RICE UNIVERSITY

Eliminating Incoherence from Subjective Estimates of Chance

by

Spiridon Tsavachidis

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Science

Approved, Thesis Committee:

Moshe Vardi, Professor, Chair
Computer Science

Devika Subramanian, Associate Professor
Computer Science

Daniel Osherson, Professor
Psychology

Houston, Tx
February, 2003

Abstract

Eliminating Incoherence from Subjective Estimates of Chance

by

Spiridon Tsavachidis

Human expertise is a significant source of information about environments with inherent uncertainty. However, it is well documented that subjective estimates of chance tend to violate the mathematical axioms of probability, that is, they are *incoherent*. This fact makes the use of such estimates problematic for statistical inference, decision analysis, economic modelling or aggregation of expert opinions. In order for the subjective probability estimates to be used in a correct and meaningful way, they must be reconstructed so that they are coherent. The proposed algorithms for coherent reconstruction are based on heuristic search methods , namely, Genetic Algorithms and Simulated Annealing. These algorithms are combined with efficient data structures that compactly represent probability distributions. The reconstructed estimates are coherent and close to the initial judgments with respect to some distance measure, maintaining the insight of the expert. Empirical studies shown that the coherent approximations are more stochastically accurate than the original subjective estimates.

Acknowledgements

I am very thankful to Moshe Vardi and Daniel Osherson for their guidance throughout my years in the department of Computer Science. I also thank Devika Subramanian for being a member of my thesis committee and for transmitting her enthusiasm about science.

Contents

List of Figures

List of Tables

Abbreviations

ADD	Algebraic Decision Diagram
GAADD	Genetic Algorithm over Algebraic Decision Diagrams
GAAR	Genetic Algorithm over probability Arrays
LINPOOL	Linear Pooling
LP	Linear Programming
MAD	Mean Absolute Deviation
MQD	Mean Quadratic Deviation
QP	Quadratic Programming
SIMADD	Simulated Annealing over Algebraic Decision Diagrams
SIMAR	Simulated Annealing over probability Arrays
SIMARAGG	SIMAR applied to aggregate data

Chapter 1

Introduction

Probabilistic knowledge has always been an important issue in Artificial Intelligence. The elegance of probability theory enabled it to survive and in many cases dominate over other theories of uncertainty. In this thesis, human judgment as a source of information about uncertain events is examined. Subjective estimates of chance, however, often conflict with the axioms of probability when logically related events must be evaluated. Estimates that are incompatible 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).

One way to ensure probabilistic consistency is to pair each query to a judge with a precalculated response set containing all coherent possibilities (Druzdzel and 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 can be achieved only if the judgments are coherent to begin with).

The principal challenge for such a method 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 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). Searching through the class of probability arrays via simulated annealing allowed us to efficiently approximate target sets of simulated judgments over 20 - 50 variables. The same method was applied to sets of 46 judgments over 10 variables in five empirical studies involving 181 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 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.

In place of probability arrays, we have also experimented with *algebraic decision diagrams*, which have proven useful in other problems involving numerous variables (Bahar et al., 1997). Similarly, we have explored genetic algorithms in place

of simulated annealing. We consider these alternative techniques in Section 3.3.2.

To proceed, the optimization problem posed by off line reconstruction of probability estimates is specified in Chapter 2. Details about the proposed optimization methods are presented in Chapter 3. Experimental results are described in Chapter 4. The parameter space of the proposed algorithms is explored in Chapter 5.

1.1 Related work

It is often the case in expert systems that both the knowledge as well as the entailment rules are known with some uncertainty. For a typical expert system, facts and rules are stated as logical sentences, thus it has been of interest to generalize logic to engulf uncertainty.

There have been proposals that diverge from classical probability theory, examples being the Dempster-Shafer theory (Shafer, 1976), fuzzy set theory (Zadeh, 1971), or certainty/confidence factor (Pearl, 1986). For a survey of the theories see (Genesereth and Nilsson, 1987).

The first comprehensive attempt to combine logic and probability was Lukasiewicz's original work on probability logic (Lukasiewicz, 1913). His exploration of the logical foundations of probability produced original ideas that anticipated much of the work that followed many years later, (Halpern, 1990). Another important moment in the history of probabilistic logic was Nilsson's publication of his logic for probabilistic reasoning in his paper "Probabilistic Logic" (Nilsson, 1986). In contrast with Lukasiewicz's work, Nillson's approach has become integrated into AI as a logic for probabilistic reasoning, see (Torsun, 1995). In his 1986 paper, Nilsson takes a new approach to the semantics of probability statements. Probabilities are assigned to sentences and a probability evaluation is made by a reference to *possible worlds*. In particular, the probability of a sentence is given by summing the probabilities assigned to all possible worlds (truth assignments of the variables) in which the sentence if true.

Nilsson addressed the main problem of probabilistic satisfiability (PSAT), that is, the problem of checking for the consistency of the logical sentences and their assigned probabilities. He showed that PSAT can be expressed as a linear program with a number of columns exponential to the number of variables (facts).

Computational complexity issues were explored in (Georgakopoulos et al., 1988) and (Kavvadias and Papadimitriou, 1990). NP-completeness of PSAT is shown even when all clauses contain at most two literals (2PSAT). They propose solving PSAT by column generation and they prove that the selection of the entering column can be done by solving an appropriate weighted maximum satisfiability (MAXSAT) problem. Since MAXSAT is also NP-complete, they propose the use of heuristics to solve it.

Extensions of Nilsson's work as well as further exploration of the ideas of (Kavvadias and Papadimitriou, 1990) can be found in (Jaumard et al., 1991). In particular, the authors extend Nilsson's model allowing the assignment of intervals of probability values to the logical sentences. The second contribution is the description of a column generation algorithm for solving PSAT for fixed or interval probability values. Probabilistic satisfiability is referred to as *generalized coherence* (g-coherence) in the literature when intervals of probability values are involved (imprecise probability assessments) instead of probability values (precise probability assessments), see (Biazzo et al., 2001). In this paper, it is shown that coherence is equivalent to avoiding uniform loss when probabilities are interpreted as random gain rates for a set of bets between a bookmaker and a gambler. Thus, coherence prohibits the possibility of sure loss for both the gambler and the bookmaker. Similarly, g-coherence is equivalent to avoiding sure loss for both the upper and lower betting rates (corresponding to lower and upper probabilities). A thorough exploration of the notions of coherence and avoiding uniform loss can be found in (Shafer et al., 2002) and (Walley, 1997).

Chapter 2

Judgment reconstruction as optimization

2.1 Formulation of the problem

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, the variables yield 2^n mappings (called *truth assignments*) from $\{v_1 \cdots v_n\}$ to $\{true, false\}$. A truth assignment α *satisfies* a formula φ if φ evaluates to *true* 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 $Est(\varphi) = x$ to indicate the judgment that the probability of φ is x, and $Est(\varphi : \psi) = y$ for the judgment that the conditional probability of φ assuming ψ is y. *Est* is thus a finite function from formulas and pairs of formulas to real numbers in $[0, 1]$. If it coincides on its domain with some distribution Pr over $v_1 \cdots v_n$ then *Est* is called *coherent,* otherwise *incoherent.* In the typical case, *Est* is incoherent, and we seek to reconstruct it via a close distribution. The resulting optimization problem can be stated as follows.

(1) Optimization Problem: Let *Est* map formulas $\varphi_1 \cdots \varphi_k$, and pairs of formulas $(\chi_1, \psi_1) \cdots (\chi_j, \psi_j)$, into numbers. Find a map *Prob* with the same domain as *Est* such that *Prob* is coherent, and

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

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

is minimized.

We examine (1) for $p = 1$ and $p = 2$, corresponding to absolute and quadratic deviation from *Est* . In either case, it is evident that a solution to (1) leads to a solution of PSAT, an NP-complete problem (Georgakopoulos et al., 1988) implying that the existence of an efficient algorithm is highly unlikely. [1]

[1] In addition to the problem of intractability is the possibility that there may not even be a minimum distance between *Est* and a coherent approximation *Prob* to it. The incoherent judgments

Note that distinct approximations *Prob* can yield the same deviation from *Est* 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 $Est(p) = .3$, $Est(\neg p) = .6$ are equally well reconstructed with respect to their absolute deviations from the given estimates 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 (1), for example, maximizing entropy, see (Lukasiewicz and Isberner, 1999) for entropy considerations in probabilistic logic. For simplicity in the current investigation, only deviation from *Est* appears in (1). Absolute and conditional events, moreover, are given equal weight in calculating deviation from *Est*, and only point estimates of probability are considered. Obviously, such matters can be settled differently within specific applications.

2.2 Sparse distributions

We rely the following fact (also observed by (Fagin et al., 1990), and exploited in a similar way).

(2) 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 $\mathsf{Pr'}$ such that:

(a) $\mathsf{Pr'}$ 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: Let k formulas and j pairs of formulas be given as in (2). 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.

$Est(p : q) = .5$, $Est(q) = 0$, for example, can be approximated by setting $Prob(p : q) = .5$ and $Prob(q)$ arbitrarily close to 0, but not 0 itself. We deal with this problem by requiring that the probability of conditioning events is greater then a small constant

Hence, Pr maps the 2^n truth assignments $\tau_1, \tau_2, ...\tau_{2^n}$ into non-negative real numbers $x_1, x_2, ..., x_{2^n}$ such that $\sum_{i=1}^{2^n} x_i = 1$. Let

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

It then follows that:

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

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

$$\begin{aligned} \textstyle\sum_{\ell \in S_i} x_\ell &= \mathsf{Pr}(\varphi_i) \quad \text{for } i \leq k, \\ \textstyle\sum_{\ell \in Y_i \cap Z_i} x_\ell - \mathsf{Pr}(\chi_i : \psi_i) \cdot \sum_{\ell \in Z_i} x_\ell &= 0 \quad \text{for } i \leq j, \\ \textstyle\sum_{i=1}^{2^n} x_i &= 1. \end{aligned}$$

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. ■

This observation allows the use of search algorithms for solving problem (1) restricted only on sparse distributions. This fact is exploited by using efficient data structures such as probability arrays and algebraic decision diagrams described in

Chapter 3.

2.3 Special cases

In its general form, (1) is a nonconvex nonlinear optimization problem with 2^n continuous variables, where n is the number of boolean variables. Special cases of the problem can be solved by applying linear and quadratic programming techniques. These cases involve only absolute judgments. Depending on the choice of absolute or squared deviation this special family of problems can be solved using linear and quadratic programming (LP, QP), respectively. LP and QP provide useful benchmarks of success in simple settings but it does not embody a general solution to (1). For one thing, LP and QP are impractical for more than 20 variables because their running times are polynomial in the number of continuous variables which is equal to 2^n. It is noted that LP can become efficient by using a column generation technique, however, using the standard simplex algorithm is inefficient. More importantly, LP and QP cannot be applied to judgments involving ratios of probabilities, notably, when estimates are given for conditional events.

2.3.1 Linear Programming

If we are interested in minimizing absolute deviation ($p = 1$) and only absolute judgments are involved, (1) can be solved by linear programming (LP). 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. A final constraint sets the sum of all the x_α's to unity. Note that this LP formulation of (1) 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.This formulation of the linear programming problem is often used when the objective is to minimize the sum of the absolute values of weighted sums (Nemhauser et al., 1989). 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), see (Rustagi, 1994). [2] An example of the linear programming approach is illustrated in the following example.

Suppose there are two events, represented by the boolean variables p and q. The following estimates are given: $Est(p) = .75$, $Est(q) = .5$, $Est(p \wedge q) = .1$.
The corresponding linear program is shown in Table 2.1

	States												
Judgment	pq		$p\bar{q}$		$\bar{p}q$		$\bar{p}\bar{q}$		Slack				Target
$Est(p)$	x_1	$+$	x_2					$+$	y_p	$\geq$			.75
	x_1	$+$	x_2					$-$	y_p	$\leq$			.75
$Est(q)$	x_1			$+$	x_3			$+$	y_q	$\geq$			.5
	x_1			$+$	x_3			$-$	y_q	$\leq$			.5
$Est(p \wedge q)$	x_1							$+$	$y_{p\wedge q}$	$\geq$			.1
	x_1							$-$	$y_{p\wedge q}$	$\leq$			.1
	x_1	$+$	x_2	$+$	x_3	$+$	x_4			$=$			1.0
						x_i	$\geq$	0		$i = 1, .., 4$			

Table 2.1: Example of a linear program for reconstructing probability estimates. The optimal solution is found by minimizing the sum of the slack variables given the above constraints

Column generation techniques The inefficiency of the LP formulation of the problem has been addressed to some extent by using column generation methods, see (Chvátal, 1983). It was suggested in (Georgakopoulos et al., 1988) the use of column generation methods for solving instances of PSAT. The problem of coherent rectification by minimizing the absolute deviations as shown in (1) is referred to as *Restore Satisfiability* (RSAT) in (Jaumard et al., 1991). The authors proposed a

[2]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.

combination of heuristic and exact algorithms for finding an entering column at each iteration of the revised simplex method, see (Nemhauser et al., 1989) and (Chvátal, 1983). They showed that RSAT instances with up to 70 variables and 100 constraints can be solved efficiently with their techniques. It is evident that column generation can scale well with the number of variables contrary to the standard LP formulation. It is noted, however, that these methods are applicable only to absolute events.

2.3.2 Quadratic Programming

If the square deviation is minimized ($p = 2$) then quadratic programming (QP) can be applied to reconstruct optimally the given absolute judgments. The cost function can be written as $\sum_{i=1}^{k}(p_i - \sum_{j \in A_i} x_j)^2$, where $A_i = \{a_{i1}, ..a_{ij_i}\}$ is the set of indices that correspond to the j_i truth assignments that render the i^{th} formula true. The constraints for the variables $x_1, ...x_{2^n}$ are $x_i \geq 0$ and $\sum_i x_i = 1$. The cost function can be trivially rewritten as a sum of quadratic, linear, and constant terms. Its convexity is determined solely by the quadratic terms $\sum_{i=1}^{k}(\sum_{j \in A_i} x_j)^2$. It is known that a quadratic form is convex iff it is greater than or equal to zero for all real values of its variables, see (Skutella, 2001). This is obviously the case for the above sum of squared terms, thus the cost function is convex. It is also trivial to see that the constraints define a convex set. Thus, the optimization problem is convex and it can be solved in polynomial time in the number of continuous variables $x_1, ...x_{2^n}$, see (Vavasis, 1993).

To illustrate the difference of the QP approach to the LP approach, an example is shown in Table 2.2. The event variables p and q and the probability estimates are the ones used in Table 2.1

	$Est(p)$		$Est(q)$		$Est(p \wedge q)$
$Minimize:$	$(x_1 + x_2 - .75)^2$	$+$	$(x_1 + x_3 - .5)^2$	$+$	$(x_1 - .1)^2$
	$x_1 + x_2 + x_3 + x_4 = 1.0$		$x_i \geq 0$	$i = 1, .., 4$	

Table 2.2: Example of a convex quadratic program for reconstructing probability estimates

Chapter 3

Optimization methods

The intractability of the optimization problem (1) leads us to a strategy involving data structures that efficiently represent probability distributions in conjunction with heuristic search methods, namely, simulated annealing and genetic algorithms.

The first approach is to represent candidate distributions using probability arrays, and to search through them via simulated annealing. An alternative approach it to employ genetic algorithms for searching and Algebraic Decision Diagrams to represent efficiently probability distributions.

3.1 Data structures

3.1.1 Probability Arrays

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 (1)a, below. The first three rows of (1)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

(1) (*a*)

1	0	∗	0
0	∗	∗	0
1	0	1	0
.2	.1	.4	.3

(*b*)

1	1	1	1
0	0	0	0
1	1	1	1
.2	.1	.4	.3

(*c*)

∗	∗	∗	∗
∗	∗	∗	∗
∗	∗	∗	∗
.2	.1	.4	.3

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 (1)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 .3/4. The sole truth-assignment 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 (1)a, the truth-assignment v_1 = true, v_2 = false, v_3 = true is represented in columns 1 and 3. Its probability according to (1)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 (1)a, the probability of v_1 = true, v_2 = true, v_3 = false is 0. As an aid to intuition, the extreme arrays (1)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:

(2) 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).

From (2) it follows that our search for an approximating distribution [in the sense of (1)] can be limited to sparse distributions. Hence (2) implies that 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.

3.1.2 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 labelled either 0 or 1, and non-terminal nodes with out-degree two labelled 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 3.1. An ADD maps each truth assignment to one of its terminal nodes in the obvious way. For example, the ADD of Figure 3.1 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 3.1 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. To manipulate ADDs, we rely on a subset of operations defined in the software package CUDD[1] from Colorado University.

The function *Matrix_Multiply* computes the product of two ADDs with re-

[1]The package is available via `http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html`

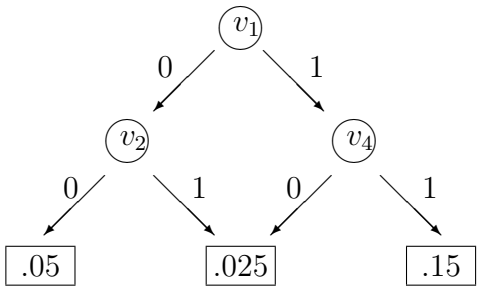

Figure 3.1: ADD representation of a probability distribution

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

spect to a set of summation variables. That is, each ADD is seen as the representation of a rectangular matrix whose rows and columns are encoded by two subsets X_1 and X_2 of the variables. It is noted that X_1 and X_2 constitute a partitioning of the set of variables. In order to calculate the matrix product of two ADDs the variables that encode the columns of the first must coincide with the ones encoding the rows of the second. These are the summation variables. If the set of the summation variables contains all the variables, *Matrix_Multiply* computes the inner product of the vectors that correspond to the two ADDs.

Inner product is used as follows to calculate the probability of a given formula φ according to 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. Other important functions of the CUDD package include the if-then-else operator *ITE(f,g,h)=fg+f'h* and the *Apply(g,h,op) = g op h* that implements arithmetic operators on ADDs, where g, h are ADDs and f is a BDD (kept as a 0-1 ADD in the CUDD package). These functions will be used in the implementation of the genetic operators of the Genetic Algorithm. The time complexity of the preceding operators is linear in the product of the number of nodes of the operands. Thus, ADD manipulation is efficient as long as they do not grow

too large. For more information concerning ADD manipulation and time complexity issues see (Bahar et al., 1997).

3.2 Algorithmic schemes

3.2.1 Simulated Annealing

Simulated annealing is a general heuristic search method that has been applied successfully to several optimization problems (Davis, 1987). It is inspired by the physical process of *annealing* . If the annealing process is carried out slowly enough, the *ground state* of a solid can be found, that is, the molecules are arranged in a minimum energy configuration. This phenomenon motivated people to devise the generic method of simulated annealing. The goal is to find a configuration that minimizes a given cost function C. First, a initial solution to the problem s_0 is found and the initial value of the control parameter T is defined. The cost function C and control parameter T are analogous to the energy and temperature of the annealing process. At the i^{th} step of the algorithm, a configuration r is randomly selected from the set of 'neighbors' of the current configuration s_i. The new configuration s_{i+1} is set to r if $C(r) \leq C(s_i)$, otherwise it is set equal to r with probability $e^{-\frac{C(r)-C(s_i)}{T_i}}$. The sequence $T_0, ...T_i...$ depends on the initial value of the control parameter T, and the *cooling schedule*. The presented algorithm uses the decrement rule $T_{i+1} = \alpha T_i$ for some constant $\alpha < 1$. At each temperature step, the algorithm iterates several times in order to let the Markov chain represented by the algorithm to converge to its limit distribution, see (van Laarhoven, 1988) for the mathematical analysis of Simulated Annealing.

3.2.2 Genetic Algorithms

Genetic Algorithms are a type of stochastic search technique inspired by evolution. They rely on the encoding of possible solutions of a problem to data structures called *chromosomes* and their transformation to new improved solutions via *genetic operators*. The application of a genetic algorithm to a problem requires the following components.

- A chromosomal representation of solutions to the problem.
- A way to generate a population of initial solutions.
- An evaluation function that rates the chromosomes in terms of their *fitness*.
- Genetic operators that recombine the chromosomes to yield the next generation of the population.

The representation of the chromosomes is typically bit strings but other data structures such as graphs can be appropriate for certain problems (Mitchell, 1996). The initial population is usually randomly generated , taking into account possible constraints of the problem. The genetic operators typically used in genetic algorithms are *crossover* and *mutation*. Crossover is considered the key to GA's power. It combines two chromosomes to produce their offspring. Typically, chromosomes with high fitness tend to produce highly fit children. If bit strings are used as chromosomes, than the *one-point* crossover operator works as follows. Let c_1, c_2 be two chromosomes. Then we randomly partition the two strings to $c_1 = s_1 s_2$, $c_2 = s_1^{'} s_2^{'}$. The results of crossover are the two offspring $c_1^{'} = s_1 s_2^{'}$ and $c_2^{'} = s_1^{'} s_2$. The mutation operator introduces random variations to a chromosomes. In the case of bit strings, mutation simply involves random flipping of randomly chosen bits. Occasionally, these variations are beneficial to the gene pool. For a thorough presentation of the key notions of GAs see (Mitchell, 1996).

3.3 The reconstruction algorithms

3.3.1 Simulated Annealing over Probability Arrays

The SIMAR algorithm uses simulated annealing to search over probability arrays and finds one that approximates well the input estimates. It starts by generating randomly a probability array (n, m), where n is the number of variables and $m = N_{columns}$, an algorithm parameter analyzed later. The entries of its first n rows are randomly filled with 1, 0 and $*$ (don't care). The probability of setting an entry to $*$ is equal to P_{star}. The last row of the $(n, N_{columns})$ array contains uniformly distributed random floating-point numbers (generated by the function *random*() from the Java package Math) , normalized to sum to 1. At each step of the algorithm we have to create 'neighbors' to an array A. The method we used is the following

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

The parameter ϵ controls the relative change of the entries of the last columns of A. In order to determine the initial 'temperature' we define the parameter *initial acceptance rate*, I_{ar}, that is, the average percentage of the accepted configurations versus the total number of configurations generated from a run of the simulated annealing algorithm for a fixed number of steps and constant temperature. The initial temperature is found statistically by the algorithm. A description of the exact methodology for statistically defining the initial temperature from the initial acceptance rate can be found in (van Laarhoven, 1988). The parameters associated with the running time are N_T, the number of temperature decreasing steps, and N_{iter} ,

the number of iterations at each temperature value. The cooling schedule is defined by the equation $T_{i+1} = \alpha T_i$ where T_i is the temperature at the i^{th} temperature step and α is a constant defined so that the initial temperature is 1000 times bigger than the final temperature. Thus, α is a function of the number of temperature steps, N_T. It is easy to see that it is given by the equation $\alpha = e^{-ln(1000)/N_T}$. The computation of the cost function given by (1) requires the calculation of probabilities of events given by formulas with respect to a given probability array. This is straightforward for events given by simple formulas, as is the case with the experimental analysis presented in the present paper. Specifically, the calculation of the probability of an event associated with a formula is done by counting the number of satisfying assignments of the formula for each column, multiplying with the weight of the column, and finally, summing over all columns. The presented algorithm can run with multiple starting points, manipulated according to the 'go with the winners' strategy described in (Aldous and Vazirani, 1994). An analysis of the parameter space of the algorithm is presented in section 5.1.

3.3.2 Genetic Algorithms and ADDs

To find a coherent approximation to input judgments, we searched through the space of ADDs using genetic algorithms (GAADD). We now provide details about the genetic algorithm used to produce the results presented in the sequel. We describe (a) the starting population, (b) the processes of crossover and mutation, and (c) the construction of the next generation. The values of the parameters of the algorithm cited below were determined by the experimental analysis described in section 5.2. The same parameters are used in all the experiments reported later.

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 N_{prod} ternary sequences of length n is randomly generated. The parameter N_{prod} is set equal to 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. 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. The random generation of an ADD can be also seen as a conversion of a probability array to an equivalent ADD. 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 order to minimize the storage requirements of the entire set of ADDs. (The same variable ordering governs all the ADDs in the population.)

Genetic operators Crossover between two ADDs A_1, A_2 proceeds as follows. Each truth assignment determines a base-2 numeral 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. This procedure is described by the following CUDD commands. $A_1^{'} = ITE(f, A_1, A_2)$, $A_2^{'} = ITE(f, A_2, A_1)$, where f is the 1-0 ADD representing the *s*plitting function $f(v_1, \ldots, v_n) = \begin{cases} 1 & \text{if } r < \overline{v_1 \cdots v_n} \\ 0 & \text{otherwise} \end{cases}$, where r is a randomly generated n-digit binary number and $\overline{v_1 \cdots v_n}$ is a binary number with digits determined by $v_1 \ldots v_n$.

Mutation of a given ADD starts from another choice of a number between

0 and $2^n - 1$. The terminal nodes of all paths in the ADD that correspond to a truth assignment below (or above) the chosen number are multiplied by $1 + e_m$. The remaining terminals are multiplied by $1 - e_m$. The ADD is then renormalized. The parameter e_m is set equal to 0.1. Mutation of an ADD A is described by the following commands. $A_1 = \mathit{Apply}(A, 1 + pe_m, \times)$,$A_2 = \mathit{Apply}(A, 1 - pe_m, \times)$, and $A = ITE(f, A_1, A_2)$. The number p is randomly chosen from the set $\{-1, 1\}$ and f is the splitting function described above. The mutation operation splits the 2^n probability vector represented by an ADD into 2 vectors using the function f. It then proceeds by multiplying the elements of one vector by $1 \pm e_m$ and the other by $1 \mp e_m$. Renormalization is finally performed to make the probabilities sum to 1.

Construction of successive generations Let *Est* 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 (or quadratic) deviation between *Est* and *Prob* in the sense of (1)[2] . We take the *fitness* of A to be $1/(dev + .01)$. Thus, greater match to *Est* 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 $P_c = .95$ the pair undergoes crossover. Whether crossed or not, the pair then undergoes mutation with probability $P_m = .04$. The two chromosomes then enter the next generation (again yielding a population of N). Optimal variable ordering is once again sought for the entire population.

Across generations, the ADD with highest fitness (lowest *dev*) is retained, and its distribution is used to approximate *Est*. The impact of the total number of generations *Gennum* as well as the size of the population *Popul* is analyzed in section

[2]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 $\mathit{Prob}(\psi) = 0$). In this case (which is extremely rare), we augment *dev* to make the proliferation of such an ADD highly unlikely .

5.2.

3.3.3 Methods that did not perform well

The proposed algorithms, SIMAR and GAADD, are the amalgamation of efficient data structures and the generic search methods of simulated annealing and genetic algorithms. It is evident that there are two more possibilities that require investigation; genetic algorithms applied to probability arrays and simulated annealing applied to ADDs. We call these algorithms GAAR and SIMADD.

Genetic algorithms and probability arrays The GAAR algorithm starts by generating an initial population of probability arrays. Each member of the population is randomly generated as described in section 3.3. The mutation operation is the same as finding a neighbor, see section 3.3. Crossover of two (m, n) probability arrays A_1 and A_2 is performed as follows: A uniformly distributed random integer k between 1 and m is generated. Their offspring are two new probability arrays B_1 and B_2. The first k columns of B_1 are copied from the corresponding columns of A_1 and the rest from A_2. The array B_2 has the k first columns of A_2 and the last $m - k$ columns of A_1. Normalization is required after the application of the genetic operators.

This algorithm proved to be considerably slower than SIMAN since it involves computations over a population of probability arrays. The crossover operation requires extensive memory copying , thus increases further the running times. Typically, the running times of GAAR are 10 times bigger than those of SIMAR. It is also noted that for several parameter value configurations we experimented with, the accuracy of the resulting coherent approximations was significantly worse than the accuracy achieved with SIMAR or GAADD.

Simulated annealing and ADDs Simulated annealing over ADDs uses an ADD as a starting point. At each step, it transforms the ADD to create a "neighbor".

The initial ADD is generated using the same procedure used to create a population of ADDs for GAADD, see 3.3.2. In order to find a neighbor of the ADD we used the mutation operator of GAADD, see 3.3.2. This implementation of SIMADD had several problems. Failure of convergence in several cases was the most important. Problems also arose from the repeated creation of neighbors. Even though the algorithm deals with a single ADD, after a few steps the ADD grew very large to handle efficiently. It is not clear how one should alter the ADD at each step so that its size does not grow excessively. In the cases where the algorithm did converge, its accuracy was very poor compared to SIMAR or GAADD.

Chapter 4

Tests of the methods

4.1 Scalability studies

As a test of the computational feasibility of our methods in large problems, we created sets of simulated estimates of chance and tried to rectify 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 $k < .5$. (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.

Twelve tests were carried out, involving distributions with 20, 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 $(0 \leq i \leq 2^n - 1)$ 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 (DNF). 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 20, 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. Specifically, for a conditional event, two formulas in DNF p, q are constructed as described above. Then,the DNF formula $p \wedge q$ is constructed. The conditional probability $P(p|q)$ is calculated by the formula. The only constraint here is that the formula of $p \wedge q$ is composed of at most 5 conjunctions. This is done because the calculation of probabilities of DNF formulas in SIMAR is time consuming for big formulas. It is noted however, that GAADD does not have this problem due to the efficiency of ADDs to represent and manipulate such formulas. $P(p|q) = P(p \wedge q)/P(q)$.

For each of the 12 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 annealing over arrays of size $(n, 100)$, with one starting point, 150 temperature steps, 25 iterations per step, and probability of * set to .6,

Results are shown in Table 4.1. 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 devi-

ation 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. Table 4.2 shows the results of GAADD applied to simulated data. It is evident that GAADD achieves accuracy similar to SIMAR with less running time. This is partially due to the very efficient manipulation of the DNF formulas by ADDs. SIMAR suffered from the probability calculations of 300 formulas (100 for the absolute and 200 for the conditional events) at each step. This is not the case for the experimental data since the formulas involved in these sets are very simple and SIMAR vastly outperforms GAADD in terms of running time. Another important reason is that while GAADD is written is C, SIMAR is written exclusively in Java, thus making true comparisons difficult. These results provide evidence for the scalability of the two approaches to reconstructing incoherent estimates of chance.

vars.	20		30		40		50	
Inc.	MAD	Min	MAD	Min	MAD	Min	MAD	Min
0	.0045	100	.0063	132	.0054	154	.0062	140
.1	.0769	64	.0638	135	.0638	140	.0680	144
.2	.1478	60	.1428	85	.1472	104	.1454	215

Table 4.1: Mean absolute deviation (MAD) and time in minutes (Min) achieved using SIMAR on simulated data generated from sparse distributions with $20,000$ positive states. Times are relative to a Pentium III (533 MHz) processor running the Java Virtual Machine for Windows '98.

vars.	20		30		40		50	
Inc.	MAD	Min	MAD	Min	MAD	Min	MAD	Min
0	.0024	20	.0037	56	.0029	106	.0033	127
.1	.0858	18	.0832	43	.0826	90	.0884	139
.2	.1662	11	.1687	26	.1699	46	.1711	75

Table 4.2: Mean absolute deviation (MAD) and time in minutes (Min) achieved using GAADD 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.

4.2 Empirical Studies

Consider the following event.

(1) 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 event (1) 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.[1] The 36

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

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."[2]

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

We performed four other experiments of identical design. One experiment with 38 participants concerned weather prediction and variables like:

[2]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.

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 with 29 participants 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.

Another experiment also involved the NBA, this time a game between the Rockets and the Dallas Mavericks.[3]

The final experiment, involves predictions for the fourth quarter of 2001. There are 30 variables in total, but each of the participants was asked to estimate probabilities of 46 simple events on 10 randomly chosen variables. The 30 variables included:

- The NASDAQ Composite Index increases.
- Amazon.com's stock price increases.

Let us call the five experiments stocks, weather, Suns, Mavericks, and finance respectively. There were 38 participants in weather, 29 in Suns, 36 in Mavericks, and 47 in finance . All were Rice undergraduates with the exception of finance in which 26 participants were undergraduates and the rest were graduate students. Except for some overlap between Suns and Mavericks, the sets of participants in the four experiments were disjoint.

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

4.2.1 Incoherence

Not one of the participants of the five experiments offered a coherent set of estimates. The average number of violations (out of 6 possible) of the following constraint on the probabilities of conjunctions of form (c) are shown in the first row of Table 1.

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

Similarly, the average number of violations of the following constraint on disjunctions of form (e) are shown in the second row of Table 1.

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

Simple constraints corresponding to (a), (b), (d), and (f) yielded comparable rates of violation.

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
(6)	3.0	2.92	4.21	3.64	3.0
(7)	3.23	2.16	3.86	2.67	3.13

Table 4.3: Average number of violations of (6) and (7) for the five data sets

4.2.2 Approximation via SIMAR and GAADD

We applied our algorithms to each participant separately, represented by his/her data set of 46 judgments. For both algorithms, we used the best parameter values we found, see Appendices I and II. It is noted that SIMAR required 15-20 seconds per dataset , while GAADD requires 40-60 minutes for the same task. Table 4.4 shows the results for SIMAR and GAADD. The first and last two rows of the table show the achieved average MAD and MQD respectively.

The difference in speed is due to the overhead of operating on a population of ADDs in GAADD. However, GAADD achieves smaller running times when complex

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
SIMAR/MAD	.095	.085	.115	.092	.088
GAADD/MAD	.102	.090	.124	.095	.092
SIMAR/MQD	.021	.019	.025	.020	.019
GAADD/MQD	.023	.021	.029	.023	.022

Table 4.4: Average MAD and MQD for the 5 data sets achieved by SIMAR and GAADD

judgments on many variables are considered, see Table 4.2. From Table 4.4, it is evident that SIMAR outperforms GAADD in the sense that it achieves lower deviations from the input judgments.

4.2.3 Comparison to linear and quadratic programming

Perfect approximation (MAD or MQD equal to 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 or quadratic programming (LP/QP) as discussed in Section 2 above. In particular, LP is used for minimizing the MAD, while QP is used to find the optimum MQD. We applied LP and QP to the individual data of each participant in all five data sets. Then, SIMAR and GAADD were applied only on the absolute judgments for each data set. The first two rows of Table 4.5 compare the optimum MAD values found via LP with the results achieved by SIMAR and GAADD applied on absolute judgments only. The last two rows show how well SIMAR and GAADD performed on minimizing the MQD. Here, the benchmark is the optimum MQD values found by QP. In both tables only relative differences are shown. Again, SIMAR outperforms GAADD and comes as close as 106.5% to the optimum.

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
SIMAR/LP	10.5%	11.5%	6.5%	9.2%	8.9%
GAADD/LP	12%	13.5%	8%	10%	10.5%
SIMAR/QP	11%	7%	7%	12.5%	11%
GAADD/QP	12%	8.5%	9%	13%	14.5%

Table 4.5: Performance of SIMAR and GAADD relative to LP and QP

4.2.4 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. Two measures of accuracy are considered. Brier (quadratic) score and slope.

Quadratic score as a measure of accuracy The accuracy in a judgment can be measured via its quadratic score, defined as follows (see (von Winterfeldt and Edwards, 1986) for general discussion of scoring rules). It is noted that the results of this section were based on SIMAR minimizing absolute and quadratic deviation. Suppose that *Est* represents the estimates of a given judge. Let E be an event in the domain of *Est*, and let $(G : F)$ be a pair of events in the domain of *Est*.

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

The *overall quadratic score* of *Est* is the average of all the scores incurred by *Est* 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. The following analysis refers to the stocks experiment. The results for all five data sets are summarized in Table 4.6. After the facts were established about the third quarter performance of the 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 achieved by our algorithm minimizing the MAD. (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 was 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). When MQD is minimized by our algorithm, the resulting coherent approximations have a quadratic score of .226 (S.D. = .055), which again is reliable lower than the original scores. Improvement in quadratic score was achieved for 24 of the 31 participants.

Overall, the participants of the stocks experiment 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. Similar results were found by analyzing the other four data sets.

An interesting result concerning absolute events is the following theorem due to de Finetti, (de Finetti, 1972).

(4) For every incoherent set I of probability estimates over absolute events (no conditional probabilities) there is a coherent set C of estimates over the same

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
Est.	.254	.232	.237	.228	.309
Model/MAD	.232	.201	.203	.200	.285
Model/MQD	.226	206	196	.198	.284

Table 4.6: Quadratic score comparison. Original estimates (Est.) vs. coherent approximations via minization of MAD (Model/MAD) and MQD (Model/MQD)

events such that the quadratic penalty for C is lower than the penalty for I no matter which state is true.

In other words, de Finetti showed that incoherent estimates over a set of *absolute* events can always be replaced by coherent estimates that have lower penalties in every state. The following example clarifies the meaning of the theorem. Consider the incoherent estimates in (a), below, and the coherent replacements (b).

(a) $Est(p \wedge q) = .5, \quad Est(p : q) = .9, \quad Est(q) = .75$

(b) $Est(p \wedge q) = .564, \quad Est(p : q) = .802, Est(q) = .703$

Since only two variables occur in these events, four states represent all possibilities. We see from the following calculations that the quadratic penalty for the corpus (b) is lower than the penalty for (a) in all cases. (The last line in each table gives the two quadratic penalties.)

p	true
q	true
(a)	(b)
108	.106

p	true
q	false
(a)	(b)
271	.270

p	false
q	true
(a)	(b)
374	.350

p	false
q	false
(a)	(b)
271	.270

Theorem (4) can be extended to some sets of estimates involving conditional events; see, for example, (Bernardo and Smith, 1994) . The theorem cannot be extended, however, to *every* corpus of estimates because of examples like the following.

(5)

$Est(r : p \wedge q) = .8,$	$Est(p) = .5,$	$Est(p \wedge \neg q) = .5$

The second and third estimates in (5) imply that the conditioning event $p \wedge q$ has probability zero. So no distribution that satisfies the last two equalities assigns any probability to the conditional event at the left. Hence, the three estimates are incoherent. A simple argument shows that no coherent estimates lower the quadratic score in all states. The example thus demonstrates that no method of coherent approximation uniformly lowers the quadratic score of judgments involving arbitrary conditional events.

Returning to absolute events, de Finetti's (1974) proof of Theorem (4) describes a method for constructing coherent approximations for incoherent estimates of absolute events that lowers the quadratic penalty in all states. It turns out that the method of de Finetti computes a set of coherent probabilities whose Euclidean distance from the original estimates is minimum. Thus, the problem is equivalent to problem (1) when quadratic deviation is considered ($p = 1$). Since we are dealing only with absolute judgments, quadratic programming can be used, as shown in subsection 2.3.2.

Note that even for absolute events, quadratic programming is not guaranteed to deliver lower quadratic penalties than a rival method like SIMAR. Theorem (4) only guarantees lower quadratic penalties than the incoherent target.

Slope as a measure of accuracy Accurate judgment corresponds to low quadratic penalties. A contrasting measure of stochastic accuracy (for which accuracy is reflected in high scores) is also employed. Following the discussion in (Yates, 1990, Ch. 3), we define the *slope* of a corpus of judgments to be the average probability assigned to events that come true minus the average probability assigned to events that do not come true. In this definition, a conditional event $(A : B)$ is considered to be undefined if B does not occur. Otherwise, it is considered to occur if and only if A occurs. The average slope for the 31 participants of the stocks experiment was .129

(S.D. = .176) whereas the average slope for the 31 coherent approximations was .180 (S.D. = .156), when SIMAR is set to minimize MAD. Similar results were obtained when MQD was minimized. Once again, this difference is reliable ($p < .001$, by correlated t test). The slope for 26 of the 31 participants improved in the transition from raw to reconstructed estimates. As in the case of using quadratic penalty as a measure of accuracy, the lack of insight of the participants into the stock market is evident. However, coherent reconstruction produced a reliable increase in accuracy. Similar results were produced for the other four experiments. The results are summarized in Table 4.7. It is noted that the improvements of the slope in all cases are reliable ($p < .001$, by correlated t test). It is worth noting that there is not a theorem for slope equivalent to Theorem (4). That is, there are cases where uniform improvement of slope for all possible outcomes is not possible.

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
Est.	.129	.154	.138	.133	.053
Model/MAD	.180	.199	.224	.216	.115
Model/MQD	.177	.197	.226	.224	.106

Table 4.7: Slope comparison. Original estimates (Est.) vs. coherent approximations via minization of MAD (Model/MAD) and MQD (Model/MQD)

4.2.5 Selective reconstruction of judgments and accuracy

This section explores the accuracy of the original and reconstructed judgments with respect to their types for the 5 data sets Suns, Mavericks, stocks, weather, and finance. In particular, we wanted to examine the accuracy of the probabilities assigned to simple events (represented by single variables) versus the accuracy of the probabilities assigned to complex events (conditional events, conjunctions and disjunctions of variables). We performed two experiments. First, SIMAR was applied only on the 36 complex events per participant (46 total minus 10 estimates for the basic variables). The algorithm constructed a probability distribution that ap-

proximated well only the complex events, and this probability distribution produced probabilities for the simple events. The accuracy of these probabilities was examined (10 per data set, Mod-simple-complex in Table 4.8). The corresponding quadratic scores of the original estimates (Sub-simple in Table 4.8), as well as the quadratic scores of the probabilities of the simple events produced by SIMAR approximating all 46 judgments (Mod-simple-all in Table 4.8) were calculated for comparison. By applying correlated t-test, we found that the only cases where we had significant statistical differences the following. For Mavericks, the significant differences were between Mod-simple-complex and Sub-Simple (t(35)=2.22, $p < 0.04$) and between Mod-simple-complex and Mod-simple-all (t(35)=2.11, $p < 0.05$). In these two cases, the quadratic scores deteriorated for 22 and 19 out of 36 participants, respectively (not significant by binomial test). For finance, Sub-simple-all was better than Sub-Simple (t(46)=3.13, $p < 0.003$), as was Mod-simple-complex (t(46)=2.78, $p < 0.01$). The number of participants for which there was improvement were 35 ($p < 0.001$) and 33 ($p < 0.004$) respectively.

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
Sub-simple	.255	.247	.239	.235	.363
Mod-simple-all	.262	.245	.240	.241	.341
Mod-simple-complex	.268	.261	.238	.258	.329

Table 4.8: Quadratic score comparison. Original simple event estimates (Sub-simple) vs. simple event probabilities from SIMAR applied on all judgments (Mod-simple-all) and simple event probabilities from SIMAR applied only on complex events (Mod-simple-complex)

The second experiment concentrated on the accuracy of the complex event estimates and their coherent approximations. Similarly, SIMAR was applied on all judgments and then on the complex judgments only. The corresponding quadratic scores are named Mod-complex-all and Mod-complex-complex in Table 4.9. In this experiment, the differences between Mod-complex-all and Mod-complex-complex are

not statistically significant (by correlated t-test.

However, in all cases, both are reliably better than Sub-complex. For stocks, we have t(30)=3.56 ($p < 0.002$) and t(30)=3.77 ($p < 0.001$). Improvement was seen in 24 participants out of 31 in both cases ($p < 0.002$).

For weather, we have t(37)=8.53 ($p < 0.0001$) and t(37)=6.39 ($p < 0.0001$). Improvement was seen in 36 and 33 participants out of 38, respectively ($p < 0.0001$).

For Suns, we have t(28)=5.95 ($p < 0.0001$) and t(28)=5.92 ($p < 0.0001$). Improvement was seen in 20 and 24 participants out of 29, respectively ($p < 0.03$, $p < 0.001$).

For Mavericks, we have t(35)=7.64 ($p < 0.0001$) and t(35)=7.16 ($p < 0.0001$). Improvement was seen in 31 and 34 participants out of 36, respectively ($p < 0.0001$).

For finance, we have t(46)=5.16 ($p < 0.0001$) and t(46)=7.01 ($p < 0.0001$). Improvement was seen in 33 and 39 participants out of 47, respectively ($p < 0.004$, $p < 0.0001$).

Data	*stocks*	*weather*	*Suns*	*Mavericks*	*finance*
Sub-complex	.254	.226	.237	.226	.288
Mod-complex-all	.228	.198	.196	.189	.257
Mod-complex-complex	.227	.202	.196	.199	.253

Table 4.9: Quadratic score comparison. Original complex event estimates (Sub-complex) vs. complex event probabilities from SIMAR applied on all judgments (Mod-complex-all) and complex event probabilities from SIMAR applied only on these complex events (Mod-complex-complex)

From Table 4.8, it is evident that, in most cases, there is a small deterioration of the quadratic scores of the reconstructed probabilities compared to the original estimates of the basic variables. On the other hand, Table 4.9 shows a uniform improvement of the quadratic scores after reconstruction. However, selective reconstruction of the estimates of only the complex events produced identical results to reconstructing the estimates of both complex and simple events.

4.3 Aggregation of opinion

4.3.1 The aggregation problem, and one approach

Suppose that geopolitical experts employed by the U.S. State Department were willing to express themselves by assigning point probabilities to events. One expert might be knowledgeable about oil production, another about Middle Eastern politics, and a third about foreign military potential. Each would assign probabilities to events over distinct sets of variables representing their individual expertise. Yet there would likely be overlap. The first two experts, for example, might both consider events involving the variable *oil prices fall by at least 10% next quarter*. The second and third might both consider *the Saudi dynasty collapses next year*. The Secretary of State receives from each expert a list of events (absolute and conditional) with probabilities attached. In exploiting these reports, the Secretary will need to confront three potential problems:

(a) Individual reports might be probabilistically incoherent.

(b) Different experts evaluating the same event might assign it different probabilities (*inconsistency among experts*).

(c) Different experts evaluating logically related events might assign probabilities that cannot all be coherently accepted. For example, one expert might assign an event of form $p \wedge q$ a higher probability than the other expert assigns to p. In this case, there is *incoherence among experts*.

A potential solution is to pool the experts' reports to create a single set of estimates over all the variables and then apply SIMAR to ensure consistency and coherence. The individual reports will have thus been *aggregated* into one coherent corpus of opinion. The present section examines the consequences of this aggregation strategy on stochastic accuracy.

Our perspective differs from earlier work on aggregation inasmuch as a multiplicity of complex events are considered at once. The usual goal is to aggregate opinions about the same event, or about the same partition of events. The major schemes for aggregation in the latter contexts involve Bayesian models (Morris, 1977), (Clemen et al., 1996) and linear averaging (Wagner, 1984). Literature reviews in this area include (Clemen, 1989), and (Genest and Zidek, 1986). These approaches seem not to apply straightforwardly to the situation envisioned here, where events relate to each other in complex ways, are evaluated by partially overlapping sets of judges, and are plagued by intra- and inter-judge incoherence. For this reason, we rely on the optimization strategy summarized in (1), which seeks the closest coherent approximation to all of the events in play.

The question immediately arises how to weight individual judges when approximating the pooled estimates. Perhaps some opinions should be more closely approximated than others because the judge who offered them is more reliable. Alternative schemes for weighting judges are reviewed in (Ferrell, 1985). Here we adopt the simple strategy of weighting everyone equally. Similarly, all events will be treated equally, in conformity with statement (1) of our optimization problem. One detail merits clarification, however. If two judges estimate the probability of an event φ, then both estimates of φ will appear in the pooled list of all estimates. Coherent approximation must produce a unique probability for the two copies of φ, and the absolute deviation between this probability and *both* of the original estimates will figure in the overall MAD achieved by the algorithm. Thus, φ will be doubly important to the optimization process since it occurs twice (and similarly for conditional events). Our approach may thus be summarized as follows.

(6) To aggregate the opinions of a collection of judges, pool their estimates to form a single corpus, then apply SIMAR to create a coherent set of probabilities that approximate all the original judgments at once.[1] The latter probabilities

are taken to be the group opinion.

This procedure will be called SIMARAGG in what follows.

Before examining SIMARAGG empirically, we observe that its use potentially allows every judge to influence the coherent probabilities constructed for every other judge. To illustrate, consider three judges, the first of whom faces events over the variables p, q, the second over q, r, and the third over r, s. Suppose that they offer the following estimates.

$$(7)\quad \boxed{\begin{array}{lrcl} \text{Judge 1:} & \mathsf{Pr}(p \wedge q) & \approx & 1 \\ \text{Judge 2:} & \mathsf{Pr}(r : q) & \approx & 1 \\ \text{Judge 3:} & \mathsf{Pr}(s : r) & \approx & 1 \end{array}}$$

Suppose also that Judge 3 offers an estimate of the probability of the variable s. Then, if the estimates (7) are retained in the aggregate, coherence requires that $\mathsf{Pr}(s) \approx 1$. There is no such constraint on the probability of s in the absence of the estimate offered by Judge 1. Thus, in the aggregate, the estimates offered by Judge 1 influence the approximation of the estimates of Judge 3 even though the two judges face events over disjoint sets of variables.

4.3.2 SIMARAGG in the stocks, weather, Suns, Mavericks, and finance experiments

For each of the five data sets, we pooled all estimates of chance offered by any of the participants in each experiment. The resulting sets of judgments may be called the *aggregate judges*. Recall that each participant made estimates for events defined from ten variables (in the finance experiment, these variables were randomly chosen for him or her from a stock of thirty). The aggregate judge therefore works in a probability space built from thirty variables. SIMAR was used to find a close approximation in this space to all judgments at once. The computation required about an hour on average.

For each of the judges in a data set, we compared the quadratic penalties arising from:

(i) the judge's original estimates;

(ii) the judge's estimates after revision by individual application of SIMAR; and

(iii) the judge's estimates after revision by SIMARAGG, as in (6).

For (iii), the judge's estimate of the probability of an event φ was replaced by the estimate of φ obtained by applying SIMAR to the aggregate judge (and similarly for conditional events).

For the finance experiment, the average quadratic penalty for SIMARAGG is .254, as seen in line (c) of Table 4.10 (finance part). This is lower than the average penalties for raw estimates and SIMAR; see line (a) of Table 4.10. The differences are reliable by correlated t test [$t(46) = 5.84$, respectively]. For 32 out of the 47 subjects, the penalty is lower if their estimates were corrected as part of the group (SIMARAGG) compared to individually (SIMAR) ($p < .01$ by a binomial test). So it seems that coherent approximation squeezes more information out of human judgment if opinion is pooled rather than left atomized.

Fifteen pooled judges seem to be enough to get all the benefit of aggregation, at least in these data. When random subsets of 15 are drawn from our pool of 47 participants, we observe lower quadratic penalties for the aggregate compared to the 15 individuals. Beyond 15, further improvement for the aggregate judge is not detectable. This finding is consistent with other experimental studies on quantitative estimates; small panels suffice to obtain most of the benefit of aggregated judgment, see (Ashton and Ashton, 1985).

The foregoing results concern quadratic penalty. When slope is used as a measure of accuracy there is only small benefit of aggregation. As shown in Table 4.11, line (c) (finance part), the average slope after application of SIMARAGG is .123.

This is superior to slopes from raw estimates and SIMAR, but the difference is reliable by correlated t-test only in the comparison to raw estimates [$t(46) = 4.37, .53$]. Thirty-six of the 47 participants have higher slopes after SIMARAGG compared to their raw estimates ($p < .001$ by binomial test). Only 26 of 47 have higher slopes after SIMARAGG compared to SIMAR (not significant).

The same pattern emerges when data from the other four experiments are aggregated, see Tables 4.10 and 4.11. Again we compare SIMARAGG estimates with raw estimates and with SIMAR. In Suns, SIMARAGG penalties are reliably lower than for both raw estimates and SIMAR [$t(28) = 7.30, 5.09$]. Improvement is seen for 28 and 26 of the 29 participants, respectively ($p < .001$). Similarly, SIMARAGG slopes are reliably higher than for raw estimates and SIMAR [$t(28) = 6.13, 3.3$]. For slope, improvement is seen for 26 and 25 of the 29 participants, respectively.

In Mavericks, SIMARAGG quadratic penalties are again reliably lower than for raw estimates and SIMAR [$t(35) = 4.80, 2.18$]. Improvement is seen for 27 and 24 of the 36 participants, respectively ($p < .03$). SIMARAGG slopes are reliably higher than raw slopes [$t(35) = 4.59$ and the improvement is seen in 26 of the 36 participants ($p < .006$). There is no reliable slope advantage, however, when SIMARAGG is compared to SIMAR. The difference between average slopes in these two cases yields $t(35) = 1.07$, and is seen in only 20 of the 36 participants (not significant).

In stocks, SIMARAGG quadratic penalties are once more reliably lower than for raw estimates and SIMAR [$t(30) = 5.36, t(30) = 3.54$]. Improvement is seen for 26 and 24 of the 31 participants, respectively ($p < .001$). SIMARAGG slopes are reliably higher than for raw estimates [$t(30) = 2.14$ and is seen in 21 of the 31 participants ($p < .04$). Again, however, there is no reliable slope advantage for SIMARAGG compared to SIMAR, and the advantage is seen in only 19 of the 31 participants (not significant).

In weather, SIMARAGG quadratic penalties are again reliably lower then for

the raw estimates and SIMAR$[t(37) = 5.17, t(37) = 4.81]$. Improvement is seen for 35 and 33 out of 38 participants, respectively ($p < .001$). SIMARAGG slopes are reliably higher than for raw estimates $[t(37) = 2.25]$ and there is improvement in 27 of the 38 participants ($p < .04$). Again, however, there is no reliable slope advantage for SIMARAGG compared to SIMAR, and the advantage is seen in only 21 of the 38 participants (not significant).

Overall, the data show reliable improvement in quadratic penalty when coherent approximation is applied to estimates in the aggregate rather than individually. The same tendency is consistently seen for slope, but the effect is smaller and usually not statistically significant.

4.3.3 Comparison to linear pooling of probability arrays

A plausible alternative to the aggregation strategy (6) is to average the probability arrays that emerge from use of SIMAR on individual participants. Technically, the weighted average of n probability arrays (over the same set of variables) is achieved by concatenating them then renormalizing the probabilities in the last row by dividing by n. (The number of columns in the resulting array can be somewhat reduced by merging columns that agree at every row corresponding to a variable, then assigning it the sum of the probabilities in the last row of the merged columns.) This procedure assigns a given state the average probability it receives in the n probability arrays. Such concatenated probability arrays represent the simplest *linear pool* of opinion. Linear pooling has been analyzed in several axiomatic studies, see (Wagner, 1984), (Genest and Zidek, 1986). Its use here may be summarized as follows.

(8) To aggregate the opinions of a collection of judges, concatenate the probability arrays that result from applying SIMAR to each judge separately, and renormalize. Then use the concatenated probability array to assign probabilities to each of the estimated events.

	Quadratic penalty computed for:	Mean	(S.D.)
	finance ($N = 47$)		
(a)	raw estimates	.314	.077
(b)	after individual approximation by SIMAR	.285	.076
(c)	after aggregate approximation by SIMARAGG	.254	.042
	Suns ($N = 29$)		
(a)	raw estimates	.237	.053
(b)	after individual approximation by SIMAR	.205	.042
(c)	after aggregate approximation by SIMARAGG	.167	.021
(d)	after aggregate approximation by LINPOOL	.190	.023
	Mavericks ($N = 36$)		
(a)	raw estimates	.228	.064
(b)	after individual approximation by SIMAR	.200	.067
(c)	after aggregate approximation by SIMARAGG	.176	.020
(d)	after aggregate approximation by LINPOOL	.199	.021
	stocks ($N = 31$)		
(a)	raw estimates	.254	.057
(b)	after individual approximation by SIMAR	.232	.055
(c)	after aggregate approximation by SIMARAGG	.199	.025
(d)	after aggregate approximation by LINPOOL	.209	.028
	weather ($N = 38$)		
(a)	raw estimates	.232	.053
(b)	after individual approximation by SIMAR	.201	.051
(c)	after aggregate approximation by SIMARAGG	.178	.033
(d)	after aggregate approximation by LINPOOL	.185	.035

Table 4.10: Quadratic penalty comparisons for SIMAR and SIMARAGG

We denote the technique in (8) by LINPOOL in what follows.

Observe that LINPOOL cannot be applied to the data in finance since different participants evaluated events over distinct sets of variables (each was randomly assigned 10 variables from a starting set of 30). This obstacle to LINPOOL would be typical in the kind of application discussed at the beginning of the present section (involving judges with overlapping expertise). Linear pooling does apply to Suns, Mavericks stocks and weather, however, since each was based on a single set of 10 variables. It is therefore instructive to compare the stochastic accuracy of LINPOOL to SIMARAGG.

	Slope computed for:	Mean	$(S.D.)$
	finance $(N = 47)$		
(a)	raw estimates	.053	.130
(b)	after individual approximation by SIMAR	.115	.129
(c)	after aggregate approximation by SIMARAGG	.123	.092
	Suns $(N = 29)$		
(a)	raw estimates	.138	.133
(b)	after individual approximation by SIMAR	.221	.106
(c)	after aggregate approximation by SIMARAGG	.276	.057
(d)	after aggregate approximation by LINPOOL	.208	.053
	Mavericks $(N = 36)$		
(a)	raw estimates	.133	.130
(b)	after individual approximation by SIMAR	.214	.131
(c)	after aggregate approximation by SIMARAGG	.237	.049
(d)	after aggregate approximation by LINPOOL	.180	.048
	stocks $(N = 31)$		
(a)	raw estimates	.129	.176
(b)	after individual approximation by SIMAR	.179	.154
(c)	after aggregate approximation by SIMARAGG	.196	.053
(d)	after aggregate approximation by LINPOOL	.156	.058
	weather $(N = 38)$		
(a)	raw estimates	.053	.120
(b)	after individual approximation by SIMAR	.134	.112
(c)	after aggregate approximation by SIMARAGG	.155	.064
(d)	after aggregate approximation by LINPOOL	.150	.056

Table 4.11: Slope comparisons for SIMAR and SIMARAGG

Comparison of lines (c) and (d) in Table 4.10, and (c) and (d) in Table 4.11 shows that LINPOOL is inferior to SIMARAGG. In all four experiments, LINPOOL has higher quadratic penalty and lower slope compared to SIMARAGG. The difference in quadratic penalty is reliable in all four studies [$t(28) = 11.04$, $t(35) = 14.70$, $t(30) = 4.54$, and $t(37) = 6.72$], and is seen in 28/29, 36/36, , 24/31 and 32/38 participants, respectively ($p < .001$). The difference in slope is also reliable in all three studies [$t(28) = 14.2$, $t(35) = 15.03$, $t(30) = 8.14$, and $t(37) = 10.32$], and is seen in 29/29, 36/36, 29/31 and 37/38 participants ($p < .001$).

What accounts for the superiority of SIMARAGG compared to LINPOOL? It may be that individual estimates are too anonymous when whole distributions are av-

eraged. In such a procedure, estimates made by different judges interact only through their impact on the distributions (probability arrays) computed for the judges separately. In contrast, pooling all the judgments before coherent approximation allows more direct confrontation between estimates of logically related events, including events evaluated by different judges.

4.4 Discussion

4.4.1 Coherent approximation and stochastic accuracy

What explains the improvement in stochastic accuracy following coherent approximation by SIMAR? As a preliminary remark, let us recall that simulated annealing is a randomized algorithm so SIMAR's behavior depends on its random seed. We have therefore applied SIMAR to our data multiple times starting from alternative seeds. All the results in this paper are stable through change in seed.

It can also be seen more directly why coherent approximation will often reduce quadratic penalty. Suppose that the judge offers $\mathsf{Pr}(p) = .6$ and $\mathsf{Pr}(p \wedge q) = .8$. The closest coherent approximation revises both estimates to .7, which lies between the two original estimates. Since quadratic penalty is a convex function, the penalty is guaranteed to be lower for the two copies of .7 compared to .6 and .8. The foregoing explanation does not extend to improvements seen in slope.

Improvement in stochastic accuracy when opinion is aggregated presumably results from reduction of error variance through averaging. A rigorous formulation of this phenomenon (which we do not attempt here) is complicated by (a) the presence of incoherence within and between judges and (b) variability regarding which events were evaluated by different judges.

Chapter 5

Parameter space analysis of the algorithms

5.1 Parameter analysis of SIMAR

In this section an analysis of the parameter space of SIMAR is presented. It is important to know not only parameter values leading to increased accuracy, but also the robustness of the algorithm when one tries input data of various types as well as different penalty functions (i.e. absolute or quadratic deviation). The parameters of the algorithm are described in section 3.3.1 and are listed below.

- N_T is the number of temperature decreasing steps.
- N_{iter} is the number of iterations per temperature step.
- Iar is the initial acceptance rate.
- $N_{columns}$ is the number of columns of the probability array.
- P_{star} is the probability of *don't care* for the probability array.
- ϵ determines the relative change of the probability assigned to a column of the array.

5.1.1 Experimental Data

First, the study is based on the 5 sets of experimental data , Suns, Mavericks, stocks, weather, and finance. The results shown below are the average MADs among the 5 sets. Since it is impractical to perform analysis by trying out all possible values of the parameters, we group certain parameters and keep the rest constant. First we set $N_{columns} = 50$, $\epsilon = 0.1$, $P_{star} = 0.0$, $Iar = 0.9$ and we try different values for N_T and N_{iter}. It is evident from Table 5.1 that the results are quite insensitive to changes to the parameters N_T and N_{iter}. It is noted that the running time is proportional to the product $N_T N_{iter}$, thus one reasonable choice is $N_T = 150, N_{iter} = 25$. These values are used for the rest of the analysis. Table 5.2 shows the effect of the parameters $N_{columns}$ and P_{star}. The rest of the parameters are fixed to $N_T = 150$, $N_{iter} = 25$, $\epsilon = 0.1$, $Iar = 0.9$. The running time is proportional to the number of columns $N_{columns}$ and independent of P_{star}, thus a good choice is $N_{columns} = 50$ and $P_{star} = 0.0$.

The parameter Iac has limited effect to the performance as shown in Table 5.3. The parameter ϵ is set equal to 0.1 as indicated by Table 5.4. The results of Table 5.5 show the optimal values of $N_{columns}$ and P_{star} for the aggregate data. Here we have a big number of judgments, thus it is expected that the best results will be obtained by probability arrays representing dense distributions and the use of many columns is justified. It is finally noted, that the use of multiple starting points in conjunction with the "Go-with-the-winners" strategy, further reduced the best MAD 2.5% for 5 starting points and 3% for 10. It noted however that the running time are increased linearly with the number of starting points, thus we use 1 starting point in all experiments.

5.1.2 Simulated Data

Similarly, experiments have been performed to see the effect of $N_{columns}$ and P_{star} when simulated data are considered, see Tables 5.6-5.9. In this case, the results

are quite robust and one can select the smallest possible value for $N_{columns}$ to achieve fastest running times. Roughly, the running time is linear to the product of the number of variables and the number of columns $N_{columns}$. It is noted that the algorithm did not converge for some data sets when $P_{star} = 0.0$ and $N_{columns} = 100, 200$. It is evident, that the best compromise between running time and performance of the minimization is to choose a large value for P_{star} (.6) and use as few as 100 columns.

5.2 Parameter analysis of GAADD

In this section an analysis of the parameter space of GAADD is presented. Similarly to the analysis of section 5.1, several experiments have been performed to achieve best accuracy and to examine the robustness of the algorithm. The parameters of the algorithm are described in section 3.3.2 and are listed below.

- *Gennum* is the number of generations.
- *Popul* determines the size of the population.
- P_c is the probability of crossover.
- P_m is the probability of mutation.
- N_{prod} controls the number of products that are added to form an ADD of the initial population.
- P_{dc} the probability of *don't care* while forming the products that constitute the initial ADDs.
- ϵ_m is a constant used to perturb probabilities during mutation.

5.2.1 Experimental data

Similarly to the analysis of SIMAR, the 5 sets of experimental data , Suns, Mavericks, stocks, weather, and finance are considered. The results shown in Table

5.10 are the average MAD and MQD among the 5 sets. First we set $N_{prod} = 50$, $\epsilon = 0.1$, $P_{dc} = 0.0$, and we try different values for P_c and P_m. Table 5.11 shows the effect of the parameters N_{prod} and P_{dc}. The rest of the parameters are fixed to $Popul = 200$, $Gennum = 200$, $P_c = 0.95$, $P_m = 0.04$, $\epsilon_m = 0.1$. The impact of the population size $Popul$ is demonstrated in Table 5.12. A population of 400 chromosome yields considerable improvement but the penalty is excessively large running times. Typical running times for populations of 100, 200 and 400 chromosomes are 1,2 and 7 minutes per individual (46 judgments) respectively. The effect of the parameter ϵ_m that controls the magnitude of mutation is shown in Table 5.13. The algorithm is insensitive to that parameter and we set it to .1 for all experiments. The results in Table 5.14 show the optimal values of N_{prod} and P_{dc} for the aggregate data. Here we have a big number of judgments, thus it is expected that the best results will be obtained by probability arrays representing dense distributions, see (2).

5.2.2 Simulated data

Similarly, experiments have been performed to see the effect of the population size $Popul$ as well as of the parameters N_{prod} and P_{dc} when simulated data are considered. In this case, the results are quite robust and one can select the smallest possible value for $Popul$ and N_{prod} to achieve fastest running times. From Tables 5.15-5.18, it is evident that it is sufficient to have as few as 25 products/ADD and to have a population between 100 and 200. It is noted that the running times vary significantly. For $Popul = 100$, $N_{prod} = 25$ running times vary from 20 minutes for 20 variables to 140 for 50 variables. On the other hand for $Popul = 200$, $N_{prod} = 50$ the running times vary from 200 minutes for 20 variables to 2000 minutes for 50 variables. It is evident that one should use small populations with few products per initial ADD. The robustness of the results with respect to these parameters will ensure no loss in accuracy.

N_T	100	150	200	250	300
N_{iter}	MAD	MAD	MAD	MAD	MAD
10	.0981	.0942	.0937	.0930	.0918
25	.0926	.0914	.0910	.0909	.0910
50	.0926	.0918	.0934	.0926	.0942
N_{iter}	MQD	MQD	MQD	MQD	MQD
10	.0220	.0207	.0202	.0196	.0195
25	.0196	.0191	.0190	.0189	.0190
50	.0191	.0189	.0187	.0187	.0188

Table 5.1: Average MAD and MQD achieved for various values of N_T and N_{iter}

$N_{columns}$	25	50	100	200
P_{star}	MAD	MAD	MAD	MAD
0.0	.0941	.0914	.0911	.0915
0.2	.0940	.0923	.0920	.0924
0.4	.0974	.0955	.0944	.0958
P_{star}	MQD	MQD	MQD	MQD
0.0	.0198	.0191	.0189	.0187
0.2	.0200	.0194	.0193	.0190
0.4	.0202	.0201	.0199	.0196

Table 5.2: Average MAD and MQD achieved for various values of $N_{columns}$ and P_{star}

Iac	.95	.9	.8	.7	.6	.5
MAD	.0932	.0914	.0915	.0914	.0914	.0914
MQD	.0210	.0201	.0196	.0191	.0192	.0191

Table 5.3: Average MAD and MQD achieved for various values of Iac

ϵ	.05	.1	.2	.3
MAD	.0922	.0914	.0920	.0931
MQD	.0192	.0191	.0202	.0215

Table 5.4: Average MAD and MQD achieved for various values of ϵ

$N_{columns}$	50	100	200	400
P_{star}	MAD	MAD	MAD	MAD
0.0	.1916	.1913	.1913	.1912
0.2	.1915	.1913	.1912	.1912
0.4	.1921	.1922	.1914	.1913
P_{star}	MQD	MQD	MQD	MQD
0.0	.0595	.0594	.0590	.0589
0.2	.0594	.0590	.0583	.0584
0.4	.0591	.0590	.0589	.0590

Table 5.5: Average MAD and MQD achieved for various values of $N_{columns}$ and P_{star} for aggregate data

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0045	.0033	.0063	.0037	.0054	.0029	.0062	.0042
.1	.0769	.0289	.0638	.0276	.0638	.0269	.0680	.0277
.2	.1478	.0389	.1428	.0398	.1472	.0378	.1454	.0386

Table 5.6: SIMAR on simulated data, $P_{star} = 0.6, num\_of\_columns = 100$

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0056	.0035	.0044	.0038	.0053	.0027	.0056	.0025
.1	.0734	.0285	.0648	.0294	.0656	.0277	.0677	.0291
.2	.1536	.0387	.1506	.0371	.1473	.0383	.1457	.0390

Table 5.7: SIMAR on simulated data, $P_{star} = 0.6, num\_of\_columns = 200$

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0036	.0028	.0046	.0021	.0048	.0029	.0062	.0034
.1	.0715	.0267	.0641	.0263	.0671	.0275	.0707	.0270
.2	.1525	.0366	.1509	.0371	.1419	.0357	.1391	.0349

Table 5.8: SIMAR on simulated data, $P_{star} = 0.6, num\_of\_columns = 400$

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0104	.0037	.0103	.0045	.0113	.0044	.0129	.0041
.1	.0636	.0287	.0587	.0271	.0570	.0266	.0597	.0277
.2	.1397	.0368	.1302	.0371	.1269	.0357	.1245	.0365

Table 5.9: SIMAR on simulated data, $P_{star} = 0.0, num\_of\_columns = 400$

P_c	.8	.9	.95	1
P_m	MAD	MAD	MAD	MAD
.02	.1008	.0998	.0994	.0994
.03	.1005	.0995	.0985	.0987
.04	.0996	.1015	.0983	.0984
.05	.0997	.1017	.0985	.0984
P_m	MQD	MQD	MQD	MQD
.02	.0242	.0231	.0234	.0240
.03	.0237	.0231	.0227	.0226
.04	.0229	.0220	.0218	.0219
.05	.0233	.0223	.0220	.0220

Table 5.10: Average MAD and MQD achieved by GAADD for various values of P_c and P_m

N_{prod}	25	50	100
P_{dc}	MAD	MAD	MAD
0.0	.1010	.0984	.0993
0.2	.1021	.1001	.0996
0.4	.0995	.1012	.1002
P_{dc}	MQD	MQD	MQD
0.0	.0227	.0218	.0211
0.2	.0226	.0225	.0215
0.4	.0227	.0231	.0227

Table 5.11: Average MAD and MQD achieved by GAADD for various values of N_{prod} and P_{dc}

$Popul$	50	100	200	400
MAD	.1024	.1011	.0983	.0935
MQD	.0234	.0228	.0218	.0194

Table 5.12: Average MAD and MQD achieved by GAADD for various population sizes

ϵ_m	.05	.1	.2	.3
MAD	.0982	.0983	.0989	.0993
MQD	.0221	.0218	.0218	.0224

Table 5.13: Average MAD and MQD achieved by GAADD for various values of ϵ_m

N_{prod}	50	100	200	400
P_{dc}	MAD	MAD	MAD	MAD
0.0	.1923	.1920	.1931	.1937
0.2	.1926	.1927	.1933	.1938
0.4	.1931	.1937	.1933	.1941
P_{star}	MQD	MQD	MQD	MQD
0.0	.0601	.0591	.0596	.0588
0.2	.0607	.0601	.0598	.0591
0.4	.0605	.0596	.0591	.0595

Table 5.14: Average MAD and MQD achieved for various values of N_{prod} and P_{dc} for aggregate data

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0054	.0067	.0057	.0072	.0090	.0077	.0072	.0062
.1	.0751	.0317	.0755	.0328	.0766	.0314	.0822	.0333
.2	.1481	.0407	.1482	.0411	.1472	.0408	.1468	.0397

Table 5.15: GAADD on simulated data, $Popul = 200, N_{prod} = 25$

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0055	.0085	.0074	.0091	.0114	.0082	.0088	.0073
.1	.0803	.0335	.0826	.0315	.0856	.0326	.0898	.0349
.2	.1491	.0428	.1494	.0431	.1510	.0417	.1507	.0410

Table 5.16: GAADD on simulated data, $Popul = 100, N_{prod} = 25$

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0036	.0035	.0046	.0033	.0048	.0027	.0062	.0021
.1	.0715	.0289	.0641	.0297	.0671	.0301	.0707	.0273
.2	.1525	.0417	.1509	.0428	.1419	.0413	.1391	.0409

Table 5.17: GAADD on simulated data, $Popul = 200, N_{prod} = 50$

vars.	20		30		40		50	
Inc.	MAD	MQD	MAD	MQD	MAD	MQD	MAD	MQD
0	.0104	.0085	.0103	.0083	.0113	.0077	.0129	.0079
.1	.0636	.0342	.0587	.0344	.0570	.0331	.0597	.0337
.2	.1397	.0434	.1302	.0417	.1269	.0429	.1245	.0421

Table 5.18: GAADD on simulated data, $Popul = 100, N_{prod} = 50$

Chapter 6

Concluding remarks

The problem of restoring coherence of subjective estimates of chance has been addressed. It is formulated as an optimization problem and two heuristic-based algorithms have been introduced. They are combined with data structures that enable the compact representation of joint probability distribution on several variables. This leads to scalable methods that efficiently reconstruct estimates of both absolute and conditional events. The proposed methods produce coherent approximations of the input judgments that maintain the insight of the expert(s). In addition , it is shown, through empirical studies, that the stochastic accuracy (measured by quadratic penalty and slope) of the reconstructed probabilities is reliably better than that of the original estimates. Aggregation of opinion was explored and it is shown that aggregation improves stochastic accuracy. Comparisons with the standard approach of linear pooling were presented.

Bibliography

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

Ashton, A. H. and Ashton, R. H. (1985). Aggregating subjective forecasts: some empirical results. *Management Science*, 31:1499–1508.

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

Bernardo, J. and Smith, A. (1994). *Bayesian Theory*. New York, John Wiley and Sons.

Biazzo, V., Gilio, A., Lukasiewicz, T., and Sanfilippo, G. (2001). Probabilistic logic under coherence: Complexity and algorithms. *Second International Symposium on Imprecise Probabilities and Their Applications*.

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

Chvátal, V. (1983). *Linear Programming*. San Francisco CA, W. H. Freeman.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5:559–583.

Clemen, R. T., Jones, S. K., and Winkler, R. L. (1996). Aggregating forecasts: An empirical evaluation of some bayesian methods. In D. A Berry, K. M. C. and Geweke, J. K., editors, *Bayesian Analysis in Statistics and Economics*. New York NY, John Wiley & Sons.

Davis, L. (1987). *Genetic Algorithms and Simulated Annealing*. Los Altos CA, Morgan Kaufman Publishers.

de Finetti, B. (1972). *Probability, Induction, and Statistics*. New York, Wiley.

Druzdzel, M. J. and van der Gaag, L. C. (1995). Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In *Uncertainty in Artificial Intelligence (95): Proceedings of the 11th conference*, Los Altos CA. Morgan Kaufmann.

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

Ferrell, W. R. (1985). Combining individual judgments. In Wright, G., editor, *Behavioral Decision Making*, pages 111–145. New York NY, Plenum Press.

Genesereth, M. and Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA.

Genest, C. and Zidek, J. (1986). Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–135.

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

Halpern, J. (1990). An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350.

Jaumard, B., Hansen, P., and Aragao, M. (1991). Column generation methods for probabilistic logic. *ORSA Journal on Computing*, 3(2):135–148.

Kavvadias, D. and Papadimitriou, C. (1990). A linear programming approach to reasoning about probabilities. *Annals of Mathematics and Artificial Intelligence*, 1:189–205.

Lukasiewicz, J. (1913). *Die logische grundlagen der wahrscheinlichkeitsrechnung*. English version publised as: Logical foundations of probability theory. In I. Berkowski (editor) *Jan Lukasiewicz Selected Works*, 1970.

Lukasiewicz, T. and Isberner, G. (1999). Probabilistic logic programming under maximum entropy. *IFIG Research Report 9903*.

Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge MA, MIT Press.

Morris, P. (1977). Combining expert judgments: A bayesian approach. *Management Science*, 23:679–693.

Nemhauser, G., Kan, A. R., and Todd, M. (1989). *Optimization*. Amsterdam, North-Holland.

Nilsson, N. (1986). Probabilistic logic. *Artificial Intelligence*, 28(1):71–87.

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

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288.

Rustagi, J. S. (1994). *Optimization Techniques in Statistics*. New York NY, Academic Press.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University, Princeton, NJ.

Shafer, G., Gillett, P., and Scherl, R. (2002). A new understanding of subjective probability and its generalization to lower and upper prevision. Game-Theoretic Probability Project Working Paper #3, to appear in the *International Journal of Approximate Reasoning*. URL: http://www.glennshafer.com/articles.

Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the Association for Computing Machinery*, 42:206 – 242.

Torsun, I. (1995). *Foundations of intelligent knowledge-based systems*. Academic Press, New York.

van der Gaag, L., Renooij, S., Witteman, C., Aleman, B., and Taal, B. (1999). How to elicit many probabilities. In Laskey, K. B. and Prade, H., editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco. Morgan Kaufmann Publishers.

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

Vavasis, S. (1993). *Complexity Issues in Global Optimization: A Survey*. url: citeseer.nj.nec.com/vavasis95complexity.html.

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

Wagner, C. (1984). Aggregating subjective probabilities: Some limitative theorems. *Notre Dame Journal of Formal Logic*, 25(3).

Walley, P. (1997). Coherent upper and lower previsions. The Imprecise Probabilities Project, URL: http://ensmain.rug.ac.be/ ipp.

Yates, J. F. (1990). *Judgment and Decision Making*. Englewood Cliffs NJ, Prentice Hall.

Zadeh, L. (1971). Fuzzy sets. *Information and Control*, 8:338–353.