to the Boltzmann factor $\exp(-L^{\mathscr{M}}(\gamma))$ times the product of local edge activities $\mathscr{M}(l[e];$

$\iota[e])\mu(dl[e])$, $e \in \text{Edges}(\gamma)$. Observe that this construction should be regarded as a specific version of the general polygonal model given by Arak and Surgailis [11, 2.11] and an extension of the non-homogeneous polygonal fields considered in Schreiber [379] at their consistent regime (inverse temperature parameter fixed to 1). It should be also noted at this point that if the typical edge length for $\mathscr{A}_D^{\mathscr{M}}$ is much smaller than the characteristic scale for oscillations of $\mathscr{M}$, which is often the case in our applications below, then $\mathscr{M}(l[e];\cdot)$ is usually approximately constant along the corresponding edge e and the formal dependency of the factor $\mathscr{M}(l[e];\iota[e])\mu(dl[e])$ on the choice of initial segment for e becomes negligible in large systems. The finiteness of the partition function

$$\mathscr{Z}_D^{\mathscr{M}} := \sum_{k=0}^{\infty} \frac{1}{k!} \int_{\Gamma_D[k]} \exp(-L^{\mathscr{M}}(\gamma)) \prod_{e \in \text{Edges}(\gamma)} [\mathscr{M}(l[e];\iota[e])\mu(dl[e])] \tag{15.3}$$

is not difficult to verify, see [379], and in fact it will be explicitly calculated in the sequel.

The so-defined locally specified polygonal fields enjoy a number of striking features inherited from the previously developed polygonal models, see [11, 379]. One of these is the two-dimensional germ-Markov property stating that the conditional behaviour of the field $\mathscr{A}_D^{\mathscr{M}}$ inside a smooth closed curve θ depends on the outside field configuration only through the trace it leaves on θ, consisting of intersection points and the respective line directions, see [11] for details. This is where the term *polygonal Markov field* comes from. Further properties of the locally defined polygonal fields are going to be discussed in the next section, where their algorithmic construction is provided.

15.3 Dynamic Representation for Locally Specified Polygonal Fields

The present section is meant to extend the so-called *generalised dynamic representation* for consistent polygonal fields as developed in Schreiber [379] to cover the more general class of locally specified polygonal fields defined in Sect. 15.2 above. The name *generalised representation* comes from the fact that it generalises the original construction of homogeneous polygonal fields introduced by Arak and Surgailis [11]. In the sequel we will often omit the qualifier *generalised* for the sake of terminological brevity. To describe the generalised representation, fix the convex field domain D and let $(D_t)_{t\in[0,1]}$ be a time-indexed increasing family of compact convex subsets of $\bar{D}$, eventually covering the entire $\bar{D}$ and interpreted as a *growing window* gradually revealing increasing portions of the polygonal field under construction in the course of the time flow. In other words, under this interpretation, the portion of a polygonal field in a bounded open convex domain D *uncovered* by time t is precisely its intersection with D_t. To put it in formal terms, consider $(D_t)_{t\in[0,1]}$ satisfying

(D1) $(D_t)_{t\in[0,1]}$ is a strictly increasing family of compact convex subsets of $\bar{D} = D \cup \partial D$.
(D2) D_0 is a single point x in $\bar{D} = D \cup \partial D$.
(D3) D_1 coincides with $\bar{D}$.
(D4) D_t is continuous in the usual Hausdorff metric on compacts.

Clearly, under these conditions, for μ-almost each $l \in [\![D]\!]$ the intersection $l \cap D_{\tau_l}$ consists of precisely one point $\mathbb{A}(l)$, where $\tau_l = \inf\{t \in [0,1],\ D_t \cap l \neq \emptyset\}$. The point $\mathbb{A}(l)$ is chosen to be the *anchor point* for l, which induces the *anchor* mapping $\mathbb{A} : [\![D]\!] \to D$ as required for our construction in Sect. 15.2. Note that this choice of the anchor mapping implies that at each point of a line l the direction away from its anchor point $\mathbb{A}(l)$ coincides with the outwards direction with respect to the *growing window* (D_t). Consider now the following dynamics in time $t \in [0,1]$, with all updates, given by the rules below, performed independently of each other, see Fig. 15.1.

(*GE:Initialise*) Begin with empty field at the time 0.
(*GE:Unfold*) Between critical moments listed below, during the time interval $[t, t+dt]$ the unfolding field edges in D_t reaching ∂D_t extend straight to $D_{t+dt} \setminus D_t$.
(*GE:BoundaryHit*) When a field edge hits the boundary ∂D, it stops growing in this direction (note that μ-almost everywhere the intersection of a line with ∂D consists of at most two points).
(*GE:Collision*) When two unfolding field edges intersect in $D_{t+dt} \setminus D_t$, they are not extended any further beyond the intersection point (stop growing in the direction marked by the intersection point).
(*GE:DirectionalUpdate*) A field edge extending along $l \in [\![D_t]\!]$ updates its direction during $[t, t+dt]$ and starts unfolding along $l' \in [\![l^{[t,t+dt]}]\!]$, extending away from the anchor point $\mathbb{A}(l')$, with probability $\mathscr{M}(l'; l \cap l')\mu(dl')$, where $l^{[t,t+dt]} := l \cap (D_{t+dt} \setminus D_t)$. Directional updates of this type are all performed independently.
(*GE:LineBirth*) Whenever the anchor point $\mathbb{A}(l)$ of a line l falls into $D_{t+dt} \setminus D_t$, the line l is born at the time t at its anchor point with probability $\mathscr{M}(l; \mathbb{A}(l))\mu(dl)$, whereupon it begins extending in both directions with the growth of D_t (recall that l is μ-almost always tangential to ∂D_t here).
(*GE:VertexBirth*) For each intersection point of lines l_1 and l_2 falling into $D_{t+dt} \setminus D_t$, the pair of field lines l_1 and l_2 is born at $l_1 \cap l_2$ with probability $\mathscr{M}(l_1; l_1 \cap l_2)\mathscr{M}(l_2; l_1 \cap l_2)\mu(dl_1)\mu(dl_2)$, whereupon both lines begin unfolding in the directions away from their respective anchor points $\mathbb{A}(l_1)$ and $\mathbb{A}(l_2)$.

Observe that the evolution rule (*GE:VertexBirth*) means that pairs of lines are born at birth sites distributed according to a Poisson point process in D with intensity measure given by the *intersection measure* $\langle\!\langle \mathscr{M} \rangle\!\rangle$ of $\mathscr{M}$:

$$\langle\!\langle \mathscr{M} \rangle\!\rangle(A) := \frac{1}{2}\int_{\{(l_1,l_2),\ l_1 \cap l_2 \subset A\}} \mathscr{M}(l_1; l_1 \cap l_2)\mathscr{M}(l_2; l_1 \cap l_2)\mu(dl_1)\mu(dl_2). \quad (15.4)$$

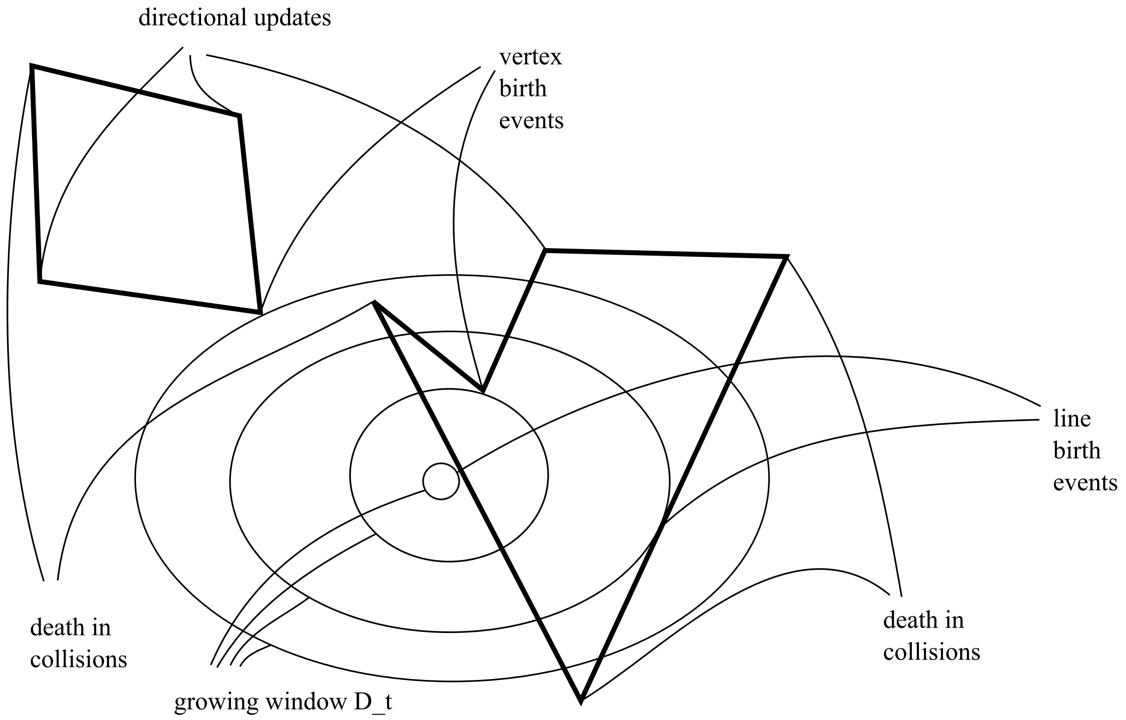


Fig. 15.1 Dynamic representation

Likewise, the evolution rule (*GE:LineBirth*) implies that individual lines are born at their anchor points according to a Poisson point process in $[\![D]\!]$ with intensity measure given by the *anchor measure* $\langle \mathscr{M} \rangle$ of $\mathscr{M}$:

$$\langle \mathscr{M} \rangle(B) := \int_{\{l,\ \mathbb{A}(l) \in B\}} \mathscr{M}(l, \mathbb{A}(l))\mu(dl). \tag{15.5}$$

The main theorem of this section is the following extension of Theorem 3 in [379].

Theorem 15.1 *The random contour ensemble resulting from the above construction* (*GE*) *coincides in law with* $\mathscr{A}_D^{\mathscr{M}}$. *Moreover, we have*

$$\log \mathscr{Z}_D^{\mathscr{M}} = \langle\!\langle \mathscr{M} \rangle\!\rangle(D) + \langle \mathscr{M} \rangle([\![D]\!]). \tag{15.6}$$

Proof We pick some $\gamma \in \Gamma_D$ and calculate the probability that the outcome of the above dynamic construction falls into $d\gamma$. To this end, we note that:

- Each edge $e \in \mathrm{Edges}(\gamma)$ containing the anchor point $\mathbb{A}(l[e])$ and hence resulting from a line birth event due to the rule (*GE:LineBirth*), contributes to the considered probability the factor $\mathscr{M}(l[e]; \mathbb{A}(l[e]))\mu(dl[e])$ (line birth probability for $l[e]$) times $\exp[-\int_{[\![e]\!]} \mathscr{M}(l; l \cap e)\mu(dl)]$ (no directional updates along e).
- Each of the two edges $e_1, e_2 \in \mathrm{Edges}(\gamma)$ stemming from a common interior birth vertex $l[e_1] \cap l[e_2] = \iota[e_1] = \iota[e_2]$ yields the factor $\mathscr{M}(l[e_i]; \iota[e_i])\mu(dl[e_i]),\ i = 1, 2$, (coming from the vertex birth probability due to the rule (*GE:VertexBirth*)) times $\exp[-\int_{[\![e_i]\!]} \mathscr{M}(l; l \cap e_i)\mu(dl)]$ (no directional updates along e_i).

- Each of the edges $e \in \mathrm{Edges}(\gamma)$ arising in (*GE:DirectionalUpdate*) yields the factor $\mathscr{M}(l[e]; \iota[e])$ (directional update probability) times $\exp[-\int_{[\![e]\!]} \mathscr{M}(l; l \cap e)\mu(dl)]$ (no directional updates along e).
- The absence of interior birth sites in $D \setminus \gamma$ yields the factor $\exp[-\langle\!\langle \mathscr{M} \rangle\!\rangle(D)]$.
- Finally, the absence of line birth events for all lines in $[\![D]\!]$ except for the finite collection $\{l[e],\ e \in \mathrm{Edges}(\gamma),\ \mathbb{A}(l[e]) \in e\}$ yields the additional factor $\exp[\langle -\mathscr{M} \rangle([\![D]\!])]$.

Putting these observations together we conclude that the probability element of γ resulting from the generalized construction above is

$$\frac{\exp(-L^{\mathscr{M}}(\gamma)) \prod_{e \in \mathrm{Edges}(\gamma)} [\mathscr{M}(l[e]; \iota[e])\mu(dl[e])]}{\exp[\langle\!\langle \mathscr{M} \rangle\!\rangle(D)] \exp[\langle \mathscr{M} \rangle([\![D]\!])]}$$

and thus, upon comparing with (15.2) and (15.3), the field obtained by this construction coincides in law with $\mathscr{A}_D^{\mathscr{M}}$ as required and (15.6) follows as well. This completes the proof of the theorem. □

15.4 Disagreement Loop Dynamics

In this section we discuss a random dynamics on the space Γ_D of admissible polygonal configurations which leaves the law of the field $\mathscr{A}_D^{\mathscr{M}}$ invariant and reversible. This dynamics will be used in the sequel as a mechanism for update proposal generation in stochastic optimization schemes for image segmentation. We build upon [377, 379] in our presentation of the dynamics based on an important concept of a *disagreement loop*.

To proceed we place ourselves within the context of the dynamic representation discussed in Sect. 15.3 above and suppose that we observe a particular realisation $\gamma \in \Gamma_D$ of the polygonal field $\mathscr{A}_D^{\mathscr{M}}$ and that we modify the configuration by adding an extra (*GE:VertexBirth*) vertex birth site at $x_0 \in D$ to the existing collection of vertex births for γ, while keeping unchanged the remaining evolution rules (*GE*) for all the edges, including the two newly added ones. Denote the resulting new (random) polygonal configuration by $\gamma \oplus x_0$. A simple yet crucial observation is that for $x_0 \in D$ the symmetric difference $\gamma \triangle [\gamma \oplus x_0]$ is almost surely a single loop (a closed polygonal curve), possibly self-intersecting and possibly chopped off by the boundary (becoming a path then). Indeed, this is seen as follows. Each point in $x \in D$ can be attributed its *time coordinate* which is just the time moment at which x is first hit by ∂D_t. Then the chronologically initial point of the loop $\gamma \triangle [\gamma \oplus x_0]$ is of course x_0. Each of the two *new* polygonal curves p_1, p_2 initiated by edges e_1, e_2 emitted from x_0 unfold independently, according to (*GE*), each giving rise to a *disagreement path*. The initial segments of such a disagreement path correspond to the growth of the curve, say p_1, before its annihilation in the first collision. If this is a collision with the boundary, the disagreement path gets chopped off and terminates there. If this is a collision with a segment of the original configuration γ corresponding to a certain *old* polygonal curve p_3 emitted from a prior vertex birth site, the *new* curve p_1 dies but the disagreement path continues along the part of the trajectory of p_3 which is contained in γ but not in $\gamma \oplus x_0$. At some further moment p_3 dies itself in γ, touching the boundary or killing another polygonal curve p_4 in γ. In the second case, however, this collision only happens for γ and not for $\gamma \oplus x_0$ so the polygonal curve p_4 survives (for some time) in $\gamma \oplus x_0$ yielding a further connected portion of the disagreement path initiated by p_1, which is contained in $\gamma \oplus x_0$ but not in γ etc. A recursive continuation of this construction shows that the disagreement path initiated by p_1 at x_0 consists alternately of connected polygonal sub-paths contained in $[\gamma \oplus x_0] \setminus \gamma$ (call these *creation phase* sub-paths) and in $\gamma \setminus [\gamma \oplus x_0]$ (call these *annihilation phase* sub-paths). Note that this disagreement path is self-avoiding and, in fact, it can be represented as the graph of some piecewise linear function $t \mapsto x(t) \in \partial D_t$. Clearly, the same applies for the disagreement path initiated by p_2 at x_0. An important observation is that whenever two *creation phase* or two *annihilation phase* sub-paths of the two disagreement paths hit each other, both disagreement paths die at this point and the disagreement loop closes (as opposed to intersections of segments of different phases which do not have this effect). Obviously, if the disagreement loop does not close in the above way, it gets eventually chopped off by the boundary. We shall write $\Delta^{\oplus}[x_0; \gamma] = \gamma \triangle [\gamma \oplus x_0]$ to denote the (random) disagreement loop constructed above. A similar argument shows that an extra (*GE:LineBirth*) line birth event added for $l \in [\![D]\!]$ at its anchor point $\mathbb{A}(l)$, while keeping the remaining evolution rules unchanged, also gives rise to a disagreement loop $\Delta^{\oplus}[l; \gamma]$ which coincides with the symmetric difference $\gamma \triangle [\gamma \oplus l]$, where $\gamma \oplus l$ is the polygonal configuration resulting from γ upon adding the line birth site at $\mathbb{A}(l)$.

Likewise, a disagreement loop arises if we *remove* one vertex birth site $x_0 \in D$ from the collection of vertex birth sites of an admissible polygonal configuration $\gamma \in \Gamma_D$, while keeping the remaining evolution rules. We write $\gamma \ominus x_0$ for the configuration obtained from γ by removing x_0 from the list of vertex birth sites, while the resulting random disagreement loop is denoted by $\Delta^{\ominus}[x_0; \gamma]$ so that $\Delta^{\ominus}[x_0; \gamma] = \gamma \triangle [\gamma \ominus x_0]$. In full analogy, we define $\gamma \ominus l$ and $\Delta^{\ominus}[l; \gamma]$ where $l = l[e]$ is the field line extending an edge $e \in \mathrm{Edges}(\gamma)$ with $\mathbb{A}(l) \in e$ and $\gamma \ominus l$ is the configuration obtained from γ upon killing the line l at its anchor $\mathbb{A}(l)$ whereas $\Delta^{\ominus}[x_0; \gamma]$ is the resulting disagreement loop. We refer the reader to Sect. 2.1 in [377] for further discussion.

With the above terminology we are in a position to describe a random dynamics on the configuration space Γ_D, which leaves invariant the law of the polygonal process $\mathscr{A}_D^{\mathscr{M}}$. Particular care is needed, however, to distinguish between the notion of time considered in the dynamic representation of the field as well as throughout the construction of the disagreement loops above, and the notion of time to be introduced for the random dynamics on Γ_D constructed below. To make this distinction clear we shall refer to the former as to the *representation time* (r-time for short) and shall reserve for it the notation t, while the latter will be called the *simulation time* (s-time for short) and will be consequently denoted by s in the sequel.

Consider the following pure jump birth and death type Markovian dynamics on Γ_D, with $\gamma_s = \gamma_s^D$ standing for the current configuration

(*DL:Birth*) With intensity $\langle\langle \mathscr{M} \rangle\rangle(dx)ds$ for $x \in D$ and with intensity $\langle \mathscr{M} \rangle(dl)ds$ for $l \in [\![D]\!]$ set $\gamma_{s+ds} := \gamma_s \oplus x$ and $\gamma_{s+ds} := \gamma_s \oplus l$ respectively.

(*DL:Death*) For each vertex birth site x in γ_s with intensity ds set $\gamma_{s+ds} := \gamma_s \ominus x$. For each line birth site $\mathbb{A}(l[e]) \in e,\ e \in \mathrm{Edges}(\gamma)$ with intensity ds set $\gamma_{s+ds} := \gamma_s \ominus l[e]$.

If none of the above updates occurs we keep $\gamma_{s+ds} = \gamma_s$. It is convenient to perceive the above dynamics in terms of generating random disagreement loops λ and setting $\gamma_{s+ds} := \gamma_s \triangle \lambda$, with the loops of the type $\Delta^{\oplus}[\cdot,\cdot]$ corresponding to the rule (*DL:Birth*) and $\Delta^{\ominus}[\cdot,\cdot]$ to the rule (*DL:Death*).

As a direct consequence of the dynamic representation of the field $\mathscr{A}_D^{\mathscr{M}}$ as developed in Sect. 15.3, we obtain

Theorem 15.2 *The distribution of the polygonal field* $\mathscr{A}_D^{\mathscr{M}}$ *is the unique invariant law of the dynamics given by* (DL:Birth) *and* (DL:Death). *The resulting s-time stationary process is reversible. Moreover, for any initial distribution of* γ_0 *the laws of the polygonal fields* γ_s *converge in variational distance to the law of* $\mathscr{A}_D^{\mathscr{M}}$ *as* $s \to \infty$.

The uniqueness and convergence statements in the above theorem require a short justification. They both follow by the observation that, in finite volume, regardless of the initial state, the process γ_s spends a non-null fraction of time in the empty state (no polygonal contours). Indeed, this observation allows us to conclude the required uniqueness and convergence by a standard coupling argument, e.g. along the lines of the proof of Theorem 1.2 in [289].

15.5 Adaptive Optimization Scheme for Image Processing

To provide a formal description of our image segmentation procedure we represent the image processed by a continuously differentiable function $\phi : D \to [-1,1]$ defined on an open bounded convex image domain D. By segmentations of ϕ we shall understand admissible polygonal configurations $\gamma \in \Gamma_D$. Interpreting the contours of γ as curves separating regions of different signs in D we associate with γ two natural sign-functions $s_\gamma^+ : D \to \{-1,+1\}$ and $s_\gamma^- = -s_\gamma^+$. The quality of a segmentation is quantified in terms of an *energy function* $\mathscr{H}[\gamma] := \mathscr{H}[\gamma|\phi]$ which in our case is a positive linear combination of a L_1-type distance (multiple of pixel misclassification ratio), the length element and the number of edges, that is to say

$$\mathscr{H}[\gamma] := \alpha_2 \min\left(\int_D |\phi(x) - s_\gamma^+(x)|dx,\ \int_D |\phi(s) - s_\gamma^-(x)|dx\right) \\ + \alpha_1 \,\mathrm{length}(\gamma) + \alpha_0 \,\mathrm{card}(\mathrm{Edges}(\gamma)), \quad \alpha_i > 0,\ i = 0,1,2, \quad (15.7)$$

although clearly many other natural options are also possible, such as L_p-type metrics or various weighed versions thereof. Our optimization scheme (*OPT*) presented below is based on the (*DL*) evolution as described in Sect. 15.4 above, combined with the following ideas.

- The initial local activity function encodes the gradient information for ϕ.
- In the course of the dynamics, the local activity function undergoes adaptive updates in the spirit of the celebrated Chen algorithm, see [73] and 10.2.4.c in [333].
- The segmentation update proposals are accepted or rejected depending on the energy changes they induce, conforming to the simulated annealing scheme, see [1] for a general reference.

At each time moment $s \geq 0$ in the course of the (OPT) dynamics the local activity function $\mathscr{M}(\cdot\,;\cdot)$ is given by

$$\mathscr{M}_s(l;x) := |\mathbf{e}[l] \times \mathbf{G}_s(x)|, \tag{15.8}$$

where $\mathbf{e}[l]$ is a unit vector along l and $\times$ stands for the usual vector cross product. The vector field $\mathbf{G}_s$ evolves in (OPT) time together with the polygonal configuration γ_s as specified below, with the initial condition

$$\mathbf{G}_0(x) := \nabla\phi(x), \tag{15.9}$$

for practical reasons possibly modified by convolving ϕ with a small variance Gaussian kernel at the pre-processing stage. When combined, the relations (15.8) and (15.9) mean that we promote edges in directions perpendicular to local gradients and proportionally to the gradient lengths. In precise terms, our algorithm admits a description in terms of the following (non-homogeneous) pure-jump Markovian dynamics (OPT) unfolding in time $s \geq 0$.

(*OPT:Initialise*) At time 0 set the initial activity function $\mathscr{M}_0(\cdot\,;\cdot)$ as specified by (15.8) and (15.9) and generate γ_0 according to $\mathscr{A}_D^{\mathscr{M}_0}$.

(*OPT:Birth*) For $\mathscr{M}_s$ given as in (15.8), with intensity $\langle\langle\mathscr{M}_s\rangle\rangle(dx)ds$ for $x \in D$ and with intensity $\langle\mathscr{M}_s\rangle(dl)$ for $l \in [\![D]\!]$ do

[*GenerateDisagreementLoop*] Set $\delta := \gamma_s \oplus x$ and $\delta := \gamma_s \oplus l$ respectively, with λ standing for the respective disagreement loop $\Delta^{\oplus}[x;\gamma]$ or $\Delta^{\oplus}[l;\gamma]$ and with λ^+ and λ^- denoting its respective creation and annihilation phase sub-paths. Note that the disagreement loop is generated according to the current activity measure $\mathscr{M}_s$. Let $\Delta := \mathscr{H}(\delta) - \mathscr{H}(\gamma_s)$ be the energy difference between the current configuration γ_s and its update proposal δ.

[*ActivityUpdate*] Put

$$\begin{aligned}\mathbf{G}_{s+ds}(x) &:= \mathbf{G}_s(x)\\ &+ \frac{\exp(-K_s\Delta)-1}{2\pi\sigma_s^2}\int_{\lambda^+}\mathbf{n}[y]\langle\mathbf{n}[y],\mathbf{G}_s(x)\rangle\exp\left(-\frac{\mathrm{dist}^2(x,y)}{2\sigma_s^2}\right)dy\\ &+ \frac{\exp(K_s\Delta)-1}{2\pi\sigma_s^2}\int_{\lambda^-}\mathbf{n}[y]\langle\mathbf{n}[y],\mathbf{G}_s(x)\rangle\exp\left(-\frac{\mathrm{dist}^2(x,y)}{2\sigma_s^2}\right)dy,\end{aligned}$$

where $\mathbf{n}[y]$ stands for the unit normal to λ at $y \in \lambda$, defined almost everywhere; whereas K_s and σ_s are positive deterministic parameter functions discussed in more detail below.

[*ConfigurationUpdate*] If $\Delta < 0$ then set $\gamma_{s+ds} := \delta$. Otherwise set $\gamma_{s+ds} := \delta$ with probability $\exp(-\beta_s \Delta)$ (*accept update*) and keep $\gamma_{s+ds} = \gamma_s$ with the complementary probability (*reject update*). The parameter function β_s, referred to as the inverse temperature according to the usual terminology, increases in time following the cooling protocol of our simulated annealing.

(*OPT:Death*) With $\mathscr{M}_s$ as given by (15.8), for each vertex birth site x in γ_s with activity ds, and for each line birth site $\mathbb{A}(l[e]) \in e,\ e \in \mathrm{Edges}(\gamma)$ with intensity ds, do

[*GenerateDisagreementLoop*] Set $\delta := \gamma_s \ominus x$ and $\delta := \gamma_s \ominus l[e]$ respectively, with λ standing for the respective disagreement loop $\Delta^{\ominus}[x;\gamma]$ or $\Delta^{\ominus}[l[e];\gamma]$ and with λ^+ and λ^- denoting its respective creation and annihilation phase sub-paths. Note that the disagreement loop is generated according to the current activity measure $\mathscr{M}_s$. Let $\Delta := \mathscr{H}(\delta) - \mathscr{H}(\gamma_s)$ be the energy difference between the current configuration γ_s and its update proposal δ.

[*ActivityUpdate*] Put

$$\begin{aligned}\mathbf{G}_{s+ds}(x) := \mathbf{G}_s(x) \\ &+ \frac{\exp(-K_s\Delta)-1}{2\pi\sigma_s^2}\int_{\lambda^-}\mathbf{n}[y]\langle\mathbf{n}[y],\mathbf{G}_s(x)\rangle\exp\left(-\frac{\mathrm{dist}^2(x,y)}{2\sigma_s^2}\right)dy \\ &+ \frac{\exp(K_s\Delta)-1}{2\pi\sigma_s^2}\int_{\lambda^+}\mathbf{n}[y]\langle\mathbf{n}[y],\mathbf{G}_s(x)\rangle\exp\left(-\frac{\mathrm{dist}^2(x,y)}{2\sigma_s^2}\right)dy.\end{aligned}$$

[*ConfigurationUpdate*] If $\Delta < 0$ then set $\gamma_{s+ds} := \delta$. Otherwise set $\gamma_{s+ds} := \delta$ with probability $\exp(-\beta_s \Delta)$ (*accept update*) and keep $\gamma_{s+ds} = \gamma_s$ with the complementary probability (*reject update*).

Roughly speaking, our optimization dynamics (*OPT*) generates successive updates according to the disagreement loop dynamics (*DL*) driven by the current local activity function $\mathscr{M}_s$, whereupon it updates the activity function in the spirit of the Chen algorithm in [*ActivityUpdate*] phase, and then accepts or rejects the configuration update proposal for γ_s in [*ConfigurationUpdate*] conforming to the simulated annealing paradigm. Note that the activity update is carried out regardless of whether the configuration update proposal has been accepted or not. This is natural because in the activity update step the original and new configuration are compared for quality and then, along the disagreement segments present in the better of the two configurations, the normal component of the local gradient field is reinforced and, likewise, the normal component is subject to fading along the disagreement segments present in the worse configuration. The strength of this reinforcement/fading depends exponentially on the energy difference between the original configuration and its update, with rate controlled by time-dependent parameter K_s, which should increase over time starting from a low level to avoid erratic reinforcements induced by the initially chaotic nature of the early stage polygonal configurations γ_s. To keep the local activity function smooth we smear the activity updates over the domain by convolving them with a Gaussian kernel of time-dependent standard deviation parameter σ_s, as made precise in the [*ActivityUpdate*] formulae above. The

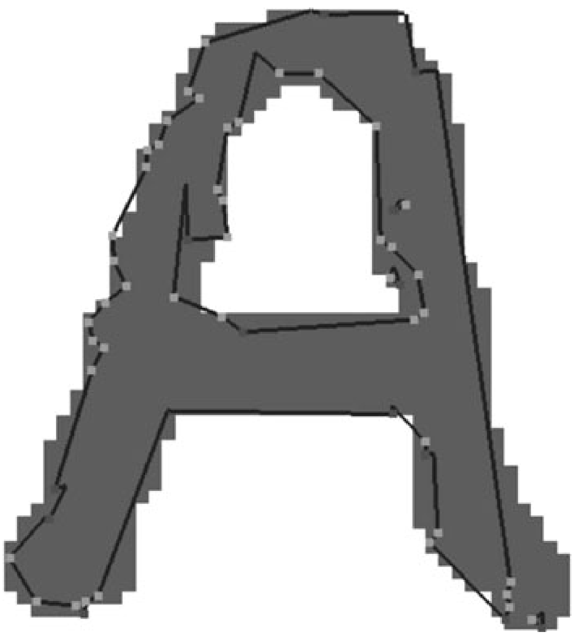


Fig. 15.2 Segmented handwritten A (30000 updates)

parameter σ_s should decrease over time to pass from global shape approximation to fine detail tuning at the later stages of the (OPT) dynamics. The update proposals for the polygonal configurations are accepted or rejected according to the standard simulated annealing scheme with time-dependent inverse temperature parameter β_s which increases over time—to be precise, our software employs a linear cooling schedule $\beta_s = \beta s$ for some constant $\beta > 0$.

15.6 Results and Discussion

In this final section we present applications of our algorithm on sample images. The software, implemented in D programming language, is in a rather early stage of development and will be further optimised. The segmentations shown in Figs. 15.2, 15.3 and 15.4 have been obtained after about 30000 (accepted) updates under a linear cooling schedule, with mean execution time 0.05 sec per single update on Intel Pentium M 2 GHz CPU and 2 GB RAM memory.

A large number of segmentation techniques are available in the literature. But, there does not exist a general algorithm that can perform the segmentation task for all images. Classification of image segmentation methods can be divided into several categories. Starting with the simplest one, the thresholding method, that uses a global property of the image, usually intensity, to classify individual pixels from the image as object pixels, if the value of the pixels property exceeds threshold value, or as background pixels otherwise. The main disadvantage of the method is a narrow range of application, because it works only for a subclass of images in which objects are distinct from background in intensity. The adjustment of the threshold parameter is also a nontrivial task and often requires human interaction. Another well known method is the K-means algorithm [293]. It is an unsupervised clustering algorithm that classifies the pixels from the image into multiple classes based on their inherent

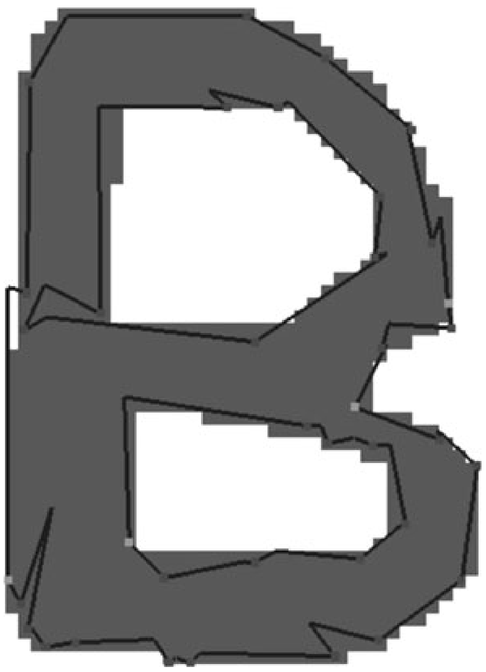


Fig. 15.3 Segmented handwritten B (30000 updates)

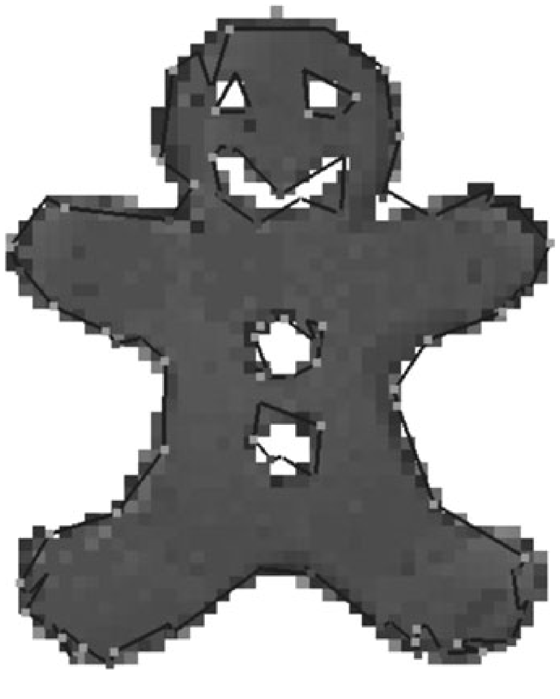


Fig. 15.4 Segmented gingerbread-man (30000 updates)

distance from each other. For small values of k the algorithm gives good results, but for larger values of k, the segmentation is very coarse, many clusters appear in the images at discrete places. Selection of parameter k is crucial in that algorithm and inappropriate choice may yield wrong results.

One of the most popular methods in segmentation uses morphological approach: the watershed transformation. That approach was introduced in [34] and consists of placing a spring of water in each selected region, the water will relief from sources, and construct barriers when water from different sources meet. The resulting barriers are the segmentation of the image. The main disadvantage of the algorithm is the need of human interaction to locate points where flooding should start. Another group of algorithms: the Markov random fields (MRF)-based methods are of great importance, for their ability to model a prior belief about the continuity of image features such as textures, edges or region labels [457], but obtain unsatisfied results when the prior knowledge is taken seriously.

The approach described here uses models that operate on the pixel level. Alternative intermediate level methods focus on the partition of the image that is the outcome of a segmentation. Green [190] and Møller and Skare [313] propose Voronoi-based models, and [318] suggests triangulations. One of the main advantage of our method is a higher conceptual level than most of listed algorithms i.e. the real world is not a collection of pixels and as we do not know what is in the image we cannot model the objects. The algorithm achieves reasonable global behaviour. Another benefit from our algorithm is easy and fast implementation. The drawbacks of our method can be seen around the edges, which will require more finetuning in the future.

Acknowledgements We gratefully acknowledge the support from the Polish Minister of Science and Higher Education grant N N201 385234 (2008–2010).

References

10. ARAK, T.: On Markovian random fields with finite number of values, *4th USSR-Japan symposium on probability theory and mathematical statistics, Abstracts of Communications*, Tbilisi (1982).
11. ARAK, T., SURGAILIS, D.: Markov Fields with Polygonal Realizations, *Probab. Th. Rel. Fields* **80**, 543-579 (1989).
12. ARAK, T., SURGAILIS, D.: Consistent polygonal fields, *Probab. Th. Rel. Fields* **89**, 319-346 (1991).
13. ARAK, T., CLIFFORD, P., SURGAILIS, D.: Point-based polygonal models for random graphs, *Adv. Appl. Probab.* **25**, 348-372 (1993).
34. BEUCHER, S., LANTUJOUL, C.: Use of watersheds in contour detection. *In International workshop on image processing, real-time edge and motion detection* (1979).
73. CHEN, K.: Simple learning algorithm for the traveling salesman problem, *Phys. Rev. E* **55**, 7809-7812 (1997).
79. CLIFFORD, P., AND MIDDLETON, R.D.: Reconstruction of polygonal images. *J. Appl. Stat.* **16**, 409–422 (1989).
190. GREEN, P.: Reversible jump MCMC computation and Bayesian model determination. *Biometrika* 82(4), 711732 (1995).
256. KLUSZCZYŃSKI, R., LIESHOUT, M.N.M. VAN, SCHREIBER, T.: An algorithm for binary image segmentation using polygonal Markov fields. In: F. Roli and S. Vitulano (Eds.), Image Analysis and Processing, Proceedings of the 13th International Conference on Image Analysis and Processing. *Lecture Notes in Comput. Sci.* **3615**, 383-390 (2005).
257. KLUSZCZYŃSKI, R., LIESHOUT, M.N.M. VAN, SCHREIBER, T.: Image segmentation by polygonal Markov fields. *Ann. Inst. Statist. Math.*, **59**, 465-486 (2007).
289. LIGGETT, T.: *Interacting particle systems*. Springer-Verlag, New York (1985).
293. LLOYD, S.P.: Least square quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2), 129137 (1982).
313. MOLLER, J., SKARE, O.: Bayesian image analysis with coloured Voronoi tesselations and a view to applications in reservoir modelling. *Stat. Model.* 1, 213232 (2001).
318. NICHOLLS, G.K.: Bayesian image analysis with Markov chain Monte Carlo and coloured continuum triangulation models. *J. R. Stat. Soc., Ser. B, Stat. Methodol.* 60, 643659 (1998).
319. NICHOLLS, G.K.: Spontaneous magnetization in the plane, *Journal of Statistical Physics*, **102**, 1229-1251 (2001).
333. PERETTO, P.: *An Introduction to the modeling of neural networks*, Collection Aléa-Saclay: Monographs and Texts in Statistical Physics 2, Cambridge University Press (1992).
377. SCHREIBER, T.: Random dynamics and thermodynamic limits for polygonal Markov fields in the plane, *Advances in Applied Probability* **37**, 884-907 (2005).
378. SCHREIBER, T.: Dobrushin-Kotecký-Shlosman theorem for polygonal Markov fields in the plane, *Journal of Statistical Physics*, **123**, 631-684 (2006).
379. SCHREIBER, T.: Non-homogeneous polygonal Markov fields in the plane: graphical representations and geometry of higher order correlations, *Journal of Statistical Physics*, **132**, 669-705 (2008).
380. SCHREIBER, T., LIESHOUT, M.N.M. VAN: Disagreement loop and path creation/annihilation algorithms for binary planar Markov fields with applications to image segmentation, *submitted* (2008).
395. SURGAILIS, D.: Thermodynamic limit of polygonal models, *Acta applicandae mathematicae*, **22**, 77-102 (1991).
424. LIESHOUT, M.N.M. VAN, SCHREIBER, T.: Perfect simulation for length-interacting polygonal Markov fields in the plane, *Scand. Journal of Statistics*, **34**, 615-625 (2007).
448. WINKLER, G.: Image analysis, random fields and Markov chain Monte Carlo methods, A mathematical introduction, *2nd ed. Applications of Mathematics, Stochastic Modelling and Applied Probability* **27**. Springer-Verlag, Berlin (2003).
457. YANG, F., JIANG, T.: Pixon based image segmentation withMarkov random fields. *IEEE Trans. Image Process.* 12(12), 15521559 (2003).

Paper D

Matuszak, M., Miękisz, J., Schreiber, T.

Solving Ramified Optimal Transport Problem in the Bayesian Influence Diagram Framework

Solving Ramified Optimal Transport Problem in the Bayesian Influence Diagram Framework

Michal Matuszak[1], Jacek Miękisz[2], and Tomasz Schreiber*[1]

[1] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopina 12/18, 87–100 Torun, Poland
{gruby, tomeks}@mat.umk.pl
[2] Institute of Applied Mathematics and Mechanics, University of Warsaw, Banacha 2, 02–097 Warsaw, Poland
miekisz@mimuw.edu.pl

Abstract. The goal of the ramified optimal transport is to find an optimal transport path between two given probability measures. One measure can be identified with a source while the other one with a target. The problem is well known to be NP–hard. We develop an algorithm for solving a ramified optimal transport problem within the framework of Bayesian networks. It is based on the decision strategy optimisation technique that utilises self–annealing ideas of Chen–style stochastic optimisation. Resulting transport paths are represented in the form of tree–shaped structures. The effectiveness of the algorithm has been tested on computer–generated examples.

Keywords: Optimal Transport Path, Transport Network, Branching Structure, Bayesian Influence Diagrams, Optimal Decision Strategies

1 Introduction

The transport problem was introduced by G. Monge in a very famous paper, *Mémoire sur la théorie des déblais et des remblais* [10,4]. Recently this classical problem has gained an extensive popularity [1,14]. The original problem is to move a pile of soil from one place to another with the minimal effort. In 1942, Kantorovich introduced his formalization of a relaxed version of the Monge's problem [7,2]. The task of finding optimal paths was transformed into the problem of transporting a positive measure μ_s onto another positive measure μ_d with the same mass. The Monge–Kantorovich approach assumes that the transport cost is proportional to the distance and the transported mass. It favours thin routes rather than wide ones. Unfortunately it is not practical from the economic point of view.

In most transport networks, sending each particle straight to the destination is economically unrealistic. The preferable solution is to aggregate particles and move them together as it happens in tree leafs or on highways. We should mention

* Deceased author (1975 – 2010).

the Steiner tree problem where one minimizes only the total length of a network [13] and omits the cost of constructing edges. His model is not appropriate for our purposes because it does not discriminate the cost of high or low capacity edges; constructing a high–capacity highway is more expensive than constructing a backroad.

The first model taking into account the cost of edges was introduced by Gilbert and it has been extensively investigated in [14]. He showed that in shipping two objects from nearby cities to the same far away city, see Fig. 1(c), it may be more optimal to first transport them to a common location and then transport them together to the target. In this case, a Y shaped path is preferable to a V shaped path, see Fig. 1. In general, resulted paths form leaf–like structures. Biological leafs tend to maximize an internal efficiency by developing an efficient transport system for water and nutrients [16]. We should note that the presented problems are NP–hard [4,13].

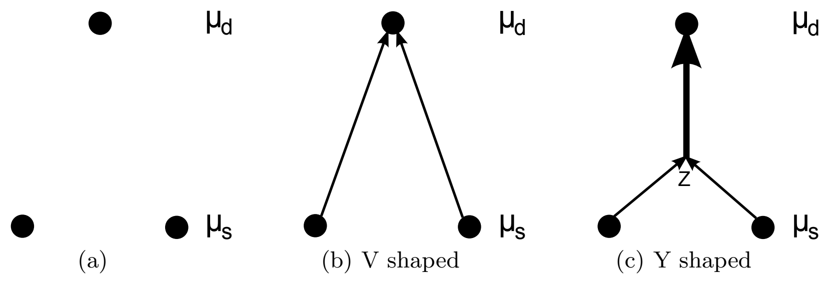


Fig. 1. (a) Three cities, two of them at the bottom are the source and the third city at the top is the destination for transported goods, (b) Monge–Kantorovich solution, (c) Gilbert solution, the interior vertex z can be determined analytically [14].

An influence diagram [6,11,12] is an extension of a Bayesian network [8,6,11,12], in which not only a probabilistic inference occurs but also decision making problems are solved. Influence diagrams are built on a directed acyclic graphs (DAGs) whose nodes and edges have standard interpretations stemming from and extending those used for Bayesian networks.

An influence diagram, similar to a Bayesian network, can be built with the use of chance nodes, which we represent as ovals. Also two additional types of nodes are introduced: decision nodes corresponding to available decisions (rectangles) and utility nodes (rhombi) specifying payoff functions (utilities) to be maximized by suitable choices of decision policies.

If the network is well designed, then the arcs leading to chance nodes specify direct causal relationships not necessarily corresponding to any temporal ordering. The arcs leading to decision nodes indicate the information available at the moment of decision making, thus feeding input to decision policies. The influence diagrams can be considered as generalizations of (symmetric) decision trees, see [6].

Finding an optimal decision strategy for an influence diagram is an NP–hard task. One can show this easily by reducing the traveling salesman problem (that is NP–complete) to our task.

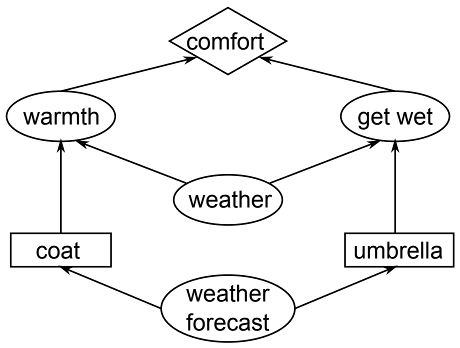


Fig. 2. Illustration of a simple influence diagram that includes two decision nodes: whether or not to take an umbrella and/or a coat for a journey. The decisions have an impact on the warmth and can prevent from getting wet from the rain, but if it does not rain, then carrying an umbrella has a negative impact on the mood.

In Fig. 2, an example of an influence diagram is shown. The decision node *umbrella* represents the choice whether or not to take an umbrella. Taking an umbrella does increase the chance of not getting wet during rain, yet it also causes negative effects, such as the need of carrying an additional weight. Further, wearing a *coat* decreases the chance of getting chilled but also negatively affects our comfort if the outside temperature is too high. All these effects are jointly taken into account in the utility node *comfort*.

There exists a number of algorithms for finding an optimal decision strategy for an influence diagram. For a detailed discussion we refer the reader to Chapter 10 in [6] and Subsection 5.2.2 in [11]. We focus our attention on the Chen–style [5] stochastic optimisation algorithm described in [9] which is well suited for our task.

In Section 2, we formalize the transport problem. Then we outline the transformation of the problem into influence diagrams. Results, technical details, and a discussion are contained in following sections.

2 Optimal transport problem

In this section we recall some concepts of Xia [14,15,16] concerning optimal transport paths between measures.

Let (X, d) be a metric space. We define an atomic measure on X as follows

$$a = \sum_{i=1}^{k} a_i \delta_{x_i} \tag{1}$$

for some integer k and points $x_i \in X$, a_i are positive numbers and δ_{x_i} is the Dirac mass located at the point x_i. We will work with the probability measures, i.e. we assume that $\sum_{i=1}^{k} a_i = 1$.

Let $A(X)$ be the space of all atomic probability measures on X. For measures on X,

$$\mu_s = \sum_{i=1}^{k} s_i \delta_{x_i} \text{ and } \mu_d = \sum_{j=1}^{n} d_j \delta_{y_j} \tag{2}$$

a **transport path** from μ_s (source) to μ_d (destination) is defined as a weighted directed acyclic graph (DAG) $G = (V_G, E_G)$, where V_G is a set of vertices such that $\{x_1, x_2, \ldots, x_k\} \cup \{y_1, y_2, \ldots, y_n\} \subset V_G$ and E_G is a set of directed edges with a weight function

$$w : E_G \to (0, +\infty). \tag{3}$$

Hence V_G consist of source, destination and intermediate vertices, see for example Fig. 1(c) and Fig. 3. The value $w(e)$ can be identified with the amount of goods transported along the edge e.

The balance equation for every $v \in V_G$

$$\sum_{e \in E_G, e^- = v} w(e) = \sum_{e \in E_G, e^+ = v} w(e) + \begin{cases} s_i \text{ , if } v = x_i \text{ for some } i = 1, \ldots, k \\ -d_j \text{ , if } v = y_j \text{ for some } j = 1, \ldots, n \\ 0 \text{ , otherwise} \end{cases} \tag{4}$$

where e^- denotes the first vertex of the edge $e \in E_G$ and e^+ is the second vertex. It simply means that the total mass flowing into $v \in V_G$ equals to the total mass flowing out of v.

For any $0 \leq \alpha \leq 1$ and any transport path G, we define the **path cost** function $w_p(G, \alpha)$ as follows

$$w_p(G, \alpha) = \sum_{e \in E_G} \|e\| * [w(e)]^\alpha \tag{5}$$

where $\|e\|$ denotes the length of the edge e.

The ramified optimal transport problem focuses on finding a transport path from μ_s to μ_d which minimizes $w_p(G, \alpha)$. The minimizer is called an optimal transport path. In other words, for a given G, and α we have to create a weight function such that (4) is satisfied and (5) is minimized.

3 The algorithm

In this section, the formal description of our algorithm is given. First, we define the total cost function which is minimized during the optimization phase. Then we present the transformation of the optimal transport problem into an influence diagram and finally we translate an optimized decision policy into an optimal transport path.

So far we have assumed that each destination node receives a specific amount of mass. Such a strict constraint prevents us from applying many optimization techniques so we relax the above assumption and introduce a **disagreement cost** for a DAG G,

$$w_d(G) = \sum_{j=1}^{n} \left[d_j - \sum_{e \in E_G, e^+ = v} w(e) \right]^2 \tag{6}$$

which characterizes the difference between a shipped and expected mass.

We define the **total cost function** which is based on (5) and (6),

$$w(G, \alpha, c_1, c_2) = c_1 w_p(G, \alpha) + c_2 w_d(G), \tag{7}$$

where c_1 and c_2 are weights which control the importance of the **disagreement cost** and the **path cost**. The objective is to minimize the total cost function in (7) which is identified with the $-$payoff ("minus" because it is assumed that we maximize the payoff function) in the influence diagram.

Let us assume that an influence diagram $(\mathcal{S}, \mathcal{P}, \mathcal{U})$ is given, which is built on a connected DAG $\mathcal{S}$, with conditional probability tables (CPTs) $\mathcal{P}$ and utility functions $\mathcal{U}$. The set of nodes in $\mathcal{S}$ splits into chance nodes $C_\mathcal{S}$, decision nodes $D_\mathcal{S}$ and utility nodes $U_\mathcal{S}$. An influence diagram that describes an optimal transport problem is constructed in the following way:

– $C_\mathcal{S} = \emptyset$
– $D_\mathcal{S} = V_G \backslash \{y_j \| j = 1, \ldots, n\}$
– $U_\mathcal{S} = \{y_j \| j = 1, \ldots, n\}$

In addition, for each decision node $D \in D_\mathcal{S}$, a *randomised policy* τ_D is attached. It assigns to each configuration $\bar{w}$ of $\mathrm{pa}(D)$ (where $\mathrm{pa}(D)$ is a set of parents of node D) a probability distribution on possible decisions to be taken, that is to say $\tau_D(d|\bar{w})$ stands for the probability of choosing a decision d given that $\mathrm{pa}(D) = \bar{w}$. These randomised policies will evolve in the course of the optimisation process, eventually to become (sub)optimal deterministic policies which collectively determine the utility maximizing strategy for the influence diagram considered. The initial choice of $\tau_D,\ D \in D_\mathcal{S}$ can be either *uniform*, with all routes equiprobable, or *heuristic*, provided some additional knowledge is available allowing us to make a good *first guess* about the optimal path.

The connections in $\mathcal{S}$ are replicated from the set of edges E_G. If V_G consists only of source and destination vertices and does not have intermediate ones, then we can add them either

– uniformly – producing a regular grid of vertices
– heuristically – an additional knowledge about preferred paths is provided
– randomly – all parts of the space are treated with equal probabilities

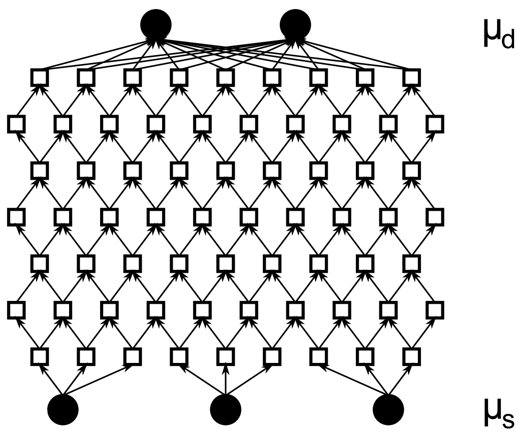


Fig. 3. Representation of an influence diagram used in the algorithm. Squares describe intermediate vertices where junctions can occur and dots represent source and destination of the mass.

Next we have to define connections between them. To preserve an acyclic property and equality of routes we assume that the atomic measures (the source and the destination) can be spatially separated by a hyperplane. Edges can by defined as follows,

1. For each decision node we define a maximal number of children k_d
2. $Q = \{x_i \| i = 1, \dots, k\}$ and $R = \emptyset$
3. For each $q \in Q$, find k_q nearest neighbours of q, set them as children of q, and add to R.
4. If $R \neq \emptyset$ then $Q = R$ and $R = \emptyset$ and go to 3.

For such an influence diagram it is feasible to use the stochastic optimisation algorithm from [9]. Description of the algorithm falls beyond the scope of the present article. Results of the algorithm will be stored in the *randomised policy* τ_D. Using computed policies we can easily determine the optimal paths. Each policy describes where and how we should transport the incoming mass. Starting from roots of DAG we transport the source mass to the children according to computed policies.

4 Examples

In the first example, presented in Fig. 4, we reproduced the Gilbert solution from Fig. 1(c). The angle between merging edges was computed in [14] and is equal to $arccos(2^{2\alpha-1} - 1)$. Expected solution for $\alpha = 0.7$ is 71.36 degrees and the experimental value obtained from presented algorithm is equal to 73.5 degrees and highly depends on the distribution of decision nodes. The second example is presented in Fig. 5. Simulation of the influence diagram from Fig. 5 required 160 ms time per epoch of the algorithm [9]. The resulted transport path follows the expectations and results from [16]. It favours high capacity roads over narrow ones.

The programme has been implemented in language `D` [3], currently gaining popularity as a natural successor of `C++`. The implementation, aimed so far mainly at algorithm evaluation purposes, can be described as careful but not fully performance–optimised, with the total utility evaluation performed using the standard Monte–Carlo rather than a more refined and effective scheme. All tests were performed on a machine with Intel Core 2 Q9300 2.50 GHz CPU and 4GB RAM.

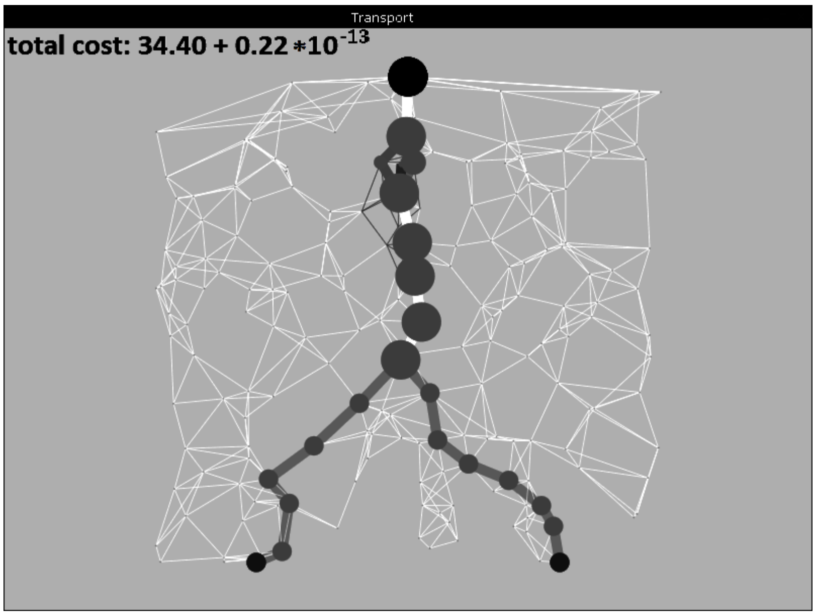
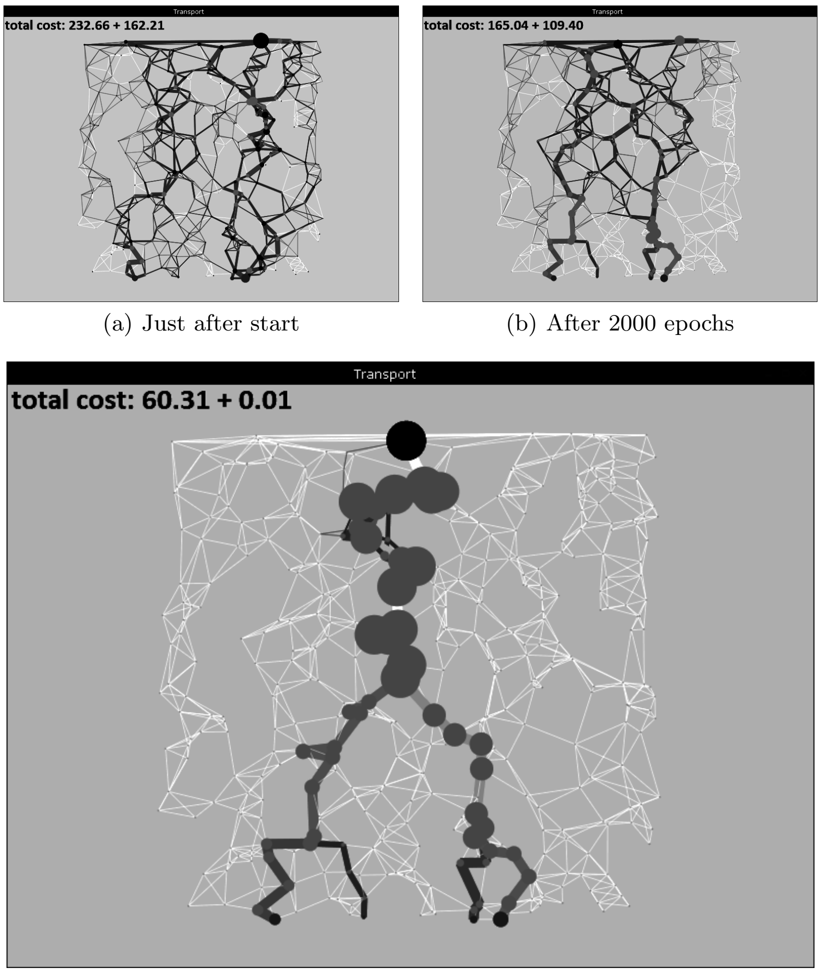


Fig. 4. Results of the algorithm on a graph that has 2 source nodes, one destination node and 200 randomly distributed decision nodes. Resulted transport path follows the Gilbert solution, see Fig. 1(c). In the upper left corner the total cost is presented in the form given by Eq. 7. Parameter α was set to 0.7, $s_1 = s_2 = 0.5$ and $d_1 = 1$.

5 Conclusions

A new stochastic algorithm for solving transportation problem has been presented. The main advantage of the introduced method is its innovative application of Bayesian influence diagrams. Experimental results indicated the correctness of the algorithm. In the first test, the analytical result has been reproduced



(a) Just after start (b) After 2000 epochs

(c) After 5000 epochs

Fig. 5. Results of the algorithm on a graph that has 4 source nodes, one destination node and 400 randomly distributed decision nodes. In the upper left corner the total cost is presented in the form given by Eq. 7. Parameter α was set to 0.7, $s_1 = 0.2$, $s_2 = 0.4$, $s_3 = 0.3$, $s_4 = 0.1$ and $d_1 = 1$.

and in the second one our expectation for the solution has also been met. The Chen's style algorithm for solving Bayesian influence diagram has been shown as a powerful tool able to find other applications in machine learning related problems.

Acknowledgements

This research has been supported by the National Science Centre grant 2011/01/N/ST6/00573 (2011-2014). The authors gratefully acknowledge the access to the PL–Grid[3] infrastructure that is co–funded by the European Regional Development Fund as a part of the Innovative Economy program. The work of M. Matuszak has also been supported by the European Social Fund as a part of the Sub–measure 4.1.1 (National PhD Programme in Mathematical Sciences).

References

1. Ambrosio, A. Lecture Notes on Optimal Transport Problems, Scuola Normale Superiore, Pisa (2000).
2. Ambrosio, A. Optimal transport maps in Monge–Kantorovich problem, *Proceedings of the ICM, Beijing* 3: 131–140 (2002).
3. Bell, K., Igesund, L.I., Kelly, S. Parker, M. Learn to Tango with D, Apress (2008).
4. Bernot M., Caselles V., Morel J.-M. Optimal Transportation Networks, *Lecture Notes in Mathematics 1955* (2009).
5. Chen, K. Simple learning algorithm for the traveling salesman problem,*Phys. Rev. E* 55: 7809–7812 (1997).
6. Jensen, F.V., Nielsen, T.D. Bayesian Networks and Decision Graphs, 2nd Ed., *Springer* (2007).
7. Kantorovich, L.V. On the transfer of masses, *Dokl. Akad. Nauk. SSSR* 37: 227–229 (1942).
8. Koski, T., Noble, J. Bayesian Networks: An Introduction, *John Wiley & Sons, Ltd* (2009).
9. Matuszak, M., Schreiber, T. A new stochastic algorithm for strategy optimisation in Bayesian influence diagrams, LNAI 6114: 574–581 (2010).
10. Monge, G. Mémoire sur la théorie des déblais et des remblais, Histoire de l'Académie Royale des Sciences de Paris, 666–704 (1781).
11. Neapolitan, R. E. Learning Bayesian Networks, *Prentice Hall Series in Artificial Intelligence*, *Pearson Prentice Hall* (2004).
12. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, *Morgan Kaufmann Publishers Inc.* (1988).
13. Vazirani, V. V. Approximation Algorithms *Springer–Verlag*, Berlin (2001).
14. Xia, Q. Optimal paths related to transport problems, *Communications in Contemporary Mathematics* 5: 251–279 (2003).
15. Xia, Q. Ramified optimal transportation in geodesic metric spaces, *Adv. Calc. Var.*4: 277–307 (2011).
16. Xia, Q. The formation of a tree leaf, *ESAIM. COCV* 13: 359–377 (2007).

[3] `http://www.plgrid.pl`

Paper E

Matuszak, M., Miękisz, J.

Stochastic Techniques in Influence Diagrams for Learning Bayesian Network Structure

Stochastic Techniques in Influence Diagrams for Learning Bayesian Network Structure

Michal Matuszak[1] and Jacek Miękisz[2]

[1] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopina 12/18, 87–100 Torun, Poland
gruby@mat.umk.pl

[2] Institute of Applied Mathematics and Mechanics, University of Warsaw, Banacha 2, 02–097 Warsaw, Poland
miekisz@mimuw.edu.pl

Abstract. The problem of learning Bayesian network structure is well known to be NP–hard. It is therefore very important to develop efficient approximation techniques. We introduce an algorithm that within the framework of influence diagrams translates the structure learning problem into the strategy optimisation problem, for which we apply the Chen's self–annealing stochastic optimisation algorithm. The effectiveness of our method has been tested on computer–generated examples.

Keywords: Bayesian Networks, Structure Learning, Chen Adaptive Optimisation, Influence Diagrams.

1 Introduction

Bayesian networks represent probabilistic relationships among given variables. They are built on directed acyclic graphs (DAG) in which nodes represent random variables (ovals in our figures below) and direct edges between nodes represent the probabilistic dependencies between them. Conditional probabilities for variables are stored in conditional probability potentials (or tables) attached to dependent nodes.

There are two basic learning problems in Bayesian networks: learning the structure of a graph and learning the conditional probability potentials. It is fairly easy to learn parameters of a given DAG (see [6,8,9,12]). One approach is to compute frequencies that are optimal with respect to the maximum likelihood estimation (MLE). Here we focus on the task of learning the structure of a DAG from a given dataset. It has many important applications in various fields like classification and variable selection, and bioinformatics, where it is used for locating gene regulatory pathways (check [9] for more applications).

Learning Bayesian network structure is NP–hard even for networks with two parents [4]. Chow and Liu [5] showed that trees can be learned in a polynomial time but it has been shown in [7] that even learning 2–polytrees is an NP–hard problem. A polytree is a DAG with the property that if the directions on edges are ignored, it results in an undirected graph with no cycles.

There are many learning algorithms (see [6,8,9,12]) for building structure of Bayesian networks. Generally, they can be divided into three main groups: `constraint based` algorithms, `search and score` techniques, and `hybrid` methods which combine the first two methods. The constraint based algorithms perform a study of the dependence and independence relationships among variables of the Bayesian network. They are performed by conditional independence tests. For large enough datasets, χ^2 or G^2 tests can be performed and for the smaller ones exact tests could be done. Main problems of those algorithms are: the time complexity of the independence tests, unreliable results of the independence tests, and also the fact that the most widely used algorithms require an existence of a faithful graph [9].

The search and score techniques attempt to find a graph structure that maximizes the value of a given scoring function. A brief description of these techniques is provided in the next section.

An influence diagram is an extension of a Bayesian network. They both are built on DAG's and consist of chance nodes, while influence diagrams also have decision and utility nodes. They not only provide tools for a probabilistic inference but also provide a language for sequential decision making problems, where there is a fixed order among the decisions.

In [3], a simple stochastic optimisation algorithm for the traveling salesman problems (TSP) has been proposed. Then in [11], Chen's ideas have been substantially extended in order to construct an algorithm for solving general influence diagrams. It shows a strong performance in optimal transport problems [10]. In this paper, we translate the structure learning problem into the one involving influence diagrams and we show that it can be solved with an extended version of the algorithm presented in [11].

2 Learning Bayesian Networks Structure

Formally, a Bayesian network is a pair (G, θ), where $G = (C, E)$ is a directed acyclic graph (DAG), $C = \{X_1, \ldots, X_n\}$ consists of n random variables, E represents direct dependencies between variables, and θ represents a set of parameters for each variable in C, which defines their conditional probability distributions. Each random variable X_i has values in a finite domain K_i.

The problem of learning a Bayesian network structure is given as follows: For a set of random variables $C = \{X_1, \ldots, X_n\}$ and a database of m cases $M = \{M_1, \ldots, M_m\}$, where each case contains observations of all variables in C, i.e. $M_i = (x_1, \ldots, x_n)^T$ is a vector of instances of variables $X_1, \ldots, X_n$, find a DAG (that is a set of directed edges) which best matches M.

As stated before, we focus our attention on `search and score` techniques. A search space of possible DAG's grows super–exponentially [14], so testing all possible DAG patterns is computationally unfeasible. Scoring functions can be divided into two main classes: `Bayesian scoring` functions such as K2, the mutual information test (MIT), the Bayesian Dirichlet test and its variants (BD, BDe, BDeu), and `information-theoretic` scoring functions such as the log likelihood (LL), the Bayesian information criterion (BIC), and the Akaike information criterion (AIC).

Here we will use a modification of the Cooper–Herskovits likelihood (belonging to the Bayesian scoring class) [6] for a DAG G and a dataset M with $P(G)$ as a prior probability of G. It has the following form,

$$P(G,M) = P(G) \times \prod_{k=1}^{n} \prod_{j=1}^{|\phi_k|} \frac{(s_k-1)!}{(s_{kj}+s_k-1)!} \prod_{l=1}^{s_k} \alpha_{kjl}! \tag{1}$$

where s_k is a number of states of the variable X_k, ϕ_k is a variable describing joint configurations of variables in $\pi(X_k)$ ($\pi(X_k)$ denotes the set of parents of X_k), $|\phi_k|$ is a number of states of ϕ_k, and α_{kjl} is a number of cases in M in which X_k is at the l–th state and ϕ_k is at the j–th state. Also $s_{kj} = \sum_{l=1}^{s_k} \alpha_{kjl}$.

The evaluation of a broad spectrum of scoring functions can be found in [2]. It is stated there that there are only small differences between various scoring functions and all of them behave in a similar way (only the BIC score was clearly the worst). Thus, we use the K2 metric [6] which is a slight modification of the Cooper–Herskovits likelihood from Eq. 1. Its objective is to find the most probable network structure, with a given data set, which maximizes the posterior probability distribution. It assumes a uniform prior $P(G)$ and instead of $P(G,M)$ uses $log(P(G,M))$.

3 The Algorithm

Now we give a formal description of our algorithm. Let us assume that a set of chance variables $C = \{X_1, \ldots, X_n\}$ and a database of cases $M = \{M_1, \ldots, M_m\}$ are given. Both objects are fixed during the execution of the algorithm. In addition, to each chance variable $X_i \in C$ a decision node D_i is attached. Decision nodes play crucial role in the algorithm. During the optimization procedure, they unfold a structure, that is a set of directed edges E of the Bayesian network. Our algorithm is based on an extension of ideas from [3,11].

If nodes ordering is not given, then each decision node D_i has n states. First $n-1$ states describe available connections with chance variables $C \setminus X_i$ and the $n-$th state is used during the optimization process and is applied to disable connections of further children. The proposed algorithm does not require a node ordering, however, it may benefit from a predetermined ordering as the search space will be reduced.

For each decision node D_i, a *randomised policy* τ_i is attached. It assigns probabilities to all possible decisions that may be taken, with $\tau_i^{X_j}$ standing for the probability of adding a direct edge from X_i to X_j. In the course of the optimisation process, these randomised policies evolve and include an (sub)optimal structure of the network. The initial choice of τ_i can be either *uniform* or *heuristic*. In the *uniform* choice, all decisions are equiprobable and in the *heuristic* case, some additional knowledge is provided allowing us to make a good first guess about the optimal structure. For example if we know that an edge is more probable than others starting from the same node, then we can increase it's probability and thus making the edge appear with a higher probability in the optimization procedure. If we have a knowledge that some edges exist in the network, then we can add them as *permanent edges* and they will always be included in the network structure.

[**Permanent edge condition**] During the iterative procedure, the *randomised policies* should converge to Dirac deltas which results in an almost deterministic selection of edges. It means for the node $X_i \in C$, that if

$$1 - \tau_i^{X_j} < \epsilon \tag{2}$$

for some node $X_j \neq X_{None}$ and a small fixed ϵ, a direct edge $X_i \rightarrow X_j$ will be added to E. However, if $X_j = X_{None}$, then X_i is permanently removed from active nodes, resulting in no further addition of children. This does not however restricts its possibility of being a child.
It is possible that two (or more) decisions are equiprobable and then τ_i is almost a uniform distribution over them, while the probabilities of other decisions are near 0. Therefore, if Eq. 2 is not satisfied after a fixed number of steps Q, we should randomly choose a decision (that is a vertex to connect) according to the probability distribution τ_i and if decision X_{None} is drawn, then we deactivate X_i, and in other cases we add a direct edge $X_i \rightarrow X_j$ to E.

Each decision node has also an *active* field with states $\{enabled, disabled\}$ which describe whether the node is subject to the optimization procedure. At the initialization of the algorithm, all nodes are *active*. When node's optimization is over (see *permanent edge condition*), then it becomes passive and it can not be reactivated.
The algorithm works as follows.

1. Set the iteration counter $j = 0$.
2. Attach and initialize decision nodes as described above.
3. Generate an instance of the network:
   (a) If predefined edges are given, include them in the network and follow the actions from *permanent edge condition.*
   (b) Select randomly (with a uniform distribution without replacement) an active node D_i.
   (c) According to the distribution τ_i draw a new vertex X_h and if $X_h \neq X_{None}$ and a direct edge from X_i to X_h preserve acyclicity of the graph, then add that edge.
   (d) Go back to Step 3b until each active node is chosen.
4. Select randomly (with a uniform distribution) an active node D_i and denote its state as h_0. It is an index that can be used to select the h_0–th chance node (X_{h_0}) or h_0–th decision node (D_{h_0}).

(a) If h_0 corresponds to an existing edge $(X_i \rightarrow X_{h_0})$, then remove the edge.
(b) According to the probability distribution τ_i draw a new vertex X_{h_1} and if $X_{h_1} \neq X_{None}$, then add a direct edge from X_i to X_{h_1}. In other words, during the search space step we can add or remove an edge.
(c) Accept the modification if the following conditions are met:
  – a new vertex X_{h_1} is different from the current vertex X_{h_0}.
  – a graph is acyclic.
  – the weakly connected property is preserved i.e. the skeleton (undirected graph obtained from replacing directed edges with undirected ones) is connected. The property can be verified with using either depth-first or breadth-first search algorithm.

  else go to Step 4.

5. Evaluate the utility function $U^{(j)}$ using, for example, the K2 metric.
6. Check the *permanent edge condition* and if $X_i \rightarrow X_{h_1}$ has been added to E, remove from the support of τ_i the decision to link to the node X_{h_1} and from the support of τ_{h_i} the decision to link to the node X_i . Reinitialize the values of τ_i with the uniform distribution (or apply a predefined knowledge to the distribution). For τ_{h_1}, normalize the weights (by dividing them by their sum) to achieve a probability distribution.
7. IF j mod $B \neq 0$ set
$$\Delta := U^{(j)} - U^{(j-1)} \tag{3}$$
then update the randomised policy like in the Chen's algorithm,
$$\tau_i^{X_{h_1}} = \exp(\beta\Delta)\tau_i^{X_{h_1}}, \tag{4}$$
and renormalise τ_i so that it remains a probabilistic distribution. The important parameter of the algorithm is constant β which describes the rate of learning (larger β speeds up learning but decreases the optimization's stability). β represents the noise level in the algorithm, it corresponds to the inverse of the temperature in physical systems.
8. Set $j = j + 1$ and if there still exists an active node then:
   – if j mod $B = 0$ return to Step 3. For small enough β the initial order of D_i has only a small impact on the algorithm. However, in rare situations even for small β impact of the first selection of the nodes could cause divergence of the algorithm. To resolve that issue and to allow for larger values of β (which speed up the optimization) we implement *restarting after fixed number of steps* i.e a new order of nodes is generated after B steps.
   – else return to Step 4

In Fig. 1, we present an application of our algorithm to a network with three chance nodes: X_1, X_2, and X_3. After the initialization phase the decision nodes and randomised policies have been attached. The utility node U, which computes the score function, has been added and connected with chance nodes.

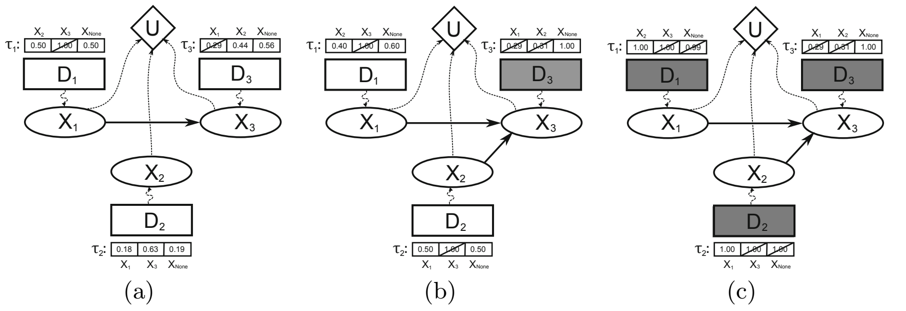

τ1: X2 0.50, X3 1.00, XNone 0.50
τ3: X1 0.29, X2 0.44, XNone 0.56
τ2: X1 0.18, X3 0.63, XNone 0.19
U
D1
D2
D3
X1
X2
X3
(a)
τ1: X2 0.40, X3 1.00, XNone 0.60
τ3: X1 0.29, X2 0.31, XNone 1.00
τ2: X1 0.50, X3 1.00, XNone 0.50
(b)
τ1: X2 1.00, X3 1.00, XNone 0.99
τ3: X1 0.29, X2 0.31, XNone 1.00
τ2: X1 1.00, X3 1.00, XNone 1.00
(c)

Fig. 1. An example of the algorithm executed on a network with three chance nodes

In our example, no prior knowledge is available, thus the randomised policies are represented by uniform distributions. All nodes are active and the algorithm can proceed. An instance of the network is generated and during the iterative procedure the search space of possible DAG's is explored, and randomised policies τ_i are modified with the use of the Chen's formula [3]. Fig. 1(a) presents the network state after $\tau_1^{X_3}$ converged to 1. A permanent direct edge $X_1 \rightarrow X_3$ is added, the policies ($\tau_1^{X_3}$ and $\tau_3^{X_1}$) associated with it are deleted, τ_1 is initialized with the uniform distribution and weights in τ_3 are renormalized.

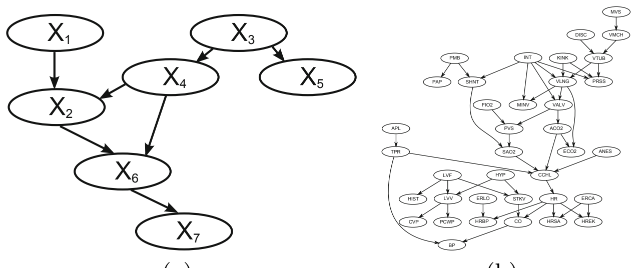
In Fig. 1(b), $\tau_2^{X_3}$ converges to 1. A permanent direct edge $X_2 \rightarrow X_3$ is added and the policies $\tau_2^{X_3}$, and $\tau_3^{X_2}$ are deleted. The only available policy in τ_3 is $\tau_3^{X_{None}}$ so no further children can be connected to X_3 and the decision node D_3 can be *deactivated* (excluded from the optimisation procedure). Weights in τ_1 are initialized with the uniform distribution and we return to the iterative procedure. Fig. 1(c) presents the network status after $\tau_2^{X_{None}}$ converged to 1, so no edge is added and D_2 is *deactivated.* The only possible edge that still can be added is $X_1 \rightarrow X_2$, but $\tau_1^{X_{None}}$ also converged to 1, and D_1 is *deactivated.* All decision nodes are inactive, thus the algorithm stops and returns a set of directed edges E.

4 Numerical Examples

The programme has been implemented in the language `C++`, with the implementation aimed so far mainly at algorithm evaluation purposes, it can be described as careful but not fully performance–optimised.

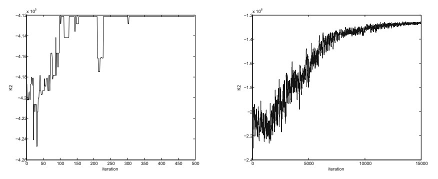
We have selected two networks for numerical experiments: a *simple Bayesian network* with 7 nodes and 7 edges (Fig. 2(a)) and *ALARM network* [1] which contains 37 nodes and 46 edges (Fig. 2(b)). Each network has been used to generate a database, which contains 100000 instances.

For the *simple Bayesian network* with $\beta = 0.001$, $\epsilon = 0.01$, $B = 100$ and the number of iterations limited to $Q = 500$, a final network differs from the optimal



(a) (b)

Fig. 2. (a) Simple Bayesian network; (b) ALARM network



(a) (b)

Fig. 3. Convergence of the algorithm for (a) Simple Bayesian network and (b) ALARM network

	Simple net	ALARM net
K2	0.195	99
Chow–Liu	0.1	12
Our algorithm	0.9	917

Table 1. Computational times (in seconds)

one in K2 metric by 0.11% (for the Chow–Liu tree it is 0.73%, and for classical K2 algorithm [6] it is 1%), and for the *ALARM network* with $\beta = 9 \times 10^{-6}$, $\epsilon = 0.01$, $B = 250$ and iterations limited to $Q = 15000$, the difference in K2 metric is 20% (16% for the Chow–Liu tree, and for classical K2 algorithm it is 33%).

In Fig. 3, we show the convergence of our algorithm, that is the maximization of the K2 metric. In Table 1, we compare computational times of our and other algorithms.

5 Conclusions

A new stochastic algorithm for finding Bayesian network structure has been presented. Our method is based on an innovative application of Bayesian influence diagrams for structure optimization. The main advantage of the introduced method is the use of an algorithm that can determine an optimal decision strategy for a different problem. Numerical results indicate the correctness of the presented algorithm. Although computational times of our algorithm are not optimal (see Table 1) our results are competitive as compared to classical ones.

Acknowledgements. This research has been supported by the National Science Centre grant 2011/01/ N/ST6/00573 (2011-2014). The authors gratefully acknowledge the access to the PL–Grid[1] infrastructure, that is co–funded by the European Regional Development Fund as a part of the Innovative Economy program. The work of M. Matuszak has also been supported by the European Social Fund as a part of the Sub–measure 4.1.1 (National PhD Programme in Mathematical Sciences).

References

1. Beinlich, I., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In: Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine, pp. 247–256 (1989)
2. de Campos, L.M.: A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. Journal of Machine Learning Research 7, 2149–2187 (2006)
3. Chen, K.: Simple learning algorithm for the traveling salesman problem. Phys. Rev. E 55, 7809–7812 (1997)
4. Chickering, D.M.: Learning Bayesian networks is NP–complete, Learning from Data: Artificial Intelligence and Statistics V, pp. 121–130. Springer (1996)
5. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Info. Theory 14(3), 462–467 (1968)
6. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks. Data Machine Learning 9, 309–347 (1992)
7. Dasgupta, S.: Learning polytrees. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 131–141. Morgan Kaufmann, Stockholm (1999)
8. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer (2007)
9. Koski, T., Noble, J.: Bayesian Networks: An Introduction. John Wiley & Sons, Ltd. (2009)
10. Matuszak, M., Miękisz, J., Schreiber, T.: Solving Ramified Optimal Transport Problem in the Bayesian Influence Diagram Framework. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS (LNAI), vol. 7268, pp. 582–590. Springer, Heidelberg (2012)
11. Matuszak, M., Schreiber, T.: A New Stochastic Algorithm for Strategy Optimisation in Bayesian Influence Diagrams. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part II. LNCS, vol. 6114, pp. 574–581. Springer, Heidelberg (2010)
12. Neapolitan, R.E.: Learning Bayesian Networks. Prentice Hall Series in Artificial Intelligence. Pearson Prentice Hall (2004)
13. Peretto, P.: An Introduction to the Modeling of Neural Networks, Collection Aléa–Saclay. Cambridge University Press (1992)
14. Robinson, R.W.: Counting unlabelled acyclic digraphs. In: Little, C.H.C. (ed.) Combinatorial Mathematics V. Lecture Notes in Mathematics V, pp. 28–43. Springer (1977)

[1] http://www.plgrid.pl