



# Minimization of Testing Costs in Capacity-Constrained Database Migration

K. Subramani<sup>1</sup>(✉), Bugra Caskurlu<sup>2</sup>, and Alvaro Velasquez<sup>3</sup>

<sup>1</sup> LCSEE, West Virginia University, Morgantown, WV, USA  
k.subramani@mail.wvu.edu

<sup>2</sup> CE, TOBB University of Economics and Technology, Ankara, Turkey  
caskurlu@gmail.com

<sup>3</sup> RISC, Air Force Research Laboratory, Rome, NY, USA  
alvaro.velasquez@us.af.mil

**Abstract.** Database migration is an ubiquitous need faced by enterprises that generate and use vast amount of data. This is due to database software updates, or from changes to hardware, project standards, and other business factors [1]. Migrating a large collection of databases is a way more challenging task than migrating a single database, due to the presence of additional constraints. These constraints include capacities of shifts, sizes of databases, and timing relationships. In this paper, we present a comprehensive framework that can be used to model database migration problems of different enterprises with customized constraints, by appropriately instantiating the parameters of the framework. We establish the computational complexities of a number of instantiations of this framework. We present fixed-parameter intractability results for various relevant parameters of the database migration problem. Finally, we discuss a randomized approximation algorithm for an interesting instantiation.

## 1 Introduction

The database migration problem entails the movement of data between different databases. Such migration is often necessary due to database software updates, or from changes to hardware or project standards, and other business factors [1]. As per established rules of software reliability, when a database is migrated, every application that is dependent upon it *must* be tested (i.e., run through regression suites [2]). It is known that testing an application is an expensive aspect of maintaining the application [3]. Consequently, the principal goal in the migration process is to minimize the application testing cost [4]. This is a

---

K. Subramani—This research was supported in part by the Air Force Research Laboratory Information Directorate, through the Air Force Office of Scientific Research Summer Faculty Fellowship Program and the Information Institute®, contract numbers FA8750-16-3-6003 and FA9550-15-F-0001.

pervasive issue in cloud computing clusters, where a pay-per-use infrastructure and unpredictable workloads necessitate frequent allocation and movement of data [5]. Thus, there is a pressing need for efficient procedures to minimize the resource overhead involved in data migration. Interest in this area has also been fueled in recent years by the massive generation of data in what is now being called the Big Data age [6]. Consequently, there has been a proliferation of resource-intensive data centers and the adoption of cloud computing and storage as a service in order to manage said data [7].

While the ubiquity of computational and storage capabilities of cloud computing are undeniable, there remain open challenges with regards to resource allocation and data management [8]. In fact, the migration of data in data centers remains a significant problem, due to the massive throughput of data and the limited bandwidth of communication channels [9]. This problem is further exacerbated by the overhead incurred from retesting applications after data migration and by Quality-of-Service (QoS) requirements, which demand minimal interruptions to end-user applications [10]. As such, minimizing the cost associated with bandwidth-constrained database migration is of great interest. Patil et al. [11] introduced the first systematic study on the database migration problem, and proved that the problem is **NP-hard** for some special cases. They provided an integer programming formulation, which can only be used for small instances, and a greedy heuristic that can be used for the large instances of the problem. In this paper, we define a very general framework that subsumes the model in [11]. Our framework accommodates the modelling of database migration needs of various enterprises with customized constraints. We present hardness results for all the models in our framework, as well as fixed-parameter intractability for various relevant parameters. We also present a randomized approximation algorithm for a simple but interesting special case of the problem.

The rest of the paper is organized as follows:

- In Sect. 2, we formally define the capacity-constrained database migration (CCDM) problem and introduce the notation used in the paper.
- In Sect. 3, we study the computational complexity of the (CCDM) problem and prove that the problem is **NP-hard** for all the models of the problem defined in Sect. 2.
- In Sect. 4, we study fixed parameter tractability of the CCDM problem with respect to the four most relevant parameters of the problem.
- In Sect. 5, we present a randomized  $(\frac{3}{2} + \epsilon)$ -approximation algorithm for a special case of the CCDM problem.
- In Sect. 6, we summarize our contributions and point out avenues for future research.

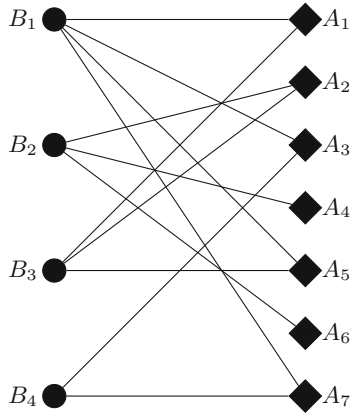
## 2 Notations and Problem Formulation

In this section, we provide a formal definition of the framework for the capacity constrained database migration problem (CCDM) and introduce the terminology used in this paper.

Assume we have a collection of  $m$  applications  $\mathcal{A} = \{A_1, A_2, \dots, A_m\}$  and  $n$  databases  $\mathcal{B} = \{B_1, B_2, \dots, B_n\}$ , with each application calling one or more databases. The call relationship is stored in the  $n \times m$  matrix  $\mathbf{D} = [d_{ij}]$ , where

$$d_{ij} = \begin{cases} 1, & \text{if application } A_i \text{ calls database } B_j \\ 0, & \text{otherwise.} \end{cases}$$

The matrix  $\mathbf{D} = [d_{ij}]$ , which represents a bipartite graph as shown in Fig. 1, is part of the input. Associated with the set of applications  $\mathcal{A}$ , is a cost-vector  $\mathbf{c} = [c_1, c_2, \dots, c_m]^T$ , where  $c_i$  represents the cost of testing application  $A_i$  once. For each application  $A_i$ , we let  $x_i$  be an integer variable that denotes the number of times  $A_i$  will have to be tested in the migration schedule. The size-vector  $\mathbf{w} = [w_1, w_2, \dots, w_m]^T$  represents the size of databases, with  $w_i$  representing the size of database  $B_i$ .



**Fig. 1.** The bipartite graph shows the relationship between the databases and the applications. The nodes in the left partition, represent the databases in the system, while the nodes in the right partition, represent the applications. An edge  $(b, a)$  exists in the graph if application  $a$  calls database  $b$ . This means application  $a$  must be tested immediately after database  $b$  migrates. We note that each database is associated with a nonempty set of applications, and each application is associated with a nonempty set of databases.

In the CCDM problem, the set of databases  $\mathcal{B}$  is to be clustered into disjoint subsets which we call shifts. The databases in each shift are migrated at the same time. When a shift of databases migrates, each application that calls at least one database in that shift needs to be tested immediately. For example, if the set of databases called by an application  $A_i$  are scheduled to 5 distinct shifts, then application  $A_i$  is to be tested 5 times throughout the migration process, i.e.,  $x_i = 5$ . The cumulative size of the databases migrated in shift  $i$  (i.e., the size of shift  $i$ ) is denoted by  $l_i$ . The shift size-vector  $\mathbf{l} = [l_1, l_2, \dots, l_m]^T$  is also

part of the input. In the worst case, when the size of each shift is smaller than the sum of the sizes of any two databases, we may have to assign each database to a separate shift.

Thus, the input to the CCDM problem must contain the 4-tuple  $\langle \mathbf{c}, \mathbf{w}, \mathbf{D}, \mathbf{l} \rangle$ . For instance, consider the following 4-tuple:

$$\left\langle \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 7 \\ 10 \\ 12 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 10 \\ 15 \\ 12 \\ 7 \\ 28 \\ 34 \end{pmatrix} \right\rangle$$

In this example, we have seven applications and four databases, since the matrix  $\mathbf{D}$  has seven columns and four rows. We label the applications as  $A_1, A_2, \dots, A_7$ , and the databases as  $B_1, B_2, B_3, B_4$ . Matrix  $\mathbf{D}$  indicates which applications call which databases. We observe that database  $B_1$  is called by applications  $A_1, A_3, A_5$ , and  $A_7$ . The database  $B_2$  is called by applications  $A_2, A_4$ , and  $A_6$ . The database  $B_3$  is called by applications  $A_1, A_2$ , and  $A_5$ , and database  $B_4$  is called by applications  $A_3$ , and  $A_7$ . The sizes of the databases  $B_1, B_2, B_3$ , and  $B_4$  are given as 5, 7, 10, and 12, respectively, in  $\mathbf{w}$ . The database migration operation in this example is to be completed in at most 6 shifts since  $|\mathbf{l}| = 6$  in the input. The cumulative size of the databases that migrate in each shift is constrained by 10, 15, 12, 7, 28, and 34, respectively, in  $\mathbf{l}$ .

There are several parameters associated with the CCDM problem:

- (i) Application Testing cost ( $\alpha$ ) - The testing cost of an application depends primarily on two factors, viz., the time it takes to test the application, and the skills required to test a given application. We consider the following three cost models associated with testing an application in order to take these factors into account:
  - (a) Constant (*const*) - In this model, the cost of testing each application is the same and is equal to some known fixed constant  $C$ . Note that this model captures the scenario in which each application requires the same level of skills and takes roughly the same time to test.
  - (b) Proportional (*prop*) - In this model, the cost of testing an application after migrating a corresponding database is proportional to the sizes of the migrated databases it calls. Note that this model captures the situation where each application in the system requires the same level of skills, but the testing time may vary from application to application.
  - (c) Arbitrary (*arb*) - In this model, there is no relation among the costs of testing different applications. This model captures the problem of companies that use application software from several different companies such that each application requires personnel with different skill sets.
- (ii) Size of Databases ( $\beta$ ) - The size of a database is a factor in the amount of time required for its migration. This is because the database migration

operation must read the database at the original location, write it at the new location, and then delete the original database. Consider a bank that maintains data for credit card accounts, savings accounts, and checking accounts in different databases. Typically, a bank will have more customers with a checking account than a savings account. Similarly, the number of credit card customers will be significantly more than the number of savings account customers, since a typical customer has one savings account but several credit cards. This means we need two size models associated with the databases:

- (a) Constant (*const*) - In this model, all databases have the same size and are equal to some fixed constant  $W$ . This allows us to model the database migration operation for companies whose databases have more or less the same size.
  - (b) Arbitrary (*arb*) - In this model, the sizes of the databases are arbitrary. This lets us model the database migration operation for companies whose databases may significantly vary in size.
- (iii) Shift size ( $\gamma$ ) - During the database migration operation, some parts of the database will be inaccessible. For some companies, there is no ideal time to make a database unavailable. For instance, Facebook and Youtube have users all over the world which means that the database access rate is roughly uniform. In this case, regardless of when a database becomes inaccessible, there will be a subset of users who cannot access the database until the migration is complete. It is critical for these companies to perform the database migration operation in small shifts to minimize user dissatisfaction. For companies (e.g., banks) that operate during regular business hours, it is preferable for the databases to be unavailable when the companies are closed rather than when they are open. In order to model the needs of several different companies, we use two size models associated with shifts.
- (a) Uniform (*unif*) - In this model, the total size of the databases migrated in the same shift is the same and is equal to a constant  $L$ , for all shifts (i.e.,  $\mathbf{l} = \langle L, L, \dots, L \rangle$ ). We note that this model is better suited for the database migration needs of companies that have uniform database access rates.
  - (b) Non-uniform (*non-unif*) - In this model, the total size of the databases migrated in each shift is arbitrary. We note that this model is better suited for the database migration needs of companies that have non-uniform database access rates.

Thus, a model of the capacity-constrained database migration problem has three parameters, and it is specified as a triple  $\langle \alpha \mid \beta \mid \gamma \rangle$ . For instance,  $\langle arb \mid const \mid unif \rangle$  refers to the capacity-constrained database migration problem in which the application testing costs are arbitrary, all databases have the same size, and the shift sizes are uniform. For notational convenience we use  $*$  as an entry of the triple when we present a statement that is true for all the models for that entry. For instance, the notation  $\langle arb \mid * \mid * \rangle$  refers to all 4 models

of the CCDM problem in which the application testing costs are arbitrary. The following is formal definition of the CCDM problem:

**CCDM:** *Given a 4-tuple  $\langle \mathbf{c}, \mathbf{w}, \mathbf{D}, \mathbf{l} \rangle$ , cluster the databases into shifts so that the total application testing cost is minimized, while respecting the shift size constraints.*

Given that we have 3 different models for application testing costs, 2 different models for sizes of databases, 2 different models for shift sizes; the CCDM problem formulation gives us a framework with a total of 12 different models each of which is suitable for the database migration needs of different companies.

### 3 Computational Complexity of the CCDM Problem

The formulation given for the database migration problem in [11] corresponds to our CCDM problem under the model  $\langle arb \mid arb \mid unif \rangle$ . In [11], it is proven that the CCDM problem is **NP-hard** under the model  $\langle arb \mid * \mid * \rangle$ . In this section, we strengthen the result in [11] via Theorem 1, which states that the CCDM problem is **NP-hard** for all the models in our framework, even under the restriction that there are only 2 shifts and each application calls at most 2 databases.

**Theorem 1.** *The CCDM problem in models  $\langle * \mid * \mid * \rangle$  is **NP-hard** even under the restrictions that there are only two shifts, and each application calls at most two databases.*

*Proof.* Since the set of instances of the capacity-constrained database migration problem under the model  $\langle const \mid const \mid unif \rangle$  is a subset of the instances of any model captured by the notation  $\langle * \mid * \mid * \rangle$ , all we need to do is to prove that the CCDM problem is **NP-hard** under the model  $\langle const \mid const \mid unif \rangle$ , when there are only two shifts, and each application calls at most two databases. We will do this via a polynomial time reduction from the classical MINIMUM-BISECTION problem [12], whose definition is given below.

**Definition 1 (MINIMUM-BISECTION).** *Given an undirected graph  $G = (V, E)$ , partition  $V$  into two subsets  $V_1$  and  $V_2$  of equal size such that the number of edges with one endpoint in  $V_1$  and one endpoint in  $V_2$  are minimized. It is assumed  $|V|$  is even.*

For a given instance  $G = (V, E)$  of the MINIMUM-BISECTION problem, we construct the corresponding CCDM instance in model  $\langle const \mid const \mid unif \rangle$  as follows:

- For every vertex  $i$  of the graph of the MINIMUM-BISECTION instance, the CCDM instance has a corresponding database  $B_i$  with unit size, i.e.,  $w_i = 1$ ,
- For every edge  $e = (i, j)$  of the graph of the MINIMUM-BISECTION instance, the CCDM instance has a corresponding application  $A_e$  with unit application testing cost ( $c_e = 1$ ) that calls databases  $B_i$  and  $B_j$ ,

- The CCDM instance has only two shifts and the size of each shift is  $\frac{|V|}{2}$ .

Notice that the constructed CCDM instance has  $|V|$  databases and  $|E|$  applications such that each application calls exactly two databases. In any feasible solution to the CCDM instance exactly  $k = \frac{|V|}{2}$  databases are assigned to the first shift and the remaining  $k$  databases are assigned to the second shift. If both of the databases called by an application  $A_i$  are assigned to the same shift then application  $A_i$  needs to be tested only once ( $x_i = 1$ ) in the database migration process, and it needs to be tested twice ( $x_i = 2$ ) otherwise. Let  $d$  denote the number of applications that calls one database from each shift, and thus needs to be tested twice. Then, the total application testing cost of the CCDM instance is  $|V| + d$ , since  $d$  of the  $|E|$  applications are to be tested twice whereas all other applications are to be tested only once. Since  $|V|$  is fixed, the optimal solution to the CCDM instance is the one that minimizes  $d$ .

Given a solution to constructed CCDM instance, consider the following solution to the given MINIMUM-BISECTION instance:

- If database  $B_i$  of the constructed CCDM instance is assigned to the first shift, then assign vertex  $i$  of the MINIMUM-BISECTION instance to  $V_1$ ,
- If database  $B_i$  of the constructed CCDM instance is assigned to the second shift, then assign vertex  $i$  of the MINIMUM-BISECTION instance to  $V_2$ .

Notice that an edge  $e = (i, j)$  of the given MINIMUM-BISECTION instance has one endpoint in  $V_1$  and one endpoint in  $V_2$  if and only if the databases  $B_i$  and  $B_j$  of the CCDM instance are assigned to different shifts and thus the application  $A_e$  is to be tested twice. Therefore, the number of edges with one endpoint in  $V_1$  and one endpoint in  $V_2$  of the given MINIMUM-BISECTION instance is equal to the number of applications that needs to be tested twice in the constructed CCDM instance, which is denoted by  $d$ .  $\square$

## 4 Fixed Parameter Intractability of the CCDM Problem

In this section, we study the fixed parameter tractability of the CCDM problem for various parameters of the problem such as the number of applications, the number of shifts, the maximum number of databases that is called by an application, and the maximum number of applications that calls a database.

Notice that Theorem 1 establishes fixed parameter intractability for all 12 models of the CCDM problem, where the parameter is the number of shifts, or the maximum number of databases that is called by an application. In the rest of the section, we prove intractability results for the remaining parameters.

Theorem 2 establishes fixed parameter intractability for all 6 models of the CCDM problem captured by the notation  $\langle * | arb | * \rangle$ , when the parameter is the number of applications. Theorem 3 establishes fixed parameter intractability for all 6 models of the CCDM problem captured by the notation  $\langle * | * | non - unif \rangle$ , when the parameter is the maximum number of applications that calls a database.

**Theorem 2.** *The CCDM problem is **NP-hard** under the 6 models captured by the notation  $\langle * | arb | * \rangle$ , even when the number of applications  $|\mathcal{A}|$  is 2.*

*Proof.* Since the set of instances of the database migration problem under the model  $\langle const | arb | unif \rangle$  is a subset of the instances of any model captured by the notation  $\langle * | arb | * \rangle$ , all we need to do is to prove that the CCDM problem is **NP-hard** under the model  $\langle const | arb | unif \rangle$ , when  $|\mathcal{A}|$  is 2. We will do that via a polynomial reduction from the classical PARTITION problem [12], whose definition is given below.

**Definition 2 (PARTITION).** *Given a multiset  $S$  of positive integers, can  $S$  be partitioned into two subsets  $S_1$  and  $S_2$  such that the sum of the numbers in  $S_1$  equals the sum of the numbers in  $S_2$ ?*

Given a PARTITION instance  $S = \{s_1, s_2, \dots, s_n\}$ , we construct a corresponding CCDM instance under the model  $\langle const | arb | unif \rangle$  with 2 applications as follows:

- For every integer  $s_i$  in the multiset  $S$  of the PARTITION instance, the CCDM instance has a corresponding database  $B_i$  with size  $s_i$ , i.e.,  $w_i = s_i$ ,
- The CCDM instance has two applications  $A_1$  and  $A_2$  with unit testing costs, each of which calls all of the  $n$  databases,
- The CCDM instance has sufficiently many shifts and the size of each shift is  $\frac{\sum_{i=1}^n s_i}{2}$ .

Since both of the applications of the CCDM instance calls all the databases, the total application testing cost is two times the number of shifts with at least one database assigned. Thus, the CCDM instance has a solution with total application testing cost of 4 if the databases can be clustered into two shifts. The theorem holds since the databases can be clustered into two shifts if and only if the answer to the PARTITION instance is yes.  $\square$

**Theorem 3.** *CCDM is strongly **NP-hard** for the 6 models captured by the notation  $\langle * | * | non - unif \rangle$ , even if each database is called by two applications.*

*Proof.* Since the set of instances of the capacity-constrained database migration problem under the model  $\langle const | const | non - unif \rangle$  is a subset of the instances of any model captured by the notation  $\langle * | * | non - unif \rangle$ , all we need to do is to prove that the CCDM problem is **NP-hard** under the model  $\langle const | const | non - unif \rangle$ , when each database is called by at most two applications. We will do that via a polynomial reduction from the classical CLIQUE problem, whose definition is given below.

**Definition 3 (CLIQUE).** *Given a graph  $G = (V, E)$  and an integer  $k$ , is there a fully connected subgraph  $G' \subseteq G$  consisting of  $k$  vertices?*

Given a CLIQUE instance  $\langle G = (V, E), k \rangle$ , we construct a corresponding CCDM instance under the model  $\langle const | const | non - unif \rangle$  as follows:



- For every vertex  $v_i$  of the CLIQUE instance, the CCDM instance has a corresponding application  $A_i$  with unit application testing cost, i.e.,  $c_i = 1$ .
- For every edge  $e$  of the CLIQUE instance, the CCDM instance has a corresponding database  $B_e$  with unit size, i.e.,  $w_i = 1$ ,
- For every edge  $e = (v_i, v_j)$  of the CLIQUE instance, the applications  $A_i$  and  $A_j$  of the CCDM instance calls database  $B_e$ . Notice that each database is called by exactly 2 applications.
- The CCDM instance has  $|E| - k(k-1)/2 + 1$  shifts. The size of the first shift is  $k(k-1)/2$ , and the size of each of the remaining  $|E| - k(k-1)/2$  shifts is 1.

We next show that there is a  $k$ -clique in  $G$  if and only if our CCDM instance yields a solution with a total application testing cost of  $2|E| - k^2 + 2k$ .

Let  $c^*$  denote the cost of the minimum-cost solution for our CCDM instance. Since each database is connected to 2 applications, we have  $c^* > 2(|E| - k(k-1)/2) = 2|E| - k^2 + k$ . This follows from the fact that each database placed in one of the  $|E| - k(k-1)/2$  shifts with capacity of 1 must have both of its connected applications tested, leading to a test cost of  $2(|E| - k(k-1)/2) = 2|E| - k^2 + k$ . For the shift with capacity  $k(k-1)/2$ , we have a total application test cost of  $k$  if and only if the vertices in  $G$  corresponding to the databases in this shift induce a clique of size  $k$ . This follows trivially from the fact that the number of edges in a  $k$ -clique is  $k(k-1)/2$ . Thus,  $c^* \geq 2|E| - k^2 + 2k$  and  $c^* = 2|E| - k^2 + 2k$  if and only if  $G$  has a clique of size  $k$ . Since all values in the reduction are polynomially bounded, it follows that this problem is **NP-hard** in the strong sense.  $\square$

## 5 Approximation Algorithm for a Special Case of the CCDM Problem

In this section, we present Algorithm 5.1 for the CCDM problem under the model  $\langle \text{const} \mid \text{const} \mid \text{unif} \rangle$ , when there are only two shifts and each application calls at most two databases. Algorithm 5.1 is a randomized  $(\frac{3}{2} + \epsilon)$ -approximation algorithm for any given  $\epsilon > 0$  by Theorem 4.

**Function** MIN-TEST-COST( $\langle \mathbf{c}, \mathbf{w}, \mathbf{D}, \mathbf{l} \rangle, \epsilon$ )

- 1: **if**  $n < 1 + \frac{1}{2\epsilon}$  **then**
- 2:     Find optimal solution by brute force
- 3: **else**
- 4:     Select half of the databases by simple random sampling without replacement
- 5:     Assign the selected databases to the first shift
- 6:     Assign the remaining databases to the second shift
- 7: **end if**

**Algorithm 5.1.** Randomized  $(\frac{3}{2} + \epsilon)$ -approximation algorithm for CCDM problem under the model  $\langle \text{const} \mid \text{const} \mid \text{unif} \rangle$ , when there are only two shifts and each application calls at most two databases.

**Theorem 4.** *For any given  $\epsilon > 0$ , Algorithm 5.1 returns a solution whose total application testing cost is at most  $(\frac{3}{2} + \epsilon)$  times that of the optimum, for the CCDM problem under the model  $(\text{const} \mid \text{const} \mid \text{unif})$ , when there are only two shifts and each application calls at most two databases.*

*Proof.* Since Algorithm 5.1 finds the optimal solution in polynomial time by brute force if  $n < 1 + \frac{1}{2\epsilon}$ , in the rest of the proof, we will assume the contrary, i.e.,  $\epsilon \geq \frac{1}{2n-2}$ .

Since there are only two shifts, each application  $A_i$  is to be tested only once or twice. If both of the databases that are called by  $A_i$  are assigned to the same shift or if it calls only one database it will be tested once, otherwise it will be tested twice. Let  $C$  denote the application testing cost of any application. Then the total application testing cost is  $C$  times the total number of application tests. Note that  $m \cdot C$  is a lower bound for the cost of the optimal solution to this CCDM instance, since each application is to be tested at least once. (Recall that  $m$  is the number of applications and  $n$  is the number of databases.)

Let  $X_i$  be a random variable denoting the number of times application  $A_i$  is to be tested with respect to the migration schedule generated by Algorithm 5.1. The total cost of the migration schedule generated by Algorithm 5.1 is then  $C \cdot \sum_{i=1}^m X_i$ . To complete the proof, all that we need to do is to show that  $\mathbb{E}(\sum_{i=1}^m X_i) \leq (\frac{3}{2} + \epsilon) \cdot m$ . Since  $\mathbb{E}(\sum_{i=1}^m X_i) = \sum_{i=1}^m \mathbb{E}(X_i)$  due to linearity of expectations, it suffices to show that  $\mathbb{E}(X_i) \leq (\frac{3}{2} + \epsilon)$  for any  $i$ . If  $A_i$  calls only one database then this inequality is trivially satisfied since  $\mathbb{E}(X_i) = 1$ . So, we focus on applications that calls exactly two databases.

Let  $A_i$  be an application that calls databases  $B_j$  and  $B_k$ . Let  $E_j$  and  $E_k$  denote the events that databases  $B_j$  and  $B_k$  are assigned to the first shift respectively. Accordingly,  $\overline{E}_j$  and  $\overline{E}_k$  are the events that the databases  $B_i$  and  $B_j$  are assigned to shift 2.

It follows that

$$\begin{aligned}
\mathbb{E}(X_i) &= 1 \cdot \Pr((E_j \cap E_k) \cup (\overline{E}_j \cap \overline{E}_k)) + 2 \cdot \Pr((E_j \cap \overline{E}_k) \cup (\overline{E}_j \cap E_k)) \\
&= \Pr(E_j \cap E_k) + \Pr(\overline{E}_j \cap \overline{E}_k) + 2 \cdot \Pr(E_j \cap \overline{E}_k) + 2 \cdot \Pr(\overline{E}_j \cap E_k) \\
&= \Pr(E_j) \Pr(E_k | E_j) + \Pr(\overline{E}_j) \Pr(\overline{E}_k | \overline{E}_j) \\
&= 2 (\Pr(E_j) \Pr(\overline{E}_k | E_j) + \Pr(\overline{E}_j) \Pr(E_k | \overline{E}_j)) \\
&= \frac{1}{2} \cdot \frac{\frac{n}{2} - 1}{n - 1} + \frac{1}{2} \cdot \frac{\frac{n}{2} - 1}{n - 1} + 2 \left( \frac{1}{2} \cdot \frac{\frac{n}{2}}{n - 1} + \frac{1}{2} \cdot \frac{\frac{n}{2}}{n - 1} \right) \\
&= \frac{3}{2} + \frac{1}{2n - 2} \\
&\leq \frac{3}{2} + \epsilon, \text{ as desired.}
\end{aligned}$$

## 6 Conclusion and Future Research Directions

This paper presented a general framework that is suitable for modelling the database migration requirements of a variety of enterprises. We showed that the CCDM problem is **NP-hard** for all the models considered, even under the very restricted scenario, where there are only 2 shifts and each application calls at most 2 databases. We also studied the parameterized complexity of the CCDM problem for four relevant parameters and presented fixed parameter intractability results for all of them. Finally, we presented a  $(\frac{3}{2} + \epsilon)$ -approximation algorithm for an interesting but a quite restricted special case of the CCDM problem. Every model of the CCDM problem is an interesting combinatorial optimization problem by itself, and it would be interesting to know for which models of the CCDM problem there are low factor approximation algorithms, and for which models there are not. From our perspective, the following avenues of research are interesting:

1. Derandomizing the randomized approximation algorithm.
2. Designing approximation algorithms and/or obtaining inapproximability results for all the models of the CCDM problem.

## References

1. Ravikumar, Y.V., Krishnakumar, K.M., Basha, N.: Oracle database migration. Oracle Database Upgrade and Migration Methods, pp. 213–277. Apress, Berkeley (2017). [https://doi.org/10.1007/978-1-4842-2328-4\\_5](https://doi.org/10.1007/978-1-4842-2328-4_5)
2. Harrold, M.J., et al.: Regression test selection for java software. In: ACM SIGPLAN Notices, vol. 36, pp. 312–326. ACM (2001)
3. Vergilio, S.R., Maldonado, J.C., Jino, M., Soares, I.W.: Constraint based structural testing criteria. *J. Syst. Softw.* **79**(6), 756–771 (2006)
4. Eric Wong, W., Horgan, J.R., Mathur, A.P., Pasquini, A.: Test set size minimization and fault detection effectiveness: a case study in a space application. *J. Syst. Softw.* **48**(2), 79–89 (1999)
5. Elmore, A.J., Das, S., Agrawal, D., El Abbadi, A.: Zephyr: live migration in shared nothing databases for elastic cloud platforms. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 301–312. ACM (2011)
6. Lohr, Steve: The age of big data. *New York Times* **11**, 2012 (2012)
7. Bahrami, M., Singhal, M.: The role of cloud computing architecture in big data. In: Pedrycz, W., Chen, S.-M. (eds.) *Information Granularity, Big Data, and Computational Intelligence*. SBD, vol. 8, pp. 275–295. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-08254-7\\_13](https://doi.org/10.1007/978-3-319-08254-7_13)
8. Nascimento, D.C., Pires, C.E., Mestre, D.: Data quality monitoring of cloud databases based on data quality SLAs. In: Trovati, M., Hill, R., Anjum, A., Zhu, S.Y., Liu, L. (eds.) *Big-Data Analytics and Cloud Computing*, pp. 3–20. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-25313-8\\_1](https://doi.org/10.1007/978-3-319-25313-8_1)
9. Ping, L., Zhang, L., Liu, X., Yao, J., Zhu, Z.: Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks. *IEEE Network* **29**(5), 36–42 (2015)

10. Xiaonian, W., Deng, M., Zhang, R., Zeng, B., Zhou, S.: A task scheduling algorithm based on qos-driven in cloud computing. *Procedia Comput. Sci.* **17**, 1162–1169 (2013)
11. Patil, S., et al.: Minimizing testing overheads in database migration lifecycle. In: *COMAD*, p. 191 (2010)
12. Papadimitriou, C.H.: *Computational Complexity*. Addison-Wesley, New York (1994)