Eliminating Incoherence from Subjective Estimates of Chance

Randy Batsell
Rice University
Houston TX 77251
batsell@rice.edu

Lyle Brenner
University of Florida
Gainesville FL 32611
lbrenner@ufl.edu

Daniel Osherson
Rice University
Houston TX 77251
osherson@rice.edu

Moshe Y. Vardi
Rice University
Houston TX 77251
vardi@rice.edu

Spyros Tsavachidis
Rice University
Houston TX 77251
spy@rice.edu

Abstract

Human judgment is an essential source of Bayesian probabilities but is plagued by incoherence when complex or conditional events are involved. We consider a method for adjusting estimates of chance over Boolean events so as to render them probabilistically coherent. The method works by searching for a sparse distribution that approximates a target set of judgments. (We show that sparse distributions suffice for this purpose.) The feasibility of our method was tested by randomly generating sets of coherent and incoherent estimates of chance over 30 to 50 variables. Even with 50 variables, good approximations were computed within a few hours. Empirical test was provided by asking people to estimate the chances of events relating to the stock market. The estimates of each participant were incoherent but well approximated by the coherent distribution constructed for him. In addition, the quadratic scores of the reconstructed estimates were reliably superior to the scores of the original estimates. In this sense, our correction method improves the objective accuracy of the judgments that it renders coherent. The judgments of all participants were then pooled to form an "aggregate judge," and a coherent approximation to these 1426 aggregated judgments was computed. The quadratic scores for the resulting (coherent) probabilities were superior to the original estimates and also to the coherent approximations computed for subjects individually. Our method thus offers a new approach to aggregating opinions, an important problem in decision theory.

1 Introduction

Human judgment is a fertile source of information about uncertainty in the environment. Intuitive estimates of chance, however, often conflict with the axioms of probability when logically related events must be evaluated. Estimates that cannot be reconciled with any probability distribution are called *incoherent*. The tendency for human judgment to stray from coherence is well documented (Yates, 1990; Osherson, 1995) and is commonly encountered when eliciting probabilities for the construction of Bayesian networks (van der Gaag et al., 1999). Expert judgment in fields like medicine and physics is not immune to such errors, as revealed in experimental studies (e.g., Tversky & Kahneman, 1983; Viale & Osherson, 2000).[1] The difficulty people experience in maintaining coherence may stem in part from the intrinsic complexity of probability manipulations (Georgakopoulos et al., 1988).

One way to ensure probabilistic consistency is to pair each query to a judge with a precalculated response set containing all coherent possibilities (cf. Druzdzel & van der Gaag, 1995). Such structured elicitation can be tedious, however, since verifying coherence may involve large calculations. The estimates of chance finally obtained, moreover, may depend on the order in which they were generated because later estimates are more highly constrained than earlier ones. As a complement to structured elicitation, it is therefore desirable to possess a method for adjusting estimates of chance off line, rendering them coherent after the judge has left the scene. An obvious constraint on the method is to return coherent estimates that are as close as possible to the original judgments (an exact match

[1] Indeed, it is striking to observe how many well educated people find nothing exceptional about ascribing probability 30% to Man reaching Mars by 2020, 80% probability of a sustained global economic downturn starting in 2010, and 5% probability to the conjunction of these events.

can be achieved only if the judgments are coherent to begin with).

The present paper advances such a method. It presents an algorithm that accepts a corpus of probabilistic judgments (typically incoherent) and returns a revised corpus that is consistent with some joint distribution over all the events in play. The goal of the algorithm is to choose probabilities that are close to the original estimates. A successful algorithm of this kind can be used not only to amend inconsistent estimates of chance from a single judge. It can also suggest a consensus view among a group of experts whose judgments cannot be coherently pooled. The pooled judgments may be submitted to the algorithm in view of producing a single corpus of probabilistic estimates that minimally distorts the views of the individual judges. (Results on "aggregating" opinion in this way are reported below in Section 4.1.5.)

The principal challenge for the algorithm we seek is manipulating large probability distributions. Recall that a distribution over n Boolean variables assigns probabilities to 2^n elementary states (called "truth assignments" in what follows). Since practical applications can involve scores of variables, it is necessary to represent underlying distributions in compact form. For this purpose we rely on a simple data structure called a *probability array* (or *array*), described below. It will be shown that small arrays suffice to optimally approximate virtually any set of incoherent estimates of chance. To search for an array that approximates a target set of incoherent judgments, we rely on simulated annealing (van Laarhoven, 1988). As reported below, searching through the class of probability arrays via simulated annealing allowed us to efficiently approximate target sets of simulated judgments over 30 to 50 variables. The same method was applied to sets of 46 judgments over 10 variables in four empirical studies involving 134 human judges. Close approximations were again achieved efficiently. In the empirical studies, the *quadratic scores* of the reconstructed estimates were reliably superior to the scores of the original estimates. (The quadratic score, defined below, is a familiar measure of the objective accuracy of an estimate of chance.) Even better quadratic scores resulted from pooling the judgments of all participants in a given study into an "aggregate judge," and then applying our algorithm to achieve a coherent approximation. Thus, in both the individual and aggregate sense, our correction method improves the objective accuracy of the judgments that it renders coherent. Thus, the approach advocated here is both feasible and shown to improve the predictive power of judges, as well as to enable aggregation of the judges' estimates (a well known problem is decision theory, cf. Adler & Ziglio, 1996).

In place of probability arrays, we have also experimented with *algebraic decision diagrams* (ADDs), which have proven useful in other problems involving numerous variables (Bahar et al., 1997). Similarly, we have explored genetic algorithms as an alternative to simulated annealing. We compared the performance of the two approaches (probability arrays with simulated annealing and ADDs with genetic algorithms). The two approaches construct coherent approximations of similar quality, but differ markedly in their running times.

To proceed, we first specify the optimization problem posed by off line reconstruction of probability estimates. Then we give details about our optimization method involving arrays and simulated annealing. Experimental results are described for this method. We then describe the use of algebraic decision diagrams in conjunction with genetic algorithms.

2 Off line reconstruction as optimization

Finding a coherent approximation to incoherent estimates of chance amounts to solving an optimization problem. To state the matter formally, let $v_1 \cdots v_n$ be Boolean variables, representing the occurrence or non-occurrence of n logically independent events. Syntactically, the variables give rise to an infinity of formulas built up in the usual way from sentential connectives like $\neg, \wedge, \vee$. The formulas serve to describe absolute events whereas pairs $(\varphi : \psi)$ of formulas describe conditional events. Semantically, we have 2^n mappings (called *truth assignments*) from $\{v_1 \cdots v_n\}$ to $\{\mathit{true}, \mathit{false}\}$. A truth assignment α *satisfies* a formula φ if φ evaluates to *true* under α via standard propositional logic semantics. A *(probability) distribution* over $v_1 \cdots v_n$ is a mapping of the 2^n truth assignments into nonnegative numbers that sum to unity. Distribution Pr is extended to formulas φ via: $\mathsf{Pr}(\varphi) = \Sigma\{\ \mathsf{Pr}(\alpha) : \alpha \text{ satisfies } \varphi\ \}$. It is extended to pairs $(\varphi : \psi)$ of formulas via:

$$\mathsf{Pr}(\varphi : \psi) = \frac{\Sigma\{\ \mathsf{Pr}(\alpha) : \alpha \text{ satisfies both } \varphi \text{ and } \psi\ \}}{\Sigma\{\ \mathsf{Pr}(\alpha) : \alpha \text{ satisfies } \psi\ \}}$$

provided that $\mathsf{Pr}(\psi) > 0$. If $\mathsf{Pr}(\psi) = 0$ then $\mathsf{Pr}(\varphi : \psi)$ is undefined.

Consider a judge who is estimating the probabilities of some events and conditional events over $v_1 \cdots v_n$. We write $\mathit{Prob}(\varphi) = x$ to indicate the judgment that the probability of φ is x, and $\mathit{Prob}(\varphi : \psi) = y$ for the judgment that the conditional probability of φ assuming ψ is y. *Prob* is thus a finite function from formulas and pairs of formulas to numbers. If it coincides on its domain with some distribution Pr over $v_1 \cdots v_n$ then *Prob* is called *coherent,* otherwise *incoherent.* In the typical case, *Prob* is incoherent, and we seek to reconstruct it via a close coherent distribution. The resulting COHERENT APPROXIMATION PROBLEM is:

(1) *Let Prob map formulas $\varphi_1 \cdots \varphi_k$, and pairs of formulas $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$, into $[0,1]$. Find a map Prob* with the same domain as Prob such that Prob* is coherent, while minimizing*

$$\sum_{i \leq k} \mid Prob(\varphi_i) - Prob^*(\varphi_i) \mid +$$

$$\sum_{i \leq j} \mid Prob(\chi_i : \psi_i) - Prob^*(\chi_i : \psi_i) \mid .$$

Squared deviation from *Prob* could be substituted for the absolute deviation appearing in (1). In either case, it is easy to see how to convert an efficient solution to the Coherent Approximation Problem — or to its squared deviation version — into a method for testing the satisfiability of Boolean formulas, implying that the optimization problem is NP-hard.[2] We prefer absolute to squared deviation because it allows the use of linear programming in special cases (see below).

Note that distinct approximations *Prob** can yield the same distance to *Prob* yet assign different probabilities to the target events $\varphi_1 \cdots \varphi_k$, and $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$. For example, the incoherent judgments $Prob(p) = .3$, $Prob(\neg p) = .6$ are equally well reconstructed as $Prob^*(p) = .4$, $Prob^*(\neg p) = .6$ or as $Prob^*(p) = .3$, $Prob^*(\neg p) = .7$. Such multiplicity invites additional desiderata in the formulation of the Coherent Approximation Problem, for example, maximizing entropy. For simplicity in the current investigation, only deviation from *Prob* appears in our formulation (1). Absolute and conditional events, moreover, are given equal weight in calculating deviation from *Prob*, and only point estimates of probability are considered. Obviously, such matters can be settled differently within specific applications.

A special case of the Coherent Approximation Problem can be solved by linear programming (LP). If only the probabilities of absolute events are estimated, then LP can be used as follows to calculate the closest possible approximation. For each truth assignment α we have a variable x_α, and for each (absolute) event e we have a variable k_e. For each estimate p associated with an event e, we have two constraints. One has the form $k_e + \Sigma x_\alpha \geq p$, where the summation ranges over the truth assignments α that satisfy e. The other one has the form $-k_e + \Sigma x_\alpha \leq p$, with the same summation. The two constraints thus amount to $\mid p - \Sigma x_\alpha \mid \leq k_e$. A final constraint sets the sum of all the x_α's to unity. Note that this LP formulation of the Coherent Approximation Problem has an exponential number of variables, but the number of constraints is linear in the number of probability estimates. The objective is to minimize the sum of the k_e's. The resulting distribution (given by the x_α's) minimizes this sum and can be shown to yield a best approximation in the sense of (1).[3]

LP provides a useful benchmark of success in simple settings but it does not embody a general solution to the Coherent Approximation Problem. For one thing, LP is impractical for more than 30 variables (requiring a constraint matrix with more than a billion columns). More importantly, it cannot be applied to judgments involving ratios of probabilities, notably, when estimates are given for conditional events. We now describe a more general approach to the Coherent Approximation Problem. and thus a more general approach to off line reconstruction of incoherent judgments.

3 Optimization method

To confront the Coherent Approximation Problem, our strategy is to represent candidate distributions using probability arrays, and to search through them via simulated annealing. We consider arrays and annealing in turn.

A *probability array* of size (n, m) (for $n, m > 0$) is a set of m vectors each of form $\alpha_1^i \cdots \alpha_n^i, \beta^i$ $(1 \leq i \leq m)$, where $\alpha_k^i \in \{1, 0, *\}$, $\beta^i \in [0,1]$, and $\sum_{i=1}^m \beta^i = 1$. Letting the vectors be columns, one array of size $(3,4)$ may be pictured as in (2)a, below.

[2] In addition to the problem of intractability is the possibility that there may not even be a minimum distance between *Prob* and a coherent approximation *Prob** to it. The incoherent judgments $Prob(p : q) = .5$, $Prob(q) = 0$, for example, can be approximated by setting $Prob^*(p : q) = .5$ and $Prob^*(q)$ arbitrarily close to 0, but not 0 itself.

[3] Neither the distribution nor the optimizing k_e's are unique. Many choices of the k_e may produce minimum deviation from the target estimates.

(2)

$$(a)\ \begin{array}{|cccc|}\hline 1 & 0 & * & 0 \\ 0 & * & * & 0 \\ 1 & 0 & 1 & 0 \\ \hline .2 & .1 & .4 & .3 \\ \hline\end{array} \qquad (b)\ \begin{array}{|cccc|}\hline 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ \hline .2 & .1 & .4 & .3 \\ \hline\end{array}$$

$$(c)\ \begin{array}{|cccc|}\hline * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ \hline .2 & .1 & .4 & .3 \\ \hline\end{array}$$

The first three rows of (2)a correspond to three (Boolean) variables v_1, v_2, v_3, respectively. The first column represents the truth-assignment v_1 = true, v_2 = false, v_3 = true, and assigns it probability .2. The second column represents two truth-assignments, namely, v_1 = false, v_2 = true, v_3 = false, and v_1 = false, v_2 = false, v_3 = false. In other words, the symbol $*$ functions as "don't care." The two truth-assignments represented by the second column of (2)a share the value .1, each receiving .05. The other two columns are interpreted similarly. (Four truth-assignments are represented in the third column, and are each assigned .4/4. The sole truth-assignement represented by the fourth column is assigned .3.) Notice that the same truth-assignment may appear in more than one column. The probability of such a truth-assignment is the sum of the values it receives in each column where it appears. For example, in (2)a, the truth-assignment v_1 = true, v_2 = false, v_3 = true is represented in columns 1 and 3. Its probability according to (2)a is $.2 + (\frac{1}{4} \times .4) = .3$. Some truth-assignments may not appear anywhere in the array, and are therefore assigned probability zero. For example, in (2)a, the probability of v_1 = true, v_2 = true, v_3 = false is 0. As an aid to intuition, the extreme arrays (2)b,c may be helpful. The first represents a highly sparse distribution, placing all probability on one truth-assignment. The second represents the uniform distribution.

It should be clear that every array of size (n, m) represents a distribution over n variables. It should also be clear that:

(3) FACT: *Every distribution over n variables that assigns positive probability to no more than m truth-assignments is represented by some array of size (n, m).*

Let us now note that following fact (exploited in a similar way by Fagin et al., 1990).

(4) FACT: *Let formulas $\varphi_1 \cdots \varphi_k$, and pairs $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$ of formulas be given. For every distribution* Pr *there is a distribution* Pr' *such that:*

(a) Pr' *assigns positive probability to at most $k+j+1$ truth assignments;*

(b) $\mathsf{Pr'}(\varphi_i) = \mathsf{Pr}(\varphi)$ *for every* $1 \leq i \leq k$;

(c) $\mathsf{Pr'}(\chi_i : \psi_i) = \mathsf{Pr}(\chi_i : \psi_i)$ *for every* $1 \leq i \leq j$ *with* $\mathsf{Pr}(\chi_i : \psi_i)$ *defined;*

Proof of Fact (4): Let k formulas and j pairs of formulas be given as in (4). Suppose that all the formulas are written over the same set of n variables. Let Pr be a probability distribution for the formulas and pairs of formulas. Hence, Pr maps the 2^n truth assignments $\tau_1, \tau_2, \ldots \tau_{2^n}$ into non-negative real numbers $x_1, x_2, \ldots, x_{2^n}$ such that $\sum_{i=1}^{2^n} x_i = 1$. Let

$$\begin{array}{rcll} S_i & = & \{\ell : \tau_\ell \text{ satisfies } \varphi_i\} & \text{for } i \leq k, \\ Y_i & = & \{\ell : \tau_\ell \text{ satisfies } \chi_i\} & \text{for } i \leq j, \\ Z_i & = & \{\ell : \tau_\ell \text{ satisfies } \psi_i\} & \text{for } i \leq j. \end{array}$$

It then follows that:

$$\begin{array}{rcll} \mathsf{Pr}(\varphi_i) & = & \sum_{\ell \in S_i} x_\ell & \text{for } i \leq k, \\ \mathsf{Pr}(\chi_i : \psi_i) & = & \dfrac{\sum_{\ell \in Y_i \cap Z_i} x_\ell}{\sum_{\ell \in Z_i} x_\ell} & \text{for } i \leq j. \end{array}$$

The latter equalities imply that the following equations have a nonnegative real solution.

$$\sum_{\ell \in S_i} x_\ell = \mathsf{Pr}(\varphi_i), \quad \text{for } i \leq k,$$

$$\sum_{\ell \in Y_i \cap Z_i} x_\ell - \mathsf{Pr}(\chi_i : \psi_i) \cdot \sum_{\ell \in Z_i} x_\ell = 0, \quad \text{for } i \leq j,$$

$$\sum_{i=1}^{2^n} x_i = 1.$$

It is well known that if a system of m linear equations in p unknowns has a nonnegative solution then it has a solution with at most m nonnegative values (see, e.g., Chvátal, 1983, Thm 9.3). It thus follows immediately that there is another solution to the last equations with at most $k+j+1$ non-negative entries. The desired sparse probability distribution Pr' consists of these values. ■

From (4) it follows that our search for an approximating distribution (in the sense of (1)) can be limited to sparse distributions. Hence (3) implies that the goal of the search can be restricted to the class of probability arrays. Specifically, if the judge has estimated probabilities for m events and conditional events over n variables, then an optimal coherent approximation is represented by some array of size $(n, m+1)$. The optimal array need not include *, but the presence of *'s allows optimal approximation in some cases by even smaller arrays. Furthermore, the presence of *'s allows the search to go through intermediate points that cannot be represented by a *-free array of size $(n, m + 1)$. We come back to the issue of *'s later.

We now turn to the search algorithm. Simulated annealing is described in many places (e.g., van Laarhoven, 1988). It suffices here to explain the selection of initial points, and also how neighbors to a given point were constructed. As initial points for a problem involving m events over n variables, we randomly generated a set of probability arrays of size $(n, \hat{m})$, where $\hat{m}$ was determined empirically in light of our experience finding good approximations (specific choices of $\hat{m}$ are reported below). Each cell of an array had probability $\hat{p}$ of being assigned *, and otherwise equal probability of being assigned either 1 or 0 ($\hat{p}$ was also determined empirically). The last row of each column (the probabilities) were uniformly randomly chosen from $[0, 1]$ (using double-precision floating-point numbers), then renormalized to sum to 1.0. For creating neighbors, the algorithm relied on the following routine.

> Routine for creating neighbors to an array $\mathcal{A}$: for every column in $\mathcal{A}$, with 10% probability (a) randomly choose one of the top n entries and replace its entry with a randomly chosen member of $\{0, 1, *\}$, and (b) multiply the last entry (namely, the probability) by a random choice between .9 or 1.1, then renormalize across columns (so that they sum again to 1.0).

The initial and final temperatures, as well as the cooling schedule for the algorithm conformed to the recommendations in van Laarhoven (1988). The number of temperature steps in the algorithm was determined empirically, as was the number of iterations at a given step. When we employed multiple starting points, they were managed via the procedure known as "go with the winners" (Aldous & Vazirani, 1994).

4 Tests of the method

We first tested our method by soliciting estimates of chance for logically simple and complex events over sets of 10 variables. To assess its scalability, we then tested the method against simulated judgments over 30+ variables. These tests are described in turn. All experiments were run on a Pentium III PC running at 533 MHz within the Java Virtual Machine for Windows '98.

4.1 Empirical studies

4.1.1 Data collected

Consider the following event: "The change in value of United Airlines stocks in the third quarter of 2000 will be more favorable than the change in value of the S&P 500 composite." Nine other events of the same form were constructed involving the companies Continental, American, Southwest, Exxon, Chevron, Texaco, British Petroleum, Enron, and Schlumberger. The above event was abbreviated to "United Airlines outperforms the S&P 500," and similarly for the other companies.

In the summer of 2000, 26 MBA students at the Jones School of Management (Rice University) along with 5 professional stock traders estimated the probabilities of the ten events plus the probability of 36 complex events built from them.[4] The 36 complex events included an individually randomized selection of 6 events from each of the following categories.

(a) CONDITIONAL EVENTS like "Exxon outperforms the S&P 500 *assuming that* Chevron outperforms the S&P 500."

(b) CONDITIONAL EVENTS WITH NEGATION like "Exxon outperforms the S&P 500 *assuming that* Chevron does not outperform the S&P 500."

(c) CONJUNCTIVE EVENTS like "Exxon outperforms the S&P 500 *and* Chevron outperforms the S&P 500."

(d) CONJUNCTIVE EVENTS WITH NEGATION like "Exxon outperforms the S&P 500 *and* Chevron does not outperform the S&P 500."

(e) DISJUNCTIVE EVENTS like "Exxon outperforms the S&P 500 *or* Chevron outperforms the S&P 500."

(f) DISJUNCTIVE EVENTS WITH NEGATION like "Exxon outperforms the S&P 500 *or* Chevron does not outperform the S&P 500."[5]

The six events in each category were chosen individually randomly for each participant from the 45 non-trivial possibilities (thus, different events were chosen

[4] The data from MBA students and traders were not distinguishable so they are treated together.

[5] The inclusive meaning of *or* was explained to all participants prior to collecting their estimates. Other instructions clarified the conditional reading of the expression *assuming that*, as well as the exact financial significance of outperforming the S&P 500 index.

for different participants). The participants first made probability estimates for the 10 variables, then for the 36 complex events in individualized random order under the constraint that the six events in a given category appear as a block. The 46 queries were administered via a web interface at the participant's convenience and required about half an hour to collect.

4.1.2 Incoherence

Not one of the 31 participants offered a coherent set of estimates. For example, they averaged 3.0 violations (out of 6 possible) of the following constraint on the probabilities of conjunctions of form $p \wedge q$.

(5) $\mathsf{Pr}(p)+\mathsf{Pr}(q)-1 \leq \mathsf{Pr}(p\wedge q) \leq \min\{\mathsf{Pr}(p),\mathsf{Pr}(q)\}$.

Similarly, they averaged 3.23 violations (out of 6) of the following constraint on disjunctions of form $p \vee q$.

(6) $\max\{\mathsf{Pr}(p),\mathsf{Pr}(q)\} \leq \mathsf{Pr}(p \vee q) \leq \mathsf{Pr}(p) + \mathsf{Pr}(q)$.

Constraints corresponding to conditional events yielded comparable rates of violation.

4.1.3 Approximation via simulated annealing over arrays

We applied our algorithm to each participant separately, represented by his/her dataset of 46 judgments. There were ten starting points, 150 temperature steps, and 25 iterations per step. Arrays were of size $(10, 50)$. The probability of $*$ was set to .2. With these parameters, the algorithm required less than 20 seconds per dataset. We experimented with different number of columns (25, 50, and 100) and different probabilities of $*$ (0, 0.2, and 0.4). The running time scales roughly linearly with the number of columns, but was not affected by the probability of $*$. Neither the number of columns nor the probability of $*$ had a measureable impact on the quality of the approximation, indicating the robustness of the technique.

For each participant we calculated the *mean absolute deviation* (MAD) between her 46 probability estimates and those offered by the best approximating array discovered for her. Across the 31 participants, the average MAD was .095 (S.D. = .043). Thus, to render coherent our judges' estimates of chance, we modified them by adding or subtracting roughly .1, on the average.

4.1.4 Comparison to linear programming

Perfect approximation (MAD of 0) by a probability distribution is ruled out because of the incoherence that characterizes the estimates. For just the 34 estimates of absolute events the closest possible approximation can be calculated using linear programming (LP) as discussed in Section 2 above. We applied LP to each of the 31 participants individually. The average MAD obtained was .081 (S.D. = .049). For comparison purposes, we again applied our simulated annealing algorithm to each participant after deleting the estimates of conditional events. (Both LP and the second application of simulated annealing involved 34 absolute estimates per participant.) The average MAD achieved was .093 (S.D. = .049). Thus, for absolute events, the simulated annealing algorithm came within 15% of producing an optimal approximation.

4.1.5 Impact on accuracy

We desire to improve the estimates of a judge by making them coherent. But if the process robs the judgments of their accuracy, the judge might prefer that we left her estimates in their original state. The accuracy in a judgment can be measured via its quadratic score, defined as follows (see von Winterfeldt & Edwards, 1986, for general discussion of scoring rules).

Suppose that *Prob* represents the estimates of a given judge. Let E be an event in the domain of *Prob*, and let $(G : F)$ be a pair of events in the domain of *Prob*.

- The quadratic score incurred by *Prob* for E is $(1 - Prob(E))^2$ if E is true. It is $Prob(E)^2$ if E is false.
- The quadratic score incurred by *Prob* for the pair $(G : F)$ is $(1 - Prob(G : F))^2$ if both G and F are true. It is $Prob(G : F)^2$ if G is false and F is true. It is not defined if F is false.

The *overall quadratic score* of *Prob* is the average of all the scores incurred by *Prob* for events and pairs of events in its domain. (Pairs of events for which the quadratic score is not defined do not figure in this average.) Notice that the quadratic score is best interpreted as a penalty; low scores signal accurate judgment. If a judge estimated every probability at .5, her overall quadratic score would be .25. While such estimates would be incoherent in our experiment, an overall score below .25 is a convenient benchmark of knowledge about the domain in question.

After the facts were established about the third quarter performance of our 10 stocks, we computed the overall quadratic score for each of the 31 participants separately. The average, overall score was .254 (S.D. = .056). For each participant, we then computed the overall quadratic score associated with the reconstructed (coherent) estimates due to our algorithm. (The same events and conditional events figure in the scores associated with the original estimates and with their coherent reconstruction.) The average, overall quadratic score for the coherent approximations was .232 (S.D. = .057), which is reliably lower than the original scores ($p < .001$ by correlated t test). The scores for 23 of the 31 participants were lower when computed with the coherent approximation rather than the original estimates. A majority of this size is unlikely to arise by chance ($p < .01$ by a binomial test).

For another assessment of accuracy, define a participant's *discrimination index* to be the average probability assigned to true events minus the average probability assigned to false ones. Conditional events figure in the discrimination index provided that the conditioning events are true. Good judges assign higher probabilities to events that come true, hence have higher discrimination indexes. (See Yates, 1990, Ch. 3 for discussion.) The average discrimination index for the 31 participants was .129 (S.D. = .176) whereas the average discrimination index for the 31 coherent approximations was .180 (S.D. = .156). Once again, this difference is reliable ($p < .001$, by correlated t test). The discrimination index for 26 of the 31 participants improved in the transition from raw to reconstructed estimates.

Overall, our participants seem not to have enjoyed much insight into the stock market. (Thus, their average quadratic score was worse than what could be achieved by responding with .5 to each query.) It is nonetheless clear that our method of coherent reconstruction did not make their judgment worse. Rather, it produced a reliable increase in accuracy.

Next we pooled all the judgments from our 31 participants to form an "aggregate judge" with $31 \times 46 = 1426$ judgments. Applying our algorithm[6] to the aggregate judge resulted in MAD = .203 (after 2 minutes of computation).[7] Three quadratic scores were then compared, namely, (a) the average quadratic score for the original judgments of the individual judges, (b) the average quadratic score for the coherent judgments that result from applying our algorithm to the individual judges, and (c) the quadratic score for the coherent judgments that result from applying our algorithm to the aggregate judge. The respective quadratic scores were .253, .231, and .198. By a correlated t-test, the quadratic score in (c) (that is, for the corrected aggregate judge) was reliably lower than the two others ($p < .001$). For 23 of the 31 participants, the aggregate quadratic score (c) was lower than both the scores associated with original judgments (a) and with individual correction (b). Thus, our set of judges has statistically significant "collective wisdom", even though as individuals their estimates are generally poor.

4.2 Other judgment experiments

We performed three other experiments of identical design. Each involved 10 variables and 36 complex events of forms (a) - (f). One experiment concerned weather prediction and variables like:

> One week from today at noon, it will be at least 58 degrees in Philadelphia.

Five cities and two weather conditions (temperature and precipitation) gave rise to the ten variables. Another experiment required predicting the outcome of a National Basketball Association game between the Houston Rockets and the Phoenix Suns. The ten variables included:

- The Rockets lead at halftime.
- The Rockets have fewer turnovers than the Suns.

The last experiment also involved the NBA, this time a game between the Rockets and the Dallas Mavericks.[8] Let us call the four experiments weather, Suns, Mavericks, and stocks, respectively. There were 38 participants in weather, 29 in Suns, and 36 in Mavericks. All were Rice undergraduates. Except for some overlap between Suns and Mavericks, the sets of participants in the four experiments were disjoint.

We performed the same analyses for weather, Suns, and Mavericks as we did for stocks. The results are now summarized, starting with the incoherence of the participants' estimates. The average numbers of violations of (5) in weather, Suns, and Mavericks were 2.92, 4.21, and 3.64, respectively. For (6), these numbers are 2.16, 3.86, and 2.67. The average MAD between a participant's estimates and those supplied by our coherent approximations was .085 for weather, .115 for Suns, and .094 for Mavericks. Applying LP to just the 34 absolute events yielded coherent approximations with

[6] Again, the results are robust with respect to the number of columns and the probability of $*$; 100 columns suffice, even though we have 1426 judgments.

[7] Higher MAD for the aggregate subject compared to the average MAD for the individual subjects is virtually inevitable since the aggregate involves a proper superset of the judgments of each individual.

[8] Monetary incentives were offered for accurate probabilities in the two basketball experiments.

MADs of .070 for weather, .107 for Suns, and .081 for Mavericks. In comparison, our method applied just to absolute events gave rise to MADs of .083 for weather, .118 for Suns, and .092 for Mavericks. Simulated annealing thus performs within roughly 16% of optimum for absolute events. The average quadratic score for the participants in weather was .232. For Suns and Mavericks the average quadratic scores were .237 and .228, respectively. The quadratic scores for the coherent approximations in weather, Suns, and Mavericks were .201, .203, and .200, respectively. In weather, 30 of the 38 participants improved their quadratic scores in the transition from raw to reconstructed estimates. Improvement occurred for 26 of the 29 participants in Suns, and for 34 of the 36 participants in Mavericks. The average discrimination indexes for the raw estimates in weather, Suns, and Mavericks were .154, .138, and .133, respectively. The average discrimination indexes for the reconstructed estimates in the three experiments were .199, .224, and .216, respectively. Discrimination indexes improved for 34 of the 38 participants in weather, for 25 of the 29 participants in Suns, and for 34 out of 36 participants in Mavericks.

To determine the impact on accuracy of aggregation, for each of weather, Suns, and Mavericks, we pooled all the judgments from the respective participants to form an "aggregate judge." Applying our algorithm to the aggregate judges resulted in respective MADs of .175, .205 and .183. For each study the same three quadratic scores were compiled as before, namely, (a) for the original judgments of the aggregate judge, (b) for the coherent judgments that result from applying our algorithm to the participants individually, and (c) for the coherent judgments that result from applying our algorithm to the aggregate judge. In the case of weather, the respective quadratic scores were .231, .210, and .191, all reliably different by a correlated t-test ($p < .001$). For 30 of the 38 participants, the aggregate quadratic score (c) was lower than the raw quadratic score (a), and (c) was lower than the individually corrected quadratic score (b) for 27 of the 38 participants. These proportions are greater than expected by chance ($p < .01$ by a binomial test). In the case of Suns, the respective quadratic scores were .237, .203, and .166, all reliably different by correlated t-test ($p < .001$). For 28 of the 29 participants, the aggregate quadratic score (c) was lower than the raw quadratic score (a), and (c) was lower than the individually corrected quadratic score (b) for 23 of the 29 participants ($p < .01$ by binomial test). Finally, for Mavericks, the respective quadratic scores were .228, .200, and .172, all reliably different by correlated t-test ($p < .001$). For 30 of the 36 participants, the aggregate quadratic score (c) was lower than the raw quadratic score (a), and (c) was lower than the individually corrected quadratic score (b) for 28 of the 36 participants ($p < .01$ by binomial test).

4.3 Scalability studies

As a test of the computational feasibility of our method in large problems, we created sets of simulated estimates of chance and tried to match them with our algorithm. In some cases the target estimates were constructed to be coherent because the ideal level of accuracy in this case is known (namely, a MAD of zero). To simulate incoherent estimates, we started with coherent probabilities, then perturbed them randomly by adding or subtracting a constant k. (A coin toss determined whether to add or subtract under the restriction that the resulting number remain in the unit interval.) In different tests, k was chosen to be either .1 or .2 (or else zero in the coherent case). Note that the value of k provides an upper bound on the lowest MAD that can be achieved between the original judgments and their coherent approximations.

Nine tests were carried out, involving distributions with 30, 40 and 50 variables. For each distribution, $20,000$ truth-assignments were randomly chosen to carry positive probability. To ensure that the distribution was strongly nonuniform (the only challenging case), we proceeded as follows. Truth assignments can be naturally ordered by the binary numbers they encode (relative to a prior ordering of variables). Let α_i ($1 \leq i \leq 2^n$) be one such ordering. For each α_i obtaining positive probability, a number r was uniformly randomly selected in the interval $(0, 1)$, then multiplied by $1+(.01 \times i)$ and assigned to α_i. Normalization then ensured that the distribution sums to unity. Multiplying by $1 + (.01 \times i)$ in the prenormalization step skews the distribution.

Events were represented by randomly constructed formulas in disjunctive normal form. Specifically, for each event, we randomly chose a number between 1 and 5 to serve as the number of conjunctions, then for each conjunction we randomly chose between 1 and 10 variables with randomly determined polarity to serve as conjuncts. (The variables were drawn from a set of size 30, 40, or 50, depending on the simulation.) Conditional events were constructed in the same way from pairs of events, the second serving as conditioning event.

For each of the 9 tests, we generated 100 events and 100 conditional events. Their probabilities were computed relative to a distribution of the kind described above, then perturbed if incoherence was involved. For events involving n variables, we performed simulated

vars.	30		40		50	
Inc.	MAD	Min	MAD	Min	MAD	Min
0	.0063	132	.0054	154	.0062	140
.1	.0638	135	.0638	140	.0680	144
.2	.1428	85	.1472	104	.1454	215

Table 1: Mean absolute deviation (MAD) and time in minutes (Min) achieved on simulated data generated from sparse distributions with 20,000 positive states. Running times are relative to a Pentium III (533 MHz) processor and the Java Virtual Machine for Windows '98.

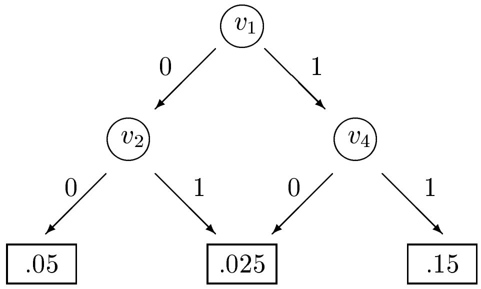

Figure 1: An ADD for a distribution over the variables v_1, v_2, v_3, v_4. According to the distribution, $\mathsf{Pr}(\bar{v}_1\bar{v}_2\bar{v}_3\bar{v}_4) = .05$, $\mathsf{Pr}(\bar{v}_1\bar{v}_2 v_3\bar{v}_4) = .05$, $\mathsf{Pr}(v_1\bar{v}_2\bar{v}_3 v_4) = .15$, $\mathsf{Pr}(v_1\bar{v}_2\bar{v}_3\bar{v}_4) = .025$, etc.

annealing over arrays of size $(n, 100)$, with one starting point, 150 temperature steps, 25 iterations per step, and probability of * set to .6. (Again, the algorithm was quite robust with respect to the number of columns. Here, however, the probability of $*$ played an important role; setting it to zero resulted in the algorithm not converging.)

Results are shown in Table 1. To interpret the table, consider the simulation involving coherent judgments and 50 variables. The simulated annealing algorithm produced a coherent approximation within 140 minutes whose mean absolute deviation from the original 200 judgments was .0062. For another example, consider again 50 variables, with probabilities that are rendered incoherent by adding or subtracting .1. Within 144 minutes, the algorithm produced a coherent approximation whose mean absolute deviation from the original 200 judgments was .0680. These results provide evidence for the scalability of our approach to reconstructing incoherent estimates of chance.

5 An alternative optimization technique

Probability distributions may also be compactly represented by *algebraic decision diagrams* (ADDs) instead of probability arrays. Indeed, ADDs have already proven to be a useful data structure in several contexts (Bahar et al., 1997). In this section, we describe how ADDs can be used to find a coherent approximation to incoherent judgments.

5.1 Algebraic decision diagrams

An ADD is a generalization of a *reduced, ordered, binary decision diagram* or ROBDD (Bryant, 1986). The latter structure is a DAG with terminal nodes labeled either 0 or 1, and non-terminal nodes with out-degree two labeled by variables. One outgoing edge from a given variable has label 0, the other 1. The DAG is ordered in the sense that all its paths respect a fixed linear ordering of the variables. It is reduced in the sense that (a) two nodes with the same label and having the same 0- and 1-successors (if any) are identified, and (b) there is no nonterminal node whose 0- and 1-successors are identical. An ADD is like an ROBDD except that its terminals can assume any real value. See Figure 1. An ADD maps each truth assignment to one of its terminal nodes in the obvious way. For example, the ADD of Figure 1 maps the uniform assignment of *true* to .15 via its rightmost edges (v_2 and v_3 are treated as "don't cares" on this path). Given an ordering of the variables, a given mapping of truth assignments to numbers is represented by a unique ADD. See Bahar et al. (1997) for the theory of ADDs, and applications. (To avoid misunderstanding, we stress that Figure 1 does not represent a Bayesian network.)

In the present context, we restrict attention to ADDs that yield probability distributions, namely, whose terminals are nonnegative and such that $\Sigma\,\mathsf{terminal}\,(\alpha) = 1$ where the sum is over all truth assignments α, and $\mathsf{terminal}\,(\alpha)$ is the value of the terminal node on the path that represents α. In principle, every probability distribution represented by an ADD can also be represented by a probablity array, but the ADD representation might be much more compact. To manipulate ADDs, we rely on a subset of operations defined in the software package CUDD from Colorado University, primarily, on inner product (via matrix multiplication) and on variable reordering.[9] The latter operation seeks to minimize the storage requirements of a set of ADDs through choice of a single ordering of variables for them all. (This operation is heuristic in character.) Inner product is used as follows to

[9]The package may be downloaded via: `http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html`

calculate the probability of a given formula φ according to a probability distribution represented by a given ADD A. We first represent φ as a Boolean function by means of an ROBDD B (i.e, an ADD with terminals limited to 0 and 1). A and B can be conceived as vectors over the same set of truth assignments. The desired probability is therefore obtained via their inner product. The time-complexity of this operation is linear in the product of the number of nodes in A and B.

5.2 Genetic algorithms applied to ADDs

To find a coherent approximation to input judgments, we search through the space of probability distributions represented by ADDs, using genetic algorithms.[10] We now provide details about the genetic algorithm used to produce the results presented in the sequel. (For general properties of genetic algorithms, see Mitchell, 1996.) We describe (a) the starting population, (b) the processes of crossover and mutation, and (c) the construction of the next generation. The numerical parameters cited below (e.g., for mutation and crossover) were determined by trial-and-error. The same parameters are used in all the experiments reported later.

5.2.1 Initial population

To form the initial population, a set of ADDs is generated randomly. A given ADD is created as follows. Suppose there are n variables, $v_1 \cdots v_n$. First, a set of m ternary sequences of length n is randomly generated (we chose $m = 50$). Each coordinate of a given sequence holds either *true, false*, or *don't care*. For each sequence, a randomly chosen probability p is chosen. By identifying coordinate i of a given sequence with v_i, each ternary sequence represents a set of truth assignments, all with probability p. In essence, such a sequence is a single column of a probability array. An ADD is created for each such set of truth assignments (one ADD for each ternary sequence). The different ADDs corresponding to the different ternary sequences are then summed into one larger ADD whose terminal nodes are normalized to ensure that the ADD represents a probability distribution. Thus, each inital ADD represents a probability array of size (n, m). The number of nodes of such an ADD is bounded from above by mn. After all the ADDs in the starting population are created, the variables $v_1 \cdots v_n$ are reordered by routines in the software package CUDD in view of minimizing the storage requirements of the entire set of ADDs. (The same variable ordering governs all the ADDs in the population.)

5.2.2 Crossover and mutation

One-point crossover between two ADDs A_1, A_2 proceeds as follows. Each truth assignment determines an binary number by associating truth with 1 and falsity with 0 (the ordering of variables determined by CUDD orders the digits of these numerals). One of these 2^n numbers is chosen uniformly randomly, and two distributions are created. In one distribution, the truth assignments below the chosen point are given probabilities according to A_1, the remaining truth assignments get probabilities according to A_2. The other distribution receives the complementary probabilities. In both cases, the resulting ADDs are renormalized to sum to unity. Mutation of a given ADD starts from another choice of a number between 1 and 2^n. With probability 0.5, the terminal nodes of all paths in the ADD that correspond to a truth assignment below the chosen number are multiplied by .9, and the remaining terminals are multiplied by 1.1. With probability 0.5, the terminal nodes of all paths in the ADD that correspond to a truth assignment below the chosen number are multiplied by 1.1, and the remaining terminals are multiplied by .9. The ADD is then renormalized.

5.2.3 Construction of successive generations

Let *Prob* represent the set of target estimates to be approximated, and let $Prob^*$ be the distribution represented by a given ADD A in our population. Define *dev* to be the absolute deviation between *Prob* and $Prob^*$ in the sense of (1)[11] We take the *fitness* of A to be $1/(dev + .01)$. Thus, greater match to *Prob* yields higher fitness. (Adding .01 prevents division by zero.) Between two generations, the probability of being selected for mating is proportional to fitness. For a population of (even) size N, $N/2$ pairs of ADDs are selected with replacement on this basis. With probability .95 the pair undergoes crossover. Whether crossed or not, the pair then undergoes mutation with probability .04. The two chromosomes then enter the next generation (again yielding a population of N). Better variable ordering is once again sought for the entire population.

[10] We did not achieve good results when ADDs were searched using simulated annealing. Similarly, searching through arrays with genetic algorithms performed poorly. We are unable to explain why certain combinations of data structures and search techniques work well whereas others do not.

[11] It can happen that there is a conditional event $(\varphi : \psi)$ among the target estimates for which the ADD's distribution $Prob^*$ is undefined (because $Prob^*(\psi) = 0$). In this case (which is extremely rare), we augment *dev* by a small value.

vars	30		40		50	
Inc	MAD	Min	MAD	Min	MAD	Min
0	.0037	56	.0029	106	.0033	127
.1	.0832	43	.0826	90	.0884	139
.2	.1687	26	.1699	46	.1711	75

Table 2: Mean absolute deviation (MAD) and time in minutes (Min) achieved using genetic algorithms and ADDs on simulated data generated from sparse distributions with 20,000 positive states. Times are relative to a Pentium III (533 MHz) processor running C code.

Across generations, the ADD with highest fitness (lowest *dev*) is retained, and its distribution is used to approximate *Prob*. We were prepared to discourage the growth of large ADDs by incorporating size into the measure of fitness. But we observed manageable growth of ADDs across generations, so *dev* alone was used to determine fitness.

5.3 Test of the new method

Let us use the abbreviation "GAADD" to denote the algorithm based on genetic algorithms and ADDs, and "SIMARR" to denote simulated annealing applied to arrays. We applied GAADD to the four experimental studies described in Sections 4.1 and 4.2 above. For each participant in the experiment on stock market forecasting we calculated the mean absolute deviation (MAD) between her 46 probability estimates and those offered by the best approximating ADD discovered for her. Across the 31 participants, the average MAD was .097, compared to .095 for SIMARR. Regarding the other experiments, the average MAD between a participant's estimates and those supplied by the best approximating ADD was was .085 for weather, .121 for Suns, and .094 for Mavericks. The MADs achieved with SIMARR were .085, .115, and .094, respectively. The quadratic scores for the coherent approximations delivered by GAADD were as low or lower than those for SIMARR.

The two methods thus constructed coherent approximations of similar quality. GAADD required more time than SIMARR, however. Whereas only a few minutes were required for SIMARR to complete its work on all the data in a given experiment, GAADD required hours.[12] A different picture emerges when GAADD is applied to the simulated data described in Section 4.3. The results are shown in Table 2, which can be interpreted analogously to Table 1 for SIMARR. GAADD here shows levels of performance that are slightly superior to those for SIMARR and in less time. The difference in the two comparisons between GAADD and SIMARR is likely connected to the format of the judgments in the two data sets. For the simulated data, we relied on disjunctive normal form, easily exploited by routines in the CUDD software package. Thus, computing the probabilities of formulas is much easier relative to ADDs than relative to probability arrays. In contrast, the judgments in the experimental data had a simpler structure. From these contrasts we surmise that in the present context simulated annealing is a faster search algorithm than genetic algorithms but that ADDs are more efficient than arrays for coding complex judgments.

[12] Moreover, GAADD was implemented in C whereas SIMARR relied on the JAVA virtual machine.

6 Discussion

We have described two methods for finding coherent approximations to incoherent probability estimates by human judges. The coherent approximations were shown to be objectively more accurate than the judges themselves. Furthermore, our algorithms provide a means to aggregate the opinions of distinct judges, and evidence was presented that enhanced accuracy is the result. The scalability of our methods was also demonstrated using synthetic data. We are currently testing scalability on real data by collecting probability estimates about stock and oil markets involving 30 variables. Algorithms are also being designed to exploit the factorization of a distribution that results from conditional independence among subsets of variables (Castillo et al., 1997). Such factorization might be built into the structure of a domain or else revealed by expert judgment.

Let us conclude by underlining the potential use of our technique for finding consensual agreement among experts. Each expert can offer estimates of events (or conditional events) that only partially overlap the events assessed by others. Provided that every pair of experts makes judgments over some shared variables, each expert will indirectly interact with every other (through the constraints imposed by the probability axioms on the coherence of the pooled estimates). The aggregation procedure can be iterated by allowing the experts to review and discuss the output of the algorithm, possibly modifying their original assessments at the end of each cycle. In this sense, our technique is an extension of traditional approaches to group consensus (see Adler & Ziglio, 1996). The algorithm can also be adapted to allow judges to attach varying levels of confidence to their estimates, and to allow higher authorities to attach varying levels of confidence to judges (or even to specific judgments). It is worth noting that without any such modifications, our algorithm already respects pre-existing consensus in the following sense. Suppose that many judges assign nearly the same estimate to the chance of a specific event E. Then (under mild conditions) the algorithm will tend to change the estimate less than if the estimates for E were more varied.

Acknowledgements

The work of Osherson, Tsavachidis, and Vardi is partially supported by NSF grant IIS-9978135. We thank three anonymous reviewers for helpful comments on the first draft of this paper.

References

[1] M. Adler and E. Ziglio. *Gazing into the Oracle: The Delphi Method and its Application to Social Policy and Public Health.* Jessica Kingsley Publishers, Amsterdam, 1996.

[2] D. Aldous and U.V. Vazirani. "Go With the Winners" Algorithms. In *IEEE Symposium on Foundations of Computer Science*, pages 492–501, 1994.

[3] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Journal of Formal Methods in Systems Design*, 10(2/3):171–206, 1997.

[4] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, c-35(8), 1986.

[5] E. Castillo, J. Gutiérrez, and A. Hadi. *Expert systems and probabilistic network models.* Springer, New York, 1997.

[6] V. Chvátal. *Linear Programming.* W. H. Freeman, San Francisco CA, 1983.

[7] M. J. Druzdzel and L. C. van der Gaag. Elicitation of probabilities for belief networks: combining qualitative and quantitative information. In *Proc. 11th Conference on Uncertainty in Artificial Intelligence*, pages 141–148, Los Altos, CA, 1995. Morgan Kaufmann Publishers.

[8] R. Fagin, J. Halpern, and N. Megiddo. A Logic for Reasoning about Probabilities. *Information and Computation*, 87:78 – 128, 1990.

[9] G. Georgakopoulos, D. Kavvadias, and C. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4:1–11, 1988.

[10] M. Mitchell. *An introduction to genetic algorithms.* MIT Press, Cambridge MA, 1996.

[11] D. Osherson. Probability judgment. In E.E. Smith and D. Osherson, editors, *Invitation to Cognitive Science: Thinking (2nd Edition).* M.I.T. Press, Cambridge MA, 1995.

[12] A. Tversky and D. Kahneman. Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review*, 90:293–315, 1983.

[13] L. van der Gaag, S. Renooij, C. Witteman, B. Aleman, and B. Taal. How to elicit many probabilities. In K. B. Laskey and H. Prade, editors, *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, pages 647–654, Los Altos, CA, 1999. Morgan Kaufmann Publishers.

[14] P. van Laarhoven. *Theoretical and computational aspects of simulated annealing.* Center for Mathematics and Computer Science, Amsterdam, 1988.

[15] R. Viale and D. Osherson. The Diversity Principle and the Little Scientist Hypothesis. *Foundations of Science*, 5:239–253, 2000.

[16] D. von Winterfeldt and W. Edwards. *Decision analysis and behavioral research.* Cambridge University Press, New York NY, 1986.

[17] J. F. Yates. *Judgment and Decision Making.* Prentice Hall, Englewood Cliffs NJ, 1990.