Master Thesis Proposal

Decision Chain Graphs

Jose M. Peña
STIMA, IDA, LiU
jose.m.pena@liu.se

1 Introduction

Reasoning under uncertainty is ubiquitous in fields like statistics and artificial intelligence. Decision theory provides a formal framework to compute decision strategies yielding maximum expected utility for an agent when observations and outcomes of actions are uncertain. In the beginning, these decision problem were solved by applying dynamic programming to decision trees. Later, influence diagrams were developed for the task. Influence diagrams can be seen as an extension of Bayesian networks with decision and utility variables. Influence diagrams are more compact, intuitive, structured and scalable than decision trees. Solving an influence diagram (i.e., finding the strategy with maximum expected utility) is typically done via the Lauritzen-Spiegelhalter algorithm developed for making inference on Bayesian networks. In this project, we focus on limited memory influence diagrams (LIMIDs). The main advantage of LIMIDs over influence diagrams is that they can model decision problems where the no-forgetting assumption does not hold, i.e. LIMIDs do not require that all the previously made observations and decisions are known when making a new decision. This is particularly suitable for decision problems involving an agent (e.g., a team or corporation) composed of subagents (e.g., individuals or robots) which may not be aware of each other's previous decisions and observations. However, LIMIDs do not allow for simultaneous and coordinated decisions, e.g. a doctor may have to decide the dosage of several drugs simultaneously and coordinately to maximize the effectiveness of a treatment. This project aims to address this problem.

2 LIMIDs and DCGs

Formally, a LIMID consists in a directed and acyclic graph (DAG) G over $V \cup D \cup U$ where

- $V = \{V_i\}$ is a set of discrete random variables, i.e. controlled by nature
- $D = \{D_i\}$ is a set of discrete decision variables, i.e. controlled by the agent
- $U = \{U_i\}$ is a set of continuous utility variables, i.e. the agent's preferences

and such that

- $pa(D_i)$ are the random/decision variables to observe/make and take into consideration before deciding on D_i
- $pa(U_i)$ are the variables determining U_i

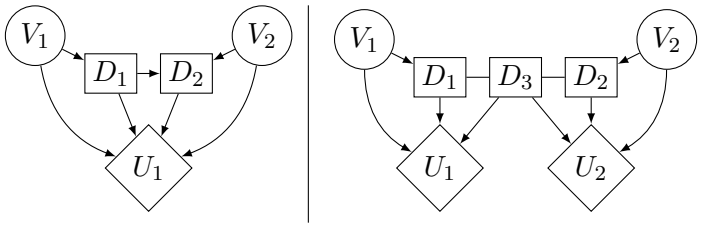

Figure 1: Left: LIMID. Right: DCG.

- $ch(U_i) = \varnothing$ due to the semantics of utilities.[1]

See Figure 1 (left) for an example of a LIMID. A policy for D_i is a probability distribution of the form $p(D_i|pa(D_i))$. A policy is typically (but not necessarily) deterministic. A policy prescribes an agent's behavior. A strategy is a set of policies, one for each decision variable. Note that a strategy turns the decision variables into random variables. Therefore, all the strategies are Markovian with respect to $G_{V \cup D}$. Therefore, it seems worth considering graphical models more expressive than DAGs in order to enlarge the set of representable strategies. In this work, we propose to consider Lauritzen-Wermuth-Frydenberg chain graphs. This gives rise to a new graphical model for decision making under uncertainty, which we dub decision chain graph (DCG) and which consists in a graph G over $V \cup D \cup U$ such that

- $G_{V \cup D}$ is a chain graph
- $pa(D_i)$ are the random/decision variables to observe/make and take into consideration before deciding on D_i
- $ne(D_i)$ are the decisions to make simultaneously and coordinately with D_i
- $pa(U_i)$ are the variables determining U_i
- $ch(U_i) = ne(U_i) = \varnothing$ due to the semantics of utilities
- $ne(D_i) \cap V = \varnothing$ due to unclear semantics otherwise.

See Figure 1 (right) for an example of a DCG.

3 Objective

The goal of this project is to devise, implement and test some simple heuristic technique to solve DCGs, preferably by adapting the single policy updating or SPU algorithm that exists for LIMIDs. This is simply a hill-climbing algorithm that finds a locally optimal strategy in terms of expected utility. SPU starts with the uniform policy for all the decision variables. Then, it chooses a decision variable and updates its policy so as to maximize the expected utility keeping the rest of the policies fixed. Finally, it repeats the previous step until no change occurs in two consecutive iterations. Using Bayesian optimization may be an option too. Implementing the Lauritzen-Spiegelhalter algorithm may also be an option, albeit one that may require more programming. For more information, see `https://www.dropbox.com/s/sw7kp5iolh6djnx/DCGs2.pdf?dl=0`.

[1] $pa(X)$ denote the parents of X (i.e. the nodes with a directed edge to X), $ch(X)$ denote the children of X (i.e. the nodes with a directed edge from X), and $ne(X)$ denote the neighbors of X (i.e. the nodes with an undirected edge to X).