# Querying Priced Information in Databases: The Conjunctive Case

RENATO CARMO
*Universidade Federal do Paraná*

TOMÁS FEDER
*Stanford University*

YOSHIHARU KOHAYAKAWA
*Universidade de São Paulo*

EDUARDO LABER
*Pontifícia Universidade Católica do Rio de Janeiro*

RAJEEV MOTWANI AND LIADAN O'CALLAGHAN
*Stanford University*

RINA PANIGRAHY
*Cisco Systems*

AND

DILYS THOMAS
*Stanford University*

Abstract. Query optimization that involves expensive predicates has received considerable attention in the database community. Typically, the output to a database query is a set of tuples that satisfy certain conditions, and, with expensive predicates, these conditions may be computationally costly

---

Some of the results in this article appeared as an extended abstract at the *Latin American Theoretical Informatics (Latin) Conference*, 2004 [Laber et al. 2004].

R. Carmo was partially supported by CAPES (PICDT) and CNPq (Proc. 476817/2003-0). E. Laber was partially supported by FAPERJ (Proc. E-26/150.715/2003) and CNPq (Proc. 476817/2003-0). Y. Kohayakawa was partially supported by FAPESP and CNPq through ProNEx projects (Proc. CNPq 664107/1997–4 and Proc. FAPESP 2003/09925–5) and CNPq (Proc. 306334/2004–6, 468516/2000–0, and Proc. 479882/2004–5).

Authors' addresses: R. Carmo, Departamento de Informática da Universidade Federal do Paraná, Centro Politécnico da Universidade Federal do Paraná, 81531-990, Curitiba, PR Brasil; T. Feder, R. Motwani, L. O'Callaghan, D. Thomas, Computer Science Department, Stanford University, Stanford, CA 94305; Y. Kohayakawa, Instituto de Matemática e Estatística da Universidade de São Paulo, R. do Matao, 1010, Cidade Universitária, 05508-090, São Paulo, Brasil; E. Laber (contact author), Departamento de Informática PUC-Rio, Rio Data-Centro, Sala 518, e-mail: laber@info.puc-rio.br; R.Panigrahy, Cisco Systems, Inc., 170 West Tasman Dr., San Jose, CA 95134.

to verify. In the simplest case, when the query looks for the set of tuples that simultaneously satisfy $k$ expensive predicates, the problem reduces to ordering the evaluation of the predicates so as to minimize the time to output the set of tuples comprising the answer to the query. We study different cases of the problem: the *sequential case*, in which a single processor is available to evaluate the predicates, and the *distributed case*, in which there are $k$ processors available, each dedicated to a different attribute (column) of the database, and there is no communication cost between the processors.

For the sequential case, we give a simple and fast deterministic $k$-approximation algorithm, and prove that $k$ is the best possible approximation ratio for a deterministic algorithm, even if exponential time algorithms are allowed. We also propose a randomized, polynomial time algorithm with expected approximation ratio $1 + \sqrt{2}/2 \approx 1.707$ for $k = 2$, and prove that $3/2$ is the best possible expected approximation ratio for randomized algorithms. We also show that given $0 \leq \varepsilon \leq 1$, no randomized algorithm achieves approximation ratio smaller than $1 + \varepsilon$ with probability larger than $(1 + \varepsilon)/2$.

For the distributed case, we consider two different models: the *preemptive model*, in which a processor is allowed to interrupt the evaluation of a predicate, and the *nonpreemptive model*, in which the evaluation of a predicate must be completed once started. We show that $k$ is the best possible approximation ratio for a deterministic algorithm, even if exponential time algorithms are allowed. For the preemptive model, we introduce a polynomial time $k$-approximation algorithm. For the nonpreemptive model, we introduce a polynomial time $O(k \log^2 k)$-approximation algorithm.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; *sequencing and scheduling*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*; H.2.4 [**Database Management**]: Systems—*Query processing, relational databases, multimedia databases*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, search process, selection process*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Competitive analysis, online algorithms

**ACM Reference Format:**

Carmo, R., Feder, T., Kohayakawa, Y., Laber, E., Motwani, R.,O'Callaghan, L., Panigrahy, R., and Thomas, D. 2007. Querying priced information in databases: The conjunctive case. ACM Trans. Algor. 3, 1, Article 9 (Feb. 2007), 22 pages. DOI = 10.1145/1186810.1186819 http://doi.acm.org/10.1145/1186810.1186819.

## 1. *Introduction*

The main goal of *query optimization in databases* is to determine how a query over a database should be processed in order to minimize the user response time. A typical query extracts the tuples from a database relation that satisfy a set of conditions, or *predicates*, in database terminology. For example, consider the set of tuples $D = \{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_1)\}$ (see Figure 1(a)) and a conjunctive query that seeks to extract the subset of the tuples $(a_i, b_j)$ for which $a_i$ satisfies predicate $P_1$ and $b_j$ satisfies predicate $P_2$. These predicates can be viewed together as a 0/1-valued function $\delta$ defined on the set of tuple elements $\{a_1, a_2, b_1, b_2, b_3\}$, with the convention that $\delta(a_i) = 1$ if and only if $P_1(a_j)$ holds and $\delta(b_j) = 1$ if and only if $P_2(b_j)$ holds. The answer to the query is the set of pairs $(a_i, b_j)$ with $\bar{\delta}(a_i, b_j) = \delta(a_i)\delta(b_j) = 1$. The query optimization problem that we consider is that of determining a strategy for evaluating $\bar{\delta}$ so as to compute this set of tuples by evaluating as few values of the function $\delta$ as possible (or, more generally, with the total cost for evaluating the function $\bar{\delta}$ minimal).

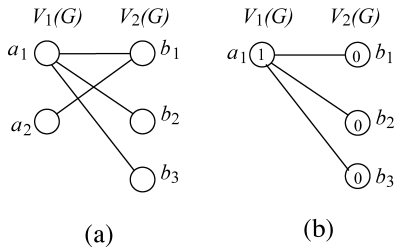


FIG. 1. (a) The set of tuples $\{(a_1, b_1), (a_1, b_2), (a_1, b_3), (a_2, b_1)\}$; and (b) an instance of D2O.

It is usually the case that the cost (measured as the computational time) to evaluate the predicates of a query can be assumed to be bounded by a constant so that the query can be answered by just scanning through all the tuples in $D$, while evaluating the corresponding predicates.

This is not the case with computationally expensive predicates. For instance, when the database holds complex data as images, tables, long sequences etc., this constant may happen to be so large as to render the usual strategy impractical. In such cases, the different costs involved in evaluating each predicate must also be taken into account in order to keep user response time within reasonable bounds.

Among several proposals to model and solve this problem (see, e.g., Bouganim et al. [2001], Chaudhuri and Shim [1993], and Hellerstein [1998]), we focus on improvement of the approach proposed in Porto [2001] where, differently from others, the query evaluation problem is reduced to an optimization problem on a hypergraph (see Figure 1). We study different cases of the problem: the *sequential case*, in which a single processor is available to evaluate the predicates, and the *distributed case*, in which there are $k$ processors available, each dedicated to a different attribute (column) of the database and there is no communication cost between the processors.

1.1. PROBLEM STATEMENT. A *hypergraph* is a pair $G = (V(G), E(G))$, where $V(G)$, the set of *vertices* of $G$, is a finite set and each *edge* $e \in E(G)$ is a nonempty subset of $V(G)$.

The *rank of* $G$, denoted $r(G)$, is the maximum cardinality of an edge in $G$. A hypergraph $G$ is said to be *$r$-uniform* if each edge has $r(G)$ vertices. Hypergraph $G$ is said to be $k$-partite if there is a partition $\{V_1, \ldots, V_k\}$ of $V(G)$ such that no edge contains more than one vertex in the same partition class. Unless otherwise noted, every hypergraph $G$ in this work will be uniform, $r(G)$-partite and $\{V_i(G)\colon\ 1 \leq i \leq r(G)\}$ denotes the partition of $V(G)$ under consideration.

A *matching* in a hypergraph $G$ is a set $M \subseteq E(G)$ with no two edges in $M$ sharing a common vertex. A hypergraph $G$ is said to be a matching if $E(G)$ is a matching. A *cover* for $G$ is a set $C \subseteq V(G)$ such that every edge of $G$ has at least one vertex in $C$.

Given a hypergraph $G$ and a function $\delta : V(G) \to \{0, 1\}$, we define an *evaluation* of $(G, \delta)$ as a set $\mathsf{E} \subseteq V(G)$ such that knowing the value of $\delta(v)$ for each $v \in \mathsf{E}$, we may determine, for each $e \in E(G)$, the value of

$$\bar{\delta}(e) = \prod_{v \in e} \delta(v). \tag{1}$$

Note that an evaluation for $(G, \delta)$ must be a cover for $G$, otherwise there will be an edge for which the value of $\bar{\delta}$ cannot be determined.

Given a hypergraph $G$ and a function $\gamma : V(G) \to \mathbb{R}$, we associate a *cost* $\gamma(X)$ to each subset $X$ of $V(G)$ as a function of the values of $\gamma(x)$, $x \in X$, for example,

$$\gamma(X) = \sum_{v \in X} \gamma(v).$$

See the following for the different definitions of $\gamma(X)$ to be used in later sections.

Given an integer $k > 0$, an instance of the *Dynamic k-Partite Ordering* problem (D$k$O) is a $k$-partite, $k$-uniform hypergraph $G$, together with functions $\delta$ and $\gamma$, as before. The goal in D$k$O is to determine an evaluation of minimum cost for $(G, \delta, \gamma)$. Observe that while the value of $\gamma(v)$ is known in advance for each $v \in V(G)$, the function $\delta$ is 'unknown' to us at first. More precisely, the value of $\delta(v)$ becomes known only when $\delta(v)$ is actually evaluated, and this evaluation costs $\gamma(v)$.

Before we proceed, let us observe that D$k$O models our database problem as follows: The vertex classes $V_i(G)$, $1 \leq i \leq k$ correspond to the $k$ different attributes of the relation that is being queried. Each vertex of $G$ corresponds to a distinct attribute value (tuple element). Edges correspond to tuples in the relation, $\gamma(v)$ is the cost to evaluate $\delta$ on $v$, and $\delta(v)$ corresponds to the result of a predicate evaluated at the corresponding tuple element. For this reason, we say that a vertex $v$ is a *false vertex* or a *true vertex* when $\delta(v) = 0$ or $\delta(v) = 1$, respectively.

As described up to this point, D$k$O stands for a whole family of related computational problems, rather than a unique problem, since each definition of the cost $\gamma(\mathsf{E})$ of an evaluation $\mathsf{E}$ may define a different problem. In every case, we will refer to a minimum cost evaluation as an *optimal evaluation*. If $\mathcal{I}$ is an instance of D$k$O, we will denote an optimal evaluation for $\mathcal{I}$ by $\mathsf{E}^*_{\mathcal{I}}$. We are interested in modeling our database problem in two different contexts.

The case in which there is only one processor available to evaluate the predicates will be called the *sequential case* of D$k$O, denoted sD$k$O. In this case, the *cost of an evaluation* $\mathsf{E}$ is defined as

$$\gamma(\mathsf{E}) = \sum_{v \in \mathsf{E}} \gamma(v).$$

The case in which there are $k$ processors available, one dedicated to evaluate the predicates referring to each attribute of the database, will be called the distributed case of D$k$O, and will be denoted by dD$k$O. In the distributed case, we address both the *preemptive* and *nonpreemptive* models. In the former, we are allowed to interrupt the evaluation of a vertex before completed, while in the latter, such an interruption is not permitted.

Figure 1(b) shows an instance of sD2O. The value of $\delta(v)$ is indicated inside each vertex $v$. Suppose that $\gamma(a_1) = 3$ and $\gamma(b_1) = \gamma(b_2) = \gamma(b_3) = 2$. In this case, any strategy that starts evaluating $\delta(a_1)$ will return the evaluation $\{a_1, b_1, b_2, b_3\}$, of cost 9. However, the evaluation of minimum cost for this instance is $\{b_1, b_2, b_3\}$, of cost 6. This example highlights the key point: The problem is to devise a strategy for dynamically choosing, based on function $\gamma$ and the values of $\delta$ already revealed, the next vertex $v$ whose $\delta$-value should be evaluated so as to minimize the overall cost.

In the *distributed case*, an evaluation is a more complex object than a set of vertices because of the scheduling information involved. In order to remain close

to the motivating problem, we allow for processors in idle state while waiting for others to complete their respective evaluations (or for processors aborting the evaluation of a vertex before completed) to start the evaluation of another vertex (preemption).

To illustrate the subtleties involved in these scheduling issues, let us consider an instance $\mathcal{I} = (G, \delta, \gamma)$ of dD2O in which $G$ has a single edge $e = \{v_1, v_2\}$ with costs $\gamma(v_1) = 1$ and $\gamma(v_2) = 1{,}000$:

—In the nonpreemptive model, if both processors start to evaluate simultaneously, the cost would be 1, 000, whatever the values of $\delta(v_1)$ and $\delta(v_2)$; if the processor in charge of $V_2(G)$ waits for the result of the evaluation of $v_1$ before processing $v_2$, the cost of the evaluation would be either 1,001 if $\delta(v_1) = 1$, or 1 if $\delta(v_1) = 0$.

—In the preemptive model, if both processors start to evaluate simultaneously, the cost would be either 1 if $\delta(v_1) = 0$, or 1,000 if $\delta(v_1) = 1$.

The results we are concerned with here do not require us to explicitly formalize the scheduling information contained in an evaluation. It will be enough for our purposes to define the *cost of an evaluation* $\mathsf{E}$ as the time elapsed between the start of evaluation by the first processor and the end of the evaluation by the last processor.

1.2. Measuring the Performance of Algorithms for D$k$O. Let $\mathcal{A}$ be an algorithm for D$k$O and let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of D$k$O. We will denote the evaluation computed by $\mathcal{A}$ on input $\mathcal{I}$ by $\mathcal{A}(\mathcal{I})$. Establishing a measure for the performance of a given algorithm $\mathcal{A}$ for D$k$O is somewhat delicate: For example, a worst-case analysis of $\gamma(\mathcal{A}(\mathcal{I}))$ is not suitable, since any correct algorithm should output an evaluation comprising all vertices in $V(G)$ when $\delta(v) = 1$ for every $v \in V(G)$. Hence the following definition.

Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of D$k$O and let $\mathsf{E}$ be an evaluation for $\mathcal{I}$. The *deficiency of evaluation* $\mathsf{E}$ *(*with respect to $\mathcal{I}$*)* is the ratio

$$d(\mathsf{E}, \mathcal{I}) = \frac{\gamma(\mathsf{E})}{\gamma(\mathsf{E}^*_{\mathcal{I}})}.$$

Given an algorithm $\mathcal{A}$ for D$k$O, we define the *deficiency of* $\mathcal{A}$ as the worst-case *deficiency* of the evaluation $\mathcal{A}(\mathcal{I})$, where $\mathcal{I}$ ranges over all possible instances of the problem, that is,

$$d(\mathcal{A}) = \max\{d(\mathcal{A}(\mathcal{I}), \mathcal{I}) \colon \mathcal{I} \text{ is an instance of D}k\text{O}\}.$$

If $\mathcal{A}$ is a randomized algorithm, then $d(\mathcal{A}(\mathcal{I}), \mathcal{I})$ is a random variable, and the *deficiency* of $\mathcal{A}$ is defined as the maximum over all instances of the expected value of this random variable, namely,

$$d(\mathcal{A}) = \max\{\mathbb{E}\,[d(\mathcal{A}(\mathcal{I}), \mathcal{I})] : \mathcal{I} \text{ is an instance of D}k\text{O}\}.$$

We say that algorithm $\mathcal{A}$ is a *deficiency-optimal algorithm for* D$k$O if its deficiency is the best possible. Clearly, in all scenarios, we wish to devise fast algorithms whose deficiency is as close to 1 as possible. In this article, we are concerned with designing algorithms for D$k$O, analyzing them and establishing bounds for their deficiency.

1.3. STATEMENT OF RESULTS. The results in this work are organized as follows. Section 2 deals with the sequential case of D$k$O (sD$k$O) and Section 3 deals with the distributed case (dD$k$O). In Section 4, we indicate some open problems.

The study of the sequential case in Section 2 starts with the establishment of lower bounds on the deficiency of deterministic and randomized algorithms for sD$k$O (Section 2.1). These bounds apply for exponential time algorithms, as well.

We then introduce a deficiency-optimal deterministic algorithm for sD$k$O with time complexity $O(k|E(G)| \log |V(G)|)$ in Section 2.2. This algorithm does not need to know the whole hypergraph in advance in order to solve the problem, since it scans the edges (tuples), evaluating each as it becomes available. This is a convenient feature for the database application that motivates this work.

Section 2.3 focuses on the study of the sequential case of the *Dynamic* 2*-Partite Ordering* problem (sD2O). We introduce $\mathcal{R}_\varepsilon$, a randomized algorithm for sD2O with time complexity $O(|V(G)|^3)$, whose expected deficiency is, at most, $2 - \varepsilon$ for any given $0 \leq \varepsilon \leq 1 - \sqrt{2}/2$. Clearly, the best bound for the expected deficiency of $\mathcal{R}_\varepsilon$ is achieved when $\varepsilon = 1 - \sqrt{2}/2$. However, the greater the value of $\varepsilon$, the greater the probability that a particular execution of $\mathcal{R}_\varepsilon$ will return a poor result. In Section 2.3.1, we prove that the algorithm's deficiency is always bounded from above by $1 + 1/(1 - \varepsilon)$.

The deficiency $\mathcal{R}_\varepsilon$ is not assured to be highly concentrated around the expectation. In Section 2.3.2, we show that this limitation is inherent to the problem, rather than a weakness of our approach. More precisely, we show that for any $0 \leq \varepsilon \leq 1$, no randomized algorithm can have deficiency smaller than $1 + \varepsilon$ with probability larger than $(1 + \varepsilon)/2$. The proof of this fact makes use of Yao's Minimax principle (see Theorem 2.2).

The study of the distributed case in Section 3 also starts with lower bounds on the deficiency of deterministic and randomized algorithms for dD$k$O (Section 3.1). Again, these lower bounds hold for exponential time algorithms, as well.

Section 3.1 introduces deterministic algorithms for dD$k$O. In Section 3.2.1, we present a deficiency-optimal algorithm with time complexity $O(k|E(G)| \log |V(G)|)$ for the preemptive model and in Section 3.2.2, we give an algorithm with time complexity $O(k|E(G)| \log |V(G)|)$ and deficiency $O(k \log^2 k)$ for the nonpreemptive model.

1.4. RELATED WORK. The problem of optimizing queries with expensive predicates has received some attention in the database community [Avnur and Hellerstein 2000; Bouganim et al. 2001; Chaudhuri and Shim 1993; Hellerstein 1998; Laber et al. 2002; Porto 2001]. However, most of the proposed approaches [Chaudhuri and Shim 1993; Hellerstein 1998] do not take into account the fact that an attribute value may appear in different tuples. In other words, these approaches only address those instances $(G, \delta, \gamma)$ of D$k$O where $G$ is a matching.

The idea of processing the hypergraph induced by the input relation of the database appears first in Porto [2001], but no theoretical analysis is made.

The nonpreemptive version of dD2O is studied in Laber et al. [2002], where the following results are presented:

—a lower bound of 1.5 on the deficiency of any (not only polynomial) randomized algorithm;

—a randomized polynomial time algorithm with deficiency 8/3; and

—a linear time algorithm of deficiency 2 for the particular case of dD2O in which all vertices have the same cost.

In the present work, we restrict our attention to *conjunctive* queries (in the sense of (1)). However, much more general queries could be considered. For example, $\bar{\delta}\colon E(G) \to \{0, 1\}$ could be any formula in the first-order propositional calculus involving the predicates represented by $\delta$. In Charikar et al. [2002], the problem of evaluating a query on a single tuple is addressed. In particular, queries that can be represented by an "AND/OR tree" over a set of variables (where the cost of evaluating each variable may be different) are considered. This problem is further studied in Laber [2004] Cicalese and Laber [2005, 2006].

## 2. *The Sequential Case*

In this section we study sD$k$O, the sequential case of D$k$O. We start with lower bounds on the deficiency of deterministic and randomized algorithms for sD$k$O and then introduce a deficiency-optimal deterministic algorithm for sD$k$O with time complexity $O(k|E(G)| \log |V(G)|)$.

Next, we focus on the study of the sequential case of the Dynamic 2-Partite Ordering problem (sD2O) and introduce a randomized algorithm for sD2O with time complexity $O(|V(G)|^3)$, whose expected deficiency is, at most, $2 - \varepsilon$ for any given $0 \leq \varepsilon \leq 1 - \sqrt{2}/2$. We then prove that this algorithm's deficiency is always bounded by $1 + 1/(1 - \varepsilon)$. Finally, we show that for any $0 \leq \varepsilon \leq 1$, no randomized algorithm can have deficiency smaller than $1+\varepsilon$ with probability larger than $(1 + \varepsilon)/2$.

Recall that since this section deals only with the sequential case of D$k$O, the cost of an evaluation $\mathsf{E}$ is defined as $\gamma(\mathsf{E}) = \sum_{v \in \mathsf{E}} \gamma(v)$.

### 2.1. Lower Bounds.

We start with some lower bounds for the deficiency of algorithms for sD$k$O. As noted before, these bounds apply to exponential time/space algorithms, as well.

THEOREM 2.1. *Let an integer $k > 0$ be given. Given a deterministic algorithm $\mathcal{A}$ for* sD$k$O *and a hypergraph $G$ with at least one edge and $r(G) = k$, there exist functions $\gamma$ and $\delta$ such that for $\mathcal{I} = (G, \delta, \gamma)$, we have $d(\mathcal{A}(\mathcal{I}), \mathcal{I}) = k$.*

PROOF. Let $\mathcal{A}$ and $G$ be as previously given and let $e$ be an edge of $G$. Let $\gamma$ and $\delta'$ be given by

$$\gamma(v) = \delta'(v) = \begin{cases} 1, & \text{if } v \in e; \\ 0, & \text{otherwise.} \end{cases}$$

Consider the execution of $\mathcal{A}(G, \delta', \gamma)$. Let $u$ be the last vertex of $e$ evaluated in this execution and let

$$\delta(v) = \begin{cases} 0, & \text{if } v = u; \\ \delta'(v), & \text{otherwise.} \end{cases}$$

It is not difficult to see that $\gamma(\mathcal{A}(G, \delta, \gamma)) = r(G) = k$ and that $\mathsf{E}^* = (V(G) - e) \cup \{u\}$ is an evaluation of cost 1 of $\mathcal{I} = (G, \delta, \gamma)$. Therefore $d(\mathcal{A}(\mathcal{I}), \mathcal{I}) = \gamma(\mathcal{A}(\mathcal{I}))/\gamma(\mathsf{E}^*) = k$. □

To prove the result analogous to Theorem 2.1 for randomized algorithms, we will apply Yao's Minimax principle [Yao 1977], in the following form. We need some preliminaries.

In what follows, we shall consider probability distributions $\pi$ on the instances of sD$k$O, supported on a finite set ($\mathbb{P}_\pi(\mathcal{I}) > 0$ for finitely many $\mathcal{I}$ only). Given such a distribution $\pi$ and a fixed deterministic algorithm $\mathcal{A}$ for sD$k$O, we may consider the expected cost of the output of $\mathcal{A}$ when $\mathcal{A}$ is run on a random instance $\mathcal{I}$ distributed according to $\pi$. We shall denote this 'average cost' of $\mathcal{A}(\mathcal{I})$ by $\mathbb{E}_\pi(\gamma(\mathcal{A}))$.

Suppose now that we have a randomized algorithm $\mathcal{R}$ for sD$k$O and that $\mathcal{I}$ is a fixed instance. We may, of course, consider $\mathbb{E}(\mathcal{R}(\mathcal{I}))$, the expected cost of the output of $\mathcal{R}$ on the instance $\mathcal{I}$ (here, $\mathbb{E}$ denotes expectation with respect to the randomization present in $\mathcal{R}$). Having introduced the preceding notation, we may state Yao's principle in the form that is convenient for us.

THEOREM 2.2. *Let $\pi$ be a probability distribution with finite support on the set of instances of* sD$k$O. *For any randomized algorithm $\mathcal{R}$ for* sD$k$O*, we have that*

$$\begin{aligned}&\min\{\mathbb{E}_\pi[\gamma(\mathcal{A})] : \mathcal{A} \textit{ is a deterministic algorithm for } \mathsf{sD}k\mathsf{O}\}\\ &\quad\le \max\{\mathbb{E}[\gamma(\mathcal{R}(\mathcal{I}))] : \mathbb{P}_\pi(\mathcal{I}) > 0\}.\end{aligned} \tag{2}$$

With Theorem 2.2 at hand, we may prove our lower bound for the deficiency of randomized algorithms for sD$k$O.

THEOREM 2.3. *Let $k > 0$. The deficiency of a randomized algorithm for* sD$k$O *is at least $(k+1)/2$.*

PROOF. Consider the probability distribution $\pi$ on the instances $(G, \delta, \gamma)$ of sD$k$O, defined as follows: (i) The only instances with positive probability are those in which $V(G) = \{1, \ldots, k\}$ with the single edge $\{1, \ldots, k\}$, $\gamma(v) = 1$ for all $v \in V(G)$ and with exactly one false vertex; and (ii) each of these instances have the same probability, namely, $1/k$. It is immediate that the cost of an optimal evaluation for any of these instances is 1.

Let $\mathcal{A}$ be a deterministic algorithm for sD$k$O and let $\mathcal{I}$ be one of the aforementioned instances. Note that the value of $\gamma(\mathcal{A}(\mathcal{I}))$ ranges over all the possible values $1, \ldots, k$, as $\mathcal{I}$ varies over its $k$ possible values, and that $\mathcal{A}$ always evaluates the vertices of the hypergraph in the same order until it evaluates the false vertex, at which point it stops.

As there are exactly $k$ instances of positive probability, all equiprobable, the expected value of $\gamma(\mathcal{A}(\mathcal{I}))$ is

$$\mathbb{E}_\pi[\gamma(\mathcal{A}(\mathcal{I}))] = \sum_{i=1}^{k} \frac{1}{k} i = \frac{k+1}{2}. \tag{3}$$

On the other hand, if $\mathcal{R}$ is a randomized algorithm for sD$k$O, then

$$\begin{aligned} d(\mathcal{R}) &= \max\{\mathbb{E}[d(\mathcal{R}(\mathcal{I}), \mathcal{I})] : \mathcal{I} \text{ is an instance of } \mathsf{sD}k\mathsf{O}\}\\ &= \max\left\{\mathbb{E}\left[\frac{\gamma(\mathcal{R}(\mathcal{I}))}{1}\right] : \mathcal{I} \text{ is an instance of } \mathsf{sD}k\mathsf{O}\right\}\\ &= \max\{\mathbb{E}[\gamma(\mathcal{R}(\mathcal{I}))] : \mathcal{I} \text{ is an instance of } \mathsf{sD}k\mathsf{O}\}\\ &\ge \max\{\mathbb{E}[\gamma(\mathcal{R}(\mathcal{I}))] : \mathbb{P}_\pi(\mathcal{I}) > 0\},\end{aligned}$$

which, combined with Eq. (2), gives

$$d(\mathcal{R}) \geq \min\{\mathbb{E}_\pi\,[\gamma(\mathcal{A})] : \mathcal{A} \text{ is a deterministic algorithm for } \mathsf{sD}k\mathsf{O}\}.$$

From Eq. (3), we conclude $d(\mathcal{R}) \geq (k+1)/2$. □

These results can be summarized as follows.

COROLLARY 2.4. *No deterministic algorithm for* sD$k$O *can achieve deficiency smaller than* $k$ *and no randomized algorithm for* sD$k$O *can achieve deficiency smaller than* $(k+1)/2$.

2.2. A DEFICIENCY-OPTIMAL POLYNOMIAL DETERMINISTIC ALGORITHM. In this section, we introduce a deficiency-optimal polynomial time deterministic algorithm for sD$k$O, that is, one which has deficiency $k$.

The algorithm works as follows. Given an instance $(G, \delta, \gamma)$ of sD$k$O, the algorithm keeps track of a variable $\psi_e(v)$ for each $e \in E(G)$ and each $v \in e$. The main loop of the algorithm scans $E(G)$. For each examined edge $e$, it initializes $\psi_e(v)$ for every $v \in e$ and then, if the value of $\bar{\delta}(e)$ is unknown (because some of the vertices in $e$ have not yet been evaluated), the algorithm "evaluates" this edge (Algorithm EVAL). Let $E'$ be the set of edges already scanned by the main loop. The *gap at vertex* $v$ is defined as the difference

$$\Psi(v) = \gamma(v) - \sum_{e \in E' | v \in e} \psi_e(v). \tag{4}$$

The "evaluation of $e$" consists of evaluating the vertices of $e$ which have not been evaluated up to this point in increasing order of their gaps. This evaluation of $e$ finishes when either of its vertices is evaluated and found to be false, or when the last vertex of $e$ is evaluated and found to be true. During this evaluation, the values of the variables $\psi_e(v)$ are updated and, in particular, the gap at every evaluated vertex becomes 0. In Algorithm EVAL, although not indicated, whenever we modify a variable $\psi_e(v)$, we shall update $\Psi(v)$ so as to maintain Eq. (4). Our algorithm is as follows:

---

**Algorithm.** $\mathcal{P}(G, \delta, \gamma)$

---

```
for each e ∈ E(G)
    for each v ∈ e do
        ψ_e(v) ← 0
    if δ̄(e) is unknown
        EVAL(e)
```

---

**Algorithm.** EVAL($e$)

---

(1) while $\bar{\delta}(e)$ is unknown
  (a) $v \leftarrow$ a vertex of minimum gap in $e$ which has not yet been evaluated
  (b) evaluate $v$
  (c) $\psi_e(v) \leftarrow \Psi(v)$

(2) $v' \leftarrow$ last evaluated vertex in the loop (1)

(3) for each $u \in e$ not yet evaluated
  $\psi_e(u) \leftarrow \psi_e(v')$

THEOREM 2.5. *Algorithm $\mathcal{P}$ is a deficiency-optimal algorithm for* sD$k$O *with time complexity $O(k|E(G)|\log|V(G)|)$.*

PROOF. Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance for sD$k$O, let $\mathsf{E}^*$ be an optimal evaluation for $\mathcal{I}$, and let

$$J = \{e \in E(G)\colon e \text{ is passed as a parameter to EVAL}\}.$$

As $\mathsf{E}^*$ is a cover for $G$, we have $\mathsf{E}^* \cap e \neq \emptyset$. For each $e \in J$, let $v'$ be the vertex of $\mathsf{E}^* \cap e$ with the maximum gap right before the execution of EVAL($e$). We must have that either $v'$ is the last vertex processed by the loop in EVAL or that $v'$ is not processed at all, for otherwise, the vertices in $\mathsf{E}^* \cap e$ would not suffice to determine the value of $\bar{\delta}(e)$. In either case, we have

$$\psi_e(v') = \max\{\psi_e(u)\colon u \in e\}$$

right after the execution of EVAL(e) and we can conclude that for every $e \in J$,

$$\sum_{v \in \mathsf{E}^* \cap e} \psi_e(v) \geq \max\{\psi_e(u)\colon u \in e\}.$$

Furthermore, upon the termination of $\mathcal{P}$, we have that $\gamma(v) \geq \sum_{e \in J} \psi_e(v)$, and then,

$$\gamma(\mathsf{E}^*) = \sum_{v \in \mathsf{E}^*} \gamma(v) \geq \sum_{v \in \mathsf{E}^*} \sum_{e \in J} \psi_e(v) \sum_{e \in J} \sum_{v \in \mathsf{E}^* \cap e} \psi_e(v) \geq \sum_{e \in J} \max\{\psi_e(v)\colon v \in e\}. \tag{5}$$

If $v$ is indeed evaluated, then $\gamma(v) = \sum_{e \in J} \psi_e(v)$, and it follows that

$$\gamma(\mathcal{P}(\mathcal{I})) = \sum_{v \in \mathcal{P}(\mathcal{I})} \sum_{e \in J} \psi_e(v) = \sum_{e \in J} \sum_{v \in \mathcal{P}(\mathcal{I})} \psi_e(v) \leq \sum_{e \in J} |e| \max\{\psi_e(v)\colon v \in e\}. \tag{6}$$

From Eqs. (5) and (6), it follows that

$$d(\mathcal{P}(\mathcal{I}), \mathcal{I}) = \frac{\mathbb{E}[\gamma(\mathcal{P}(\mathcal{I}))]}{\gamma(\mathsf{E}^*)} \leq |e| = k,$$

and, therefore $d(\mathcal{P}) = k$.

We now approach the time complexity of algorithm $\mathcal{P}$. For each edge $e$, the algorithm spends $O(k)$ to initialize the variables $\psi_e$. If the algorithm employs a balanced search tree to store the vertices of $V(G)$, then for each edge $e$, it spends time $O(k \log|V(G)|)$ to retrieve and update the required information about the vertices of $e$ and time $O(k \log k)$ to sort these vertices according to their gaps. Therefore, the overall time complexity of algorithm $\mathcal{P}$ is $O(k|E(G)|\log|V(G)|)$. □

We remark that our implementation does not require knowledge of the whole hypergraph in advance, which is a desirable property for the database application which motivates this work.

2.3. THE BIPARTITE CASE AND A RANDOMIZED ALGORITHM. In this section we restrict our attention to sD2O, the restricted case of sD$k$O in which $G$ is a bipartite graph. The *neighborhood* of $v \in V(G)$, denoted by $\Gamma(v)$, is the set of vertices

adjacent to $v$. For any $X \subseteq V(G)$, we let

$$\Gamma(X) = \bigcup_{v \in X} \Gamma(v)$$
$$\Gamma_1(X) = \bigcup_{v \in X | \delta(v)=1} \Gamma(v).$$

Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of sD2O, and let $C$ be a cover for $G$. We define the *$C$-evaluation for $\mathcal{I}$* as the set $\mathsf{E}(C) = C \cup \Gamma_1(C)$. It is not difficult to see that the $C$-evaluation for $\mathcal{I}$ is indeed an evaluation for $\mathcal{I}$. We call an algorithm for sD2O that always outputs $\mathsf{E}(C)$ for some cover $C$ a *cover-oriented algorithm for* sD2O.

As any evaluation for $(G, \delta)$ must contain some cover for $G$ and also must contain $\Gamma_1(V(G))$, it is not difficult to conclude that a $C$-evaluation for an instance of sD2O has deficiency of (at most) 2, whenever $C$ is a minimum cover for $(G, \gamma)$. This observation appears in Laber et al. [2002] with respect to dD2O. We call a cover-oriented algorithm for sD2O that always outputs $\mathsf{E}(C)$ for some minimum cover $C$ a *minimum-cover-oriented algorithm for* sD2O. Since 2 is a lower bound for the deficiency of any deterministic algorithm for sD2O (see Section 2.2), we have that any deterministic cover-oriented algorithm for sD2O is deficiency optimal.

Moreover, when $G$ is a bipartite graph, a minimum cover $C$ for $(G, \gamma)$ may be computed in time $O(|V(G)|^3)$ by reducing it to the problem of determining a minimum cut (maximum flow) on a directed graph with $|V(G)| + 2$ vertices and $|E(G)| + |V(G)|$ arcs (see Cook et al. [1997]). The evaluation $\mathsf{E}(C)$ can then be computed in time $O(|V(G)|)$. Therefore, it is possible to devise a minimum-cover-oriented algorithm for sD2O with time complexity $O(|V(G)|^3)$.

Let $0 \leq \varepsilon \leq 1 - \sqrt{2}/2$ . In this section, we introduce $\mathcal{R}_\varepsilon$, a polynomial time randomized algorithm for sD2O whose deficiency satisfies, for every instance $\mathcal{I}$,

$$d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 2 - \varepsilon \tag{7}$$

and

$$d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 1 + \frac{1}{1 - \varepsilon}. \tag{8}$$

Algorithm $\mathcal{R}_\varepsilon$ provides a tradeoff between expected deficiency and worst case deficiency. At one extreme, where $\varepsilon = 1 - \sqrt{2}/2$ , we have expected deficiency $1.707\ldots$ and worst case deficiency of 2.41. At the other extreme, $\varepsilon = 0$, $\mathcal{R}_\varepsilon$ actually becomes a deterministic algorithm with deficiency 2.

The key idea in the design of $\mathcal{R}_\varepsilon$ comes from trying to understand under which conditions a cover-oriented algorithm shows bad performance. More exactly, given a minimum cover $C$ for $(G, \gamma)$ and $\varepsilon > 0$, we turn our attention to the instances $\mathcal{I} = (G, \delta, \gamma)$ having $d(\mathsf{E}(C), \mathcal{I}) \geq 2 - \varepsilon$.

As suggested by the proofs of Theorems 2.1 and 2.3, one family of such instances can be constructed as follows. Consider an instance $\mathcal{I} = (G, \delta, \gamma)$ of sD2O, where $G$ is a matching of $n$ edges, $\delta(v) = 1$ for every $v \in V_1(G)$, $\delta(v) = 0$ for every $v \in V_2(G)$, and $\gamma(v) = 1$ for every $v \in V(G)$. Clearly, $V_2(G)$ is an optimum evaluation for $\mathcal{I}$, with cost $n$. On the other hand, note that the deficiency of a deterministic cover-oriented algorithm for sD2O depends on which of the $2^n$ minimum covers of $G$ is

chosen. In the particular case in which $C = V_1(G)$ is chosen, we have $d(\mathsf{E}(C), \mathcal{I}) = 2n/n = 2$.

This example suggests the following idea. If $C$ is a minimum cover for $(G, \gamma)$ and nonetheless, $\mathsf{E}(C)$ is not a good evaluation for $\mathcal{I} = (G, \delta, \gamma)$, then there must be another cover $C'$ of $G$ whose intersection with $C$ is small and still $C'$ is not far from being a minimum cover for $G$. Lemma 2.7 captures this idea formally. However, before presenting this lemma, we need an auxiliary result.

LEMMA 2.6. *Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of* sD2O*, let $T \subseteq \Gamma_1(V(G))$, and let $C_T$ be a minimum cover of $G - T$. Then $\gamma(C_T) + \gamma(T) \leq \gamma(\mathsf{E}^*_{\mathcal{I}})$.*

PROOF. From $T \subseteq \Gamma_1(V(G))$, it follows that $T \subset \mathsf{E}^*_{\mathcal{I}}$. Furthermore, $\mathsf{E}^*_{\mathcal{I}} - T$ is a cover for $G - T$, for otherwise $\bar{\delta}(uv)$ cannot be determined for some $uv \in E(G - T)$. Since $C_T$ is a minimum cover for $G - T$, it follows that $\gamma(C_T) \leq \gamma(\mathsf{E}^*_{\mathcal{I}} - T)$. Therefore $\gamma(C_T) + \gamma(T) \leq \gamma(\mathsf{E}^*_{\mathcal{I}} - T) + \gamma(T) = \gamma(\mathsf{E}^*_{\mathcal{I}})$. □

LEMMA 2.7. *Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of* sD2O*, let $C$ be a minimum cover for $(G, \gamma)$, and let $0 < \varepsilon < 1$. If $d(\mathsf{E}(C), \mathcal{I}) \geq 2 - \varepsilon$, then there is a vertex cover $C_\varepsilon$ for $G$ such that*

$$\gamma(C_\varepsilon) \leq \frac{\gamma(C - C_\varepsilon)}{1 - \varepsilon}. \tag{9}$$

PROOF. Let $T = \Gamma_1(C) - C$. Since $d(\mathsf{E}(C), \mathcal{I}) \geq 2 - \varepsilon$, it follows that $2 - \varepsilon \leq (\gamma(C) + \gamma(T))/\gamma(\mathsf{E}^*_{\mathcal{I}})$.

Let $C_T$ be a minimum cover for $G - T$. Since $T \subseteq \Gamma_1(V(G))$, it follows from Lemma 2.6 that $\gamma(C_T) + \gamma(T) \leq \gamma(\mathsf{E}^*_{\mathcal{I}})$.

Simple calculations give $\gamma(T) \leq (\gamma(C) - (2 - \varepsilon)\gamma(C_T))/(1 - \varepsilon)$.

Take $C_\varepsilon = C_T \cup T$ and note that $C_\varepsilon$ is a cover for $G$ with $(C_\varepsilon \cap C) \subseteq C_T$. Then,

$$\gamma(C_\varepsilon) \leq \gamma(C_T) + \gamma(T) \leq \frac{\gamma(C) - \gamma(C_T)}{1 - \varepsilon} \leq \frac{\gamma(C) - \gamma(C_\varepsilon \cap C)}{1 - \varepsilon} = \frac{\gamma(C - C_\varepsilon)}{1 - \varepsilon},$$

as required. □

Let $\mathcal{I} = (G, \delta, \gamma)$, and let $C$ and $\varepsilon$ be as in the statement of Lemma 2.7. Let $C'$ be a minimum cover for $(G, \gamma_{C,\varepsilon})$, where $\gamma_{C,\varepsilon}$ is given by

$$\gamma_{C,\varepsilon}(v) = \begin{cases} (1 - \varepsilon)\gamma(v), & \text{if } v \notin C; \\ (2 - \varepsilon)\gamma(v), & \text{otherwise.} \end{cases}$$

Formulating the problem of finding a cover $C_\varepsilon$ satisfying Eq. (9) as a linear system, we get to

$$\sum_{v \in V(G)} \gamma(v) x_v \leq \frac{\sum_{v \in C} \gamma(v)(1 - x_v)}{1 - \varepsilon}$$
$$x_u + x_v \geq 1, \text{ for every } uv \in E(G),$$

which has a solution in $\{0, 1\}^{|V(G)|}$ if and only if such a cover $C_\varepsilon$ exists. However, through simple algebraic manipulations, we realize that the previous system is

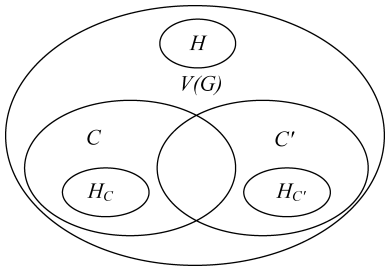


FIG. 2. Schematic view of the sets $V(G)$, $C$, $C'$, $H_C$, $H_{C'}$, and $H$.

equivalent to

$$\sum_{v \in C} (2-\varepsilon)\gamma(v)x_v + \sum_{v \in V(G)-C} (1-\varepsilon)\gamma(v)x_v \leq \gamma(C)$$
$$x_u + x_v \geq 1, \text{ for every } uv \in E(G),$$

and this linear system has a solution in $\{0, 1\}^{|V(G)|}$ if and only if $\gamma_{C,\varepsilon}(C') \leq \gamma(C)$. Furthermore, if $\gamma_{C,\varepsilon}(C') \leq \gamma(C)$, then

$$\gamma(C') \leq \frac{\gamma(C - C')}{1-\varepsilon}. \tag{10}$$

This last remark, together with Lemma 2.7, provides an efficient way to verify whether $\mathsf{E}(C)$ is a good evaluation for $(G, \delta, \gamma)$, given a minimum cover $C$ for $(G, \gamma)$.

As a cover $C'$ as previously given can be computed in polynomial time, we can devise a polynomial time randomized algorithm for sD2O which works as follows.

At the first step, the algorithm determines a minimum cover $C$ for $(G, \gamma)$ and a minimum cover $C'$ for $(G, \gamma_{C,\varepsilon})$. If $\gamma_{C,\varepsilon}(C') > \gamma(C)$, then Lemma 2.7 assures that $\mathsf{E}(C)$ will be a good evaluation for $(G, \delta, \gamma)$ and the algorithm returns $\mathsf{E}(C)$. Otherwise, the algorithm returns $\mathsf{E}(C)$ with probability $p = p(\varepsilon)$ or $\mathsf{E}(C')$ with probability $1 - p$ as the solution.

As $C$ is a minimum cover for $(G, \gamma)$, we have that $\gamma(C) \leq \gamma(C')$. It sounds reasonable to select $C$ with probability higher than $1/2$, that is, $p \geq 0.5$. Imposing this condition on the value of $p$ leads to $0 \leq \varepsilon \leq 1 - \sqrt{2}/2$.

2.3.1. *Algorithm Analysis.* The correctness of Algorithm $\mathcal{R}_\varepsilon$ follows from the fact that $\mathcal{R}_\varepsilon$ is a cover-oriented algorithm. Furthermore, the time needed for Algorithm $\mathcal{R}_\varepsilon$ is clearly asymptotically the same as that needed for computing two minimum-weight covers on a bipartite graph. Therefore, the time complexity of $\mathcal{R}_\varepsilon$ is $O(|V(G)|^3)$. Properties (7) and (8) of the evaluation computed by $\mathcal{R}_\varepsilon$, claimed at the beginning of Section 2.3, are proven in this section.

Let $\mathcal{I} = (G, \delta, \gamma)$, $C$, $C'$, and $\varepsilon$ be as in the statement of Algorithm $\mathcal{R}_\varepsilon$. It will be convenient to define the following sets (see Figure 2 for a schematic representation):

$$H = \Gamma_1(C \cap C') - (C \cup C'),$$
$$H_C = \Gamma_1(C') \cap (C - C'),$$
$$H_{C'} = \Gamma_1(C) \cap (C' - C).$$

The next result shows that any edge having an endpoint outside $C \cup C'$ must have its other endpoint in $C \cap C'$.

LEMMA 2.8. *If $C$ and $C'$ are two covers for $G$, then $\Gamma(V(G) - (C \cup C')) \subseteq C \cap C'$.*

PROOF. Let $v \in V(G) - (C \cup C')$ and $u \in \Gamma(v)$. We must have $u \in C$, otherwise $C$ would not cover $uv$. The same argument allows us to conclude $u \in C'$, and hence the result. □

---

**Algorithm.** $\mathcal{R}_\varepsilon(G, \delta, \gamma)$

---

(1) $C \leftarrow$ a minimum cover for $(G, \gamma)$
(2) $C' \leftarrow$ a minimum cover for $(G, \gamma_{C,\varepsilon})$
(3) If $\gamma_{C,\varepsilon}(C') > \gamma(C)$, return $\mathsf{E}(C)$
(4) $p \leftarrow \frac{1-3\varepsilon+\varepsilon^2}{1-2\varepsilon}$
(5) $x \leftarrow$ a random number uniformly chosen from $[0, 1]$.
(6) if $x < p$, return $\mathsf{E}(C)$, otherwise return $\mathsf{E}(C')$

---

With Lemma 2.8 in mind, we can write

$$H \cup H_{C'} = \Gamma_1(C) - C = \Gamma_1(C) - (C \cap \Gamma_1(C)),$$
$$H \cup H_C = \Gamma_1(C') - C' = \Gamma_1(C') - (C' \cap \Gamma_1(C')),$$

and, as $H, H_C, H_{C'} \subseteq \Gamma_1(V(G))$ are disjoint sets, we have

$$\gamma(\mathsf{E}(C)) = \gamma(C) + \gamma(H_{C'}) + \gamma(H), \tag{11}$$
$$\gamma(\mathsf{E}(C')) = \gamma(C') + \gamma(H_C) + \gamma(H), \tag{12}$$
$$\gamma(\mathsf{E}^*_{\mathcal{I}}) \geq \gamma(H) + \gamma(H_C) + \gamma(H_{C'}). \tag{13}$$

THEOREM 2.9. *Let $0 \leq \varepsilon \leq 1 - \sqrt{2}/2$. For any instance $\mathcal{I} = (G, \delta, \gamma)$ of* sD2O*, we have $d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 2 - \varepsilon$.*

PROOF. Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of sD2O and consider the execution of $\mathcal{R}_\varepsilon(\mathcal{I})$. If $\gamma_{C,\varepsilon}(C') > \gamma(C)$ at Step 3 of Algorithm $\mathcal{R}_\varepsilon$, it follows from Lemma 2.7 that $d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 2 - \varepsilon$.

For the case in which $\gamma_{C,\varepsilon}(C') \leq \gamma(C)$, we have that

$$\mathbb{E}[d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I})] = \mathbb{E}[\gamma(\mathcal{R}_\varepsilon(\mathcal{I}))] / \gamma(\mathsf{E}^*_{\mathcal{I}}).$$

From Eqs. (11) and (12), we have

$$\begin{aligned} \mathbb{E}[\gamma(\mathcal{R}_\varepsilon(\mathcal{I}))] &\leq p\gamma(\mathsf{E}(C)) + (1-p)\gamma(\mathsf{E}(C')) \\ &= p\gamma(C) + (1-p)\gamma(C') + (1-p)\gamma(H_C) + p\gamma(H_{C'}) + \gamma(H). \end{aligned} \tag{14}$$

Let $C_H$ be a minimum cover for $(G - H, \gamma)$. As every edge in $G - H$ is covered by $C_H$ and every edge incident to $H$ is covered by a vertex in $C \cap C'$, we conclude that $C_H \cup (C \cap C')$ is a cover for $G$.

As $C$ is a minimum cover for $G$, we have that

$$\gamma(C) \leq \gamma(C_H \cup (C \cap C')) \leq \gamma(C_H) + \gamma(C \cap C'),$$

or equivalently,

$$\gamma(C_H) \geq \gamma(C) - \gamma(C \cap C') = \gamma(C - C').$$

As $H \subseteq \Gamma_1(V(G))$, it follows from Lemma 2.6 that

$$\gamma(\mathsf{E}^*_{\mathcal{I}}) \geq \gamma(H) + \gamma(C_H) \geq \gamma(H) + \gamma(C - C'), \tag{15}$$

so that from the fact that $C$ is a minimum cover, as well as inequalities (13) and (15), we have

$$\gamma(\mathsf{E}^*_{\mathcal{I}}) \geq \max\{\gamma(C), \gamma(H) + \gamma(H_C) + \gamma(H_{C'}), \gamma(H) + \gamma(C - C')\}. \tag{16}$$

As the choice of $\varepsilon$ implies $p \geq 0.5$, it follows from Eqs. (14) and (16) that

$$\begin{aligned} d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) &\leq 2p + (1-p)\left(\frac{\gamma(C') + \gamma(H)}{\gamma(H) + \gamma(C - C')}\right) \\ &\leq 2p + (1-p) + \max\left\{\frac{\gamma(H)}{\gamma(H)}, \frac{\gamma(C')}{\gamma(C - C')}\right\}. \end{aligned}$$

Replacing $p$ by $(1 - 3\varepsilon + \varepsilon^2)/(1 - 2\varepsilon)$ and recalling inequality (10), we get $d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 2 - \varepsilon$. □

THEOREM 2.10. *Let $0 \leq \varepsilon \leq 1 - \sqrt{2}/2$. For any instance $\mathcal{I} = (G, \delta, \gamma)$ of* sD2O*, we have $d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 1 + 1/(1 - \varepsilon)$.*

PROOF. Let $C$ and $C'$ be as in Algorithm $\mathcal{R}_\varepsilon$. If $\mathcal{R}_\varepsilon(\mathcal{I}) = \mathsf{E}(C)$, then

$$d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 2 \leq 1 + \frac{1}{1 - \varepsilon}$$

because $C$ is a minimum cost cover. On the other hand, if $\mathcal{R}_\varepsilon(\mathcal{I}) = \mathsf{E}(C')$, then

$$d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq \frac{\gamma(C') + \gamma(H_C) + \gamma(H)}{\max\{\gamma(C), \gamma(H_C) + \gamma(H)\}} \leq 1 + \frac{\gamma(C')}{\gamma(C)}.$$

Thus, it follows from inequality (10) that

$$d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) \leq 1 + \frac{1}{1 - \varepsilon},$$

as required. □

To see that the bound given by inequality (7) is tight when $\varepsilon = 1 - \sqrt{2}/2$, let $\alpha \approx \sqrt{2}$ and consider the instance $\mathcal{I}_\alpha = (G, \delta, \gamma)$, where $G$ is a complete bipartite graph with $|V_2(G)| = \alpha |V_1(G)|$, $\delta(v) = 0$ for every $v \in V_1(G)$, $\delta(v) = 1$ for every $v \in V_2(G)$, and $\gamma(v) = 1$ for every $v \in V(G)$. Clearly, $V_1(G)$ is a minimum cover for $(G, \gamma)$. The set $V_2(G)$, however, is a minimum cover for $(G, \gamma_{C,\varepsilon})$ and $\gamma_{C,\varepsilon}(V_2(G)) \leq \gamma(V_1(G))$. Hence, $\mathcal{R}_\varepsilon(\mathcal{I})$ returns $\mathsf{E}(V_1(G)) = V_1(G)$ with probability $1/2$ and $\mathsf{E}(V_2(G)) = V(G)$ with probability $1/2$, so that the deficiency is $d(\mathcal{R}_\varepsilon(\mathcal{I}), \mathcal{I}) = 1 + \alpha/2 \approx 1 + \sqrt{2}/2$, as desired.

2.3.2. *Lower Bound for Randomized Algorithms.* We have proved in Section 2.3.1 that Algorithm $\mathcal{R}_\varepsilon$, with $\varepsilon = 1 - \sqrt{2}/2$, has deficiency bounded from above by $1 + \sqrt{2}/2 = 1.707\ldots$. However, $\mathcal{R}_\varepsilon$ does not achieve this deficiency with high probability. For the family of instances $\mathcal{I}_\alpha$, described at the end of the

previous section, $\mathcal{R}_\varepsilon$ attains deficiency $1+\alpha/2$ with probability $1/2$ and deficiency 1 with probability $1/2$. We can speculate whether a more dynamic algorithm would not have deficiency closer to the minimum possible value of 1.5 with higher probability. In this section, we prove that no randomized algorithm for sD2O can have deficiency smaller than $\mu$ for any given $1 \leq \mu \leq 2$ with probability close to 1 (see Theorem 2.13).

Let $1/2 \leq \lambda \leq 1$, let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of sD2O, and let $\mathcal{D}$ be a deterministic algorithm for sD2O. We define the *payoff of $\mathcal{D}$ with respect to $\mathcal{I}$* as

$$g(\mathcal{D}, \mathcal{I}) = \begin{cases} 1, & \text{if } \gamma(\mathcal{D}(\mathcal{I})) \geq \lambda |V(G)|; \\ 0, & \text{otherwise.} \end{cases}$$

The following result follows from Yao's Minimax principle.

THEOREM 2.11. *Let $\pi$ be a probability distribution with finite support on the set of instances of* sD2O. *For any randomized algorithm $\mathcal{R}$ for* sD2O*, we have that*

$$\begin{aligned} &\min\{\mathbb{E}_\pi\,[g(\mathcal{D}, \mathcal{I})] : \mathcal{D} \textit{ is a deterministic algorithm for } \mathsf{sD2O}\} \\ &\quad \leq \max\{\mathbb{E}\,[g(\mathcal{R}, \mathcal{I})] : \mathbb{P}_\pi(\mathcal{I}) > 0\}. \end{aligned} \tag{17}$$

Given that $g$ is the indicator random variable for the event $\gamma(\mathcal{D}(\mathcal{I})) \geq \lambda |V(G)|$, we can rewrite (17) as

$$\begin{aligned} &\min\{\mathbb{P}_\pi\,(\gamma(\mathcal{D}(\mathcal{I})) \geq \lambda |V(G)|) : \mathcal{D} \text{ is a deterministic algorithm for } \mathsf{sD2O}\} \\ &\quad \leq \max\{\mathbb{P}\,(\gamma(\mathcal{R}(\mathcal{I})) \geq \lambda |V(G)|) : \mathbb{P}_\pi(\mathcal{I}) > 0\}. \end{aligned} \tag{18}$$

In what follows, we define a probability distribution $\pi$ over the set of instances of sD2O and give a lower bound on the value of $\mathbb{E}_\pi\,[g(\mathcal{D}, \mathcal{I})]$ for any deterministic algorithm $\mathcal{D}$ for sD2O. This allows us to obtain a lower bound for the right-hand side of inequality (18).

Let $n$ be an even positive integer and $G$ be the complete bipartite graph with vertex classes $V_1(G) = \{1, \ldots, n/2\}$ and $V_2(G) = \{n/2+1, \ldots, n\}$. Let $\gamma(v) = 1$ for all $v \in V(G)$ and, for each $1 \leq i \leq n$, let

$$\delta_i(v) = \begin{cases} 1, & \text{if } v = i; \\ 0, & \text{otherwise.} \end{cases}$$

Consider the probability distribution $\pi$ in which the only instances with positive probability are $\mathcal{I}_i = (G, \delta_i, \gamma)$, $1 \leq i \leq n$, and all these instances to be equiprobable, with a probability of $1/n$ each. A key property of these instances is that the cost of the optimum evaluation for all of them is $n/2$, since all the vertices of the vertex class of the graph that does not contain the only true vertex must be evaluated. We have the following lemma.

LEMMA 2.12. *Let $\pi$ be the probability distribution over the set of instances of* sD$k$O*, defined earlier. For any deterministic algorithm $\mathcal{D}$ for* sD2O*:*

$$\mathbb{P}_\pi\,(\gamma(\mathcal{D}(\mathcal{I})) \geq \lambda n) \geq 1 - \lambda.$$

PROOF. Let $\mathcal{D}$ be a deterministic algorithm for sD2O. As mentioned previously, $\mathcal{D}$ must evaluate all vertices in the vertex class that does not contain the true vertex, and therefore $\gamma(\mathcal{D}(\mathcal{I}_i)) \geq n/2$ for every $1 \leq i \leq n$.

Let $j$ be the integer, determined as follows. Run $\mathcal{D}$ and reply $\delta(v) = 0$ for all vertices until it is about to evaluate all vertices from one vertex class. Let the very last vertex of this class be $j$. For convenience, also let this class be $V_1(G)$.

Let $v_1, v_2, \ldots, v_{n/2} = j$ be the vertices of $V_1(G)$ in the order they were evaluated by $\mathcal{D}$. Let $t = \lambda - 1/2$ and consider the instances $\mathcal{I}_{v_{\lceil tn \rceil}}, \mathcal{I}_{v_{\lceil tn \rceil + 1}}, \ldots, \mathcal{I}_{v_{n/2}}$. In all these instances, $\mathcal{D}$ evaluates at least $\lceil tn \rceil$ vertices in $V_1(G)$ and, as noted before, all vertices in $V_2(G)$. Thus, for at least $n/2 - \lceil tn \rceil + 1$ instances, $\mathcal{D}$ costs at least $n/2 + \lceil tn \rceil = \lceil \lambda n \rceil$ and hence,

$$\mathbb{P}_\pi \left( \gamma(\mathcal{D}(\mathcal{I})) \geq \lambda n \right) = \mathbb{E}_\pi \left[ g(\mathcal{D}, \mathcal{I}) \right] \geq \frac{n/2 - \lceil tn \rceil + 1}{n} \geq 1 - \lambda, \tag{19}$$

as required. □

THEOREM 2.13. *Let $\mathcal{R}$ be a randomized algorithm for* sD2O *and let $1 \leq \mu \leq 2$ be a real number. There is an instance $\mathcal{I}$ for which $\mathbb{P}\left(d(\mathcal{R}(\mathcal{I}), \mathcal{I}) \geq \mu\right) \geq 1 - \mu/2$.*

PROOF. Let $n > 0$ be an even number and $\pi$ be the probability distribution over the aforementioned set of instances of sD$k$O. Combining Eqs. (18) and (19), we have

$$\max \left\{ \mathbb{P}\left( \gamma(\mathcal{R}(\mathcal{I})) \geq \lambda n \right) : \mathbb{P}_\pi(\mathcal{I}) > 0 \right\} \geq 1 - \lambda.$$

Therefore, for some $1 \leq i \leq n$, the instance $\mathcal{I}_i$ satisfies

$$\mathbb{P}\left( \gamma(\mathcal{R}(\mathcal{I}_i)) \geq \lambda n \right) \geq 1 - \lambda. \tag{20}$$

As the cost of an optimal evaluation for $\mathcal{I}_i$ is $n/2$ for all $1 \leq i \leq n$, we have from (20) that

$$1 - \lambda \leq \mathbb{P}\left( \gamma(\mathcal{R}(\mathcal{I}_i)) \geq \lambda n \right) = \mathbb{P}\left( \frac{\gamma(\mathcal{R}(\mathcal{I}_i))}{n/2} \geq \frac{\lambda n}{n/2} \right) = \mathbb{P}\left( d(\mathcal{R}(\mathcal{I}_i), \mathcal{I}_i) \geq 2\lambda \right),$$

and then, taking $\mu = 2\lambda$, we get

$$\mathbb{P}\left( d(\mathcal{R}(\mathcal{I}_i), \mathcal{I}_i) \geq \mu \right) \geq 1 - \mu/2,$$

as required. □

## 3. *The Distributed Case*

In this section we study dD$k$O, the *Dynamic $k$-Partite Ordering* problem, in the setting where $k$ processors are available, each dedicated to processing the vertices of each vertex class of the input hypergraph. In this model, we assume that there is no communication cost whatsoever between the different processors.

In Section 3 we show that $k$ is a lower bound on the deficiency of any (not only polynomial) deterministic algorithm for dD$k$O. In Section 3 we introduce deterministic algorithms for dD$k$O. Section 3 introduces a slight modification to Algorithm $\mathcal{P}$ (from Section 2.2) which results in a deficiency-optimal deterministic algorithm with time complexity $O(k|E(G)| \log |V(G)|)$ for dD$k$O, under the preemptive model. In Section 3, we introduce a deterministic algorithm with time complexity $O(k|E(G)| \log |V(G)|)$ and deficiency $O(k \log^2 k)$, under the nonpreemptive model of dD$k$O.

Recall that, as discussed at the end of Section 1.1, an evaluation $\mathsf{E}$ in the distributed case is a complex object which holds, besides the vertices to be evaluated, information defining the scheduling of the evaluation of these vertices in corresponding processors. We will not need an analytic expression for the cost $\gamma(\mathsf{E})$ of an evaluation $\mathsf{E}$. Throughout this section, $\gamma(\mathsf{E})$ represents the time elapsed between the start of evaluation by the first processor and the end of evaluation by the last processor.

### 3.1. Lower Bounds

THEOREM 3.1. *Given a deterministic algorithm $\mathcal{A}$ for* dD$k$O*, there is an instance $(G_{\mathcal{A}}, \gamma_{\mathcal{A}}, \delta_{\mathcal{A}})$ such that the execution of $\mathcal{A}(G_{\mathcal{A}}, \gamma_{\mathcal{A}}, \delta_{\mathcal{A}})$ evaluates all the vertices of some vertex class.*

PROOF. Let $\mathcal{A}$ be a deterministic algorithm for dD$k$O. Let $G_{\mathcal{A}}$ be a matching and let $\gamma_{\mathcal{A}}(v) = 1$ for every $v \in V(G_{\mathcal{A}})$.

Note that since $G_{\mathcal{A}}$ is a matching, each vertex of $G_{\mathcal{A}}$ can be uniquely described by a pair $(i,\ a) \in \{1, \ldots, k\} \times E(G_{\mathcal{A}})$, and let us define $B$ as the complete bipartite graph where $V_1(B) = \{1, \ldots, k\}$ and $V_2(B) = E(G_{\mathcal{A}})$, so that there is a natural one-to-one correspondence between the edges of $B$ and the vertices of $G_{\mathcal{A}}$.

Our aim is to prove that there is a function $\delta_{\mathcal{A}}\colon V(G_{\mathcal{A}}) \to \{0, 1\}$ such that the execution of $\mathcal{A}(G_{\mathcal{A}}, \gamma_{\mathcal{A}}, \delta_{\mathcal{A}})$ evaluates all the vertices in $V_i(G)$ for some $1 \le i \le k$. As $\delta_{\mathcal{A}}$ is uniquely described by the set $\delta_{\mathcal{A}}^{-1}(0) = \{v \in V(G_{\mathcal{A}})\colon \delta(v) = 0\}$, we will think of $\delta_{\mathcal{A}}$ in terms of the set of edges of $B$ corresponding to $\delta_{\mathcal{A}}^{-1}(0)$.

To describe a worst-case scenario for $\mathcal{A}$, we define a two-player game in which the algorithm is one of the players and its adversary forces a "bad performance" of $\mathcal{A}$. For convenience, we describe our game in terms of the graph $B$, defined earlier.

As previously noted, any 0-1-function on $V(G_{\mathcal{A}})$ is uniquely described by a subset of $V(G_{\mathcal{A}})$, which, in turn, uniquely corresponds to a subset of $E(B)$. The idea of our game will be that the adversary chooses a set $F \subseteq E(B)$ and the algorithm has to discover the set $F$ by picking edges of $B$, one-by-one, and asking the adversary at each move whether this edge is in $F$.

Given a complete bipartite graph and an integer $0 < t \le |V_2(B)|$, let us describe the game $\mathcal{G}(B, t)$ between two players, the hider and the seeker. In this game, $B$ and $t$ are given to both players and the hider chooses $F \subseteq E(B)$ (which she keeps hidden from the seeker) such that

$$\deg_{B[F]}(v) = \begin{cases} t, & \text{if } v \in V_1(B); \\ 1, & \text{if } v \in V_2(B). \end{cases}$$

A *move of the game* is a pair $(e, a) \in E(B) \times \{\text{yes}, \text{no}\}$ which is interpreted as the seeker querying "*does $e \in F$?*" and the hider answering yes or no, accordingly. The game finishes when the seeker has determined $F$. A *realization of the game* is a sequence of moves in the game from beginning to end.

The goal of the seeker is to determine $F$. However, note that since $F$ does not have to be disclosed until the very end, the hider does not have to make up her mind about which $F$ to pick at the beginning; she may answer the queries as the game evolves, just making sure that her answers are consistent with *some* choice of $F$.

The following theorem is stated without proof for space considerations.

THEOREM 3.2. *Let $B$, $t$, and $\mathcal{G}$ be as previously described. There is a strategy for the* hider *in the game $\mathcal{G}(B, t)$ which forces the* seeker *to query every edge adjacent to some vertex $v \in V_1(B)$ in any realization of the game $\mathcal{G}(B, t)$.*

PROOF. For the proof, we refer the reader to Carmo et al. [2004]. □

Translated back into the setting of the original problem, Theorem 3.2 states that for every deterministic algorithm $\mathcal{A}$ for dD$k$O, there is a function $\delta_{\mathcal{A}}\colon V(G_{\mathcal{A}}) \to \{0, 1\}$ which forces $\mathcal{A}(G_{\mathcal{A}}, \gamma_{\mathcal{A}}, \delta_{\mathcal{A}})$ to evaluate $\delta_{\mathcal{A}}(v)$ for every $v \in V_i(G)$ for some $1 \leq i \leq k$.

COROLLARY 3.3. *Every deterministic algorithm for* dD$k$O*, whether in the preemptive or the nonpreemptive model, has a deficiency of at least, $k$.*

PROOF. Let $\mathcal{A}$ be a deterministic algorithm for dD$k$O. Let $t > 0$ be given and let $\mathcal{I}_{\mathcal{A}} = (G_{\mathcal{A}}, \gamma_{\mathcal{A}}, \delta_{\mathcal{A}})$ be an instance of dD$k$O, as in the proof of Theorem 3.1, with $|E(G_{\mathcal{A}})| = tk$ ($t$ will play the same role as in the game $\mathcal{G}(B, t)$ defined before). Note that $G_{\mathcal{A}}$ has $tk^2$ vertices, with exactly $tk$ of them in each vertex class. In addition, each vertex class has exactly $t$ false vertices.

Let $\mathsf{E}$ be the evaluation of $\mathcal{I}_{\mathcal{A}}$ in which only the false vertices are evaluated, all of which are in parallel with no idle processing. As each vertex class of $G_{\mathcal{A}}$ has exactly $t$ false vertices, we have that $\gamma(\mathsf{E}) = t$. However, according to Theorem 3.1, when given $\mathcal{I}_{\mathcal{A}}$ as input, Algorithm $\mathcal{A}$ evaluates $\delta_{\mathcal{A}}$ for every vertex in some of the vertex classes, say $V_1(G_{\mathcal{A}})$, and hence

$$d(\mathcal{A}) \geq d(\mathcal{A}(\mathcal{I}_{\mathcal{A}}), \mathcal{I}_{\mathcal{A}}) \geq \frac{\gamma(\mathcal{A}(\mathcal{I}_{\mathcal{A}}))}{\gamma(\mathsf{E})} \geq \frac{|V_1(G_{\mathcal{A}})|}{t} = \frac{tk}{t} = k,$$

as required. □

### 3.2. DETERMINISTIC ALGORITHMS FOR dD$k$O

3.2.1. *The Preemptive Model*The following result formalizes a natural lower bound for the cost of the optimal solution of an instance of dD$k$O in terms of the cost of the optimal solution of the same instance of sD$k$O. This bound will be used in the proof of Theorem 3.5.

LEMMA 3.4. *Let $\mathcal{I}$ be an instance of* dD$k$O *and $\gamma^*_{\mathcal{I}}$ denote the cost of an optimal evaluation for $\mathcal{I}$* as an instance of sD$k$O. *Then*

$$\gamma(E^*_{\mathcal{I}}) \geq \frac{\gamma^*_{\mathcal{I}}}{k}, \tag{21}$$

*where $E^*_{\mathcal{I}}$ is an optimal evaluation for $\mathcal{I}$ as an instance of* dD$k$O.

PROOF. For each $1 \leq i \leq k$, let $\gamma_i(\mathsf{E}^*_{\mathcal{I}})$ denote the cost incurred by the processor in charge of $V_i(G)$ in $\mathsf{E}^*_{\mathcal{I}}$. We clearly have

$$\sum_{i=1}^{k} \gamma_i(\mathsf{E}^*_{\mathcal{I}}) \geq \gamma^*_{\mathcal{I}},$$

**Algorithm.** PAREVAL(e)

while $\bar{\delta}(e)$ is unknown
(1) $v \leftarrow$ a vertex of minimum gap in $e$ which has not yet been evaluated
(2) evaluate in parallel every vertex of $e$ (whose evaluation has not completed yet) for $\Psi(v)$ units of time
(3) for each vertex $u$ evaluated in Step 2 do $\psi_e(u) \leftarrow \psi_e(u) + \Psi(v)$

and hence

$$\gamma(\mathsf{E}^*_{\mathcal{I}}) = \max\{\gamma_i(\mathsf{E}^*_{\mathcal{I}})\colon 1 \le i \le k\} \ge \frac{1}{k}\sum_{i=1}^{k}\gamma_i(\mathsf{E}^*_{\mathcal{I}}) \ge \frac{\gamma^*_{\mathcal{I}}}{k},$$

as required. □

Let us define $\mathcal{P}'$ as the algorithm resulting by replacing the procedure EVAL (see Section 2.2) by the procedure PAREVAL.

THEOREM 3.5. *Algorithm $\mathcal{P}'$ is a deficiency-optimal algorithm for* sD$k$O *in the preemptive model with time complexity* $O(k|E(G)|\log|V(G)|)$.

PROOF. Let $\mathcal{I} = (G, \delta, \gamma)$ be an instance of sD$k$O, and $\gamma^*_{\mathcal{I}}$ denote the cost of the optimal evaluation for $\mathcal{I}$ *as an instance of* sD$k$O. For each $1 \le i \le k$, let $\gamma_i(\mathcal{P}'(\mathcal{I}))$ denote the cost incurred by the processor in charge of $V_i(G)$ in the evaluation $\mathcal{P}'(\mathcal{I})$. Without loss of generality, we assume $\gamma_1(\mathcal{P}'(\mathcal{I})) = \max\{\gamma_i(\mathcal{P}'(\mathcal{I}))\colon 1 \le i \le k\}$.

Let $J = \{e \in E(G)\colon e$ is passed as a parameter to PAREVAL$\}$. Upon the termination of $\mathcal{P}'$, we have

$$\gamma_1(\mathcal{P}'(\mathcal{I})) \le \sum_{e\in J}\max\{\psi_e(x)\colon x \in e\} \le \gamma^*_{\mathcal{I}},$$

where the rightmost inequality follows from Eq. (5), since both EVAL and PAREVAL generate the same values for the variables $\psi$. Hence, with Eq. (21), we get

$$d(\mathcal{P}'(\mathcal{I}), \mathcal{I}) = \frac{\gamma(\mathcal{P}'(\mathcal{I}))}{\gamma(\mathsf{E}^*_{\mathcal{I}})} \le \frac{\gamma_1(\mathcal{P}'(\mathcal{I}))}{\gamma^*_{\mathcal{I}}/k} \le \frac{\gamma^*_{\mathcal{I}}}{\gamma^*_{\mathcal{I}}/k} = k,$$

as required. □

3.2.2. *The Nonpreemptive Model*

THEOREM 3.6. *For the nonpreemptive model of* sD$k$O*, there is a deterministic algorithm with time complexity* $O(k|E(G)|\log|V(G)|)$ *and deficiency* $k^2 - k + 1$. *This bound can be improved to* $k(2\log_5(k-1) + c_k)^2 + 1$ *for* $k \ge 13$, *with* $0.2229 < c_k < 0.5177$.

PROOF. To obtain a factor of $k^2 - k + 1$, we apply Algorithm $\mathcal{P}'$, but we delay the evaluation of the vertices until we have at least one vertex in each edge with its cost reduced to 0. We then evaluate all these vertices with total cost that does not exceed the cost for sD$k$O, and thus (at most) $k$ times the cost for sD$k$O in the preemptive model. After this first evaluation phase, the rank of the graph has been reduced to $k-1$, then the next evaluation reduces the rank of the graph to $k-2$, and so on. After $k-1$ phases, the rank of the graph is reduced to 1, and the cost incurred so far is (at most) $(k-1)k$ times the cost for sD$k$O in the preemptive model.

The last phase has edges of size 1, and thus incurs, at most, the cost for dD$k$O in the preemptive model, giving at total cost of at most $(k-1)k+1$ times the cost for dD$k$O in the preemptive model.

For the improved bound, consider the $k$ phases just described for the nonpreemptive model of dD$k$O. The last phase, with edges of size 1, evaluates vertices that must be evaluated, so if these vertices are evaluated, we incur (at most) the cost for dD$k$O in the preemptive model. Consider the first $k-1$ phases. Suppose we are considering $\lceil x \rceil \leq k-1$ consecutive phases, and a vertex $v$ with cost $\gamma(v)$ whose evaluation is done within these $\lceil x \rceil$ phases. Partition the $\lceil x \rceil$ phases into 5 consecutive subsets of, at most, $\lceil x/5 \rceil$ phases. Set $\alpha_x = \frac{1}{2}(2\log_5 x + c)$ for some constant $c$, and consider $\beta_x$ such that $1/\alpha_x + 1/\beta_x = 1$. If a fraction of at least $\gamma(v)/(4\alpha_x)$ of $\gamma(v)$ is evaluated during one of these 5 subsets without completing the evaluation of $v$, then $v$ may be fully evaluated at the end of this subset of phases. The cost of this evaluation is at most $4\alpha_x$ times the cost for sD$k$O. Otherwise, a fraction of at least $\gamma(v)/\beta_x$ is evaluated in a subset of phases that completes the evaluation of $v$. This gives the recurrence $f(x) = 4\alpha_x + f(x/5)\beta_x$, with the total cost for the algorithm given by $kf(k-1)+1$. The recurrence resolves to $f(x) = (2\alpha_x)^2$, since $(2\alpha_x)^2 = 4\alpha_x + (2\alpha_x - 2)^2(\alpha_x/(\alpha_x - 1))$. For the base case with $4 < x \leq 20$, if $x \leq rs$ for integers $r$ and $s$, then we may decompose the $\lceil x \rceil$ phases into $r$ groups of at most $s$ phases, so the recurrence gives $f(x) = (r-1)\alpha + s\beta = \left(\sqrt{r-1} + \sqrt{s}\right)^2$ for an appropriate choice of $\alpha, \beta$ with $1/\alpha + 1/\beta = 1$. At this point, we may verify that $f(x) = (2\alpha_x)^2$ for an appropriate choice of $c$ for each $4 < x \leq 20$ satisfying $0.2229 < \sqrt{6} - 2\log_5 6 \leq c \leq \sqrt{13} - 2\log_5 12 < 0.5177$. □

## 4. *Open Problems*

Several problems remain open. We single out the following:

—Is there a randomized algorithm for sD$k$O with deficiency $(k+1)/2$? We do not know the answer, even if exponential time is allowed and $k = 2$.

—Is there a polynomial time algorithm for dD$k$O with deficiency $k$ for the nonpreemptive model?

As noted in Section 1.4, in this work, we restricted our attention to conjunctive queries (in the sense of (1)). However, much more general queries could be considered. For example, $\overline{\delta}\colon E(G) \to \{0, 1\}$ could be any formula in the first-order propositional calculus involving the predicates represented by $\delta$. It would be very interesting to investigate D$k$O with such generalized queries.

ACKNOWLEDGMENTS. The authors are most grateful to the referee for the detailed and helpful comments.

## REFERENCES

AVNUR, R., AND HELLERSTEIN, J. M. 2000. Eddies: Continuously adaptive query processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (May 16–18, Dallas, TX), W. Chen et al. Eds. 261–272.

BOUGANIM, L., FABRET, F., PORTO, F., AND VALDURIEZ, P. 2001. Processing queries with expensive functions and large objects in distributed mediator systems. In *Proceedings of the 17th International Conference on Data Engineering* (Apr. 2–6, Heidelberg, Germany). 91–98.

CARMO, R., FEDER, T., KOHAYAKAWA, Y., LABER, E., MOTWANI, R., O'CALLAGHAN, L., PANIGRAHY, R., AND THOMAS, D. 2004. A two–player game on graph factors. Tech. Rep. RT-MAC 2004-05, IME-USP. available at http://www.ime.usp.br/~renato/text/A_Two-Player_Game_on_Graph_Factors.

CHARIKAR, M., FAGIN, R., GURUSWAMI, V., KLEINBERG, J., RAGHAVAN, P., AND SAHAI, A. 2002. Query strategies for priced information. *J. Comput. Syst. Sci. 64*, 4, 785–819.

CHAUDHURI, S., AND SHIM, K. 1993. Query optimization in the presence of foreign functions. In *Proceedings of the 19th International Conference on Very Large Data Bases* (Aug. 24–27, Dublin, Ireland) 529–542.

CICALESE, F., AND LABER, E. 2005. A new strategy for qurying priced information. In *Proceedings of the 37th Annual ACM Symposium on the Theory of Computing*. 136–146.

CICALESE, F., AND LABER, E. 2006. On the competitive ratio of evaluating priced information. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*. 944–953.

COOK, W. J., CUNNINGHAM, W. H., PULLEYBLANK, W. R., AND SCHRIJVER, A. 1997. *Combinatorial Optimiation*. John Wiley, new York.

HELLERSTEIN, J. M. 1998. Optimization techniques for queries with expensive methods. *ACM Trans. Database Syst. 23*, 2 (Jun.), 113–157.

LABER, E., CARMO, R., AND KOHAYAKAWA, Y. 2004. Querying priced informaaation in databases: The conjunctive casse: Extended abstract. In *LATIN: Latin American Symposium on Theoretical Informatics*.

LABER, E. 2004. A randomized competitive algorithm for evaluating priced and/or trees. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science*.

LABER, E. S., PAREKH, O., AND RAVI, R. 2002. Randomized approximation algorithms for query optimization problems on two processors. In *Proceedings of the 10th Annual European Symposium* (Rome). 136–146.

PORTO, F. 2001. Estratégias para a execuçáo paralela de consultas em bases de dados científicos distribuídos (strategies for the parallel execution of queries in distributed scientific databases). Ph.D. thesis, Departamento de Informática, PUC-Rio.

YAO, A. C. 1977. Probabilistic computations : Toward a unified measure of complexity. In *18th Annual Symposium on the Foundations of Computer Science*. 222–227.

RECEIVED JULY 2004; REVISED FEBRUARY 2006; ACCEPTED JULY 2006