A Parallel Algorithm for Statistical Belief Refinement and its use in Causal Reasoning

Jay C. Weber*
Computer Science Department
University of Rochester
Rochester NY 14627 USA
jay@cs.rochester.edu

Abstract

This paper presents a new approach to efficient parallel computation of statistical inferences. This approach involves two heuristics, *highest impact first* and *highest impact remaining,* which control the speed of convergence and error estimation for an algorithm that iteratively refines degrees of belief. When applied to causal reasoning, this algorithm provides a performance solution to the qualification problem. This algorithm has been implemented and tested by a program called HITEST, which runs on parallel hardware.

1 Introduction

A useful form of statistical inference derives a belief that an object is a member of a hypothesis set, given that the object is a known member of other sets. For example, we might wish to derive the probability that Tweety is in the set flyers, given that she is in the set birds. This probability can be obtained from the value of the *statistic* %(flyers | birds), which captures the proportion of birds that fly [Kyburg, 1987, Loui, 1987, Bacchus, 1988]. The set on the right-hand side of the conditional bar, birds in this case, is called the *reference class* [Kyburg, 1983] since it provides a reference from which to ascribe probabilities to assertions about its members belonging to other sets.

It is generally agreed that a more *specific* reference class will tend to derive a more appropriate probability[1] [Reichenbach, 1949, Bacchus, 1988, Etherington, 1987]. In the case of Tweety, if we also know that she lives in Antarctica, it is prudent to incorporate this fact in the reference class, using the value of the statistic %(flyers | birds Pi antarcticans) instead.

However, if we take the principle of specificity to its logical extreme, we must incorporate *all* available knowledge about her into our reference class, including her size, color, age, parentage, etc. In complex domains, it may not be practical for a reasoner to consider all these potentially relevant factors. When choosing a reference class, an important consideration is the cost of computing (or indexing to) the statistic corresponding to that class. This aspect has not been addressed by any of the above approaches.

This paper presents a solution to this problem. Given a set of primitive object-properties, this approach incrementally builds more specific reference classes by considering these properties in order of decreasing magnitude of their statistical impacts. As more properties are considered, the magnitudes of the impacts are used to estimate the current error, and when this is small enough and/or time is critical, the computation terminates.

The problem of reference class choice arises in causal reasoning [Weber, 1989] as the famous *qualification problem* [McCarthy, 1977]. Therefore, this paper also provides a performance solution to the qualification problem. Section 4 describes this application of highest impact heuristics, and Section 5 uses an example of causal reasoning as the principal application of a system called HITEST, a parallel implementation of these techniques.

2 Belief Refinement and Impact

A key theorem in Bayesian techniques is the following "recursive evidential updating" rule:

$$\%(h \mid r \cap f) = \%(h \mid r) \cdot \frac{\%(f \mid h \cap r)}{\%(f \mid r)}, \qquad (1)$$

which follows easily from Bayes rule. It tells us how to compute the new statistical value for a hypothesis set h, obtained when a reference class r is made more specific by intersecting it with new factor set f.

It will be more convenient to use a *logarithmic odds* version of the above update rule, obtained by dividing (1) by the update rule for $\bar{h}$ and taking the natural logarithm of both sides, *i.e.*

$$\begin{aligned} \ln\left(\frac{\%(h|r\cap f)}{\%(\bar{h}|r\cap f)}\right) &= \ln\left(\frac{\%(h|r)}{\%(\bar{h}|r)} \cdot \frac{\%(f|h\cap r)}{\%(f|\bar{h}\cap r)}\right) \\ &= \ln\left(\frac{\%(h|r)}{\%(\bar{h}|r)}\right) + \ln\left(\frac{\%(f|h\cap r)}{\%(f|\bar{h}\cap r)}\right). \end{aligned}$$

If we use the following two definitions:

$$\hat{\$}(h \mid r) = \ln\left(\frac{\%(h|r)}{\%(\bar{h}|r)}\right) \quad \hat{!}(f,h,r) = \ln\left(\frac{\%(f|h\cap r)}{\%(f|\bar{h}\cap r)}\right),$$

*This work supported in part by U.S. Army Communication Electronics Command grant no. DAAB10-87-K-022.

[1] Provided that the relevant statistics are believed with equal confidence. This paper ignores this issue in the interests of clarity and economy. See Kyburg [1989] and Loui [1987] for more details.

(where the hat (^) symbol is used as a reminder that the quantities are logarithmic), then the logarithmic odds update equation simplifies to:

$$\hat{\$}(h \mid r \cap f) = \hat{\$}(h \mid r) + \hat{!}(f, h, r). \qquad (2)$$

I call $\hat{!}(f, h, r)$ the (logarithmic) *impact* of f on h given r. Positive impact values signify that "$x \in f$" lends support for "$x \in h$" (given "$x \in r$"), and negative values signify that "$x \in f$" lends support "$x \in \overline{h}$". The impact is zero if and only if "$x \in f$" and "$x \in h$" are conditionally independent given "$x \in r$".

Knowledge of impact values provides a simple way to compute statistical values for reference classes from the values for less specific reference classes. This inspires an iterative algorithm that adds the impacts of the factors one at a time, as follows:

```
EvUpdate(hypothesis h, factors r)
belief = prior(h)
context = ∅
while r ≠ ∅ do
    pick an element f of r
    delete f from r
    belief = belief + impact(f,h,context)
    add f to context
od
return belief
```

The time complexity of this algorithm depends on the size of the set `r`, which equals the number of iterations, one per factor. Note that the impact of a factor is conditioned on all factors previously added, making the individual impacts depend on the order imposed on the factors. The final value, however, is the same for any ordering of the factors.

3 Highest Impact Heuristics

I will exploit the fact that the factors can be incrementally considered in any order, by specifying an order that tends to make the series converge quickly to its final value. When this is true, the iterative algorithm may find that a reasonable estimate of the final value can be computed from the impacts of merely a *proper subset* of the factors, allowing the iterative algorithm to terminate after relatively fewer iterations. There are two parts to this approach: how to encourage fast convergence, and how to detect when no more iterations are needed.

3.1 Highest impact first

At each stage of the iteration, there is a choice of which one of the remaining factors to add. The above discussion of impact values and their relative magnitudes suggests that more strongly positive or negative impacts are more salient than those near zero; after all, a factor with an impact of zero, being independent, will not affect the incrementally-refined belief.

I propose that at each iteration, the factor with the largest impact absolute value should be chosen to be added. This choice induces an ordering on how the factors are incorporated in the evolving belief, where the next member of the order is determined at each iteration step. I call this technique the ***highest impact first*** (HIF) heuristic.

Since these degrees of belief are computed incrementally, at each step of the iteration the reasoner has a value which can be used as the belief. The highest impact first heuristic will tend to make this value a better estimate (with respect to the value of the statistic based on all factors) as more iterations are performed. In this way, the reasoner can trade speed (number of iterations) off against accuracy (specificity of the reference class). The reasoner can respond to requests for both a "quick guess" or a "slow analysis", by performing a different number of iterations.

3.2 Highest impact remaining

It would be more interesting for the reasoner to be able to decide for itself how many iterations are necessary to provide a given accuracy. That way, the reasoner can spend only as much time computing as the situation requires. This judgment is inherently contingent on the impacts of the known dependent factors, e.g. one can predict whether the sun will rise tomorrow with great accuracy in very little time, whereas it takes much longer to make a reasonable prediction about the next hockey game. Since highest impact iteration involves scrutinizing these impacts, it provides a promising framework to make the accuracy judgment.

I propose that a reasonable estimation of the error of the prediction at a given iteration is the magnitude of the highest impact over all the factors known to contain the object in question. If this highest impact is zero, then the current prediction is *exactly* the value of the statistic in the KB with the most specific reference class. When all impacts but the highest are zero, the absolute error is exactly that highest impact. In other cases, the highest impact is merely an estimate of the error, but it will tend to be reasonable because of the way the statistic for the current reference class generalizes over the factors not currently included. The current statistic asserts that over all the possible states of knowledge the agent may have with respect to the fluents not yet considered, the average sum of all of the impacts will be zero. The current highest impact is an upper bound on how much the incorporation of a single factor can defy this average.

To use this heuristic, an error bound is attached to each request of the statistical reasoner. At the end of each iteration the reasoner examines the highest impact that was just added, and if it is less than or equal to the error bound specified, then the iteration will stop and return the current value as the posterior probability. Note that as a special case, if the error bound is zero, then highest impact iteration will continue until all knowledge about dependent factors is incorporated into the reference class.

3.3 Highest impact iteration

For example, suppose we known that Tweety is a bird, she lays eggs, swims, and lives in Antarctica, *i.e.* she is a member of each of the following sets:

birds egg-layers swimmers antaxcticans

The ideal degree of belief that she flies would be equal to the statistic conditioned on all of these facts (and possibly many more, in more realistic reasoning situations).

Highest impact iteration derives this degree of belief iteratively, as described above, starting with a reasonable prior natural log odds of $\hat{\$}$(flyers | animals) = —3 (5% of animals fly). This odds value, as well as the impact values to follow, are artificial.

On the first iteration, the impacts of the known factors are as follows:

$\hat{!}$(birds, flyers, animals)	=	4-5.2
$\hat{!}$(egg-layers, flyers, animals)	=	+1.8
$\hat{!}$(swimmers, flyers, animals)	=	—5
$\hat{!}$(antarcticans, flyers, animals)	=	—1.6

The first two impacts embody positive influences on the degree of belief that Tweety flies, and the latter two impacts embody negative influences. By HIF, the impact of birds is chosen and added to the prior odds, producing a drastic swing of the belief to -3 + 5.2 = +2.2 (90% of birds fly).

On the next iteration, new impacts are used that are relative to the reference class animals ∩ birds = birds, as follows:

$\hat{!}$(birds, flyers, birds)	=	0
$\hat{!}$(egg-layers, flyers, birds)	—	0
$\hat{!}$(swimmers, flyers, birds)	=	—.05
$\hat{!}$(antarcticans, flyers, birds)	—	—4.4

The impact of birds has fallen to zero, since any factor is independent of the hypothesis given *itself.* Due to HIF, an impact of zero will always be considered last; this shows how HIF automatically avoids the incorporation of redundant information. This is further illustrated by the fact that the impact of egg-layers also fell to zero, since all birds happen to lay eggs (other egg-layers include reptiles). Thus HIF will avoid the redundancy of considering less specific factors.

Also note that the impact of swimmers lessened considerably. To see why, consider that the previous large negative impact was due to the fact that most swimmers are fish, which in general don't fly. Now that the impact is constrained to swimming *birds,* we find that only slightly less than half of swimming birds fly (ducks do, penguins don't, etc.). The opposite effect occurs with antarcticans: its negative impact magnifies, since we know that most Antarctic birds are penguins, which happen to not fly. This last impact is the clear winner, which is added to the previous log odds value of +2.2, to produce an updated belief of —2.2 (only 10% of Antarctic birds fly). On the next iteration, the only non-zero impact left is:

$\hat{!}$(swimmers, flyers, birds ∩ antarcticans) = —.1

This impact has become more negative, since there arc some flying Antarctic birds that don't swim. It is added to produce a final odds value of —2.3.

This example has shown how the HIF heuristic leads a statistical reasoner down paths of interdependent evidence. Standard approaches to evidential updating are better suited for a set of factors that are independent given the hypothesis [Pearl, 1988, pg. 38]. Here, however, the dynamic interactions between the factors' impacts is the most interesting aspect of this approach, and the whole key behind HIF. As more factors are incorporated, the remaining impacts will tend to diminish, promoting fast convergence.

Highest impact iteration is summarized by the following algorithm:

```
HII(hypothesis h, factors r, error e)
belief = prior(h)
context = ∅
while r ≠ ∅ do
    pick f of r with max abs(impact(f,h,context))
    delete f from r
    belief = belief + impact(f,h,context)
    add f to context
    if impact(f,h,context) ≤ e then break

return belief
```

As with the EvUpdate algorithm before, the time complexity of HII depends on the number of iterations, which this time is *at most* the number of known factors. Thus this algorithm may perform substantially fewer iterations, depending on the individual impacts and the error bound specified. However, HII must produce and rank *all* impacts for each iteration, instead of just the factor chosen in EvUpdate. Section 5 shows how this apparent inefficiency can be overcome by evaluating the impacts in parallel.

4 Statistical Causal Reasoning

Having now outlined the basics of statistics and highest impact heuristics, I turn to their application to causal reasoning. Since I have described a statistical notation using sets, it will be convenient to use a set-based causal notation when combining the concepts of statistical inference and causal reasoning. This section reviews the approach to statistical causal reasoning from Weber [1989], and then goes on to show how highest impact iteration provides a new technique for efficient causal reasoning.

4.1 Statistical causal rules

We start with a totally-ordered countable set of *moments* M, also known as a discrete time line. The successor to a moment *m* is represented by m'. This paper uses the common convention of using integers to represent moment constants.

A *fluent* is simply a set of these moments; for example, the fluent ignition is the set of all moments at which the car's key is turned. The fact that a fluent, is true of a particular moment is represented by set membership, e.g. 1 ∈ ignition. A collection of these set membership assertions comprises the reasoner\s *situational knowledge.*

An agent's *statistical causal knowledge* consists of conditional statistical assertions about fluents. This knowledge represents what is known about the domain in general. Specifically, I will use statistical assertions about how fluents that contain a moment *m* influence whether other fluents contain m', e.g.

$$\hat{\$}(\{m : m' \in \textbf{running}\} \mid \textbf{ignition}) = v$$

That is, the proportion of time points in which the car is running preceded by a time point in which the key was

turned is equal to v. Since I will be using rules of this form extensively, I will use the shorthand notation h^* for the set $\{m : m' \in h\}$.

4.2 Specificity as a representational solution to the qualification problem

McCarthy [1977] introduced the most famous example of the qualification problem, called the "potato in the tailpipe" scenario. Suppose you get in your car, and turn the key. You might expect the car to start, despite not knowing (for sure) whether the battery is still charged, whether the ignition system is intact, whether there is a potato in the tailpipe, nor whether any of a long list of relevant fluents hold. The qualification problem asks how a reasoner can make a causal prediction based on a *reasonable* body of evidence. These predictions are necessarily defeasible.

It is well known that statistically-founded beliefs such as I have described have defeasible characteristics [Etherington, 1987, Neufeld and Poole, 1988, Bacchus, 1988], *i.e.* the preferred statistical belief in the proposition "$\mathbf{x} \in \mathbf{a}$" can change (increase or decrease) as a result of new knowledge about x's inclusion in other sets. For example, a reasoning system may have the following rule:

$$\hat{\$}(\text{running}^* \mid \text{ignition}) = \ln(20),$$

which says that the odds are twenty to one that the car will be running after the ignition is engaged. This statistic is used if ignition is the only contextual information known. If it is also known that there is a potato in the tailpipe, then by specificity, the following statistic will be used:

$$\hat{\$}(\text{running}^* \mid \text{ignition} \cap \text{potato}) — \ln(1/10)$$

which says that the odds are ten to one *against* the car starting. In this way, the addition of more situational knowledge can override previous assessments of statistical belief, providing a representational solution to the qualification problem.

4.3 Highest impact iteration as a performance solution to the qualification problem

Applying highest impact iteration takes the above representational solution to the qualification problem one step farther, by controlling the tradeoff between the amount of evidence that specificity says to incorporate, the time the reasoner has to act, and the accuracy required by the situation. In the car starting example, specificity says that generally irrelevant factors such as the day of the week and the current U.S. president should be incorporated, since they are known properties of the moment in question. Highest impact iteration would preclude the consideration of these factors, unless an unusual alignment of already considered factors made their impacts significant. This solution solves a stronger version of the qualification problem than addressed in previous work: that the reasoner should employ whatever evidence is *practical* given the performance demands of the reasoning situation.

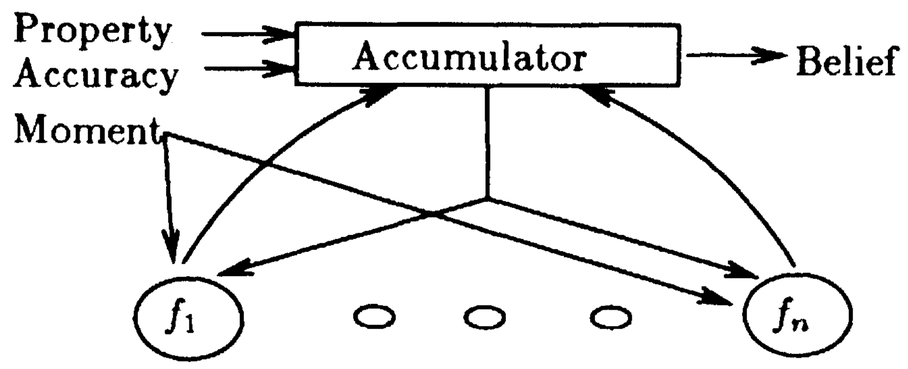

Figure 1: Flow of information within HITEST.

5 Parallel Impact Computations

Since impacts are sensitive to the nature of the current reference class, they must all be recomputed at each iteration, or at least examined to see if they must be recomputed. This would seem to be a prohibitively expensive operation. It would be senseless to use an algorithm where the complexity of a single iteration rivals the entire running complexity of more "naive" algorithms.

Fortunately, this problem is solved by the power of a key insight: *the impacts for the individual factors can be computed in parallel.* The parallel time complexity to compute the impacts is then merely the maximum time it takes to compute a single impact.

This approach is implemented by the HITEST (Highest Impact Techniques for Empirical STatistics) system, which runs on the RBN Butterfly multiprocessor. HITEST invokes parallel process control and message passing primitives from the SMP package [LeBlanc *et al.*, 1986J, developed here at the University of Rochester. The flow of information in HITEST is depicted in Figure 1. Statistical causal rules are encoded in the fluents' prior probabilities, stored in the *central accumulator,* and the reference class dependent impacts stored at the *fluent processors.* The situational knowledge is encoded by which fluent processors respond to the input moment m, *i.e.* exactly those processors whose corresponding fluents contain m.

5.1 The central accumulator

The accumulator collects incoming impacts from the fluent processors, and picks the highest. The winning impact is then added to the current degree of belief, implementing the highest impact first heuristic described in the last chapter. If the winning impact is less than or equal to the input accuracy, then the accumulator answers the original request by returning the current degree of belief, implementing the highest impact remaining heuristic. Otherwise, it broadcasts the name of the winning fluent processor to the fluent processors, and repeats the procedure in this paragraph.

5.2 The fluent processors

HITEST uses n fluent processors for n primitive fluents. A fluent processor is enabled, *i.e.* allowed to send impacts to the accumulator, if the input moment belongs to the processor's fluent. An enabled processor for fluent f starts by sending its *prior impact* $\hat{!}(f, h, \mathbf{M})$ to the accumulator. It waits until the accumulator returns the

```
[2] hitest running 0.2 +ignition -running -damp\
 +cold +heater
Timer started.
Prediction is now -1.935520
                    -damp          .012499
                    +ignition     4.733500
                    +cold         -.139392
                    -running      -1.590289
                    +heater        .001621
          +ignition has highest impact 4.733500
Prediction is now 2.797979
                    -damp          .128041
                    -running      -.465519
                    +heater        .037387
                    +cold         -.765457
                    +ignition     -.000000
          +cold has highest impact -.765457
Prediction is now 2.032521
                    -damp          .139698
                    +cold          .000000
                    +ignition      .000000
                    +heater        .331753
                    -running      -.779762
         -running has highest impact -.779762
Prediction is now 1.252759
                    -damp          .097163
                    -running       .000000
                    +heater        .514899
                    +cold          .000000
                    +ignition      .000000
          +heater has highest impact .514899
Prediction is now 1.767658
                    +damp          .178248
                    +running       .000000
                    +cold          .000000
                    +ignition      .000000
                    +heater        .000000
          -damp has highest impact .178248
Complete with final prediction of 1.945906
Elapsed time: 1.085687 seconds
```

Figure 2: An example run of HITEST.

name of the winner, and then computes the new revised impact, and repeats the send-revise loop.

The impacts are revised incrementally. A fluent processor can be thought of as a finite state machine, whose transitions are labeled by the values of the last highest impact, winner. Each transition, or equivalently, the new state the transition points to, causes a specific impact value to be output. Therefore, the state of the FSM is an encoding of the current reference class in the computation, with the start state corresponding to the vacuous reference class M. With this approach, impact revision takes constant time to index to and follow a state transition.

5.3 HITEST example run

Figure 2 shows the results from a request to know the degree of belief that the car will be running to an accuracy of 0.2, given that the car is currently not running, the ignition is engaged, the weather is cold but not damp, and the car has a device called a "block heater", which helps the car start in cold weather. The accumulator spawns the processes corresponding to these fluents, and then starts the iterative refinement of the degree of belief. At the start of each iteration, the accumulator prints out the current belief, which starts out as the prior probability of running. Thus the (non-logarithmic) prior odds starts as approximately $e^{-2} \approx 1/7$, *i.e.* the car is running during one-eighth of the moments. The data used to derive the priors and impact values are artificial; see Section 6 for more information on the derivation of these values.

The doubly-indented lines contain the impacts for the individual fluent programs (they appear in different orders on different iterations because the fluent processors are asynchronous). Notice that ignition jumps out as the winner, with running as a distant second. After ignition is chosen to be added, the belief jumps from (non-logarithmic) 1/7 to 16/1. This belief reflects that the car starts fairly reliably, and also the fact that the car will generally remain running when the ignition is engaged. On the next iteration, notice how the impact of ignition drops to zero. The fluent cold is chosen and added to the degree of belief.

On the third iteration, notice that the impact of the block heater has increased dramatically; this is because the impact is now evaluated with respect to cold, *i.e.* a block heater is only a significant factor when the weather is cold. However, its impact is obscured by the more important fluent running, so heater is not considered until the fourth iteration.

On the fifth iteration, the impact of damp has fallen below the specified error bound. The accumulator stops the execution of the fluent programs, and returns a value of 1.94, representing a 99% belief that the car will start in the given situation.

The "star" communication topology of Figure 1 is actually a special case of the topology used in HITEST. HITEST builds a tree of communicating processes, where the accumulator is at the root and the fluent programs are at the leaves. The use of a multi-level tree allows the search for the maximum impact to be conducted in parallel.

6 Compiling Impacts from Observations

A fluent program's impact table is what distinguishes it from other fluent programs. This table contains the impact of the program's fluent on every "future" fluent, with respect to every possible reference class, For n fluents the dimensions of this table are n by 3^n by 2. The last factor of two comes from the fact that a fluent program knows the impacts of both the fluent and its negation (one is not derivable from the other, although they must satisfy a consistency constraint). In general, the exponential size of the impact tables produced by FILLERUP will not be practical. In practice, however, a domain may support a large number of conditional independence assumptions that allow these tables to be reduced to a reasonable size.

Even ignoring the tedium of constructing these tables by hand, subtle errors in the table values can make the impact assignments *inconsistent.* The constraint be-

tween the impact of a fluent i_1 and the impact of its negation i_2 is as follows: there must exist two real numbers x and y such that $i_1 = x/y$ and $i_2 = (1-x)/(1-y)$. In addition, there is a constraint on impacts of different fluents that stems from the fact that the posterior probability is the same for any order of consideration of the fluents, *i.e.*

$$!(f, h, r) \cdot !(g, h, r \cap f) = !(g, h, r) \cdot !(f, h, r \cap g)$$

Thus the impacts contain some redundancy, both within a single fluent program and between fluent programs. This makes it very difficult to ensure consistency in hand-coded tables.

The program FILLERUP (Fabricated Impacts Locally . . .forget it) program performs the off-line compilation of these impact tables. The input to FILLERUP is an array of *weighted outcomes* each of which contains a *before part,* an *after part,* and a *weight.* The before and after parts are each boolean assignments to the set of primitive fluents, *i.e.* an encoding of what was true during a moment m and its successor m'. The weight expresses how many times that particular "confluence of fluents" was observed.

In a real sense, the FILLERUP program derives statistical causal rules from domain observations (simulated observations, in my examples). This removes the burden of specifying informative and consistent causal rules from the knowledge engineer. The derivation of causal rules is rarely even discussed in traditional approaches, largely due to the disparity between empirical observations and constructions like default rules. Furthermore, the impact values can be computed by scanning the outcome data in a single pass. Therefore, the impacts can be incrementally computed as new outcome data arrives. Though HITEST and FILLERUP do not attempt to do this, it is straightforward to construct a system that "trains" on domain observations, even when intermixed with requests for predictions.

7 Summary

It may not be practical for a statistical reasoner to base a degree of belief in a hypothesis on *all* factors known about the object question. The highest impact first heuristic defines an order of importance of these factors, based on their relative statistical impacts on the hypothesis. This inspires an iterative procedure where the degree of belief is incrementally refined, allowing the reasoner to trade prediction speed for accuracy. Furthermore, the highest impact remaining heuristic provides a way to estimate the current accuracy given the value of the highest impact.

Highest Impact Iteration becomes a practical approach when the statistical impacts can be computed in parallel. This suggests a parallel architecture where individual asynchronous units compute relevant impacts and send these values to a central node that accumulates the highest impacts for each iteration. HITEST is a working program on the BBN Butterfly that performs this incremental refinement of statistical beliefs from impacts computed in parallel. The priors and impacts for this computation are compiled from real or simulated empirical observations.

Causal reasoning is a promising application of highest impact iteration. The use of statistical causal rules is a powerful generalization of the traditional default rule approach, with highest impact iteration strengthening its solution to the qualification problem by guiding the search for a practical body of evidence. For more detail on all of these issues, see Weber [1989].

Acknowledgements

Special thanks to Josh Tenenberg, Henry Kyburg, and James Allen for technical discussions and moral support towards the development of the ideas in this paper.

References

[Bacchus, 1988] Fahiem Bacchus. *Representing and Reasoning with Probabilistic Knowledge.* PhD thesis, University of Alberta, Fall 1988.

[Etherington, 1987] David W. Etherington. M ore on inheritance hierarchies with exceptions: Default theories and inferential distance. In *Proceedings of AAA1-87,* pages 352-357, 1987.

[Kyburg, 1983] Henry E. Kyburg, Jr. The reference class. *Philosophy of Science,* 50:374-397, 1983.

[Kyburg, 1987] Henry E. Kyburg, Jr. Full beliefs. *Theory and Decision,* 1987.

[Kyburg, 1989] Henry E. Kyburg, Jr. Beyond specificity. Proceedings of IJCAI-89, 1989.

[LcBlanc *et a* /., 1986] Thomas J. LeBI anc, Neal M. Gafter, and Takahide Ohkami. SMP: A message-based programming environment for the BBN butterfly. Technical Report Butterfly Project Report 8, University of Rochester Computer Science Department, July 1986.

[Loui, 1987] Ronald P. Loui. *Theory and Computation of Uncertain Inference and Decision.* PhD thesis, University of Rochester Computer Science Department, September 1987.

[McCarthy, 1977] John McCarthy. Episternological problems of artificial intelligence. In *Proceedings of IJCA1-77,* pages 1034-1044, Cambridge, MA, 1977.

[Neufeld and Poole, 1988] Eric Neufeld and David L. Poole. Probabilistic semantics and defaults. In Shachter, Ross, and Levitt, editors, *The Fourth Workshop on Uncertainty in Artificial Intelligence,* pages 275-282, 1988.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, 1988.

[Reichenbach, 1949] H ans Reichenbach. *The Theory of Probability.* University of California Press, 1949.

[Weber, 1989] Jay C. Weber. *Principles and Algorithms for Causal Reasoning with Uncertainty.* PhD thesis, University of Rochester Computer Science Department, May 1989.