# On the Complexity of Learning Lexicographic Strategies

Michael Schmitt
Lehrstuhl Mathematik und Informatik, Fakultät für Mathematik
Ruhr-Universität Bochum, 44780 Bochum, Germany
mschmitt@lmi.ruhr-uni-bochum.de

Laura Martignon
Institut für Mathematik und Informatik, Pädagogische Hochschule Ludwigsburg
Reuteallee 46, 71634 Ludwigsburg, Germany
martignon@ph-ludwigsburg.de

**Abstract**

Fast and frugal heuristics are well studied models of bounded rationality. Psychological research has proposed the take-the-best heuristic as a successful strategy in decision making with limited resources. Take-the-best searches for a sufficiently good ordering of cues (or features) in a task where objects are to be compared lexicographically. We investigate the computational complexity of finding optimal cue permutations for lexicographic strategies and prove that the problem is NP-complete. It follows that no efficient (that is, polynomial-time) algorithm computes optimal solutions, unless P = NP. We further analyze the complexity of approximating optimal cue permutations for lexicographic strategies. We show that there is no efficient algorithm that approximates the optimum to within any constant factor, unless P = NP.

The results have implications for the complexity of learning lexicographic strategies from examples. They show that learning them in polynomial time within the model of agnostic probably approximately correct (PAC) learning is impossible, unless RP = NP. We further consider greedy approaches for building lexicographic strategies and determine upper and lower bounds for the performance ratio of simple algorithms. Moreover, we present a greedy algorithm that performs provably better than take-the-best. Tight bounds on the sample complexity for learning lexicographic strategies are also given in this article.

**Keywords:** bounded rationality, fast and frugal heuristic, PAC learning, NP-completeness, hardness of approximation, greedy method

# 1 Introduction

In many circumstances the human mind has to make decisions when time is scarce and knowledge is limited. Extensive reflections backed by deep reasoning are impossible in these situations. Cognitive psychology categorizes human judgments made under such constraints as being boundedly rational if they are "satisficing" (Simon, 1982) or, more generally, if they do not fall too far behind the rational standards. The modeling of bounded rationality has been considered essential for artificial intelligence. Russell and Wefald (1991), defining artificial intelligence as the problem of designing systems that "do the right thing", argue that intelligence seems linked with doing as well as possible given what resources one has.

A principal family of models for human reasoning that are studied within the context of bounded rationality are the probabilistic mental models proposed by Gigerenzer et al. (1991). To these belongs a kind of simple algorithms termed "fast and frugal heuristics" that were the topic of major research projects in psychology (Gigerenzer and Goldstein, 1996; Gigerenzer et al., 1999). Great efforts have been put into testing these heuristics by empirical means in experiments with human subjects on the one hand (Bröder, 2000; Bröder and Schiffer, 2003; Lee and Cummins, 2004; Newell and Shanks, 2003; Newell et al., 2003; Slegers et al., 2000) or in simulations on computers on the other (Bröder, 2002; Bullock and Todd, 1999; Hogarth and Karelaia, 2003; Nellen, 2003; Todd and Dieckmann, 2005). (See also the discussion and controversies documented in the open peer commentaries on Todd and Gigerenzer, 2000.) To a lesser extent, theoretical studies have been undertaken with analytical methods (Bröder, 2002; Martignon and Hoffrage, 1999, 2002; Martignon and Schmitt, 1999).

## 1.1 Take The Best

Among the fast and frugal heuristics there is an algorithm called "take-the-best"[1] (TTB) that during recent years has become one of the workhorses of research into models of bounded rationality. This algorithm is considered a process model for human judgments based on one-reason decision making. Which of the two cities has a larger population: (a) Düsseldorf, (b) Hamburg? This is the task originally studied by Gigerenzer and Goldstein (1996) where German cities with a population of more than 100,000 inhabitants have to be compared. The available information on each city consists of the values of nine binary cues, or attributes, indicating presence or absence of a feature. The cues being used are, for instance, whether the city is a state capital, whether it is indicated on car license plates by a single letter, or whether it has a soccer team in the national league.

[1] "Take-the-best" is a shortening of "take the best, ignore the rest" (Gigerenzer and Goldstein, 1996).

| | Soccer Team | State Capital | License Plate |
|---|---|---|---|
| Hamburg | 1 | 1 | 0 |
| Essen | 0 | 0 | 1 |
| Düsseldorf | 0 | 1 | 1 |
| Validity | 1 | 1/2 | 0 |

Table 1: Part of the German cities task of Gigerenzer and Goldstein (1996). Shown are cue profiles and validities. Validities are computed from the cues of the three cities as given here. The original data has different validities but yields the same ranking for the cues. The meaning of the cues and the way how to calculate validities are explained in the text.

The judgment which city is larger is made on the basis of the two binary vectors, or cue profiles, representing the two cities. TTB compares the cues one after the other and uses the first cue that discriminates as the one reason to yield the final decision. In other words, TTB performs a lexicographic strategy of comparison. For instance, if one city has a university and the other does not it would infer that the first city is larger than the second. If the cue values of both cities are equal, the algorithm passes on to the next cue.

TTB examines the cues in a certain order. Gigerenzer and Goldstein (1996) introduced ecological validity as a numerical measure for ranking the cues. (See Martignon and Hoffrage, 2002, for further criteria to order cues.) The validity of a cue is a real number in the interval $[0, 1]$ that is computed in terms of the known outcomes of paired comparisons. It is defined as the number of pairs the cue discriminates correctly (i.e., where it makes a correct inference) divided by the number of pairs it discriminates (i.e., where it makes an inference, be it right or wrong). TTB always chooses a cue with the highest validity, that is, it "takes the best" among those cues not yet considered. Table 1 gives an example showing cue profiles and validities for three cities. The data are extracted from the appendix of Gigerenzer and Goldstein (1996). The ordering defined by the population size of the cities is given by

$$\{\langle\, \text{Düsseldorf}\,, \text{Essen}\,\rangle, \langle\, \text{Düsseldorf}\,, \text{Hamburg}\,\rangle, \langle\, \text{Essen}\,, \text{Hamburg}\,\rangle\},$$

where a pair $\langle a, b\rangle$ indicates that $a$ has less inhabitants than $b$. As an example for calculating the validity, the state-capital cue distinguishes the first and the third pair but is correct only on the latter. Hence, its validity has value $1/2$.

The order in which the cues are ranked is crucial for success or failure of TTB. In the example of Düsseldorf and Hamburg, the car-license-plate cue would yield that Düsseldorf (represented by the letter "D") is larger than Hamburg

(represented by the two letters "HH"), whereas the soccer-team cue would favor Hamburg, which is correct. Thus, how successful a lexicographic strategy is in a comparison task consisting of a partial ordering of cue profiles depends on how well the cue ranking minimizes the number of incorrect comparisons. Specifically, the accuracy of TTB relies on the degree of optimality achieved by the ranking according to decreasing cue validities. For TTB and the German cities task, computer simulations have shown that TTB discriminates at least as accurate as other models (Gigerenzer and Goldstein, 1996; Gigerenzer et al., 1999; Todd and Dieckmann, 2005). TTB made as many correct inferences as standard algorithms proposed by cognitive psychology and even outperformed some of them.[2]

## 1.2 Accuracy and Complexity

Partial results concerning the accuracy of TTB compared to the accuracy of other strategies have been obtained analytically by Martignon and Hoffrage (2002). The intention of this article is to subject the problem of finding optimal cue orderings to a rigorous theoretical analysis. A conceivable approach would be to reveal conditions under which TTB performs better or worse. However, the analysis of TTB per se is not a major topic of this work. Instead, we take a different and more general road by employing methods from the theory of computational complexity (Garey and Johnson, 1979).

Obviously, TTB is an algorithm that runs in polynomial time. Given a list of ordered pairs, it computes all cue validities in a number of computing steps that is linear in the size of the list, assuming random access to the values of the cues. This observation directs our attention to studying the computational complexity of the problem of finding optimal cue permutations. Is there really an efficient algorithm that solves this problem? We define the decision problem Lexicographic Strategy as the task of determining whether for a given partial ordering, represented as a list of pairs of cue profiles, and a given threshold there exists a cue permutation such that the number of incorrect comparisons made by the lexicographic strategy does not exceed this threshold. As a fundamental result we prove that Lexicographic Strategy is NP-complete. It follows that TTB is not an algorithm for computing optimal cue permutations and, even more, that no polynomial-time algorithm exists for solving this task, unless the complexity classes P and NP are equal.

The fact that finding optimal cue permutations turns out to be practically intractable, however, does not exclude the possibility that the optimum can be efficiently approximated. The second main topic of this article is an optimization

[2]Gigerenzer and Goldstein (1996) introduced TTB with an additional feature, the recognition principle. The recognition cue indicates whether the city is recognized or not. A city that is recognized is preferred to an unrecognized one. The recognition cue is always queried first and, hence, not relevant for the problem of finding optimal cue permutations considered here.

problem called Minimum Incorrect Lexicographic Strategy denoting the task of minimizing the number of incorrect inferences for the lexicographic strategy on a given list of pairs. Many computational problems are known to be NP-complete but have efficient approximation algorithms that are good in the sense that their solutions are never more than some constant factor away from the optimum. Problems in this class, which is denoted APX, are generally considered to be approximable well and efficiently (Ausiello et al., 1999). As the second major result of this article we show that, unless P = NP, no polynomial-time approximation algorithm exists that computes solutions for Minimum Incorrect Lexicographic Strategy that are only a constant factor worse than the optimum, unless P = NP. In other words, the approximating factor, also called performance ratio, must grow with the size of the problem.

As an extension of the class of fast and frugal heuristics we introduce an algorithm for finding cue permutations that has not been considered within the context of bounded rationality. This algorithm is based on the greedy method, a principle widely used in algorithm design. The greedy algorithm runs in polynomial time and we derive tight bounds for it, showing that it approximates the optimum with a performance ratio proportional to the number of cues. An important consequence of this result is a guarantee that for those instances which have a solution that discriminates all pairs correctly, the greedy algorithm always finds a permutation attaining this minimum. We are not aware that this quality has been established for any of the previously studied heuristics for paired comparison. Moreover, we show that TTB does not have this property, concluding that the greedy method of constructing cue permutations performs provably better than TTB.

While the results mentioned so far deal with lexicographic strategies based on cue permutations, we further consider the possibility to build them by also inverting cues. We present an algorithm that greedily constructs cue inversions that are always correct on a number of pairs that is at least half the optimum. In other words, this algorithm is a constant factor approximation algorithm for the problem of maximizing the number of correct inferences. Interestingly, this algorithm does not even need to permute any cues to approximate to within a constant factor the optimum taken even over all inversions and permutations.

## 1.3 Learning

Lexicographic Strategy is a decision problem that requires to minimize a disagreement. Given a set of pairs, the question is whether a cue permutation can be found that keeps the number of incorrect comparisons, or disagreements, of the lexicographic strategy below some prescribed value. Minimizing disagreement problems play a major role in the context of a computational model of learning known as agnostic probably approximately correct (PAC) learning (see, e.g., Anthony and Bartlett, 1999). This model assumes that a learner receives a

set of examples, the sample, drawn according to some unknown probability distribution. The learner is required to output a function from a so-called hypothesis class on the condition that, with high probability, the computed hypothesis is, with respect to the distribution, close to an optimal hypothesis within the class. A fundamental result is concerned with the question whether agnostic PAC learning with a given hypothesis class can be done efficiently, in particular, if there exists an algorithm that needs only a polynomial number of computation steps to find good hypotheses. The result states that no such learner can exist if the minimizing disagreement problem for the hypothesis class is NP-complete, given that the complexity classes RP and NP are different (see, e.g., Höffgen et al., 1995; Kearns et al., 1994).

The results in this paper have immediate consequences for the question whether lexicographic strategies can be learned. Adopting the framework of agnostic PAC learning, we assume that pairs of cue profiles are drawn randomly according to some unknown distribution. The task of the learner is to find a cue permutation that, with high probability, is close to an optimal one, where closeness means that the probability of differing inferences is small. This setting seems slightly different from the original PAC model as in the latter the sample consists of labeled examples, whereas the lexicographic strategy has to be learned from pairs. However, relevant in both cases is that a hypothesis can be correct or incorrect on a given example. Therefore, applying the above-mentioned result about agnostic PAC learning and assuming that RP $\neq$ NP, by showing that Lexicographic Strategy is NP-complete we may conclude that efficient learning of lexicographic strategies is impossible. Moreover, this evidence of impossibility is reinforced by our proving that the optimization problem Minimum Incorrect Lexicographic Strategy cannot be approximated in polynomial time to within any constant factor.

A further question that models of learning are involved in is the characterization of the ability to generalize, that is, to find a good hypothesis from only a small number of examples. A principal result in agnostic PAC learning has established a combinatorial parameter of a hypothesis class, its Vapnik-Chervonenkis (VC) dimension, as the relevant measure for this sample complexity (Vapnik and Chervonenkis, 1971). In particular, to come close to the minimal generalization error it is necessary and sufficient to draw a number of examples that is proportional to the VC dimension of the hypothesis class (see, e.g., Anthony and Bartlett, 1999). In this article we determine the VC dimension of the class of lexicographic strategies exactly. In detail, we show that the class of lexicographic strategies obtained by cue permutations and inversions has a VC dimension equal to the number of cues. As a consequence, the number of cues provides a tight bound on the sample complexity for learning lexicographic strategies.

## 1.4 Related Work

Research that approaches the investigation of simple heuristics for intelligent systems via the analysis of computational complexity traces back to Simon and Kadane (1975, 1976). They provided sufficient conditions under which so-called satisficing search strategies can be proved to be optimal. Their line of study was resumed by Greiner and Orponen (1996) who obtained estimates for the sample complexity of such strategies. Regarding the issue of ordering, Greiner (1999) raised a question relevant for inductive logic programming that is similar to the problems studied here. He asked whether it is possible to efficiently revise rule-based programs by rearranging the ordering of the rules. His results include NP-completeness and nonapproximability statements for various types of logical theories.

Rivest (1987) introduced decision lists as a formalism for the representation of Boolean functions. The procedure for computing the output value of a decision list is similar to a lexicographic strategy in that both mechanisms are based on one-reason decision making. In fact, we shall show below that lexicographic strategies are a special case of so-called 2-decision lists. It will also follow from this result that the two function classes do not coincide. Thus, an algorithm that learns 2-decision lists does not necessarily learn lexicographic strategies. On the other hand, an algorithm that finds optimal cue permutations might not be good in constructing 2-decision lists.

Ordering problems have also been studied by Cohen et al. (1999). They considered the problem of putting a set of objects in a total order that maximally agrees with a specified preference function. They proved this problem to be NP-complete. We shall show later that the problem of finding cue permutations for the lexicographic strategy can be formulated as such an ordering problem. However, we shall also argue that the two problems are different, since the cue permutation problem requires the total order to be implemented as a lexicographic strategy and not every total order can be represented this way.

## 1.5 Outline

We introduce lexicographic strategies in Section 2 and provide there further definitions and properties. We then draw comparisons with decision lists and discuss the relationship of the problem of finding optimal cue permutations with the ordering problem studied by Cohen et al. (1999).

Section 3 establishes the NP-completeness of the problem Lexicographic Strategy. Additionally, we consider the complexity of this problem when the instances meet certain conditions. We obtain that the problem remains NP-complete under constraints that require the cue profiles to be sparse, impose a bound on the number of pairs, or suppose the pairs to satisfy some simple properties of orderings. In particular, we show NP-completeness to hold when

each cue profile contains no more than one 0. In contrast, if the latter condition is met and the pairs are from some partial order, the problem can be solved in linear time.

The optimization problem Minimum Incorrect Lexicographic Strategy is considered in Section 4. As the main result we show that this problem cannot be approximated in polynomial time to within any constant factor, unless P = NP. It further emerges, that this result holds even when the instances satisfy some, albeit not all, of the restrictions considered in Section 3.

Section 5 introduces the greedy algorithm for constructing cue permutations. We tightly determine the performance ratio of this algorithm, showing that it is proportional to the number of cues. The result implies that the greedy method always finds a correct cue permutation if one exists. In contrast, we show that this does not hold for TTB. Restrictions under which the lower bound for the greedy method is still valid are also determined in this section.

In Section 6 we introduce the operation of inverting cues as a means for constructing lexicographic strategies. We show that a greedy method approximates the maximum number of correct inferences to within a constant factor.

The sample complexity for learning is studied in Section 7. We determine the number of cues as the exact value for the VC dimension of the class of lexicographic strategies obtained from cue permutations and inversions. Section 8 summarizes seven major open questions arising from this article and Section 9 concludes with final remarks.

We assume that the reader is acquainted with the theory of NP-completeness as propounded, for instance, by Garey and Johnson (1979). Familiarity with the theory of computational complexity for approximation problems is not required as we shall explicate the necessary details.

**Bibliographic Note.** The main result of Section 3 (Theorem 3) was mentioned by Martignon and Schmitt (1999), but its proof has been available only in an unpublished manuscript (Schmitt and Martignon, 1999). Parts of Sections 4 and 5 appear in a contribution to a conference (Schmitt and Martignon, 2006).

# 2 Lexicographic Strategies

In the following, we introduce lexicographic strategies and the computational problem that we study in this article. After giving formal definitions in Section 2.1, we compare in Section 2.2 lexicographic strategies with a related formalism known as decision lists. The optimization problem for lexicographic strategies bears some resemblance to ranking problems that have been studied earlier. In Section 2.3, we discuss the relationship between them and demonstrate that they are different problems.

## 2.1 Definitions

A *lexicographic strategy* is a method for comparing elements of a set $B \subseteq \{0,1\}^n$ of Boolean vectors. Each component $1, \ldots, n$ of these vectors is referred to as a *cue*. Given two elements $a, b \in B$, where $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, the lexicographic strategy searches for the smallest cue index $i \in \{1, \ldots, n\}$ such that $a_i$ and $b_i$ are different. The strategy then outputs one of " $<$ " or " $>$ " according to whether $a_i < b_i$ or $a_i > b_i$ assuming the usual order $0 < 1$ of the truth values. If no such cue exists, the strategy returns " $=$ ". Formally, let $\mathrm{diff} : B \times B \to \{1, \ldots, n+1\}$ be the function where $\mathrm{diff}(a, b)$ is the smallest cue index on which $a$ and $b$ are different, or $n+1$ if they are equal, that is,

$$\mathrm{diff}(a, b) \;=\; \min\{\{i : a_i \neq b_i\} \cup \{n+1\}\}.$$

Then, the function $S : B \times B \to \{\text{" } < \text{ "}, \text{" } = \text{ "}, \text{" } > \text{ "}\}$ computed by the lexicographic strategy is

$$S(a, b) \;=\; \begin{cases} \text{" } < \text{ "} & \text{if } \mathrm{diff}(a,b) \leq n \text{ and } a_{\mathrm{diff}(a,b)} < b_{\mathrm{diff}(a,b)}, \\ \text{" } > \text{ "} & \text{if } \mathrm{diff}(a,b) \leq n \text{ and } a_{\mathrm{diff}(a,b)} > b_{\mathrm{diff}(a,b)}, \\ \text{" } = \text{ "} & \text{otherwise.} \end{cases}$$

Considering $a$ and $b$ as binary encodings of natural numbers, $S(a, b)$ is nothing else than the result of the comparison of these two numbers.

Lexicographic strategies may take into account that the cues come in an order that is different from $1, \ldots, n$. Let $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be a permutation of the cues. It gives rise to a mapping $\overline{\pi} : \{0,1\}^n \to \{0,1\}^n$ that permutes the components of Boolean vectors by $\overline{\pi}(a_1, \ldots, a_n) = (a_{\pi(1)}, \ldots, a_{\pi(n)})$. As $\overline{\pi}$ is uniquely defined given $\pi$, we simplify the notation and write also $\pi$ for $\overline{\pi}$. The *lexicographic strategy under cue permutation* $\pi$ passes through the cues in the order $\pi(1), \ldots, \pi(n)$, that is, it computes the function $S_\pi : B \times B \to \{\text{" } < \text{ "}, \text{" } = \text{ "}, \text{" } > \text{ "}\}$ defined as

$$S_\pi(a, b) \;=\; S(\pi(a), \pi(b)).$$

The problem we study is that of finding a cue permutation that minimizes the number of incorrect comparisons in a given list of element pairs using the lexicographic strategy. An instance of this problem consists of a set $B$ of elements and a set of pairs $L \subseteq B \times B$. Each pair $\langle a, b\rangle \in L$ represents an inequality $a \leq b$. Given a cue permutation $\pi$, we say that the lexicographic strategy under $\pi$ *infers* the pair $\langle a, b\rangle$ *correctly* if $S_\pi(a, b) \in \{\text{" } < \text{ "}, \text{" } = \text{ "}\}$, otherwise the inference is incorrect. The task is to find a permutation $\pi$ such that the number of incorrect inferences in $L$ using $S_\pi$ is minimal, that is, a permutation $\pi$ that minimizes

$$\mathrm{INCORRECT}(\pi, L) \;=\; |\{\langle a, b\rangle \in L : S_\pi(a, b) = \text{" } > \text{ "}\}|.$$

We recall some definitions about orders on sets. A set $L \subseteq B \times B$ is a *partial order* if it is reflexive (that is, $\langle a, a\rangle \in L$ for every $a \in B$), antisymmetric (that is, $\langle a, b\rangle \in L$ and $\langle b, a\rangle \in L$ implies $a = b$), and transitive (that is, $\langle a, b\rangle \in L$ and $\langle b, c\rangle \in L$ implies $\langle a, c\rangle \in L$). Further, $L$ is a *total order* if it is a partial order and satisfies $\langle a, b\rangle \in L$ or $\langle b, a\rangle \in L$ for every $a, b \in B$. Finally, $L$ is *irreflexive* if $\langle a, a\rangle \notin L$ for every $a \in B$.

Given some cue permutation $\pi$, consider a relation that is satisfied by a pair $\langle a, b\rangle$ if and only if $S_\pi(a, b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\}$. Clearly, this relation defines a total order on any set $B \subseteq \{0, 1\}^n$. A question that arises immediately is whether every total order has some cue permutation that represents this order using the lexicographic strategy. It is easy to see that this is not the case.

**Proposition 1.** *For every set $B \subseteq \{0, 1\}^n$ and every cue permutation $\pi$, the lexicographic strategy under cue permutation $\pi$ defines a total order on $B$. On the other hand, there are sets $B \subseteq \{0, 1\}^n$ with a total order that cannot be represented by any cue permutation.*

*Proof.* It is evident that the relation $\{(a, b) : S_\pi(a, b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\}\}$ is a total order. As a counterexample, consider a set $B$ with $\{(0, \ldots, 0), (1, \ldots, 1)\} \subseteq B$. Clearly, under every cue permutation, $(0, \ldots, 0)$ is less than $(1, \ldots, 1)$. Thus, the reverse ordering of these two elements cannot be represented by the lexicographic strategy. □

Obviously, the lexicographic strategy applied to a pair $\langle a, a\rangle$ is always correct, independently of the cue permutation. Therefore, the identical pairs of $L$ pose no obstacle for the minimization problem. Also possible were an alternative setting where $\langle a, b\rangle$ is interpreted as a strict inequality. We admit identical pairs, however, to keep the definition more general and allow $L$ to represent some “natural” relations such as partial or total orders or arbitrary subsets thereof. Nevertheless, all results presented in the following remain valid if the pairs are assumed to represent strict inequalities.

## 2.2 Lexicographic Strategies and Decision Lists

Decision lists are computing formalisms that operate quite similar to lexicographic strategies. A *decision list* represents a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ and is given by a list of pairs

$$(m_1, r_1), \ldots, (m_\ell, r_\ell),$$

where each $m_i$ is a Boolean monomial, that is, a conjunction of Boolean variables with or without negations (Rivest, 1987). Further, each $r_i$ is 0 or 1, and $m_\ell$ is the constant function 1. The Boolean function computed by the decision list is defined as follows: Given some $a \in \{0, 1\}^n$, the output value is $r_i$ where $i$ is the

smallest index such that $m_i$ evaluates to 1 on $a$. A $k$-decision list is a decision list where every monomial has size at most $k$.

In the problem of minimizing the number of incorrect comparisons the relevant question is whether the output of the lexicographic strategy is correct, and not whether it is particularly one of " $<$ ", " $=$ ", or " $>$ ". In other words, we are interested in a binary and not a ternary classification. Thus, we may consider the lexicographic strategy $S$ as a Boolean function $f$ mapping a set $L$ of pairs to $\{0,1\}$, where for every $\langle a,b\rangle \in L$ we have

$$f(a,b) = 1 \quad \text{if and only if} \quad S(a,b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\}.$$

Seen in this light, lexicographic strategies exhibit a similarity to decision lists. The following statement, which is easy to derive, makes this relationship precise.

**Proposition 2.** *Let $f : \{0,1\}^{2n} \to \{0,1\}$ be a Boolean function with variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. Then $f$ is computed by the lexicographic strategy if and only if $f$ is computed by the 2-decision list*

$$(x_1\overline{y}_1, 0), (\overline{x}_1 y_1, 1), \ldots, (x_n\overline{y}_n, 0), (\overline{x}_n y_n, 1), (1,1).$$

*Proof.* Let $a, b \in \{0,1\}^n$. Clearly, if $a = b$, all monomials of the decision list evaluate to 0, except for the constant function 1. If $a \neq b$, let $i = \mathrm{diff}(a,b)$. In the case that $a_i < b_i$, the monomial $\overline{x}_i y_i$ is the first one that evaluates to 1, and the output of the decision list is 1. Similarly, if $a_i > b_i$, this is first detected by the monomial $x_i\overline{y}_i$, and the decision list yields 0. □

The proposition shows that the lexicographic strategy has a unique characterization as a 2-decision list. Thus, finding a cue permutation for the lexicographic strategy amounts to constructing a 2-decision list with some restrictions concerning the structure of the monomials, the pattern of the output values, and the length of the list. It is also obvious from Proposition 2, however, that 2-decision lists compute a much richer class of Boolean functions than lexicographic strategies do. We conclude that cue permutations are not necessarily found using algorithms for constructing 2-decision lists. Further, an optimal cue permutation might not be an optimal 2-decision list.

## 2.3 Ranking Problems

The problem of minimizing the number of incorrect comparisons in a list of pairs exhibits some similarity with an optimization problem that occurs in the context of ordering problems and was studied by Cohen et al. (1999). In this problem, which we here call ranking problem, one receives a set $X$, a collection of functions $R_1, \ldots, R_N$ mapping $X \times X$ to the real interval $[0,1]$, and rational numbers $w_1, \ldots, w_N \in [0,1]$ whose sum is equal to 1. A solution of the problem is a total order $\rho$ of $X$ that maximally agrees with the so-called preference function

PREF : $X \times X \to [0, 1]$. The closer the value of PREF$(a, b)$ is to 1, the more $a$ is to be ranked above $b$. The preference function is defined as

$$\text{PREF}(a, b) \;=\; \sum_{i=1}^{N} w_i R_i(a, b)$$

The agreement of the total order $\rho$ with the preference function PREF is quantified by the value of

$$\sum_{\{(a,b):\rho(a)>\rho(b)\}} \text{PREF}(a, b) \tag{1}$$

and a desired total order $\rho$ is one that maximizes this value.

It is not hard to see that the instances of the cue permutation problem are particular instances of the above problem. Specifically, introduce for each pair $\langle a, b\rangle$ a function $R_{\langle a,b\rangle} : B \times B \to \{0, 1\}$ that outputs 1 on $(b, a)$, and 0 otherwise. Further, let $w_{\langle a,b\rangle} = 1/|L|$. Then, a total order $\rho$ that maximizes the value of the expression (1) is one that minimizes the number of incorrect inferences in $L$.

Cohen et al. (1999) have shown that the ranking problem is NP-complete. The question is, therefore, whether this hardness result has any implications on the complexity of finding a cue permutation that minimizes the number of incorrect inferences. However, the ranking problem is different from the cue permutation problem not only in that its instances are more general. The two problems also disagree in the type of solutions that are sought. While the ranking problem accepts any total order that maximizes the agreement with the preference function, the cue permutation problem requires that the total order can be implemented by a lexicographic strategy. Proposition 1 demonstrates, though, that not every total order can be represented as a cue permutation. Thus, the space taken by the solutions of the cue permutation problem is narrower than the solution space for the ranking problem described above. Moreover, we show in Section 3 that the cue permutation problem remains NP-complete even when the instances are known to have a total order. In contrast, imposing this restriction on the ranking problem results in a problem that is trivially solvable.

A further difference emerges if one considers the problem of approximating optimal solutions as we do in Section 4. Then the cue permutation problem is a minimization problem while the ranking problem is a maximization problem. Among the complexity classes of approximation problems several examples are known where the minimization and the maximization problem have different degrees of approximability (see, e.g., Amaldi and Kann, 1995, 1998). Consequently, despite the apparent similarity of the cue permutation problem and the ranking problem, the complexities of the two problems are obviously not related.

# 3 Complexity of Finding Optimal Cue Permutations

We consider the complexity of the problem to minimize the number of incorrect inferences under the lexicographic strategy. To show that it is computationally intractable, we formulate this search problem as a decision problem. The decision problem has as input a set of binary vectors, an ordering defined on this set in terms of a list of vector pairs, and a bound given as a natural number. The question is to decide whether the cues can be permuted such that the number of incorrect inferences made by the lexicographic strategy when applied with this cue permutation to the list of pairs is not larger than the given bound. We call this decision problem Lexicographic Strategy.

Lexicographic Strategy
Instance: A set $B \subseteq \{0,1\}^n$, a set $L \subseteq B \times B$, and a natural number $k$.
Question: Is there a permutation of the cues of $B$ such that the number of incorrect inferences in $L$ under the lexicographic strategy is at most $k$?

Clearly, any polynomial-time algorithm for finding a permutation with a minimal number of incorrect inferences can be turned into a polynomial-time algorithm that solves Lexicographic Strategy. However, we show that this problem is NP-hard. Hence, if P $\neq$ NP, no polynomial-time algorithm for the decision problem and, a fortiori, for the search problem exists. The NP-hardness proof provides a polynomial-time reduction from a problem dealing with graphs and known as Vertex Cover (Garey and Johnson, 1979).

Vertex Cover
Instance: An undirected graph $G = (V, E)$, where $V$ is the set of vertices and $E \subseteq V \times V$ is the set of edges, and a natural number $k$.
Question: Is there a vertex cover of cardinality $k$ or less for $G$, that is, a subset $V' \subseteq V$ with $|V'| \leq k$ such that for each edge $\{u, v\} \in E$ at least one of $u$ and $v$ belongs to $V'$?

**Theorem 3.** Lexicographic Strategy *is NP-complete.*

*Proof.* Obviously, a nondeterministic algorithm can generate a permutation of the cues and verify in polynomial time whether the number of incorrect inferences is at most $k$. Thus, the problem is a member of NP. To establish its NP-hardness, we construct a reduction from Vertex Cover. Let $\mathbf{1}_i$ ($\mathbf{1}_{i,j}$) denote the $n$-bit vector with a 1 in every position except for position $i$ (positions $i$ and $j$) where it has a 0. Further, $\mathbf{1}$ is the $n$-bit vector with a 1 everywhere. Given the graph $G = (V, E)$, where the set of vertices is $V = \{v_1, \ldots, v_n\}$, we define a set $B$ of Boolean vectors with $n + 1$ cues, that is $B \subseteq \{0,1\}^{n+1}$, in three steps:

1. Let $(\mathbf{1}, 0) \in B$.

2. For $i = 1, \ldots, n$, let $(\mathbf{1}_i, 1) \in B$.

3. For every $\{v_i, v_j\} \in E$, let $(\mathbf{1}_{i,j}, 1) \in B$.

The set $L \subseteq B \times B$ of pairs that represents the element ordering is defined such that the element from step 1 is less than each element constructed in step 2, and each element arising from step 3 is less than the element from step 1. Formally,

$$L \;=\; \{\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle : i = 1, \ldots, n\} \;\cup\; \{\langle(\mathbf{1}_{i,j}, 1), (\mathbf{1}, 0)\rangle : \{v_i, v_j\} \in E\}. \quad (2)$$

Finally, we let the number $k$ in the instance of Lexicographic Strategy be the same as in the instance of Vertex Cover. Clearly, the reduction is computable in polynomial time.

We establish the correctness of the reduction by proving that the graph $G$ has a vertex cover of cardinality at most $k$ if and only if the associated instance of Lexicographic Strategy has a cue permutation that results in no more than $k$ incorrect inferences. For simplicity, let us call a pair from the first and second set on the right-hand side of equation (2) a vertex pair and an edge pair, respectively.

($\Rightarrow$) Assume that $G$ has a vertex cover $V'$ of cardinality at most $k$ and, without loss of generality, let its cardinality be exactly $k$, so that $V' = \{v_{i_1}, \ldots, v_{i_k}\}$. Further, let $V \setminus V' = \{v_{i_{k+1}}, \ldots, v_{i_n}\}$. Define the permutation of the cues as

$$i_1, \ldots, i_k, n+1, i_{k+1}, \ldots, i_n.$$

We claim that this cue ranking causes no more than $k$ incorrect inferences in $L$. Consider an arbitrary edge pair $\langle(\mathbf{1}_{i,j}, 1), (\mathbf{1}, 0)\rangle$. As $V'$ is a vertex cover, at least one of $i$ and $j$ occurs in $i_1, \ldots, i_k$. This implies that the first cue that distinguishes this pair will have value 0 in $(\mathbf{1}_{i,j}, 1)$ and value 1 in $(\mathbf{1}, 0)$. Thus, the result of the lexicographic comparison is correct. Next, let $\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle$ be a vertex pair with $v_i \notin V'$. In this case, cue $n+1$ distinguishes this pair with the correct outcome. Finally, each vertex pair $\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle$ with $v_i \in V'$ is distinguished by cue $i$ with a result different from the ordering given by $L$. In summary, the only incorrect comparisons arise from vertex pairs with $v_i \in V'$. As $V'$ has cardinality $k$, we thus have no more than $k$ incorrect inferences.

($\Leftarrow$) Now, let $\pi$ be a permutation of the cues that produces at most $k$ incorrect inferences in $L$. Define the set $V'$ of vertices as follows:

1. For every incorrect vertex pair $\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle$, let $v_i \in V'$.

2. For every incorrect edge pair $\langle(\mathbf{1}_{i,j}, 1), (\mathbf{1}, 0)\rangle$, let one of $v_i, v_j \in V'$.

Clearly, $V'$ has cardinality at most $k$. It remains to show that $V'$ is a vertex cover. For the sake of a contradiction, assume that there is an edge in $E$, say $\{v_i, v_j\}$, not covered. This means that neither of $v_i, v_j$ is in $V'$, implying that we have correct comparisons for the vertex pairs corresponding to $v_i$ and $v_j$ and for the edge pair corresponding to $\{v_i, v_j\}$. The fact that the edge pair is inferred correctly implies that $\pi$ must rank cue $i$ or $j$ before cue $n+1$. But then we have that at least one of the vertex pairs for $v_i$ and $v_j$ results in an incorrect comparison. This contradicts the assertion made above that both vertex pairs have correct comparisons. We conclude that $V'$ is a vertex cover. □

The reduction constructed in the previous proof has some properties that we exploit in the following statement to establish the NP-completeness of restricted versions of Lexicographic Strategy. First, it shows that the set $B$ can be sparse in a certain sense, that is, has elements that exhibit only very constrained bit patterns. Moreover, the NP-completeness holds even when $L$ is not much larger than $B$. Finally, the problem remains intractable even if $L$ does not contain identical pairs or has some properties of a partial or total order.

**Corollary 4.** Lexicographic Strategy *is NP-complete even when the instances satisfy any (or all) of the following constraints:*

1. *Each element of $B$ contains at most two* 0*s.*
2. *The cardinality of $L$ is linearly bounded from above by the cardinality of $B$, that is, $|L|$ is $O(|B|)$.*
3. *$L$ is irreflexive.*
4. *$L$ is a subset of some partial order.*
5. *$L$ is a subset of some total order.*

*Proof.* We show that all constraints are satisfied by the instances defined in the reduction for the proof of Theorem 3. That the first condition holds is obvious from the definition of $B$. Further, the instances of Lexicographic Strategy in this reduction all satisfy $|B| = |E|+n+1$ and $|L| = n+|E|$. Thus, $|B| = |L|-1$ and the second constraint is met. Moreover, $L$ does not contain any pair $\langle a, a\rangle$ which implies that the third constraint holds. We establish the fourth condition by checking that $L$ does not violate any of the requirements for a partial order: Clearly, each $a \neq b$ does not have both $\langle a, b\rangle$ and $\langle b, a\rangle$ in $L$, and there are no three pairs $\langle a, b\rangle, \langle b, c\rangle, \langle c, a\rangle$ in $L$. Finally, it is easy to see that $L$ is consistent with the total order resulting from the following ascending arrangement of $B$: We begin with the elements $(\mathbf{1}_{i,j}, 1)$, where $\{v_i, v_j\} \in E$, in lexicographic order, followed by the element $(\mathbf{1}, 0)$, and complete this sequence at the end by the elements $(\mathbf{1}_i, 1)$, for $i = 1, \ldots, n$, again in lexicographic order. Thus, we have an ordering where any two elements of $B$ are comparable, implying that also the last constraint is satisfied. □

The first constraint of Corollary 4 gives rise to the question whether the problem is still NP-complete if each element of $B$ has no more than one 0. The following two results treat this issue. First, we show that the problem in general remains NP-complete under this restriction. To establish this we provide a reduction from the NP-complete problem Feedback Arc Set (Garey and Johnson, 1979).

Feedback Arc Set
Instance: A directed graph $G = (V, E)$, where $V$ is the set of vertices and $A \subseteq V \times V$ is the set of arcs, and a natural number $k$.
Question: Is there a subset $A' \subseteq A$ with $|A'| \leq k$ such that $A'$ contains at least one arc from every directed cycle in $G$?

**Theorem 5.** Lexicographic Strategy *is NP-complete even when restricted to instances where each element of* $B$ *contains at most one* 0.

*Proof.* Clearly, as Lexicographic Strategy is in NP, any subproblem of it is in NP as well. We establish the NP-hardness of the problem by giving a reduction that is a simple rewriting of Feedback Arc Set. Given the graph $G = (V, A)$ with $V = \{v_1, \ldots, v_n\}$ and using the notation from the proof of Theorem 3, we let

$$\begin{aligned} B &= \{\mathbf{1}_i : i = 1, \ldots, n\}, \\ L &= \{\langle \mathbf{1}_i, \mathbf{1}_j \rangle : (v_i, v_j) \in A\}, \end{aligned}$$

and define $k$ to have the same value as in the instance of Feedback Arc Set.

Obviously, $A' \subseteq A$ contains at least one arc from every directed cycle in $G$ if and only if the graph $G' = (V, A \setminus A')$ is acyclic. Further, $G'$ is acyclic if and only if $V$ has a total ordering in which $v_i$ is less than $v_j$ for each $(v_i, v_j) \in A \setminus A'$. Finally, the existence of such a total ordering is equivalent to the assertion that $B$ has a cue permutation with no incorrect comparisons in $L' = \{\langle \mathbf{1}_i, \mathbf{1}_j \rangle : (v_i, v_j) \in A \setminus A'\}$. With this chain of equivalences, the correctness of the reduction follows from the fact that $|L'| = |L| - |A'|$. □

We may also add to the assumption of Theorem 5 the restriction that $|L|$ is linearly bounded in $|B|$, so that the problem is still NP-complete. In this case, the NP-hardness follows from the fact that Feedback Arc Set remains NP-hard for directed graphs in which the degree of the vertices is bounded by some constant (Garey and Johnson, 1979). However, if we include the constraint that $L$ is a subset of some partial order, the complexity of the problem changes drastically, as we see in the following statement.

**Corollary 6.** *The problem of finding a cue permutation with a minimal number of incorrect comparisons under the lexicographic strategy is solvable in linear time for instances where* $B$ *contains at most one* 0 *and* $L$ *is a subset of some partial order.*

*Proof.* As was argued in the proof of Theorem 5, the problem is the same as the problem of finding a total order that is consistent with the partial order given by $L$ (which is always possible). Such a total order can be constructed by topological sorting. Algorithms for this sorting problem exist that run in linear time (see, e.g., Skiena, 1997). □

It is not difficult—and we leave it to the reader—to establish dual formulations of Theorem 5 and Corollary 6 where it is assumed that each element of $B$ contains at most one 1.

# 4 Approximability of Optimal Cue Permutations

In the previous section, we have shown that there is no polynomial-time algorithm that computes optimal cue permutations for the lexicographic strategy, unless P = NP. While it follows that this problem is as difficult as all other optimization problems that have an NP-complete decision problem, we cannot draw any conclusions for the case where we are interested in solutions that are not equal to the optimum but somehow close to it. In fact, there is a large class of optimization problems that have NP-complete decision problems, but can be solved efficiently if the solution is required to be only a constant factor worse than the optimal solution. This class of problems is denoted APX (Ausiello et al., 1999).

In this section, we show that the problem of approximating the optimal cue permutation is harder than any problem in the class APX. In particular, we prove that, if P $\neq$ NP, there is no polynomial-time algorithm whose solutions yield a number of incorrect comparisons that is by at most a constant factor larger than the minimal number possible. First, however, we state the problem as an optimization problem and introduce some definitions from the complexity theory of approximation problems (Ausiello et al., 1999).

> Minimum Incorrect Lexicographic Strategy
> Instance: A set $B \subseteq \{0,1\}^n$ and a set $L \subseteq B \times B$.
> Solution: A permutation $\pi$ of the cues of $B$.
> Measure: The number of incorrect inferences in $L$ for the lexicographic strategy under cue permutation $\pi$, that is, INCORRECT$(\pi, L)$.

Given a real number $r > 0$, an algorithm is said to approximate Minimum Incorrect Lexicographic Strategy to within a factor of $r$ if for every instance $(B, L)$ the algorithm returns a permutation $\pi$ such that

$$\text{INCORRECT}(\pi, L) \quad \leq \quad r \cdot \text{opt}(L),$$

where opt($L$) is the minimal number of incorrect comparisons achievable on $L$ by any permutation. The factor $r$ is also known as the performance ratio of the algorithm. The following optimization problem plays a crucial role in the derivation of the lower bound for the approximability of Minimum Incorrect Lexicographic Strategy.

Minimum Hitting Set
Instance: A collection $C$ of subsets of a finite set $U$.
Solution: A hitting set for $C$, that is, a subset $U' \subseteq U$ such that $U'$ contains at least one element from each subset in $C$.
Measure: The cardinality of the hitting set, that is, $|U'|$.

Similarly as above, we say that an algorithm approximates Minimum Hitting Set to within a factor of $r$ if for every instance $C$ the algorithm outputs a hitting set $U'$ that satisfies

$$|U'| \;\leq\; r \cdot \mathrm{opt}(C),$$

where opt($C$) denotes the minimal cardinality of a hitting set for $C$. (For simplicity, we use opt($\cdot$) to represent the value of an optimal solution in both problems. It shall be clear from the context to which problem it refers.)

Minimum Hitting Set is equivalent to a problem called Minimum Set Cover in the sense that every polynomial-time algorithm that approximates Minimum Hitting Set to within a certain factor can be turned into a polynomial-time algorithm that approximates Minimum Set Cover to within the same factor, and vice versa (Ausiello et al., 1980). Bellare et al. (1993) have shown that Minimum Set Cover cannot be approximated in polynomial time to within any constant factor, unless P = NP. Thus, if P $\neq$ NP, Minimum Hitting Set cannot be approximated in polynomial time to within any constant factor as well. We make use of this fact when we establish the lower bound for the approximability of the optimal cue permutation.

**Theorem 7.** *For every $r$, there is no polynomial-time algorithm that approximates* Minimum Incorrect Lexicographic Strategy *to within a factor of $r$, unless* P = NP.

*Proof.* We use the main ideas from the proof of Theorem 3 to establish an approximation preserving reduction, or AP-reduction, from Minimum Hitting Set to Minimum Incorrect Lexicographic Strategy.[3] (See Ausiello

[3]A proof of Theorem 3 can be obtained by employing this reduction as a reduction between decision problems, from the NP-complete Hitting Set to Lexicographic Strategy. However, the reduction used in the proof of Theorem 3 is more powerful since Corollary 4 cannot be inferred when reducing from Hitting Set.

et al., 1999, for a definition of the AP-reduction.) This reduction entails that every polynomial-time algorithm that approximates Minimum Incorrect Lexicographic Strategy to within some constant factor can be turned into a polynomial-time algorithm that approximates Minimum Hitting Set to within the same constant factor. Then the statement follows from the equivalence of Minimum Hitting Set to Minimum Set Cover and the lower bound on the approximability of the latter (Bellare et al., 1993).

We first define a function $f$ that is computable in polynomial time and maps each instance of Minimum Hitting Set to an instance of Minimum Incorrect Lexicographic Strategy. Let $\mathbf{1}$ denote the $n$-bit vector with a 1 everywhere and $\mathbf{1}_{i_1,\ldots,i_\ell}$ the vector with 0 in positions $i_1, \ldots, i_\ell$ and 1 elsewhere. Given the collection $C$ of subsets of the set $U = \{u_1, \ldots, u_n\}$, the function $f$ maps $C$ to $(B, L)$, where $B \subseteq \{0,1\}^{n+1}$ is defined as follows:

1. Let $(\mathbf{1}, 0) \in B$.
2. For $i = 1, \ldots, n$, let $(\mathbf{1}_i, 1) \in B$.
3. For every $\{u_{i_1}, \ldots, u_{i_\ell}\} \in C$, let $(\mathbf{1}_{i_1,\ldots,i_\ell}, 1) \in B$.

Further, the set $L$ is constructed as

$$L = \{\langle(\mathbf{1},0),(\mathbf{1}_i,1)\rangle : i = 1,\ldots,n\} \cup \{\langle(\mathbf{1}_{i_1,\ldots,i_\ell},1),(\mathbf{1},0)\rangle : \{u_{i_1},\ldots,u_{i_\ell}\} \in C\} \quad (3)$$

In the following, a pair from the first and second set on the right-hand side of equation (3) is referred to as an element pair and a subset pair, respectively. Obviously, the function $f$ is computable in polynomial time. It has the following property.

**Claim 1.** *Let $f(C) = (B, L)$. If $C$ has a hitting set of cardinality $k$ or less then $f(C)$ has a cue permutation $\pi$ where* $\mathrm{INCORRECT}(\pi, L) \leq k$.

To prove this, assume without loss of generality that $C$ has a hitting set $U'$ of cardinality exactly $k$, say $U' = \{u_{j_1}, \ldots, u_{j_k}\}$, and let $U \setminus U' = \{u_{j_{k+1}}, \ldots, u_{j_n}\}$. Then the cue permutation

$$j_1, \ldots, j_k, n+1, j_{k+1}, \ldots, j_n.$$

results in no more than $k$ incorrect inferences in $L$. Indeed, consider an arbitrary subset pair $\langle(\mathbf{1}_{i_1,\ldots,i_\ell}, 1), (\mathbf{1}, 0)\rangle$. To not be an error, one of $i_1, \ldots, i_\ell$ must occur in the hitting set $j_1, \ldots, j_k$. Hence, the first cue that distinguishes this pair has value 0 in $(\mathbf{1}_{i_1,\ldots,i_\ell}, 1)$ and value 1 in $(\mathbf{1}, 0)$, resulting in a correct comparison. Further, let $\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle$ be an element pair with $u_i \notin U'$. This pair is distinguished correctly by cue $n + 1$. Finally, each element pair $\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle$ with $u_i \in U'$ is distinguished by cue $i$ with a result that disagrees with the ordering given

by $L$. Thus, only element pairs with $u_i \in U'$ yield incorrect comparisons and subset pairs are inferred correctly. Hence, the number of incorrect inferences is not larger than $|U'|$.

Next, we define a polynomial-time computable function $g$ that maps each collection $C$ of subsets of a finite set $U$ and each cue permutation $\pi$ for $f(C)$ to a subset of $U$. Given that $f(C) = (B, L)$, the set $g(C, \pi) \subseteq U$ is defined as follows:

1. For every element pair $\langle(\mathbf{1}, 0), (\mathbf{1}_i, 1)\rangle \in L$ that is compared incorrectly by $\pi$, let $u_i \in g(C, \pi)$.

2. For every subset pair $\langle(\mathbf{1}_{i_1,\ldots,i_\ell}, 1), (\mathbf{1}, 0)\rangle \in L$ that is compared incorrectly by $\pi$, let one of the elements $u_{i_1}, \ldots, u_{i_\ell} \in g(C, \pi)$.

Clearly, the function $g$ is computable in polynomial time. It satisfies the following condition.

**Claim 2.** *Let $f(C) = (B, L)$. If* INCORRECT$(\pi, L) \leq k$ *then $g(C, \pi)$ is a hitting set of cardinality $k$ or less for $C$.*

Obviously, if INCORRECT$(\pi, L) \leq k$ then $g(C, \pi)$ has cardinality at most $k$. To show that it is a hitting set, assume the subset $\{u_{i_1}, \ldots, u_{i_\ell}\} \in C$ is not hit by $g(C, \pi)$. Then neither of $u_{i_1}, \ldots, u_{i_\ell}$ is in $g(C, \pi)$. Hence, we have correct comparisons for the element pairs corresponding to $u_{i_1}, \ldots, u_{i_\ell}$ and for the subset pair corresponding to $\{u_{i_1}, \ldots, u_{i_\ell}\}$. As the subset pair is distinguished correctly, one of the cues $i_1, \ldots, i_\ell$ must be ranked before cue $n+1$. But then at least one of the element pairs for $u_{i_1}, \ldots, u_{i_\ell}$ yields an incorrect comparison. This contradicts the assertion that the comparisons for these element pairs are all correct. Thus, $g(C, \pi)$ is a hitting set and the claim is established.

Assume now that there exists a polynomial-time algorithm $A$ that approximates Minimum Incorrect Lexicographic Strategy to within a factor of $r$. Consider the algorithm that, for a given instance $C$ of Minimum Hitting Set as input, calls algorithm $A$ with input $(B, L) = f(C)$, and returns $g(C, \pi)$ where $\pi$ is the output provided by $A$. Clearly, this new algorithm runs in polynomial time. We show that it approximates Minimum Hitting Set to within a factor of $r$. By the assumed approximation property of algorithm $A$, we have

$$\text{INCORRECT}(\pi, L) \;\leq\; r \cdot \text{opt}(L).$$

Together with Claim 2, this implies that $g(\pi, C)$ is a hitting set for $C$ satisfying

$$|g(C, \pi)| \;\leq\; r \cdot \text{opt}(L).$$

From Claim 1 we obtain $\text{opt}(L) \leq \text{opt}(C)$ and, thus,

$$|g(C, \pi)| \;\leq\; r \cdot \text{opt}(C).$$

Thus, the proposed algorithm for MINIMUM HITTING SET violates the approximation lower bound that holds for this problem under the assumption P $\neq$ NP. This proves the statement of the theorem. □

Similarly as in Corollary 4 we can state a stronger version of Theorem 7 that takes restrictions into account that may hold for the instances of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY. The proof is obtained in the same way as the proof of Corollary 4 and not given here.

**Corollary 8.** *If* P $\neq$ NP*, then for every $r$ there is no polynomial-time algorithm that approximates* MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY *to within a factor of $r$, even when the instances satisfy any (or all) of the following constraints:*

1. *The cardinality of $L$ is linearly bounded from above by the cardinality of $B$, that is, $|L|$ is $O(|B|)$.*
2. *$L$ is irreflexive.*
3. *$L$ is a subset of some partial order.*
4. *$L$ is a subset of some total order.*

The reader may have noticed that the constraint of Corollary 4 that imposes a bound on the number of 0s in the elements of $B$ is missing here. In fact, there is some evidence, that the construction of an approximation preserving reduction from MINIMUM HITTING SET to this subproblem of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY is difficult or even impossible. The case where the number of 0s is bounded by some constant corresponds to the subproblem of MINIMUM HITTING SET where the cardinality of each subset is not larger than a constant. This restricted version of MINIMUM HITTING SET is known to be approximable to within some constant factor (Bar-Yehuda and Even, 1981; Hochbaum, 1982). Of course, this apparent relationship does not prove anything about the complexity of approximating the subproblem of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY. However, it gives reason to the conjecture that this subproblem might have a constant-factor approximation algorithm.

# 5 Greedy Approximation of Optimal Cue Permutations

The so-called greedy approach to the solution of a computation or approximation problem is helpful when it is not known which algorithm performs best. This simple heuristic often provides satisfactory solutions in many situations in practice. The algorithm GREEDY CUE PERMUTATION that we introduce here is based on

**Algorithm 1** GREEDY CUE PERMUTATION

**Input:** a set $B \subseteq \{0,1\}^n$ and a set $L \subseteq B \times B$
**Output:** a cue permutation $\pi$ for $n$ cues
  $I := \{1, \ldots, n\}$;
  **for** $i = 1, \ldots, n$ **do**
    let $j \in I$ be a cue where $\mathrm{INCORRECT}(j, L) = \min_{j' \in I} \mathrm{INCORRECT}(j', L)$;
    $\pi(i) := j$;
    $I := I \setminus \{j\}$;
    $L := L \setminus \{\langle a, b\rangle : a_j \neq b_j\}$
  **end for**.

the greedy method. The idea is to select the first cue according to which single cue makes a minimum number of incorrect inferences (choosing one arbitrarily if there are two or more). After that the algorithm removes those pairs that are distinguished by the selected cue, which is reasonable as the distinctions drawn by this cue cannot be undone by later cues. This procedure is then repeated on the set of pairs left. The description of GREEDY CUE PERMUTATION is given as Algorithm 1. It employs an extension of the function INCORRECT, first defined in Section 2.1, applicable also to single cues, such that for a cue $i$ we say

$$\mathrm{INCORRECT}(i, L) \;=\; |\{\langle a, b\rangle \in L : a_i > b_i\}|.$$

It is evident that Algorithm 1 runs in polynomial time, but how good is it? The least one should demand from a good heuristic is that, whenever a minimum of zero is attainable, it finds such a solution. This is indeed the case with GREEDY CUE PERMUTATION as we show in the following result. Moreover, a general performance ratio for the approximation of the optimum is asserted here.

**Theorem 9.** *The algorithm* GREEDY CUE PERMUTATION *approximates* MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY *to within a factor of $n$, where $n$ is the number of cues. In particular, it always finds a cue permutation with no incorrect inferences if one exists.*

*Proof.* We show by induction on $n$ that the permutation returned by the algorithm makes a number of incorrect inferences no larger than $n \cdot \mathrm{opt}(L)$. If $n = 1$, the optimal cue permutation is definitely found.

Let $n > 1$. Clearly, as the incorrect inferences of a cue cannot be reversed by other cues, there is a cue $j$ with

$$\mathrm{INCORRECT}(j, L) \leq \mathrm{opt}(L).$$

The algorithm selects such a cue in the first round of the loop. During the rest of the rounds, a permutation of $n-1$ cues is constructed for the set of remaining

$\langle$ 001 , 010 $\rangle$
$\langle$ 010 , 100 $\rangle$
$\langle$ 010 , 101 $\rangle$
$\langle$ 100 , 111 $\rangle$

Figure 1: A set of lexicographically ordered pairs with nondecreasing cue validities $(1, 1/2,$ and $2/3)$. The cue ordering of TTB $(1, 3, 2)$ causes an incorrect inference on the first pair. By Theorem 9, Greedy Cue Permutation finds the lexicographic ordering.

pairs. Let $j$ be the cue that is chosen in the first round, $I' = \{1, \ldots, j-1, j+1, \ldots, n\}$, and $L' = L \setminus \{\langle a, b\rangle : a_j \neq b_j\}$. Further, let $\mathrm{opt}_{I'}(L')$ denote the minimum number of incorrect inferences taken over the permutations of $I'$ on the set $L'$. Then, we observe that

$$\mathrm{opt}(L) \;\geq\; \mathrm{opt}(L') \;=\; \mathrm{opt}_{I'}(L').$$

The inequality is valid because of $L \supseteq L'$. (Note that $\mathrm{opt}(L')$ refers to the minimum taken over the permutations of all cues.) The equality holds as cue $j$ does not distinguish any pair in $L'$. By the induction hypothesis, rounds 2 to $n$ of the loop determine a cue permutation $\pi'$ with $\mathrm{INCORRECT}(\pi', L') \leq (n-1) \cdot \mathrm{opt}_{I'}(L')$. Thus, the number of incorrect inferences made by the permutation $\pi$ finally returned by the algorithm satisfies

$$\mathrm{INCORRECT}(\pi, L) \;\leq\; \mathrm{INCORRECT}(j, L) + (n-1) \cdot \mathrm{opt}_{I'}(L'),$$

which is, by the inequalities derived above, not larger than $\mathrm{opt}(L) + (n-1) \cdot \mathrm{opt}(L)$ as stated. □

The special property of Greedy Cue Permutation that it always finds the minimum if this has value zero is not owned by TTB as demonstrated by the following result.

**Corollary 10.** *On inputs that have a cue ordering without incorrect comparisons under the lexicographic strategy,* Greedy Cue Permutation *can be better than TTB.*

*Proof.* Figure 1 shows a set of four lexicographically ordered pairs. According to Theorem 9, Greedy Cue Permutation comes up with the given permutation of the cues. The validities are $1, 1/2,$ and $2/3$. Thus, TTB ranks the cues as $1, 3, 2$ whereupon the first pair is inferred incorrectly. □

Next, we consider lower bounds on the performance ratio of Greedy Cue Permutation. We obtain bounds in terms of $n$ and $|L|$. It emerges in particular that the upper bound obtained in Theorem 9 is optimal up to the factor 2.

$\langle$ 000001 , 000000 $\rangle$
$\langle$ 100001 , 100000 $\rangle$
$\langle$ 010000 , 000001 $\rangle$
$\langle$ 001000 , 000001 $\rangle$
$\langle$ 000100 , 000001 $\rangle$
$\langle$ 000010 , 000001 $\rangle$

Figure 2: A set of pairs providing a lower bound on the performance ratio of Greedy Cue Permutation (Theorem 11).

**Theorem 11.** *The performance ratio of* Greedy Cue Permutation *is at least*

$$\max\{n/2, |L|/2\}.$$

*Proof.* We show how to construct for every $n$ an instance on which Greedy Cue Permutation has the claimed performance ratio. Let $B = \{a^{(0)}, \ldots, a^{(n)}, b\} \subseteq \{0,1\}^n$ be the set where $a^{(0)} = (0, \ldots, 0)$, $b = (1, 0, \ldots, 0, 1)$, and $a^{(i)}$, for $i = 1, \ldots, n$, is the vector with a 1 in position $i$ and 0 elsewhere. The set $L \subseteq B \times B$ is defined as

$$L \;=\; \{\langle a^{(n)}, a^{(0)}\rangle, \langle b, a^{(1)}\rangle\} \cup \{\langle a^{(i)}, a^{(n)}\rangle : i = 2, \ldots, n-1\}.$$

Figure 2 shows the set $L$ for the case $n = 6$. As can be seen, cue 1 is correct on all pairs, cue $n$ is incorrect on two pairs, and every cue $j \in \{2, \ldots, n-1\}$ satisfies $\text{INCORRECT}(j, L) = 1$. Hence, Greedy Cue Permutation selects cue 1 as the first cue. As this cue does not distinguish any pair, $L$ is left unchanged. Then, one of the cues $2, \ldots, n-1$ is selected as the second cue. After removal of the pair distinguished by this cue, the remaining cues make the same incorrect inferences as before. Thus, the algorithm keeps on choosing cues from $\{2, \ldots, n-1\}$ during rounds $2, \ldots, n-1$ of the loop until cue $n$ is selected in the last round. The resulting permutation $\pi$ has cue 1 in its first position, cues from $\{2, \ldots, n-1\}$ in positions $2, \ldots, n-1$, and cue $n$ in the last position. This implies that $\text{INCORRECT}(\pi, L) = |L|$.

On the other hand, the optimal value is 2, which is attained by any permutation that has cue $n$ as the first cue. This yields a performance ratio for Greedy Cue Permutation of at least $|L|/2$. The lower bound $n/2$ is obtained by observing that $|L| = n$. □

We conclude this section by examining the performance of Greedy Cue Permutation on subproblems, that is, when the instances are not arbitrary but meet certain constraints. It plainly arises from the proof of Theorem 11 that the lower bound holds under restrictions of the instances similar to those considered in Sections 3 and 4.

**Corollary 12.** *The lower bound* $\max\{n/2, |L|/2\}$ *for the performance ratio of* Greedy Cue Permutation *holds even when the instances satisfy any (or all) of the following constraints:*

1. *Each element of* $B$ *contains at most two* 1*s.*
2. *The set* $L$ *is smaller than the set* $B$*.*
3. $L$ *is irreflexive.*
4. $L$ *is a subset of some partial order.*
5. $L$ *is a subset of some total order.*

# 6 Lexicographic Strategies With Cue Inversion

While in the previous sections the problem was to optimize lexicographic strategies by permuting the cues, we now introduce an additional degree of freedom for building lexicographic strategies. Here, the method of construction is allowed not only to permute but also to invert cues. A *cue inversion* is a mapping $q : \{1, \ldots, n\} \to \{0, 1\}$, where $n$ is the number of cues. It uniquely defines a function $\overline{q} : \{0,1\}^n \to \{0,1\}^n$ such that for every $a \in \{0,1\}^n$,

$$\overline{q}(a_i) \quad = \quad \begin{cases} a_i & \text{if } q(i) = 0, \\ 1 - a_i & \text{otherwise.} \end{cases}$$

In other words, a value of $q(i) = 1$ indicates that the $i$-th position of every Boolean vector $a$ is to be inverted, whereas the cues with $q(i) = 0$ are left unchanged by $\overline{q}$. As the meaning is clear, we shall use $q$ also to denote $\overline{q}$. Given a set $B \subseteq \{0,1\}^n$, the *lexicographic strategy under cue inversion* $q$ is the function $S^q : B \times B \to \{\text{“}<\text{”}, \text{“}=\text{”}, \text{“}>\text{”}\}$ with

$$S^q(a, b) \quad = \quad S(q(a), q(b)).$$

Combining permutation and inversion, we obtain the *lexicographic strategy under cue permutation* $\pi$ *and cue inversion* $q$ denoted by $S^q_\pi$ and defined as

$$S^q_\pi(a, b) \quad = \quad S(\pi(q(a)), \pi(q(b))).$$

In particular, we require that the cue inversion is applied before the permutation.

A simple greedy method for inverting the cues is described as Algorithm 2. The idea is to pass through the cues and to select either the cue or its inverse, depending on which makes a larger number of correct inferences. The pairs that are distinguished by this cue are then removed. It is evident that Greedy Cue Inversion runs in polynomial time. We show that the cue inversion returned by this algorithm yields a number of correct inferences that is at least half the maximum over all cue inversions and permutations.

**Algorithm 2** GREEDY CUE INVERSION

**Input:** a set $B \subseteq \{0,1\}^n$ and a set $L \subseteq B \times B$
**Output:** a cue inversion $q$ for $n$ cues
  **for** $i = 1, \ldots, n$ **do**
    **if** $|\{\langle a, b\rangle \in L : a_i < b_i\}| \geq |\{\langle a, b\rangle \in L : a_i > b_i\}|$ **then**
      $q(i) := 0$
    **else**
      $q(i) := 1$
    **end if**
    $L := L \setminus \{\langle a, b\rangle : a_i \neq b_i\}$
  **end for**.

**Theorem 13.** *The algorithm* GREEDY CUE INVERSION *always returns a cue inversion $q$ such that $S^q$ is correct on at least opt$(L)/2$ pairs, where opt$(L)$ is the maximum number of correct inferences achievable by the lexicographic strategy under any cue permutation and any cue inversion.*

*Proof.* Let $L_i$ be the set of pairs that the algorithm removes from $L$ in round $i$ of the for-loop and let $L_{n+1}$ be the set of pairs that remains after completion of the last round. Clearly, $L_1, \ldots, L_{n+1}$ is a partition of $L$. Obviously, by the construction of $q$, $S^q$ is correct on at least half of each $L_i$, for $i = 1, \ldots, n$. Further, it is correct on all of $L_{n+1}$, as this set consists solely of identical pairs. Thus, $S^q$ correctly distinguishes at least half of all pairs in $L$. Since $\mathrm{opt}(L) \leq |L|$, it follows that $S^q$ is correct on at least $\mathrm{opt}(L)/2$ pairs. □

One remarkable aspect of this algorithm is the fact that it retains the order of the cues, while its performance guarantee is valid even over all cue permutations. It seems, at first glance, that the method of cue inversion leads much easier to a good performance guarantee than the permutation of the cues. However, the result of Theorem 13 cannot directly compared with those of the previous sections, as these apply to the problem of minimizing the number of incorrect inferences, whereas here we are concerned with the maximization of the number of correct inferences. A constant performance ratio for the one problem does not necessarily imply a constant performance ratio for the other, as can easily be seen. Assume, for instance, that the maximum number of correct inferences is $|L| - 1$. Then the algorithm that is correct on exactly $\lceil |L|/2 \rceil$ pairs has a constant performance ratio for the maximization problem, while with regard to the minimization problem its performance ratio grows linearly in $|L|$.

# 7 Sample Complexity for Learning Lexicographic Strategies

A central notion for characterizing the sample complexity of a learning problem is the VC dimension (Vapnik and Chervonenkis, 1971; Anthony and Bartlett, 1999). In the following, we calculate the VC dimension of lexicographic strategies. The definition of the VC dimension relies on the notion of shattering. A class $\mathcal{F}$ of Boolean functions is said *to shatter* a set $L \subseteq \{0,1\}^n$ if $\mathcal{F}$ induces every dichotomy of $L$, that is, if for every $(L_0, L_1)$ such that $L_0 \cap L_1 = \emptyset$ and $L_0 \cup L_1 = L$, there is some function $f \in \mathcal{F}$ satisfying $f(L_0) \subseteq \{0\}$ and $f(L_1) \subseteq \{1\}$. The *Vapnik-Chervonenkis (VC) dimension* of a class $\mathcal{F}$ of Boolean functions is the cardinality of the largest set that is shattered by $\mathcal{F}$.

We recall from Section 2.2 that we identify the lexicographic strategy $S$ with a Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$ such that for every $\langle a, b\rangle \in \{0,1\}^{2n}$,

$$f(a,b) = 1 \quad \text{if and only if} \quad S(a,b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\}.$$

In this sense, we can investigate the VC dimension of the function class

$$\mathcal{S}_n \;=\; \{S^q_\pi : \pi \text{ is a permutation and } q \text{ an inversion of } n \text{ cues}\},$$

that is, we ask what is the largest cardinality of a set $L$ of pairs that is shattered by the lexicographic strategy under all possible cue permutations and inversions.

It is evident from the definition that the VC dimension of a finite function class $\mathcal{F}$ cannot be larger than $\log |\mathcal{F}|$. Since the number of permutations is equal to $n!$ and the number of inversions is equal to $2^n$, it follows that the VC dimension of $\mathcal{S}_n$ is not larger than $n + n \log n$. We show, however, that this VC dimension is linear. Moreover, we provide the exact value.

**Theorem 14.** *The VC dimension of the class $\mathcal{S}_n$ of lexicographic strategies is equal to $n$.*

*Proof.* We first establish $n$ as upper bound. Given a cue inversion $q$, consider the lexicographic strategy $S^q \in \mathcal{S}_n$ (that is, the strategy $S^q_\pi$ where $\pi$ is the identity function). We claim that every $a, b \in \{0,1\}^n$ satisfies

$$\begin{aligned} &S^q(a,b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\} \\ &\qquad \text{if and only if } \sum_{i=1}^{n} (-1)^{q(i)} 2^{n+1-i}(b_i - a_i) \;\geq\; -1. \end{aligned} \tag{4}$$

To show this, we consider the absolute value of first term on the left-hand side of the inequality, where $i = 1$, that is,

$$|2^n(b_1 - a_1)|. \tag{5}$$

If $a_1 \neq b_1$, the value of (5) is $2^n$, whereas the absolute value of the remaining sum is not larger than $2^n - 2$. Then, the inequality in (4) is satisfied if and only if $\overline{q}(a_1) < \overline{q}(b_1)$. On the other hand, if $a_1 = b_1$, the term (5) is equal to 0, and the validity of the equivalence (4) follows by induction.

Obviously, by permuting the coefficients, every lexicographic strategy $S^q_\pi \in \mathcal{S}_n$ can be written as an inequality such as in (4). Such inequalities are evaluated by Boolean linear threshold functions. A Boolean linear threshold function $f : \{0,1\}^n \to \{0,1\}$ is a function for which there exist real numbers $w_1, \ldots, w_n$ and $t$ (the parameters of this function class ) such that for every $z \in \{0,1\}^n$,

$$f(z) = 1 \quad \text{if and only if} \quad w_1 z_1 + \cdots + w_n z_n \geq t.$$

It follows that every $S^q_\pi \in \mathcal{S}_n$ can be expressed as a Boolean linear threshold function with input variables $(y_1 - x_1), \ldots, (y_n - x_n)$ and a fixed parameter $t = -1$.

Therefore, every set $L \subseteq \{0,1\}^{2n}$ that can be shattered by $\mathcal{S}_n$ is also shattered by this class of linear threshold functions. The class of linear threshold functions in $n$ variables with $n$ parameters (that is, where $t$ is fixed) is known to have VC dimension equal to $n$ (see, e.g., Anthony and Bartlett, 1999). Thus, the VC dimension of $\mathcal{S}_n$ does not exceed $n$.

For deriving the lower bound, we show that the set $L \subseteq \{0,1\}^{2n}$ defined as

$$L \;=\; \{\langle \mathbf{1}_i, \mathbf{1}\rangle : i = 1, \ldots, n\},$$

where $\mathbf{1}$ is the vector with a 1 in every position and $\mathbf{1}_i$ has a 0 in position $i$ and 1 elsewhere, is shattered by $\mathcal{S}_n$.

Let $(L_0, L_1)$ be an arbitrary dichotomy of $L$. Define the cue inversion $q : \{1, \ldots, n\} \to \{0,1\}$ such that $q(i) = 0$ if and only if $\langle \mathbf{1}_i, \mathbf{1}\rangle \in L_1$. Obviously then, the lexicographic strategy $\mathcal{S}^q$ (without permuting the cues) yields a correct comparison for every pair in $L_1$, while the pairs from $L_0$ are inferred incorrectly. Thus, the dichotomy $(L_0, L_1)$ is induced by $S^q$. □

The lower bound in the previous result was obtained by choosing a suitable cue inversion and leaving the order of the cues unchanged. We can also obtain an almost optimal lower bound when the cues are not allowed to be inverted but only permuted. In fact, the $(n-1)$-element set

$$L \;=\; \{\langle \mathbf{1}_1, \mathbf{1}_i\rangle : i = 2, \ldots, n\}$$

can be shattered as follows. Given the dichotomy $(L_0, L_1)$, we define the permutation $\pi$ such that for $i = 2, \ldots, n$, $\pi(1) < \pi(i)$ if and only if $\langle \mathbf{1}_1, \mathbf{1}_i\rangle \in L_1$. Obviously, the dichotomy $(L_0, L_1)$ is induced by $S_\pi$.

It is easy to see that there are values of $n$ for which this lower bound of $n - 1$ cannot be improved. For $n = 1, 2$, and 3, the number of permutations of $n$ elements is $1, 2$, and 6, respectively; to shatter sets of these cardinalities, however, requires $2, 4$, and 8 functions.

# 8 Open Questions

In the following we summarize the major open questions that arise from this work hoping that they might provide fertile soil for future research. The main result of Section 3 is the NP-completeness of the decision problem Lexicographic Strategy. In that section, we have established further that the problem remains NP-complete under several restrictions. Moreover, one of the subproblems originating from such restrictions was shown to be efficiently solvable. Probably, the restrictions considered there may not be those that are "natural", that is, met in practice. It is therefore reasonable to study more subproblems and to delineate the intractable ones from those that can be solved efficiently.

- What are natural restrictions for Lexicographic Strategy under which the problem is NP-complete or efficiently solvable?

Of course, similar considerations are appropriate for Minimum Incorrect Lexicographic Strategy. In Section 4 we obtained a lower bound for the performance ratio that is still valid for various subproblems. A promising task is, therefore, to find restrictions relevant in practice under which the problem has a constant performance ratio.

- What are natural restrictions for Minimum Incorrect Lexicographic Strategy under which the problem belongs to APX?

Work by Raz and Safra (1997) implies that Minimum Hitting Set cannot be approximated in polynomial time to within some factor that grows logarithmically in $|C|$, the number of subsets. The reduction defined in the proof of Theorem 7 does not seem to allow to exploit this fact.

- Does Minimum Incorrect Lexicographic Strategy have a lower bound on the performance ratio for polynomial-time algorithms that is not bounded by some constant?

The results in Sections 4 and 5 have left a gap. While we have shown that there cannot be a polynomial-time algorithm for Minimum Incorrect Lexicographic Strategy with a performance ratio bounded by some constant (if P $\neq$ NP), the algorithm Greedy Cue Permutation has a lower bound of $\max\{n/2, |L|/2\}$.

- Are there polynomial-time algorithms for Minimum Incorrect Lexicographic Strategy that have a better performance ratio than Greedy Cue Permutation?

The algorithm Greedy Cue Permutation is a simple and obvious heuristic that has not been studied before in the context of lexicographic strategies. In

Section 5 we have derived tight bounds on the performance ratio of this algorithm. Various other procedures have been studied in the literature and become known as fast and frugal heuristics, but nothing seems to have been proven about their performance ratio.

- Which are the performance ratios of other (fast and frugal) heuristics for lexicographic strategies?

In Section 6 we have introduced cue inversion as an additional feature to build lexicographic strategies. The algorithm Greedy Cue Inversion was shown to approximate the maximum number of correct inferences to within a constant factor. While the problems of minimizing the number of incorrect inferences and maximizing the number of correct inferences give rise to equivalent decision problems, there might well be a difference with regard to the approximation problem. There seems to be no immediate way to derive a lower bound for the maximization problem from a lower bound for the minimization problem. Thus, similar questions as considered here can be raised for the problem Maximum Correct Lexicographic Strategy which is defined analogously.

- Which is the performance ratio of polynomial-time algorithms for approximating Maximum Correct Lexicographic Strategy?

While this question is meant to consider only cue permutations and not inversions for constructing lexicographic strategies, the objective of minimization is combined with both these features in a second approximation problem emerging from Section 6.

- Which is the performance ratio of polynomial-time algorithms for approximating Minimum Incorrect Lexicographic Strategy under cue permutations and cue inversions?

We can ask further what happens if the problems studied here are generalized in a certain way. One obvious possibility of generalizing is to allow cues that have more than two values. It is evident that the reductions provided in Sections 3 and 4 remain valid also in this multiple-valued case. In other words, the problem with binary cues is a subproblem of the problem with multiple-valued cues. Hence, NP-completeness and the lower bound for the approximability hold for learning lexicographic strategies on multiple-valued cues, too. Moreover, we observe that the algorithm Greedy Cue Permutation and the proof of the upper bound on its performance ratio (Theorem 9) do not make use of the two-valuedness of the cues. Thus, this algorithm has the claimed approximation property for multiple-valued cues as well. One could also generalize lexicographic strategies to the effect that more than two outcomes, correct or incorrect, of a lexicographic comparison are possible. The results of this article do not seem to yield a statement for such cases in general.

# 9 Conclusions

Computational problems that arise in learning lexicographic strategies from examples are the topic of this article. In particular, we considered the model of agnostic PAC learning. We have introduced the minimizing disagreement problem Lexicographic Strategy and shown that it is NP-complete. Thus, it has become very unlikely that lexicographic strategies can be efficiently learned. This statement was strengthened by our proving that the optimization problem Minimum Incorrect Lexicographic Strategy cannot be approximated in polynomial time to within any constant factor.

These results answer a question raised by psychological research into models of bounded rationality: How accurate are fast and frugal heuristics? We have shown that no fast, that is, polynomial-time, algorithm can compute the optimum and, moreover, not even approximate it well, under the widely accepted assumption that $P \neq NP$.

This answers also a second question concerning a specific fast and frugal heuristic: How accurate is TTB? We have introduced a greedy algorithm that provably performs better than TTB. In particular, we have shown that the greedy method always finds accurate solutions when they exist, whereas this is not the case with TTB. Tight bounds for the factor with which the greedy method approximates the optimum have also been obtained.

The lower bounds derived in this article have mostly been shown to hold even for subproblems obtained from various restrictions. We interpret this as revealing to a high degree that lexicographic strategies cannot be learned efficiently and that it might be very difficult to find satisfactory algorithms.

For the learning of lexicographic strategies using cue inversions we have provided a simple and efficient algorithm that approximates the maximum number of correct inferences to within a constant factor. Thus, it seems that cue inversions lead much easier to good performance bounds than cue permutations. However, one cannot directly compare a bound for the maximization problem with a bound for the minimization problem. This result should more be considered as a stimulating impetus for further research.

We have calculated the exact values of the VC dimension of lexicographic strategies. This result is one of the few examples where the VC dimension of a function class has been determined precisely.

While we have already presented in the previous section a couple of formal open questions for theoretical investigation, a challenge to experimental research is also given by this article: to study the relevance of the greedy method as a model for bounded rationality in psychology.

# References


Amaldi, E. and Kann, V. (1995). The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147:181–210.

Amaldi, E. and Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260.

Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and Approximation: Combinatorial Problems and Their Approximability Properties*. Springer-Verlag, Berlin.

Ausiello, G., D'Atri, A., and Protasi, M. (1980). Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153.

Bar-Yehuda, R. and Even, S. (1981). A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203.

Bellare, M., Goldwasser, S., Lund, C., and Russell, A. (1993). Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 294–304. ACM Press, New York, NY.

Bröder, A. (2000). Assessing the empirical validity of the "take-the-best" heuristic as a model of human probabilistic inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26:1332–1346.

Bröder, A. (2002). Take the best, Dawes' rule, and compensatory decision strategies: A regression-based classification method. *Quality & Quantity*, 36:219–238.

Bröder, A. and Schiffer, S. (2003). Take the best versus simultaneous feature matching: Probabilistic inferences from memory and effects of representation format. *Journal of Experimental Psychology: General*, 132:277–293.

Bullock, S. and Todd, P. M. (1999). Made to measure: Ecological rationality in structured environments. *Minds and Machines*, 9:497–541.

Cohen, W. W., Schapire, R. E., and Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA.

Gigerenzer, G. and Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103:650–669.

Gigerenzer, G., Hoffrage, U., and Kleinbölting, H. (1991). Probabilistic mental models: A Brunswikian theory of confidence. *Psychological Review*, 98:506–528.

Gigerenzer, G., Todd, P. M., and the ABC Research Group (1999). *Simple Heuristics That Make Us Smart*. Oxford University Press, New York, NY.

Greiner, R. (1999). The complexity of revising logic programs. *The Journal of Logic Programming*, 40:273–298.

Greiner, R. and Orponen, P. (1996). Probably approximately optimal satisficing strategies. *Artificial Intelligence*, 82:21–44.

Hochbaum, D. S. (1982). Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556.

Höffgen, K.-U., Simon, H.-U., and Van Horn, K. S. (1995). Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50:114–125.

Hogarth, R. M. and Karelaia, N. (2003). "Take-the-best" and other simple strategies: Why and when they work "well" in binary choice. DEE Working Paper 709, Universitat Pompeu Fabra, Barcelona.

Kearns, M. J., Schapire, R. E., and Sellie, L. M. (1994). Toward efficient agnostic learning. *Machine Learning*, 17:115–141.

Lee, M. D. and Cummins, T. D. R. (2004). Evidence accumulation in decision making: Unifying the "take the best" and the "rational" models. *Psychonomic Bulletin & Review*, 11:343–352.

Martignon, L. and Hoffrage, U. (1999). Why does one-reason decision making work? A case study in ecological rationality. In Gigerenzer, G., Todd, P. M., and the ABC Research Group, *Simple Heuristics That Make Us Smart*, pages 119–140. Oxford University Press, New York, NY.

Martignon, L. and Hoffrage, U. (2002). Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory and Decision*, 52:29–71.

Martignon, L. and Schmitt, M. (1999). Simplicity and robustness of fast and frugal heuristics. *Minds and Machines*, 9:565–593.

Nellen, S. (2003). The use of the "take the best" heuristic under different conditions, modeled with ACT-R. In Detje, F., Dörner, D., and Schaub, H., editors, *Proceedings of the Fifth International Conference on Cognitive Modeling*, pages 171–176, Universitätsverlag Bamberg, Bamberg.

Newell, B. R. and Shanks, D. R. (2003). Take the best or look at the rest? Factors influencing "One-Reason" decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29:53–65.

Newell, B. R., Weston, N. J., and Shanks, D. R. (2003). Empirical tests of a fast-and-frugal heuristic: Not everyone "takes-the-best". *Organizational Behavior and Human Decision Processes*, 91:82–96.

Raz, R. and Safra, S. (1997). A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM Press, New York, NY.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, 2:229–246.

Russell, S. and Wefald, E. (1991). *Do The Right Thing: Studies in Limited Rationality*. MIT Press, Cambridge, MA.

Schmitt, M. and Martignon, L. (1999). Complexity of lexicographic strategies on binary cues. Preprint.

Schmitt, M. and Martignon, L. (2006). On the accuracy of bounded rationality: How far from optimal is fast and frugal? In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA. To appear.

Simon, H. A. (1982). *Models of Bounded Rationality, Volume 2*. MIT Press, Cambridge, MA.

Simon, H. A. and Kadane, J. B. (1975). Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence*, 6:235–247.

Simon, H. A. and Kadane, J. B. (1976). Problems of computational complexity in artificial intelligence. In Traub, J. F., editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 281–299. Academic Press, New York, NY.

Skiena, S. S. (1997). *The Algorithm Design Manual*. Springer-Verlag, New York, NY.

Slegers, D. W., Brake, G. L., and Doherty, M. E. (2000). Probabilistic mental models with continuous predictors. *Organizational Behavior and Human Decision Processes*, 81:98–114.

Todd, P. M. and Dieckmann, A. (2005). Heuristics for ordering cue search in decision making. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 1393–1400. MIT Press, Cambridge, MA.

Todd, P. M. and Gigerenzer, G. (2000). Précis of "Simple Heuristics That Make Us Smart". *Behavioral and Brain Sciences*, 23:727–741.

Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280.