Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007)

Strategy Generation Under Uncertainty Using Bayesian Networks and Black Box Optimization

Eli Faulkner
Quantum Leap Innovations
3 Innovation Way, Suite 100
Newark DE, 19711
Email:etf@quantumleap.us

***Abstract*— We describe a mechanism for optimal strategy generation from a Bayesian Belief Network (BBN). This system takes a BBN model either created by the user or derived from data. The user then specifies a set of goals (consisting of both objectives and constraints) and the observed and actionable variables in the model. The system then applies an optimizer to develop strategies that optimally achieve the specified goals. The system can be used by either human decision makers or autonomous agents. A distinguishing feature of the system is the ability to return strategies in the form of deterministic actions that result in the highest probability of achieving the desired goals. This allows the user to execute the strategies without further reasoning. In this paper we describe the architecture of the system and show examples of developing strategies from models created either by domain experts or directly from data.**

I. Introduction

Bayesian Belief Networks have become a very important tool in probabilistic modeling and data mining over the past few decades. Using a Bayesian Belief Network (BBN) to model a system has some important advantages over other modeling paradigms such as Neural Networks or Decision Trees. BBN's are typically more re-usable than neural networks, and often do not need to be re-trained when the problem changes. The black box nature of Bayesian inference makes BBN's a very flexible and useful modeling tool. Unfortunately, it is not always easy to perform automated reasoning with Bayesian networks. This is evident from a brief survey of work in the field. Even surveys which cover decision making with BBN's typically gloss over the topic in comparison to the time that they spend covering inference and learning algorithms [1], [2], [3].

Quantum Leap's Next Generation Probabilistic Adaptive Optimization Engine (Qung PAO), is a system for representing optimization problems that have Bayesian Relationships between the Decision Variables and the Objectives and Constraints. The system employs a three stage process for generating optimal strategies from a Bayesian model.

- Define the Bayesian network.
- Define the optimization problem:
  - Actionable variables
  - Observed variables.
  - Objectives such as maximizing or minimizing the probability of a node being in a certain state.
  - Constraints stating the probability of a node being in a state must be above or below a given value.
- Generate a set of optimal strategies.

We define a strategy as a set of actions which will achieve a goal with some probability.

Using this process we can make optimal decisions where actions that the user takes affect the users goals in an uncertain way. The strategies returned by Qung PAO are completely deterministic and can be executed by a human or autonomous agent, but have associated probabilities of achieving the desired goals.

II. Background

The core elements of Qung PAO are modeling with Bayesian Belief Networks[1], inferencing with evidence using a BBN, and the principles of black box optimization.

A. Bayesian Belief Networks

There are many excellent introductions to Bayesian Belief Networks, from both on line resources [4] and various texts [1], [2], [3]. In this paper we introduce the basic terminology. The reader is encouraged to review the literature for further details.

Given a set of random variables $\mathbf{X} = \{X_1, ..., X_N\}$, a Bayesian Belief Network is a structure which efficiently encodes the joint distribution by exploiting conditional independence's among the variables. A BBN is represented by $\mathcal{B} = (G, \Theta)$, where G is a Directed Acyclic Graph (DAG), and $\Theta = \{\Theta_1, ..., \Theta_N\}$ represents conditional probabilities for $P(X_i|\Pi_i)$, where Π_i are the parents of X_i in G. Using this representation we have $P(X_1, ..., X_n) = \prod_{i=1}^{N} P(X_i|\Pi_i)$

In this paper we will assume that each variable in our network has a discrete domain. Extension to support continuous variables is straightforward but beyond the scope of the current project.

This work was funded in part by the United States Office of Naval Research under Contract N00014-02-C-0320

[1]This work can be extended to arbitrary graphical models, but the scope of this paper is restricted to Bayesian Belief Networks

Max:	$\sum w_i f_i(x_1, ..., x_n)$
Subject to:	$g_k(x_1, ..., x_n) \geq 0$
Where	w_i are priority weights
	f_i are (discontinuous, nonlinear) objectives
	g_k are (discontinuous, nonlinear) constraints
	$x_j \in (\mathbb{R}, \mathbb{N}, \text{increments, sets, permutations})$
	x_j are bounded or have enumerated values

Fig. 1. The generic description of the nonlinear optimization problem

B. Inference with evidence in a BBN

One of the most important concepts in using a BBN is the ability update the posterior probabilities of the variables when some evidence is encountered. One can take a BBN and a set of variables in that BBN whose states are known, and use this information to update the posterior probabilities of each other variable in the network.

Throughout this paper, our Bayesian inference approach follows the algorithm proposed by Lauritzen and Spiegelhalter [5]. The method described here could easily be adapted to other inference algorithms. Since we typically do not need more than a few digits of accuracy in our solutions, we have also experimented with using alternate inferencing algorithms to increase speed.

C. Black Box Optimization

Black box optimization is a process where an optimizer queries a black box function for local scores of objectives and constraints and attempts to determine an optimal solution without full knowledge of the model. This means that the user defines a black box model relating the decision variables to objective and constraint scores. The optimizer then proposes candidate solutions by assigning values to all decision variables, and the black box model that the user specifies computes the objective and constraint scores corresponding to that instantiation of the decision variables. This process is applied repeatedly as the underlying optimization methods map the search space and learn which points satisfy the constraints and maximize the objectives. This process is shown in Figure 2.

The system described in this paper uses a black box optimization technology developed at Quantum Leap Innovations, the Adaptive Optimization®Engine.

1) The Adaptive Optimization®Engine: The Quantum Leap Adaptive Optimization Engine[6], [7], [8](AOE) is a system developed to solve general nonlinear optimization problems (NLOP) and constraint satisfaction problems (CSP). The AOE is particularly suited for solving nonlinear optimization problems. The NLOP, as described in Figure 1, can be used to model many real life problems. The NLOP has enjoyed a long history in the fields of mathematics and artificial intelligence [9].

The AOE solves problems using black box optimization.

Because we do not know the structure of the surface which we are optimizing, the AOE employs over 30 different

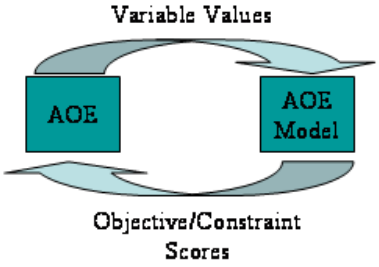

Fig. 2. The AOE evaluation loop

optimization algorithms in a cooperative-competitive paradigm to find global optima. These techniques range from weak techniques which solve a large class of problems very slowly, to strong techniques which solve a small class of problems very quickly. Examples of weak techniques are enumeration and random search. Examples of Strong techniques are Sequential Quadratic Programming and gradient methods. Because it uses many underlying optimization algorithms, the AOE is particularly suited for non-linear combinatorial optimization problems.

For the purposes of this work we only need to know that the AOE optimizes nonlinear black box models. For further details on the AOE, see [8].

III. Architecture

There are four major constructs in Qung PAO.

- Action variables
- Observed variables
- Objectives
- Constraints

Using these constructs, we define an AOE model that will use a Bayesian inference engine to relate the actions and observations to the objectives and constraints. Action variables and observed variables will be used to set up evidence that we will use in the inferencing engine. Objectives and constraints will then be read directly from the posterior probabilities computed by the inferencing engine.

A. Action Variables

Action variables are the variables in the model which represent actionable decisions, or decisions which can controlled by the executor of the strategy. The set of all action variables defines the search space for strategies. As we are searching

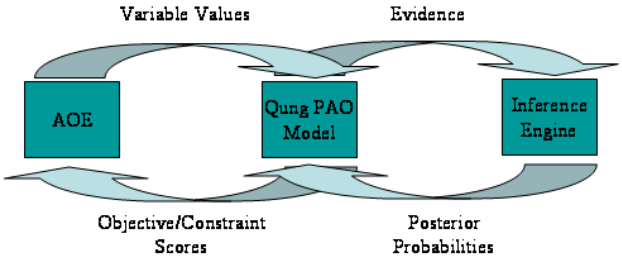

Fig. 3. The Qung PAO evaluation loop

for potential strategies, the AOE will nominate a state for each actionable variable. This will be fed into the inferencing engine as evidence[2]. Therefore, when a variable is declared as actionable we can not place an objective or constraint on it.

As an example, we may have ten machines, each with six gears. There exist 6^{10} possible choices of (machine, gear) pairs. Since it is overwhelmingly inefficient to enumerate all possible combinations of actions, we will use the AOE to search this space.

B. Observed variables

Many of the variables in the system may have their states known when the user requests a strategy. These observations can be input into the inferencing engine with the actionable variable states. Therefore, when a variable is declared as observed we can not place an objective or constraint on it.

An example of this is the effect that temperature has on a machine. We can simply place a temperature sensor on each machine in a factory. When the user requests a solve we now incorporate this knowledge into the strategy. We usually would not want to run a machine in high gear when it is overheating, so this extra information allows us to generate much more robust strategies.

C. Objectives and Constraints

Each variable that is neither an action nor observed variable can have zero or more objectives or constraints placed on it. Let V be the set of all variables in a BBN, $A \subseteq V$ be the set of actionable variables, and $E \subseteq V$ be the set of observed variables. For a variable $X \in V \setminus (A \bigcup E)$ with domain $s_0, s_1, ..., s_n$ an objective on a state in this variable takes the form $\max_{a \in A} P(X = s_k)$ or $\min_{a \in A} P(X = s_k)$. For multi-objective problems we currently take a sum of the objective scores, but weighting or other utility functions could easily be added to the system.

We can also declare constraints in the form $P(X = s_i) \leq c$ or $P(X = s_i) \geq c$ for $c \in [0, 1]$.

[2]Because we are simply viewing the causal network as an encoding of the joint probability distribution, we do not place the same restrictions on actions as Pearl[2]. This is a reason why we chose to use the word strategy instead of plan.

D. Applying the AOE

After we define our actions, observations, objectives, and constraints, we can apply the AOE to discover the optimal strategy or set of strategies. This process involves mapping the optimization constructs into the AOE, then applying the loop in Figure 2 repeatedly, where the user specifies the number of times to run the loop.

The mapping from the BBN into the AOE is very simple. In Figure 1 we see that the AOE supports decision variables whose domains are sets. For each action variable in Qung PAO, we declare a decision variable in the AOE whose domain is the set of all states that the corresponding action variable can take. We then inform the AOE of the existence of each objective and constraint in the system.

When the AOE Model receives the decision variable values, it must compute the posterior probabilities. It does this by translating the decision variable values and the observed variable information into an evidence map for the BBN, and preforming an inference to compute the posterior probabilities. We then simply need to read the posterior probabilities from the inferencing engine to get the values of the objectives and constraints. This loop is shown in Figure 3.

IV. A Simple Example

The following is a simple example to demonstrate the use of the constructs in Qung PAO.

Dave is trying to drive to a destination, but he has no money and is almost out of gas. The following conditions must be satisfied to reach his destination successfully.

- He does not get a flat tire
- He does not run out of gas
- He is not late

He also has some choices regarding his trip.

- He can have his windows up or down
- He can have his AC on or off
- He can accelerate slow, medium, or fast

A causal model of this situation is shown in Figure 4.

We can see in this simple example that the driver has control over three of the variables; Windows, AC, and Acceleration. He also has the goal of maximizing the probability that he will reach his destination. This translates into the optimization definition shown in Figure 5

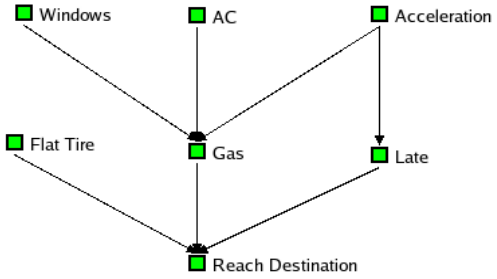

Fig. 4. The causal model from Section IV

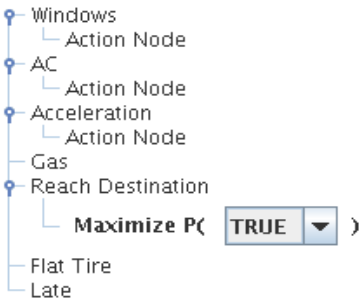

Fig. 5. The BBN Optimization definition from Section IV

When we define this problem in Qung PAO and optimize, we get the optimal actions in Table I, which in our example will result in Dave reaching the destination with a probability of .6399. Any other set of actions will result in a lower probability.

Windows	UP
AC	OFF
Acceleration	Medium

TABLE I

THE OPTIMAL STRATEGY FROM SECTION IV WITHOUT CONSTRAINTS

To extend this example, we can consider the same situation with the same BBN model. We will now say that Dave regards running out of gas as a much worse condition than being late, and thus would like to limit the probability of that happening. We will apply a constraint that $P(\text{gas} = \text{short}) \leq .02$, so any solution where the probability of running out of gas is greater than .02 is considered invalid. Generating a new strategy using Qung PAO, we get the strategy shown in Table II, which has a probability of .3959 that Dave will reach his destination.

This simple example illustrates many of the major constructs of Qung PAO, but is far more elementary than a typical problem. This simple problem only had $2 \cdot 2 \cdot 3 = 12$ possible actions. Using this as a framing example one can easily see that the search space grows combinatorially, and the problems can grow past enumerable sizes very quickly.

Windows	UP
AC	OFF
Acceleration	Slow

TABLE II

THE OPTIMAL STRATEGY FROM SECTION IV WITH $P(\text{GAS} = \text{SHORT}) \leq .02$

V. NUMERICAL RESULTS

As we can see from Figure 3, the running time of the system is bound by the time that it takes to compute an inference using the network and the number of times that the user runs the evaluation loop. We stated earlier that we can use any Bayesian inferencing algorithm in the system, but for simplicity we are currently using the algorithm due to Lauritzen and Spiegelhalter [5]. Unfortunately, this algorithm is exponential in the number of nodes and edges in the BBN.

In this section we will study the running time as a function of the network size, and the quality of solutions as a function of the number of evaluations.

A. Running time analysis

Empirically, the time to compute inferences is greatly effected by the network topology. To test Qung PAO, we have generated several random BBN's, with an various numbers of nodes and edges, and the maximum in-degree of each node less than or equal to 3. We then set a single objective and declared approximately 20% of the other nodes to be actionable.

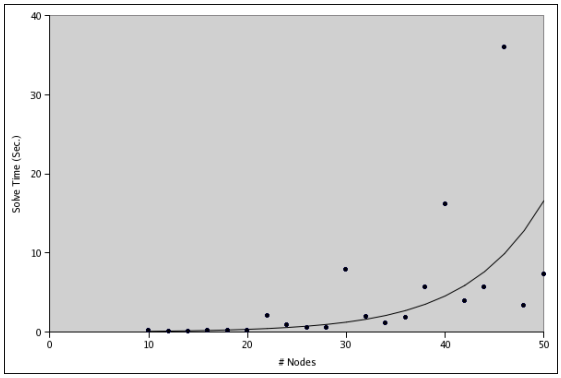

Fig. 6. The solve times (in seconds) for the networks from Section V with 1.2 times more edges than nodes. The line is an exponential regression of the data points.

We performed this process for BBN's between 10 and 50 nodes, with 3 states per node, and ran the evaluation loop 1000 times. Figures 6 and 7 show the results of this experiment where there were 1.2 and 1.3 times as many edges as nodes. These results clearly show the non-linear dependency cause by the speed of the inference algorithm.

Further investigation into the running time of the inference algorithm is outside of the scope of this paper, but the reader should understand that the inference algorithm running time has a dominating effect on the running time of the system.

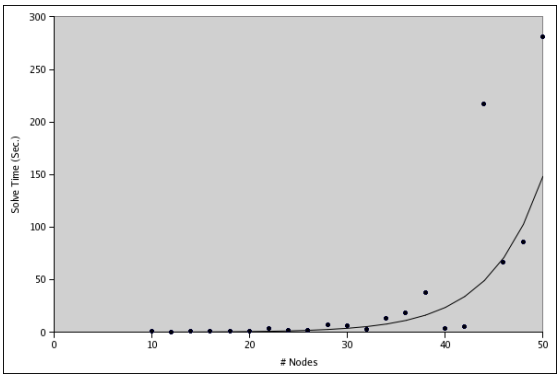

Fig. 7. The solve times (in seconds) for the networks from Section V with 1.3 times more edges than nodes. The line is an exponential regression of the data points.

B. Quality of solutions

A major feature of the Adaptive Optimization Engine is its on-line solving characteristics. During a Qung PAO solve, we can see the best solutions as they are returned from the optimizer. In this section, we will see how the quality of the solutions returned improved over time.

We performed an experiment on a randomly generated network with 200 3-state nodes and 220 edges. We defined objectives on three of these nodes, and declared 40 variables as actionable. We then ran Qung PAO for 1000 evaluation and recorded the solutions as they were incrementally returned by the optimizer. The resulting graph is shown in Figure 8.

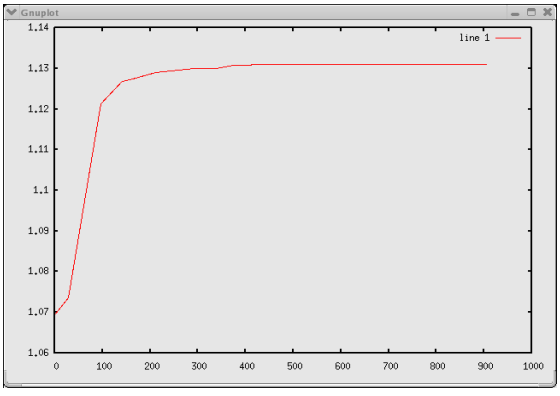

Fig. 8. The objective score over time for the experiment of Section V-B

We see that by 500 evaluations, the objective score has stabilized. We repeated the experiment running for 10000 evaluations on the above network and found a strategy with an objective score which was only $1.0343E-4$ better than the score found after 1000 evaluations. Results which find a near-optimal solution very quickly and make a few minor improvements during a long solve are typical.

VI. Comparison with State of the Art

The system described in this paper has several advantages over existing methods.

Influence Diagrams (ID) are probabilistic networks with three types of nodes; chance nodes, decision nodes, and utility nodes. See [3] for an introduction to Influence Diagrams. Viewing action nodes as decision nodes and objectives as utility nodes one can draw a comparison between Qung PAO and ID's. While there are some parallels, there are several major differences.

- All decision nodes in an ID are root nodes.
- All utility nodes in an ID are leaf nodes.
- The objective in an ID uses a utility function.
- Since ID's are extensions of BBN's, one has to be careful when learning the structure of the ID from data or modeling the ID by hand.

As Pearl says of Influence Diagrams in *Causality*, "The difficulty with this approach is that we need to anticipate in advance, and represent explicitly, all actions whose effects we might wish to evaluate in the future. This renders the modeling process unduly cumbersome, if not totally unimaginable."[2]. In Qung PAO we do not place structural restrictions on the location of actions, observations, objectives, or constraints during construction of the model [3] . These differences make Qung PAO a much more flexible system in many circumstances.

Classification problems represent an important domain for decision making and probabilistic reasoning. Popular classification methods include decision trees and support vector machines. The primary distinction between these classification techniques and our system lies in the fact that traditional classification systems deal with the forward modeling problem whose output state is predicted by a given set of input states. They are not well suited for the inverse modeling problems which is the focus of this study.

Another advantage of our system is the fact they it allows variables which are neither inputs nor outputs to affect the goals. A corollary to this is our ability to add or remove objectives and constraints without retraining our model. When we remove all objectives and constraints from a variable in the model, that variable remains in the model and will effect our solution. This is in contrast to Neural Networks [10], which would have trouble handling situations where we have uncontrollable or unmeasurable variables. Models which do not consider certain variables because they can not be measured are naturally less accurate in their representation of the system which is being modeled.

Finally, an additional advantage of BBN's for optimal strategy discovery lies in their fundamental transparency. The discovered strategies are represented directly in terms of the primary actionable variables. Methods such as neural networks

[3]It is possible to construct a model in such a way that an action node is seperated from the objectives and constraints given other action or observed nodes. If this is the true case then the action taken by the seperated node has no influence on the goals of the problem, and Qung PAO will assign a random value to that action. It is possible to extend the system to inform the user of the system when this occours.

have the undesirable property of hiding the primary variables from direct view, making it harder to elaborate strategies that are meaningful to the end user.

VII. Conclusion

Qung PAO has proven to be a powerful system for decision makers to generate optimal strategies from data.

The system could be used to enable many difficult decision making processes in many application areas. Some examples are:

- Health Care - A drug testing application could give researchers clues on which experiments to run during a drug testing process.
- Manufacturing - A maintenance planning application could allow manufacturers to avoid high risk situations as their machines degrade.
- Homeland Security - A search strategy application for airports and ships could give law enforcement and the intelligence community greater analytic power in their search strategies.
- Research and Development - A project funding application could allow research and development organizations greater risk awareness, and help increase synergy between projects.
- Marketing - A budget allocation application could give marketing organizations greater power to reach only the audience who would be their optimal target market.
- Autonomous/Multi Agent Systems - Autonomous planning. Links to process execution engines would give intelligent agents the ability to develop and execute strategies autonomously.
- Law - A policy making system based on this technology would give politicians and lobbyists great quantitative power to back their decisions.

Throughout this paper we have identified some areas of future research which would expand the capabilities of the system. Those and other areas of current and future research are:

- Combination with Bayesian model averaging techniques so we can use multiple BBN's as our model.
- Integration with goal-aware structured learning systems to allow us to build models which only include relevant variables.
- Implementation of a goal aware inferencing algorithm which only computed the required posterior probabilities.
- Implementation of alternate inferencing algorithms, like approximation methods, for inference speed increase.
- Integration of continuous variables.
- Automatic selection of inferencing algorithm based on network structure.
- Reduction of strategies based on conditional independence relationships the the BBN.

As learning and inference algorithms are improved and discovered, systems which support decision makers in their use of Bayesian networks will become increasingly important. This system fills an important missing gap in current research, as it unifies deterministic decision making with the richness of probabilistic reasoning.

References

[1] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search, Second Edition (Adaptive Computation and Machine Learning)*. The MIT Press, 2001. [Online]. Available: http://www.amazon.fr/exec/obidos/ASIN/0262194406/citeulike04-21

[2] J. Pearl, *Causality*. Cambridge Universtiy Press, 2000.

[3] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2003.

[4] K. Murphy, "An introduction to graphical models," 2001. [Online]. Available: http://www.ai.mit.edu/~murphyk/Papers/intro_gm.pdf

[5] S. Lauritzen and D. Spiegelhalter, "Local computation with probabilities in graphical structures and their applications to expert systems," *Journal of the Royal Statistical Society*, 1988.

[6] J. Elad, "System and method for representing and solving numeric and symbolic problems, united states patent 5,195,172," Patent, 1993.

[7] ——, "System and method for representing and solving numeric and symbolic problems, united states patent 5,428,712," Patent, 1995.

[8] E. Faulkner and J. Cowart, "The adaptive optimization engine," in *INFORMS Annual Meeting, November 5-8, 2006, Pittsburgh, PA*, 2006. [Online]. Available: http://www.quantumleap.us/Research\&Development/PublishedPapers/INFORMS_2006_AOE.pdf

[9] A. Neumaier, "Complete search in continuous global optimization and constraint satisfaction," *Acta Numerica 2004*, 2004.

[10] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.