Solving Hard Computational Problems Through Collections (Portfolios) of Cooperative Heterogeneous Algorithms

Eugene Santos Jr.
Dept. of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269-3155
eugene@cse.uconn.edu

Solomon Eyal Shimony
Dept. of Math. and Computer Science
Ben Gurion University of the Negev,
84105 Beer-Sheva, Israel
shimony@cs.bgu.ac.il

Edward Michael Williams
Computer Technology Division
Air Force Information Warfare Battlelab
San Antonio, TX 78243-7020
ewilliam@acm.org

Abstract

Numerous artificial intelligence schemes and applications require, at their core, a solution to intractable computational problems, such as probabilistic reasoning. Conditions for theorems guaranteeing polynomial time algorithms for special cases, do not hold for many real-world problem instances. While there are a number of highly specialized algorithms/heuristics that can solve specific problem subclasses, a great variance exists between algorithm performance over the spectrum of different problem instances. However, matching the best algorithm to a problem instance is a difficult proposition at best. Unfortunately, this is also assuming that such an individual algorithm exists which can actually solve the given problem in a reasonable amount of time and space. Harnessing several different problem-solving algorithms/methods together into a cooperative system (or portfolio) has been observed to have the potential for solving these NP-hard problems.

The need exists for an intelligent controller that is able to effectively combine radically different problem-solving techniques into a distributed, cooperative environment. In this paper, we describe the OVERMIND system which provides a framework and approach for developing such controllers. By modeling the performance/behavior of the component algorithms, especially how they cooperate and interact, this provides the means for developing a controller to select the most appropriate algorithm mix (portfolio). We applied this approach to belief revision in Bayesian networks.

Introduction

Numerous AI tasks require solving problems that are known to be computationally intractable in the general case. The latter include diverse subtasks, such as probabilistic reasoning, central to many domains from medical diagnosis to protein structure prediction. Though various algorithms exist for these NP-hard problems, their runtimes are exponential in the worst case. All of the above problems exhibit special cases - subclasses where polynomial time algorithms are known. Unfortunately, these benevolent conditions fail to hold for many real-world problem instances.

A great variance exists in individual algorithm performances over different problem instances. In fact, it is often the case that an algorithm that performs best for one problem instance, may perform much worse than another algorithm on another problem instance. However, matching the best algorithm to a problem instance can be as difficult as determining the correct solution to the problem itself. Although some indication on how to do that may be available from an inspection of the problem, such predictions are notoriously unreliable. Further complicating matters is the possibility that no single algorithm currently exists which can solve the given problem in a reasonable amount of time and space.

With the increasing use of large networks of personal workstations, the available computing power is no longer limited to a single computer. Harnessing the power of more than one computer (possibly even the entire network) into a cooperative problem solving environment where each different algorithm actively assists other algorithms can create a significant resource for working these difficult problems. To begin pursuing this idea, we conceived of and developed OVERMIND (Santos, Shimony, & Williams 1995), a distributed architecture/system for problem solving which addressed algorithm cooperation for probabilistic reasoning. The OVERMIND architecture allows the use of different inference methods in parallel and selects the initial set of algorithms to use based on the attributes of the given problem instance. In order to solve the given problem using the available resources, limitations such as processor speed, job load, memory availability, network latency, etc. are also considered.

To make a determination of which algorithms/methods to use, a model of both the methods' overall performance for a given class of networks and the methods' run-time performance profiles (Gomes & Selman 1997) are required. Furthermore, an overall algorithm interaction model must be derivable from these models since algorithms are meant to communicate amongst themselves to increase the overall effectiveness through synergy in solving the problem. The model allows OVERMIND to choose appropriate methods to begin the inferencing process in order to converge more rapidly towards the optimal solution. The models for the different inference algorithms are created (where possible) using analytical

methods instead of run-time profiling. The goal is to develop easily calculated approximate models to minimize the computational overhead for OVERMIND. Accomplishing this goal required the following effort: 1) empirical verification of the effectiveness of cooperative algorithms; 2) formulation of models of the algorithms' behavior through analytic methods; and 3) identification of requirements and development of a prototype method for selecting algorithm combinations given a specific problem instance.

Algorithm behaviour modeling and algorithm cooperation were analytically developed for reasoning algorithms over Bayesian networks (Santos, Shimony, & Williams 1995). These included best-first search (Shimony, Domshlak, & Santos 1997; Santos & Shimony 1994; Shimony & Santos 1996), stochastic simulation (Santos, Shimony, & Williams 1997), integer programming (Santos & Shimony 1994; Santos, Shimony, & Williams 1997; Shimony & Santos 1996; Santos 1993; 1991; 1994), genetic algorithms (Santos & Shimony 1994; Santos, Shimony, & Williams 1997), and clustering/conditioning methods (Williams, Santos, & Shimony 1997). With these models, we demonstrated both theoretically and empirically that cooperative algorithms were capable of solving new classes of problems previously intractable for individual algorithms.

In this paper, we discuss OVERMIND and our experimental results on belief revision for Bayesian networks. It is important to note that although the above scheme was formulated for probabilistic reasoning, it is clear that this can be applied to a larger host of problems. Observe that almost all hard computational problems can be and are naturally solvable using iterative or progressive optimization algorithms.

Anytime Algorithms

Anytime algorithms were first used by Dean and Boddy (Boddy 1991; Dean & Boddy 1988) to provide a means to balance execution time with solution quality in their work on time-dependent planning. In general, anytime algorithms are useful when problems are computationally hard; they provide a means for evaluating the progress of an algorithm during its execution. Anytime algorithms have four characteristics which differentiate them from traditional algorithms (Grass & Zilberstein 1996): 1) *Quality Measure.* It is possible to quantify the quality of a solution generated by the anytime algorithm. 2) *Predictability.* Through empirical, statistical or analytical means, we can estimate the output quality of the anytime algorithm given a particular time and input data. 3) *Interruptability.* An anytime algorithm produces the current status of its solution process when interrupted or periodically throughout its execution. 4) *Monotonicity.* The quality of the output of anytime algorithms never decreases over time; it either remains constant or increases as the algorithm is given more time to work.

Even if time is not a factor in the value of a solution, it is frequently useful to be able to predict the improvement an algorithm will make if allowed to continue. One method for making this prediction is to create a *performance profile* for the algorithm. This profile characterizes an algorithm's performance over a set of parameters; they can be generated analytically or empirically. Extensive work has been done in empirically characterizing anytime algorithms (Garvey & Lesser 1996; Grass & Zilberstein 1996), but little has been done through algorithm analysis (Santos, Shimony, & Williams 1995; Williams, Santos, & Shimony 1997).

In cases where a single algorithm is not sufficient to solve the problem, multiple algorithms would need to be used; Zilberstein and Russell have researched the use of anytime algorithms *only* as sequential components of a larger system (Zilberstein & Russell 1996).

Algorithm Combinations and Anywhere Algorithms

In problems where more than one algorithm is available, we have observed, as have others (Hogg & Huberman 1993; Hogg & Williams 1993; Huberman, Lukose, & Hogg 1997; Gomes & Selman 1997) that cooperation between algorithms can likewise result in a more rapid solution. The current issues we have identified with this approach are:

- *What to share between algorithms*: How much and what kind of information is needed to effectively help the other algorithm(s) without inducing too much overhead?
- *How to use the shared information*: How should the algorithm utilize the information imported from the other algorithm(s)?
- *When to share information*: When is sharing the information beneficial, or is it better to just let the algorithms run independently?

We call the concept underlying information sharing between algorithms, the *anywhere* property (Santos, Shimony, & Williams 1995). This property refers to an algorithm's ability to accept complete or partial solutions generated elsewhere *and* its ability to incorporate that solution into its own processing. The methods used to accomplish the incorporation of external solutions varies based on the methods used by the algorithm. The solution could be used to place bounds on the problem being worked by the algorithm, it could change the direction of the processing, or it could simply be used to affect the way the algorithm interacts with the rest of the system.

Algorithm Analysis and Modeling for Portfolio Construction

We now briefly discuss the algorithms analysis used for building algorithm models necessary to managing the overall algorithms mix/portfolio. Detailed analyses and the resulting models of the different algorithms used in the Bayesian Network testbed can be found in (Williams, Santos, & Shimony 1997; Santos, Shimony, & Williams 1995).

Algorithm portfolio selection requires the ability to predict algorithm performance over an initial short time interval. This time interval is dependent on the time required for the algorithms to initialize and begin producing results. An algorithm model consists of six factors which we have identified as being useful in predicting algorithm performance: *Time Complexity, Space Complexity, Resource Cost, Restart Cost, Level of Improvement* and *Probability of Improvement.* The model also contains one factor (*Interaction Rate*), which is calculated during run-time for each algorithm. These factors capture the expected performance of an algorithm (or combination of algorithms) in a domain-independent manner: when applied to an algorithm, the factors are calculated using characteristics of the problem domain. We use the term *solution quality* to refer to the output of the process, with the goal to produce the solution with the highest quality value. How the quality of a particular solution is determined depends on the domain: it could be related to the progress of the algorithm (how far until it is done) or possibly a value which is calculated directly from the solution itself. The important concept is that the solution quality can be used to compare solutions, with the goal to obtain the highest quality solution(s).

One factor needed by the run-time controller is the likelihood that an algorithm will produce a better solution if allowed to *continue from a known point.* This likelihood
is called the *probability of improvement*: $P_{imp}(S_i) = \sum_{S_{i+1} \in S} P_{select}(S_{i+1}|S_i) P[Q(S_{i+1}) > Q(S_i)]$ where S_i is the current anytime solution available at time i and S is the set of all possible solutions. This probability encompasses both the characteristics of the solution landscape (through the solution quality evaluation function $Q(S_{i+1})$) and the actions of the algorithm itself ($P_{select}(S_{i+1}|S_i)$), the probability that S_{i+1} is generated from S_i by the algorithm.

The other interesting characteristic of anytime algorithms is the *level of improvement* produced by the algorithm. This is the expected magnitude of the increase in solution quality that the algorithm will make in the next increment: $E_{imp} = \frac{Q(S_{i+1})}{Q(S_i)}$. With anytime algorithms the level of improvement is always greater than or equal to 1.0 because the output quality is required to be non-decreasing. For our Bayesian Networks problem domain, the joint probability of the solution is used as the solution quality.

Algorithm Interaction

With the analysis method for the different algorithms established, we can look at the impact when the algorithms are used cooperatively. Intuitively, the combination should perform at least as well as the algorithms run individually (assuming no resource limitations); but the open issue is whether to expect an improvement over the individual algorithm performance. Performance improvement due to sharing information mostly occurs when the algorithm's performance characteristic is significantly different from the rest of the collection. One algorithm with a steep performance curve can boost the performance of a slower performer; or if the performance curves for two algorithms intersect, they alternate the roles of leader and follower. Obviously, algorithms from different classes have significantly different performance profiles; but in some cases, it is possible to significantly affect the performance of an algorithm through its parameters. In such cases, it may be beneficial to use two similar algorithms, but with different parameters.

Each of the model factors are evaluated for each individual algorithm for the given problem instance attributes; however we also want to evaluate different combinations of algorithms. There are two approaches we can take: analyze each possible combination of algorithms, producing a separate model for each combination; or combine the results of the models for the individual algorithms, producing a composite result.

We note that since the interaction between the component algorithms occurs throughout the time interval t, a simple combination of the convergence factors for the individual algorithms is not adequate. We can, however, simulate the interaction by iteratively calculating the convergence factors at approximately the same rate that the algorithms would actually exchange solutions; this rate is determined by obtaining for each algorithm the length of time between production of solutions. At each iteration, each algorithm model would account for the received solution in its prediction for the next iteration.The best score received from any algorithm is retained for the next iteration.

Case Study – Belief Revision in Bayesian Networks

The overall goal of our experiments is to demonstrate the effectiveness of solving computationally hard problems with multiple cooperating algorithms using an initial algorithm selection based on our portfolio construction model. This is contrasted against using an arbitrary single algorithm system.

Bayesian Networks are a compact way of representing probabilistic uncertainty in automated reasoning. Such networks consist of directed acyclic graphs of nodes, each representing a random variable (RV) with a finite domain. Edges represent direct dependency, and the topology as a whole conveys independence assumptions. These allow the joint probability distribution over all the RVs to be fully determined by providing conditional probabilities of variable states given all their predecessor states as the product of all the conditional probabilities.

One form of uncertain reasoning on Bayesian Networks is *belief revision.* Belief revision is the process of determining the most probable instantiation of the random variables (RVs) in a network given some evidence. More formally, if W is the set of all RVs in the given

	extreme	flat	merged
GA	3.367e-05	6.085e-12	2.737e-16
best-first	6.369e-01	3.948e-12	1.681e-12
Both	6.369e-01	6.085e-12	3.790e-12
Optimal	6.369e-01	6.099e-12	3.884e-12

TABLE 0.1. Performance comparison for extreme, flat and merged extreme-flat network with GA, best-first, and cooperating GA and best-first. Numbers reflect the best probabilities found.

Network Size (RVs)	Initial Time Interval (sec) 10	20	50	100	200	300	500
30	.76	.93	.67	.67	.67	.67	.67
50	.83	.89	.67	.75	.92	.67	.67
100	.25	.42	.58	.67	.83	.92	.92
200	.08	.25	.42	.67	.58	.83	.83
300	0	0	.27	.55	.45	.80	.58
500	0	0	0	.57	.56	.78	.78

FIG. 0.1. Success rates for algorithm selection.

Bayesian Network and e is the evidence, any complete instantiations to all the RVs in W that is consistent with e is called an *explanation* or *interpretation* of e. The problem is to find an explanation w^* such that $P(w^*|e) = \max_{w \in W} P(w|e)$. Intuitively, we can think of the non-evidence RVs in W as possible hypotheses for e.

Experiment Setup and Environment The test environment utilizes a heterogeneous network of Sun workstations (both Sparc 20 and UltraSparc I) and Linux workstations (200 MHz Pentium Pro). The experiments were conducted using 75 randomly generated Bayesian Networks. The networks contain from 30 to 500 random variables, each having 2 or 3 possible states. The connectivity between the random variables was kept moderate, with the number of arcs typically ranging between twice and three times the number of random variables. The conditional probability tables were generated with varying distributions, from extremely flat (small variance between entries) through extremely spiked (most entries close to zero, with one nearly 1.0).

Multiple Algorithm Cooperation. The effectiveness of multiple cooperative algorithms were best characterized when the individual algorithms in the mix were quite different from one another. Taking a simple algorithm mix of a genetic algorithm (GA) and a best-first search method, we compared the cooperative runs against individual algorithm runs. We found that for homogeneous networks (all r.v.s in the network have the same characteristics), the combination of the two algorithms always performed at or above the level of the individual algorithms.

Where a significant performance improvement was achieved is with heterogeneous networks. For example, take a network that was constructed from two individual networks where one had a relatively flat distribution and the other with a fairly extreme distribution. On this network, the GA alone could never achieve even close to the optimal solution for the extreme network where as the best-first algorithm bogged down early in the flat network. Table 0.1 shows the results of running the individual algorithms on the composite network; neither algorithm could come close to the optimal solution by itself on the merged network. However, the combined cooperative case was much more successful. Note that the optimal solution was determined through an extremely lengthy brute force computation.

Portfolio Selection. We used a simulation approach based on behaviour approximations using a combination of the seven model factors identified for each algorithm to select from five different algorithms: a Genetic Algorithm (GA), best-first search with two different heuristics: cost-so-far and shared-cost, a hybrid stochastic search (HySS) and a Barrier Algorithm.

Figure 0.1 shows how the selected combination performed in comparison with the other combinations for that sized network. The numbers reflect the success rate/percentages of the selection process. Our definition of success is when (1) the algorithm(s) selected were contained in the combination that produced the best solution at that time *and* (2) those algorithm(s) were the ones contributing the best solutions. Clearly, in most of the cases, the selection process chose an algorithm combination that matched the best-performing combination.

Since the goal of the initial static selection process is to select a configuration to get the system off to a good start, picking the best performing combination is not essential. On the other hand, the better the initial configuration performs, the easier it will be for the future dynamic controller.

Looking again at the results, notice the increase in the success rate as you move from left to right. This is a result of the varied cycle times for the different algorithms. This effect can also be noticed when moving from top to bottom. As the network size increases, the cycle time for the algorithms increase; with the approximate models, it may take several cycles for the models to accurately predict the algorithm's performance. What is not immediately obvious is why the success rates fall off at the right edge of the table, especially for the smaller networks. This result is caused by the "near-sighted" models. Accuracy falls off as time increases, causing inaccurate predictions for the larger interval times relative to the size of the problem.

Summary

The focus of this research has been to model the process of managing a distributed system of cooperating pro-

cesses. We are particularly interested in those domains where finding the optimal solution is either intractable or impossible. In these situations, algorithms typically exist that can produce some form of sub-optimal solution, but, the problem of choosing which one to use for a given situation is often difficult as well. In fact, this selection must be made with some knowledge of the way problem characteristics affect an algorithm's performance.

We developed a method for capturing the essential information about a problem domain and the algorithms to manipulate that domain (Williams, Santos, & Shimony 1997). The resulting models encapsulate this information in the form of general performance metrics useful in the selection and evaluation of algorithms and algorithm combinations. The encapsulation makes it possible for the controller to be developed independent of any problem domain, yet still utilize the algorithm models to determine algorithm performance for the problem domain being studied. We also developed a method for selecting the initial algorithm configuration for the system based on the models' performance prediction.

The power of this approach is the use of approximate "near-sighted" models of the algorithms. They are typically easy to compute, and may be effective in modeling the algorithm's behavior in the near future.

To demonstrate the capabilities of this approach, we applied it to uncertain inference over Bayesian Networks. We characterized the problem domain, extracting the significant features from the networks that affect the performance of the algorithms. The resulting system performed as expected: when priority was given to solution quality (instead of resource usage, etc.), the system produced results comparable to the best individual algorithm. With the capability of the system to incorporate other factors into the selection process, we also demonstrated the ability of the controller to reach a reasonable compromise configuration between solution quality and resource consumption.

Acknowledgments. This work was supported in part by AFOSR Project #940006 and #9600989.

References

Boddy, M. 1991. Anytime problem solving using dynamic programming. In *Proceedings of the AAAI Conference*, 738 743.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–54.

Garvey, A., and Lesser, V. 1996. Design-to-time scheduling and anytime algorithms. *SIGART Bulletin* 7(2):16 19.

Gomes, C. P., and Selman, B. 1997. Algorithm portfolio design: Theory vs. practice. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Grass, J., and Zilberstein, S. 1996. Anytime algorithm development tools. *SIGART Bulletin* 7(2):20 27.

Hogg, T., and Huberman, B. A. 1993. Better than the best: The power of cooperation. In Nadel, L., and Stein, D., eds., *1992 Lectures in Complex Systems V: SFI Studies in the Sciences of Complexity*. Addison-Wesley. 165–184.

Hogg, T., and Williams, C. P. 1993. Solving the really hard problems with cooperative search. In *Proceedings of the National Conference on Artificial Intelligence*, 231–236.

Huberman, B. A.; Lukose, R. M.; and Hogg, T. 1997. An economics approach to hard computational problems. *Science* 275:51–54.

Santos, Jr., E., and Shimony, S. E. 1994. Belief updating by enumerating high-probability independence-based assignments. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 506–513.

Santos, Jr., E.; Shimony, S. E.; and Williams, E. 1995. On a distributed anytime architecture for probabilistic reasoning. Technical Report AFIT/EN/TR95-02. Department of Electrical and Computer Engineering, Air Force Institute of Technology.

Santos, Jr., E.; Shimony, S. E.; and Williams, E. 1997. Hybrid algorithms for approximate belief updating in bayes nets. *International Journal of Approximate Reasoning* 17(2-3):191 216.

Santos, Jr., E. 1991. On the generation of alternative explanations with implications for belief revision. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 337 347.

Santos, Jr., E. 1993. A fast hill-climbing approach without an energy function for finding mpe. In *Proceedings of the 5th IEEE International Conference on Tools with Artificial Intelligence*.

Santos, Jr., E. 1994. A linear constraint satisfaction approach to cost-based abduction. *Artificial Intelligence* 65(1):1–28.

Shimony, S. E., and Santos, Jr., E. 1996. Exploiting case-based independence for approximating marginal probabilities. *International Journal of Approximate Reasoning* 14(1):25 54.

Shimony, S. E.; Domshlak, C.; and Santos, Jr., E. 1997. Cost-sharing heuristic for bayesian knowledge-bases. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 421 428.

Williams, E.; Santos, Jr., E.; and Shimony, S. E. 1997. Experiments with distributed anytime inferencing: Working with cooperative algorithms. In *Proceedings of the AAAI Workshop on Building Resource-Bounded Reasoning Systems*, 86–91.

Zilberstein, S., and Russell, S. 1996. Optimal composition of real-time systems. *Artificial Intelligence* 82(1-2):181–213.