Matching-Based Assignement Strategies for Improving Data Locality of Map Tasks in MapReduce

Olivier Beaumont, Thomas Lambert, Loris Marchal, Bastien Thomas

Matching-Based Assignment Strategies for Improving Data Locality of Map Tasks in MapReduce

Olivier Beaumont, Thomas Lambert, Loris Marchal, Bastien Thomas

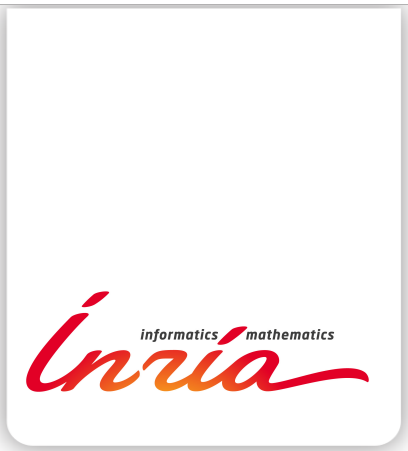
RESEARCH REPORT

N° 8968

February 2017

Project-Teams ROMA and RealOpt

ISSN 0249-6399 ISRN INRIA/RR--8968--FR+ENG



Matching-Based Assignment Strategies for Improving Data Locality of Map Tasks in MapReduce

Olivier Beaumont*, Thomas Lambert*, Loris Marchal†, Bastien Thomas‡

Project-Teams ROMA and RealOpt

Research Report n° 8968 — February 2017 — 52 pages

Abstract: MapReduce is a well-know framework for distributing data-processing computations onto parallel clusters. In MapReduce, a large computation is broken into small tasks that run in parallel on multiple machines, and scales easily to very large clusters of inexpensive commodity computers. Before the Map phase, the original dataset is split into data chunks that are replicated (a constant number of times, usually 3) and distributed randomly onto computing nodes. During the Map phase, local tasks (*i.e.* tasks whose data chunks are stored locally) are assigned in priority when processors request tasks. In this paper, we provide the first complete theoretical analysis of data locality in the Map phase of MapReduce, and more generally, for bag-of-tasks applications that behave like MapReduce. We prove that if tasks are homogeneous (in terms of processing time), as soon as the replication factor is larger than 2, FindAssignment, a matching based algorithm, achieves a quasi-perfect makespan (*i.e.* optimal up to an additive constant of one step) using a sophisticated matching algorithm. Above result is proved with high probability when the number of tasks becomes arbitrarily large, and we therefore complement theoretical results with simulations that corroborate them even for small number of tasks. We also show that the matching-based approach leads to an improvement of data locality during the Map phase and therefore decreases the amount of communications needed to achieve perfect makespan, compared to the classical MapReduce Greedy approach.

Key-words: MapReduce, Analysis of Randomized Algorithms, Matchings, Resource Allocation and Scheduling, Balls-into-bins.

* Inria & University of Bordeaux, France
† CNRS, LIP, École Normale Supérieure de Lyon, INRIA, France
‡ ENS of Rennes, France

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Stratégies d'allocations à base de couplages pour améliorer la localité des tâches Map dans MapReduce

Résumé : MapReduce est un modèle de programmation très connu pour les applications distribuées de traitement de données sur grappes de calcul. Dans ce modèle, le calcul est découpé en petites tâches qui sont lancées en parallèles sur de nombreux processeurs. Il permet d'utiliser facilement de très grandes grappes de calcul faites de processeurs standards. Avant la première phase de Map, les données sont d'abord découpées en gros fragments, qui sont répliqués et distribués sur la plate-forme. Pendant la phase de Map, les processeurs demandent du travail et se voient alloués en priorité des tâches associées aux fragments locaux (s'il y en a). Lorsque ce n'est pas (ou plus) le cas, des communications ont lieu pour transmettre des fragments de données. Dans ce rapport, nous proposont une étude théorique de la localité des données dans la phase de Map d'une application MapReduce, et plus généralement pour toutes applications de type "sac de tâches" qui se comporte de façon similaire. Nous montrons que si les tâches ont des temps de traitement homogènes, une fois que les fragments ont été répliqués et distribués aléatoirement aux processeurs avec un facteur de réplication plus grand que 2, il est possible de trouver un méchanisme de priorité pour les tâches qui obtient un temps de complétion quasi-parfait en utilisant un algorithme de couplage sophistiqué.

Mots-clés : MapReduce, Analyse d'algorithmes randomisés, couplages, allocation de ressource et ordonnancement

1 Introduction

MapReduce is a well-known framework that has been introduced by Google and has been popularized by implementations like Hadoop [29] for distributing data-processing computations onto parallel clusters. In MapReduce, a large computation is broken into small tasks that run in parallel on multiple machines. MapReduce is a very successful example of a dynamic scheduler, as one of its crucial feature is its inherent capability of handling hardware failures and processing capability heterogeneity, thus hiding this complexity to the programmer, by relying on on-demand assignments and the online detection of nodes that perform poorly.

In a classical MapReduce application, the original dataset is first split into data chunks and distributed onto the computing nodes. Then, computation is decomposed into two phases: a Map phase followed by a Reduce phase, each of them being composed of several tasks. However, MapReduce is also widely used to distribute bag-of-tasks applications, which are composed of Map tasks only. Such applications represent 77% of the MapReduce jobs studied in [18]. For these applications, data locality is the main source of communications. There have been relatively few theoretical studies of data locality in MapReduce and its impact on communications [13].

In MapReduce, minimizing the amount of communications performed at runtime is a crucial issue. The initial distribution of the chunks onto the platform is performed by a distributed filesystem such as HDFS [5]. By default, HDFS replicates randomly data chunks several (usually 3) times onto the nodes. This replication has two main advantages. First, it improves the reliability of the process, limiting the risk of losing input data. Second, replication tends to minimize the number of communications at runtime. Indeed, by default, each node is associated to a given number of Map and Reduce slots (usually two of each kind). Whenever a Map slot becomes available, the default scheduler first determines which job should be scheduled, given the job priority and history. Then, it checks whether the job has a local unprocessed data chunk on the processor. If yes, such a local task is assigned, and otherwise, a non-local task is assigned and the associated data chunk is sent from a distant node. Therefore, intuitively, having more replicas provides more opportunities for a given chunk of being processed locally.

It has been shown that depending on the size of the job [17], the fraction of non-local tasks can be around 12-17%, and their processing takes between 1.2 to 2 times longer due to communications of remote data chunks. The quality of the data locality of a scheduling policy, given a replication mechanism, is therefore a crucial issue. This paper is devoted to a precise analysis of the influence of the replication on data locality for Map tasks on the classical MapReduce assignment algorithm, and to the design of a novel assignment algorithm, named FindAssignment, that makes better usage of the initial data replication.

The problem we address is by nature bi-criteria, the first one being the makespan (total completion time) of the Map phase and the second one being the number of non-local tasks (*i.e.* tasks that are not processed on one of the

processors that holds the corresponding data chunk). In the case of homogeneous tasks, it is easy to achieve optimal makespan (never leave a processor idle by assigning non local tasks, as done in MapReduce) and conversely it is easy to perform only local tasks (keep the processor idle if it does not own unprocessed chunks anymore). The main contribution of this paper is to prove (Theorem 5.11) that a sophisticated matching algorithm FindAssignment (introduced in Section 4.2) performs local tasks only (*i.e.* to avoid communications) with quasi optimal makespan, that is, $C_{\text{opt}} + 1$ where C_{opt} is the optimal makespan (with communications). This results holds true with high probability when the number of Map tasks is large enough. We nevertheless show that it also holds for smaller task numbers through simulations.

The rest of the paper is organized as follows. We discuss related work on matching algorithms, randomized balls-into-bins algorithms and MapReduce scheduling algorithms in Section 2 and the modeling of MapReduce in Section 3. In Section 4, we first analyze the impact of the replication on the MapReduce assignment and show that it is closely related to the "power of 2 choices" in balls-into-bin games. Then, we introduce a new scheduling algorithm based on matchings. Its quasi-optimal performance is established in Section 5. Finally, simulation results assessing the actual efficiency of our algorithm are presented in Section 6 and concluding remarks in Section 7. Due to lack of space, some proofs of the paper may be found in Appendix, or more directly in the companion research report [1].

2 Related Work

In this section, we review the literature in three research areas that are close to this study

2.1 Locality in MapReduce

A number of paper have studied the data locality in MapReduce. Note that most studies that aim at minimizing the communications focus on the *Shuffle* phase of a MapReduce application: in this phase, the scheduler transfers the output data of the Map tasks to create the input data of the Reduce tasks. Minimizing the communication of this data-intensive phase has been the target of many studies, such as with coflow scheduling [7, 8, 22]. Some papers have also proposed to place Reduce tasks close to their input to reduce the communications of the shuffle phase [15, 26].

However, as outlined in the previous section, we concentrate in this paper on the Map phase. Several studies have already tried to minimize the data exchange in this phase.

Zaharia et al. [31] first proposed the *Delay* scheduler to improve data locality for several job competing on a cluster. In their strategy, if a given job has a free slot for a new task on a processor buts owns no local chunk, instead of running a non-local task, leading to data movement (as in the classical scheduler), this job

waits for a small amount of time, allowing other jobs to run local tasks instead (if they have some). The authors show that this improves data locality while preserving fairness among jobs.

Ibrahim et al. [17] also outline that, apart from the shuffling phase, another source of excessive network traffic is the high number of non-local map tasks. They propose *Maestro*, an assignment scheme that intends to maximize the number of local tasks. To this end, *Maestro* estimates which processors are expected to be assigned the smallest number of local tasks, given the distribution of the replicas. These nodes are then selected first to assign local tasks. They experimentally show that this reduces the number of non-local map tasks.

Xie et al. [30] propose a simple assignment strategy based on the degree of each processor to overcome the problem of non-local map tasks. The idea is to give priority to processors with the smallest non-zero number of unprocessed replicas, so that they could be assigned a local task. They propose a "peeling" algorithm which first serves the processor with a single unprocessed replica (and thus assigns to them their unique local task), and then switches to a classical random assignment for other processors. Using queuing theory and assuming that processing times are given by geometric distribution, they prove that their strategy offers close to optimal assignment for small to medium load. In a similar context, Wang et al. [28] propose a new queuing algorithm to simultaneously maximize throughput and minimize delay in heavy loaded conditions.

Guo et al. [13] consider the locality of map tasks. They propose an algorithm based on the Linear Sum Assignment Problem to compute an assignment with minimal communications. Unfortunately, in the case where there are more tasks than processors, there formulation is obviously wrong: they add fictitious processors to get back to the case with equal number of tasks and processors, solve the problem, and then remove the fictitious processors without taking care of the task reassignment.

2.2 Matchings in Bipartite Graphs

Many studies and algorithms have been proposed for matching problems, in particular for bipartite graph, a case with many practical applications, for example in assignment problems [6].

A first research direction deals with the existence of matchings, in particular perfect matchings, *i.e.* matchings whose size is the number of vertices. For instance, the work of Erdös and Rényi on random bipartite graph [10] proves that there exists a perfect matching of a bipartite graph of $2n$ vertices with asymptotic probability $e^{-2e^{-c}}$ when the number of edges is $n \ln n + cn + o(n)$. Walkup [27], instead of an assumption on the number of edges, uses a condition on the minimum degree of the vertices. In this model, asymptotically, a regular bipartite graph (*i.e.* whose both sets of vertices are of equal size) has a perfect matching if its minimum degree is at least 2.

A second research direction deals with the efficient computation of matchings. Many algorithms rely on augmenting paths introduced in [11]. An augmenting path is a path that induces an improvement of the current existing

flow (or matching) by permuting some edges of the path with some of the actual solution. For bipartite graphs, very efficient algorithms exist to find an optimal matching, such as the Hopcroft-Karp Algorithm [16] with a complexity of $O(\sqrt{n}m)$, where n denotes the number of vertices and m the number of edges, or the one proposed by Goel et al. [12] for regular bipartite graph, whose expected complexity is $O(n \log n)$. There also exist approximation algorithms with better computations time that no longer guarantee the computation of a perfect matching [19, 9].

2.3 Variants of Balls-into-Bins

An extreme solution for improving data locality is to forbid all communications, and only allow processors to compute the chunks they own locally. This might generate some load imbalance, since some of the processors may stop their computations early.

Such a process is closely related to balls-into-bins problems [23]. More specifically, we prove in Section 4.1 that it is possible to simulate the greedy algorithm for assigning Map tasks with a variant of a balls-into-bins game. In these randomized processes, n balls are placed randomly into p bins and the expected load of the most loaded bin is considered. In a process where data chunks are not replicated, chunks correspond to balls, processing resources correspond to bins, and if tasks have to be processed locally, then the maximum load of a bin is equal to the makespan achieved by greedy assignment. The case of weighted balls, that corresponds to tasks whose lengths are not of unitary length, has been considered in [3]. It is shown in [3] that when assigning a large number of small balls with total weight W, one ends up with a smaller expected maximum load than the assignment of a smaller number of uniform balls with the same total weight. In the case of identical tasks [23], when $n/\text{polylog}(n) \leq p \leq n \log n$, *i.e.* typically when the average number of tasks assigned to each resource is a few units, then the expected maximum load is of order $\frac{\log n}{\log\left(\frac{n \log n}{p}\right)}$, it is therefore arbitrarily large and grows as $\frac{\log n}{\log \log n}$ when the number of tasks increases.

Balls-into-bins techniques have been extended to multiple choice algorithms, where r random candidate bins are pre-selected for each ball, and then the ball is assigned to the candidate bin whose load is minimum. It is well known that having more than one choice strongly improves load balancing. We refer the interested reader to [20, 25] for surveys that illustrate the power of two choices. Typically, combining previous results with those of [2], it can be proved that whatever the expected number of balls per bin, the expected maximum load is of order $n/p + O(\log \log n)$ even in the case $r = 2$, what represents a very strong improvement over the single choice case. The combination of multiple choice games with weighted balls has been considered in [21]. In this case, each ball comes with its weight w_i and is assigned, in the r-choices case, to the bin of minimal weight, where the weight of a bin is the sum of the weights of the balls assigned to it.

3 Modeling of MapReduce

We model the assignment mechanism of MapReduce as follows. We assume that there are p *identical processors*, *i.e.* processors with the same processing capabilities and equipped with their own data storage. There are n identical *tasks* to process, and each task corresponds to the processing of a *chunk*. The set of processors is denoted by $P = \{p_1, \dots p_p\}$ while the set of tasks is denoted by $T = \{t_1, \dots t_n\}$. In the following theoretical analysis in Sections 4 and 5, we assume that all tasks have the same processing needs, *i.e.* that all tasks have unitary length. We assume that chunks are initially replicated on the processors: r replicas of each chunk are distributed uniformly at random onto the p processors. The communication cost needed for replicating the chunks is not taken into account as it is generally done much before the computation phase by the underlying file system and since the same file is in general used for more than one MapReduce job.

As presented in the introduction, the problem of scheduling tasks with replicated input chunks is bi-criteria, as the objective is both to minimize the total completion time and to reduce the number of communications. In this paper, we choose to concentrate on local schedules only: we forbid any data communications among processors once the chunks have been replicated in the initial phase (we will nevertheless consider other possible metrics in Section 6). Our objective is then to find a schedule that minimizes the expected makespan. We will show that even with the constraint of performing local tasks only, it is nevertheless possible to build a schedule with extremely good makespan.

To compare our solution to a reasonable bound, we use the concept of *perfect assignment*. A *perfect assignment* is an assignment of tasks to processors that minimizes the time needed to process all tasks, *i.e.* where each processors receives at most $\lceil n/p \rceil$ chunks. As tasks are homogeneous, this is clearly a lower bound on the makespan. Note that this also corresponds to the makespan of the classical MapReduce scheduler which allows communications in order to perform load-balancing. However, depepending on the initial data chunk distribution, such a perfect assignment may not necessarily exist in our context, as we forbid communications, especially for a small replication parameter r.

4 Finding a Better Assignment

In this section, we analyze the expected makespan of a simple dynamic strategy inspired by the MapReduce scheduler. We prove that its performance is closely related to the one of balls-into-bins algorithms with multiple choice. Then, we move to the design of a makespan-optimal scheduling algorithm based on bipartite graph matchings.

4.1 Analysis of a Simple Dynamic Scheduler

We consider here a simple dynamic scheduler, called GREEDY and inspired by the MapReduce scheduler and detailed in Algorithm 1: if there exists local not yet processed tasks, one such local task is chosen at random, performed locally and marked as processed on all processors. If no such unprocessed local task exists, then processor stops its execution. Note that it is the straightforward version of the MapReduce scheduler when communications are forbidden.

Let us consider a more general context for the analysis of this algorithm, where tasks have heterogeneous processing times (w_i is the duration of task t_i) and processors have slightly different initial availability times (ϵ_j is the availability time of processor j). However, the GREEDY scheduler has no knowledge of the task durations before their execution. We assume that this heterogeneity in task durations and processor availability times allows us to consider that no ties have to be broken, as in [3, 21]. Note that in GREEDY, the initial chunk distribution is given by n sets of r random choices: $\mathcal{C}_1, \ldots, \mathcal{C}_n$. Together with the initial processor availability times and the task durations, this entirely defines the scheduler behavior.

Algorithm 1: GREEDY$(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_p)$

Input: $\mathcal{C}_1, \ldots, \mathcal{C}_n$: sets of r random choices
Input: $\epsilon_1, \ldots, \epsilon_p$: initial processor availability times
Initial chunk distribution:
for $i = 0 \ldots n$ **do**
  Place a copy of chunk i onto processors $\{p_{i^{(1)}}, \ldots, p_{i^{(r)}}\}$ where $\mathcal{C}_i = \{i^{(1)}, \ldots, i^{(r)}\}$
Task assignment:
while *there exists an unprocessed task* **do**
  Whenever a processor p_k completes a task, assign to this processor the local task t_i with smallest index, if any

We now prove that in presence of replication, the expected makespan of the GREEDY scheduler is closely related to the balls-into-bins problem with r multiple choices. Algorithm 2 shows the process of distributing n balls of sizes $w_1, \ldots, w_n$ into p bins whose initial loads are given by $\epsilon_1, \ldots, \epsilon_p$. This distribution is done as follows: for each ball, r bins are selected at random (using the random choices $\mathcal{C}_i$) and the ball is placed in the least loaded of these r bins. The following theorem shows the relation between the simple dynamic scheduler and the balls-in-bins process.

Theorem 4.1. *Let us denote by* MAXLOAD *the maximal load of a bin using* BALLS-IN-BINS *and by* $C_{\max}$ *the makespan achieved using* GREEDY. *Then,*

$$\text{MAXLOAD}(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_p) = C_{\max}(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_p).$$

Proof. In order to prove above result, let us prove by induction on i the following Lemma.

Algorithm 2: BALLS-IN-BINS$(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_p)$

for $i = 0 \ldots n$ **do**
  Place ball b_i of weight w_i into the least loaded bin among bins $\{B_{i^{(1)}}, \ldots, B_{i^{(r)}}\}$ where $\mathcal{C}_i = \{i^{(1)}, \ldots, i^{(r)}\}$;

Lemma 4.2. *Let $j_b(i)$ denote the index of the bin where ball b_i is placed and let $j_p(i)$ denote the index of the processor where task t_i is processed, then $j_b(i) = j_p(i)$.*

Proof. Let us consider ball b_1 and task t_1. b_1 is placed in the bin such that ϵ_k is minimal, where $k \in C_1$. Conversely, t_1 is replicated onto all the processors p_k, where $k \in C_1$. Since each processor executes its tasks following their index, t_1 is processed on the first processor owning t_1 that looks for a task, *i.e.* the processor such that ϵ_k is minimal. This achieves the proof in the case $n = 1$.

Let us assume that the lemma holds true for all indexes $1, \ldots, i-1$, and let us consider the set of bins B_k and the set of processors p_k such that $k \in C_i$. By construction, at this instant, processors $p_k,\ k \in C_i$ have only processed tasks whose index is smaller than i. Let us denote by $S_i = \{t_{i_1}, \ldots, t_{i_{n_i}}\}$ this set of tasks, whose indexes i_k's are smaller than i. These tasks have been processed on the processors whose indexes are the same as those of the bins on which balls $\{b_{i_1}, \ldots, b_{i_{n_i}}\}$ have been placed, by induction hypothesis. Therefore, for each $p_k,\ k \in C_i$, the time at which p_k ends processing the tasks assigned to it and whose index is smaller than i is exactly the weight of the balls with index smaller than i placed in B_k. Therefore, the processor p_k that first tries to compute t_i is the one such that ϵ_k plus the weight of the balls with index smaller than i placed in B_k is minimal, so that $j_b(i) = j_p(i)$, what achieves the proof of the lemma. □

Therefore, the makespan achieved by GREEDY on the inputs $(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_p)$ is equal to the load of most loaded bin in BALLS-IN-BINS on the same input, which achieves the proof of the Theorem. □

Thanks to this result, we can apply known bounds on the maximum load for balls-into-bins processes derived in the literature, as related in Section 2.3. In particular, going back to the case of tasks with identical processing times, the expected makespan when $r \geq 2$ is known to be of order $n/p + O(\log \log n)$ (with high probability).

4.2 Matching-Based Assignments

We now focus on the design of a makespan-optimal assignment based on matching techniques in bipartite graphs. The initial placement of data chunks can be modeled by a bipartite graph G with a set T of n tasks nodes and a set P of p processor nodes. An edge between node task node t_i and processor node p_j indicates the presence of the chunk of task t_i on processor p_j. An assignment

of the tasks on the processors can then be described by an *assignment* in the bipartite graph,

Definition 4.3 (Assignment)**.** *Let $G = (P, T, E)$ be a bipartite graph. An* assignment *of G is a subset A of E such that:*

$$\forall t_j \in T,\ \exists \text{ a unique } p_i \in P \text{ such that } (p_i, t_j) \in A.$$

and the following metrics can be used to measure the quality of an assignment,

Definition 4.4 (Degree in an assignment and maximum degree)**.** *Let A be an assignment of $G = (P, T, E)$.*

- *The* degree in *A of a vertex p_i of P, denoted $d_A(p_i)$, is the degree of p_i in $G' = (P, T, A)$, the sub-graph of G induced by A.*
- *the* maximum degree *$d(A)$ of A is defined as $D(A) = \max_{p_i \in P} d_A(p_i)$.*

In the corresponding assignment, the degree of a processor is the number of tasks assigned to it, and thus the maximum degree of an assignment is the makespan of the corresponding assignment, since all tasks have unitary processing times. Then, finding a schedule with minimum makespan thus translates into finding an assignment with minimum maximum degree.

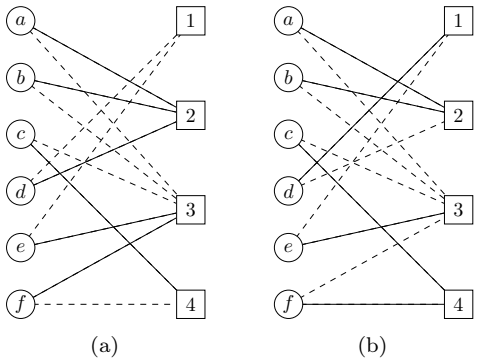


Figure 1: Two examples of assignments for the same problem.

These notions are illustrated on Figure 1, where tasks are on the left part and processors on the right. Solid edges represent an assignment and dashed ones are the initial edges that are not used in the assignment. Each task can be uniquely associated to one solid edge, but processors can be assigned to more than one task. On Figure 1(a), the maximum degree is 3 (reached on 2). On Figure 1(b), the maximum degree is 2 (reached on 2 and 4).

4.3 Reduction to a Maximal Matching

In this section, we establish the relationship between assignments and matchings. We recall that a matching of a bipartite graph $G = (P, T, E)$ is a M subset of E such that each node of $P \cup T$ is incident to at most one edge in M. There are some notable differences between an assignment and a matching. First, in an assignment, two edges may be incident to the same processor node. Second, a task node may not be covered by an edge of a matching, while it is necessary connected to a processor node in an assignment.

For a given integer m, we build an auxiliary bipartite graph G^m from the bipartite graph representation of our problem, by replicating m times the set of processors. We show in the following lemma to the existence of an assignment of maximum degree m in G is directly related to the existence of a matching of size n in G^m, as illustrated in Figure 2.

Definition 4.5 (G^m)**.** *Let $G = (P, T, E)$ be a bipartite graph and m be an integer. We define $G^m = (P^m, T, E^m)$ with:*

- $P^m = \bigcup_{p_i \in P} \{p_i^1, \ldots, p_i^m\}$.
- $(p_i^k, t_j) \in E^m$ *if and only if* $(p_i, t_j) \in E$.

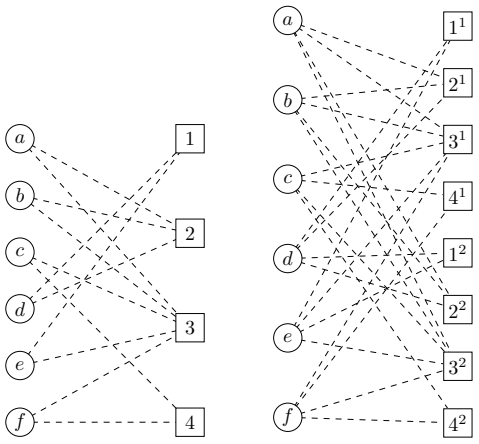


Figure 2: An example of replication of a bipartite graph. On the left G, on the right G^2.

Lemma 4.6. *Let $G = (P, T, E)$ be a bipartite graph with $|T| = n$ and let m be an integer. There exists an assignment of maximum degree m if and only if there exists a matching of size n in G^m.*

Proof. Let us first suppose that there exists an assignment A of maximum degree m of G. For $p_i \in P$ let $v_A(p_i)$ denotes its neighborhood in the sub-graph induced by A, *i.e.* $v_A(p_i) = \{t_j \in T,\ (p_i, t_j) \in A\}$ (and $|v_A(p_i)| = d_A(p_i)$). Let M be a

subset of E^m with $M = \bigcup_{p_i \in P} M_{p_i}$ with

$$M_{p_i} = \left\{(p_i^1, t_1), \ldots, (p_i^{d_A(p_i)}, t_{d_A(p_i)}), \right. \\ \left. \{t_1, \ldots, t_{d_A(p_i)}\} = v_A(p_i)\right\}.$$

Since $\forall p_i \in P$, $d_A(p_i) \leq m$, then $\forall p_i$, $\{p_i^1, \ldots, p_i^{d_A(p_i)}\}$ is a valid subset of P^m. Let (p_i^k, t_j) and $(p_{i'}^{k'}, t_{j'})$ denote any two edges of M.

- If $i \neq i'$, then $p_i^k \neq p_{i'}^{k'}$. In addition, by definition of an assignment, $v_A(p_i) \cap v_A(p_{i'}) = \emptyset$ (otherwise there is a contradiction with $\exists! p_i \in P, (p_i, t_j) \in A$) and then $t_j \neq t_{j'}$.
- If $i = i'$, then $t_j = t_{j'}$ if and only if $k = k'$.

Therefore, M is a valid matching. Moreover, since $v_A(p_i) \cap v_A(p_{i'}) = \emptyset$, then $M_{p_i} \cap M_{p_{i'}} = \emptyset$. Therefore, $|M| = \sum |M_{p_i}| = \sum d_A(p_i) = n$ by definition of an assignment. Thus, there exists a matching of cardinal n in G^m.

Let us now assume that there exists a matching M of G^m with $|M| = n$. Let us build a subset A of E such that

$$(p_i, t_j) \in A \Leftrightarrow \exists k,\ (p_i^k, t_j) \in M.$$

Let (p_i, t_j) and $(p_{i'}, t_{j'})$ be any two edges of A. There exists (k, k') such that $(p_i^k, t_j), (p_{i'}^{k'}, t_{j'}) \in M$. By definition of a matching, $t_j = t_{j'}$ if and only $p_i^k = p_{i'}^{k'}$. Hence $t_j = t_{j'}$ if and only if $p_i = p_{i'}$. Moreover, $|A| = |M| = n$ and therefore, $\forall t_j$, there exists p_i such that $(p_i, t_j) \in A$. Thus A is an assignment of G and $D(A) \leq m$ as $d_A(p_i) \leq m$ for all $p_i \in P$. □

4.4 Algorithms to Find an Optimal Assignment

In order to find an optimal assignment (*i.e.* an assignment with minimum maximum degree), we rely on Lemma 4.6 which states that finding an assignment for a bipartite graph G is equivalent to finding a maximal matching (in G^m). The method, summarized in Algorithm 3, consists in (i) building G^m, (ii) finding a maximal matching in G^m, and (iii) turning it into an assignment for G.

The complexity of NaiveAssignment is nevertheless high in practice. Indeed, if there exists very efficient algorithms to compute maximum matchings on bipartite graph [16], NaiveAssignment requires to build G^m, which is a large expansion of G, before searching for a maximum matching in G. In what follows, we propose, following classical ideas from the literature on the matching and flows problems [11], a more direct algorithm to find an assignment. Let us first adapt the notion of alternating path.

Definition 4.7. *Let $G = (P, T, E)$ be a bipartite graph and let A be a subset of E.*

Algorithm 3: NaiveAssignment(G)

Input: A bipartite graph $G = (P, T, E)$
Output: An assignment A of G of minimum maximal degree
$m = \left\lceil \frac{|T|}{|P|} \right\rceil$;
$A = \emptyset$;
while *A is not an assignment* **do**
  Find a maximal matching M of G^m ;
  if $|M| = |T|$ **then**
    $A \leftarrow$ the conversion of M into an assignment of size m ;
  $m \leftarrow m + 1$;
return A

- *A* path *of G is a sequence of vertices $(x_1, \ldots, x_k)$ such that $\forall i \in [1, k-1]$, $(x_i, x_{i+1}) \in E$. Note that in a path of a bipartite graph the vertices switch between T and P.*
- *An* alternating path *of G according to A is a path $(x_1, \ldots, x_k)$ of G such that:*
  - *If $x_i \in T$, then $(x_i, x_{i+1}) \in A$.*
  - *If $x_i \in P$, then $(x_i, x_{i+1}) \notin A$.*

An example of alternating path is described in Figure 3. On the left hand side, solid edges represent an assignment, and dashed ones the unused edges. On the right hand side, the proposed path (to improve the clarity of the scheme, edges that are not in the path has been removed) is an alternating one according to the previous assignment.

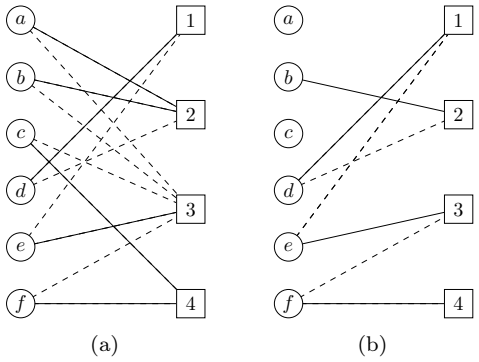


Figure 3: Example of Alternating path.

Alternating paths can be used to improve an existing assignment. Indeed, Lemma 4.8 states that if the starting and the ending vertices of an alternating

path are in P, then it is possible to build an assignment that improves the degree of the last vertex and increases by one the degree of the first one.

Lemma 4.8. *Let $G = (P, T, E)$ be a bipartite graph, let A be an assignment of G and let $x = (p_{i_1}, t_{j_1}, \ldots, p_{i_k})$ be an alternating path of G according to A. Let $\otimes$ be the xor operation ($A \otimes B = (A \cup B) \setminus (A \cap B)$) and let x be assimilated to its edges, then,*

- *$A \otimes x$ is an assignment.*
- $d_{A\otimes x}(p_{i_1}) = d_A(p_{i_1}) + 1$
- $d_{A\otimes x}(p_{i_k}) = d_A(p_{i_k}) - 1$
- $\forall p_i \in P \setminus \{p_{i_1}, p_{i_k}\},\ d_{A\otimes x}(p_i) = d_A(p_i)$.

Proof. Let $G = (P, T, E)$ be a bipartite graph, let A be an assignment of G and let $x = (p_{i_1}, t_{j_1}, \ldots, t_{j_{k-1}}, p_{i_k})$ be an alternating path of G according to A. Let $t_j \in T$:

- If $t_j = t_{j_l} \in \{t_{j_1}, \ldots, t_{j_{k-1}}\}$, then, by definition of an alternating path, $(t_{j_l}, p_{i_{l+1}})$ is in A (and, by definition of an assignment, it is the only edge from t_j in A) and (p_{i_l}, t_{j_l}) is not. Therefore (p_{i_l}, t_{j_l}) is in $A \otimes x$ (and it is the only edge from t_j in $A \otimes x$) and $(t_{j_l}, p_{i_{l+1}})$ is not.
- Otherwise, the unique edge in A from t_j is not in x (and x has no edges from t_j) and therefore is also present in $A \otimes x$.

Therefore, $\forall t_j \in T$, there is an unique edge from t_j in $A \otimes x$ that is thus an assignment.

Furthermore, let p_i be in P, then

- if $p_i = p_{i_1}$, then (p_{i_1}, t_{j_1}) is added to its neighborhood in $A \otimes x$,
- if $p_i = p_{i_k}$, then $(t_{j_{k-1}}, p_{i_k})$ is removed from its neighborhood in $A \otimes x$,
- if $p_i = p_{i_l} \in \{p_{i_2}, \ldots, p_{i_{k-1}}\}$, then (p_{i_l}, t_{j_l}) is added to its neighborhood and the edge $(t_{j_{l-1}}, p_{i_l})$ is removed from its neighborhood in $A \otimes x$,
- else there is no modification of its neighborhood from A to $A \otimes x$.

Therefore,

- $d_{A\otimes x}(p_{i_1}) = d_A(p_{i_1}) + 1$
- $d_{A\otimes x}(p_{i_k}) = d_A(p_{i_k}) - 1$
- $\forall p_i \in P \setminus \{p_{i_1}, p_{i_k}\},\ d_{A\otimes x}(p_i) = d_A(p_i)$.

□

Definition 4.9. *Let $G = (P, T, E)$ be a bipartite graph and A be an assignment of G. An* augmenting path *according to A is an alternating path $x = (p_{i_1}, t_{j_1}, \ldots, p_{i_k})$ such that $d_A(p_{i_1}) + 1 < d_A(p_{i_k})$.*

Lemma 4.8 says that an augmenting path can be used to build an assignment that can even decrease the maximum degree if p_{i_k} is the only vertex such that $d_A(p_{i_k}) = D(A)$. In fact, a stronger property holds true: if there is no augmenting path, then the assignment has a minimum maximum degree. This result is formalized in Theorem 4.10.

Theorem 4.10. *Let $G = (P, T, E)$ be a bipartite graph, let A be an assignment of G. If there is no augmenting path according to A, then A has a minimum maximum degree.*

Proof. We will rely on the following lemmas to prove Theorem 4.10.

Lemma 4.11. *(Berge's Lemma [4]) Let $G = (P, T, E)$ be a bipartite graph and M be a matching of G. M is maximal if and only if there no alternating path $x = (p_{i_1}, t_{j_1}, \ldots, t_{j_{k-1}})$ such that there is no edge from p_{i_1} and from $t_{j_{k-1}}$ in M.*

Lemma 4.12. *Let A and A' be two assignments of a same bipartite graph $G = (P, T, E)$. If there exists p_i such that $d_A(p_i) > d_{A'}(p_i)$, then there exists an alternating path $x = (p_d, \ldots, p_f)$ according to A such that $d_A(p_d) < d_{A'}(p_d)$ and p_f is the vertex verifying $d_A(p_f) > d_{A'}(p_f)$ with the largest degree.*

Proof. To prove above lemma, we need a more general definition of a replicated graph. Instead to replicate each vertex of P m times, we replicate p_i m_{p_i} times (the number or replication can be different from a vertex to another). In this case, Lemma 4.6 can be adapted: let $G = (P, T, E)$ be a bipartite graph with $|T| = n$ and $(m_i)_{1 \leq i \leq p}$ be a set of integers. Then, there exists an assignment A with, for all $p_i \in P$, $d_A(p_i) \leq m_i$ if and only if there exists a matching of size n in the replicated graph of G where each p_i is replicated m_i times.

Let now A and A' be two assignments of a bipartite graph G such that $\exists p_i$ that fulfills $d_A(p_i) > d_{A'}(p_i)$. Let p_f be the vertex satisfying $d_A(p_f) > d_{A'}(p_f)$ with the largest degree. Let us consider the graph G' that is a replicated graph of G where p_f is replicated $d_A(p_f) - 1$ times and where $p_i \in P \setminus \{p_f\}$ is replicated $max(d_A(p_i), d_{A'}(p_i))$. Let $t \in T$ be such that $(p_f, t) \in A$. Thanks to the modified version of Lemma 4.6, we can define a matching M' of G' of size $|T|$ and a matching M of size T associated to the partial assignment $A \setminus (p_f, t)$.

M is not a matching of maximum size since M' is larger. Therefore, according to Lemma 4.11, there exists an alternating path $x = (p_{i_1}, t_{j_1}, \ldots, t_{j_{k-1}})$ such that there is no edge from p_{i_1} and from $t_{j_{k-1}}$ in M. The only possible free vertex from T is t, thus this alternating path must fulfill $x = (p_d^{k_d}, \ldots, t)$. In addition, to have a free replicate in G' according to M, p_d must fulfill $d_A(p_d) < max(d_A(p_d), d_{A'}(p_d))$ and therefore $d_A(p_d) < d_{A'}(p_d)$.

We then easily check that the path $(p_d, \ldots, t, p_f)$ is an alternating path and, as $d_A(p_d) < d_{A'}(p_d)$, thus claimed result. □

Lemma 4.13. *Let $G = (P, T, E)$ be a bipartite graph and let A, A' be two assignments of G. Therefore there exist finite sequences of assignments $(A_k)_{k \leq l}$ and paths $(x_k)_{k \leq l-1}$ (l is the length of the sequence) such that*

- $A_0 = A$
- $\forall p_i \in P,\ d_{A_l}(p_i) = d_{A'}(p_i)$
- $\forall k \leq l$, *x_k is alternating according to* A_k *and* $A_{k+1} = A_k \otimes x_k$
- *If* $x_k = (p_{d_k}, \ldots, p_{f_k})$, *then* $d_{A_k}(p_{d_k}) < d_{A'}(p_{d_k})$ *and* $d_{A_k}(p_{f_k}) > d_{A'}(p_{f_k})$ *and* p_f *is the greatest such* p_i.
- $\forall p_i \in P$, *the sequence* $(|d_{A_k}(p_i) - d_{A'}(p_i)|)_{k \leq l}$ *is decreasing.*

Proof. Let us suppose that these sequences have been built until rank k. If $\forall p_i \in P,\ d_{A_k}(p_i) = d_{A'}(p_i)$, then sequences are built. Else, according to Lemma 4.12, there exists an alternating path $x_k = (p_{d_k}, \ldots, p_{d_f})$ such that $d_{A_k}(p_{d_k}) < d_{A'}(p_{d_k})$ and p_f is the vertex with the larger degree verifying $d_{A_k}(p_{f_k}) > d_{A'}(p_{f_k})$. Let us define $A_{k+1} = A_k \otimes x_k$.

Let us prove that sequences are finite. Let us consider the sequence $(u_k)_{k \in \mathbb{N}}$ defined by $u_k = \sum_{p_i \in P} |d_{A_k}(p_i) - d_{A'}(p_i)|$.

$$
\begin{aligned}
u_{k+1} &= \sum_{p_i \in P} |d_{A_{k+1}}(p_i) - d_{A'}(p_i)| \\
&= \sum_{p_i \in P \setminus \{p_{d_k}, p_{f_k}\}} |d_{A_{k+1}}(p_i) - d_{A'}(p_i)| + |d_{A_{k+1}}(p_{d_k}) - d_{A'}(p_{d_k})| + |d_{A_{k+1}}(p_{f_k}) - d_{A'}(p_{f_k})| \\
&= \sum_{p_i \in P \setminus \{p_{d_k}, p_{f_k}\}} |d_{A_k}(p_i) - d_{A'}(p_i)| + |d_{A_k}(p_{d_k}) + 1 - d_{A'}(p_{d_k})| + |d_{A_k}(p_{f_k}) - 1 - d_{A'}(p_{f_k})| \\
&= \sum_{p_i \in P \setminus \{p_{d_k}, p_{f_k}\}} |d_{A_k}(p_i) - d_{A'}(p_i)| + |d_{A_k}(p_{d_k}) - d_{A'}(p_{d_k})| - 1 + |d_{A_k}(p_{f_k}) - d_{A'}(p_{f_k})| - 1 \\
&= u_k - 2
\end{aligned}
$$

Therefore there exists an index l such that $u_l = 0$ and hence $\forall p_i \in P,\ d_{A_l}(p_i) = d_{A'}(p_i)$ and the sequences are finite. Note that similar calculations prove that $(|d_{A_k}(p_i) - d_{A'}(p_i)|)_{k \leq l}$ is decreasing for all p_i.

□

With Lemma 4.13, we can now prove Theorem 4.10. Let $G = (P, T, E)$ be a bipartite graph and let A be an assignment of G. Let us suppose that A has not a minimum maximum degree. Therefore, there exists A' such that $D(A') < D(A)$.

Thank to Lemma 4.13, we know that there are finite sequences of assignments $(A_k)_{k \leq l}$ and paths $(x_k)_{k < l}$ such that

- $A_0 = A$,
- $A_l = A'$,
- $\forall p_i \in P,\ d_{A_l}(p_i) = d_{A'}(p_i)$,

- $\forall k < m$, x_k is alternating and $A_{k+1} = A_k \otimes x_k$.

In particular $D(A_0) > D(A_l)$. Therefore, there exists k_0 such that $D(A_{k_0}) > D(A_{k_0+1}) = D(A_{k_0} \otimes x_{k_0})$. Let $x_{k_0} = (p_{d_{k_0}}, \ldots, p_{f_{k_0}})$. Since x_{k_0} is alternating

- $d_{A_{k_0} \otimes x_{k_0}}(p_{d_{k_0}}) = d_{A_{k_0}}(p_{d_{k_0}}) + 1$,
- $d_{A_{k_0} \otimes x_{k_0}}(p_{f_{k_0}}) = d_{A_{k_0}}(p_{f_{k_0}}) - 1$,
- $\forall p_i \in P \setminus \{p_{d_{k_0}}, p_{f_{k_0}}\},\ d_{A_{k_0} \otimes x_{k_0}}(p_i) = d_{A_{k_0}}(p_i)$.

Yet, $D(A_{k_0}) > D(A_{k_0} \otimes x_{k_0})$ and, since this is the only one with decreasing degree, $D(A_{k_0}) = d_{A_{k_0}}(p_{f_{k_0}})$. Moreover,

$$\begin{aligned}
&d_{A_{k_0}}(p_{d_{k_0}}) + 1 = d_{A_{k_0} \otimes x_{k_0}}(p_{d_{k_0}}),\\
&d_{A_{k_0}}(p_{d_{k_0}}) + 1 \leq D(A_{k_0} \otimes x_{k_0}),\\
&d_{A_{k_0}}(p_{d_{k_0}}) + 1 < D(A_{k_0}),\\
&d_{A_{k_0}}(p_{d_{k_0}}) + 1 < d_{A_{k_0}}(p_{f_{k_0}}).
\end{aligned}$$

Therefore, x_{k_0} is augmenting, $D(A_{k_0}) = d_{A_{k_0}}(p_{f_{k_0}})$ and $d_{A_{k_0}}(p_{d_{k_0}}) < d_{A'}(p_{d_{k_0}})$.

We have proved if there exists an index k_0 such that $D(A_{k_0}) > D(A')$, there exists an augmenting path according to A_{k_0} with some additional properties. Let us define k_0 as the smallest k such that $D(A_k) > D(A')$ and there exists an augmenting path $x = (p_d, \ldots, p_f)$ according to A_k (and that augmenting path could differ from x_k) such that $d_{A_k}(p_f) = D(A_k)$ and $d_{A_{k_0}}(p_d) < d_{A'}(p_d)$. Let us try to prove that $k_0 = 0$ in order to reach a contradiction.

Let $x = (p_d, \ldots, p_f)$ be such an augmenting path in A_{k_0}, $D(A_{k_0}) > D(A')$.

First, let us note that $D(A_{k_0-1}) \geq D(A_{k_0}) > D(A')$. Indeed, for all $p_i \neq p_{d_{k_0-1}}$, $d_{A_{k_0-1}}(p_i) \geq d_{A_{k_0}}(p_i)$.

Second, let us assume that x and x_{k_0-1} are disjoint. In this case, $d_{A_{k_0-1}}(p_d) = d_{A_{k_0}}(p_d) < d_{A'}(p_d)$ and $d_{A_{k_0-1}}(p_f) = d_{A_{k_0}}(p_f) = D(A_{k_0}) \leq D(A_{k_0-1})$. If $D(A_{k_0}) = D(A_{k_0-1})$. Then, x is a valid augmenting path and $d_{A_{k_0-1}}(p_f) = D(A_{k_0-1})$. Otherwise, $D(A_{k_0-1}) = d_{A_{k_0-1}}(p_{f_{k_0-1}})$ ($D(A_{k_0-1}) > D(A_{k_0}) > D(A')$ and thus, there exists at least one p_i such that $d_{A_{k_0-1}}(p_i) > d_{A'}(p_i)$ and by construction this is also the case for $p_{f_{k_0-1}}$), so that

$$d_{A_{k_0-1}}(p_{d_{k_0-1}}) + 1 = d_{A_{k_0}}(p_{d_{k_0-1}}) \leq D(A_{k_0}) < d_{A_{k_0-1}}(p_{f_{k_0-1}}).$$

Hence, x_{k_0-1} is an augmenting path with $D(A_{k_0-1}) = d_{A_{k_0-1}}(p_{f_{k_0-1}})$ and $d_{A_{k_0-1}}(p_{d_{k_0-1}}) < d_{A'}(p_{d_{k_0-1}})$ by construction. Thus x and x_{k_0-1} are not disjoint since otherwise, we would reach a contradiction with the definition of k_0.

Therefore, we know that x and x_{k_0-1} are not disjoint. Let v_i and v_j be two vertices of G such that

- v_i is the first vertex in x to be in x_{k_0-1},
- v_j is the last vertex in x to be in x_{k_0-1}.

Let us suppose that v_i is before v_j in x_{k_0-1}. Then, $(p_d, \ldots, v_i, \ldots, v_j, \ldots, p_f)$ is a valid alternating path in A_{k_0-1}. Because $(|d_{A_k}(p_i) - d_{A'}(p_i)|)_{k \leq l}$ is decreasing, $d_{A_{k_0-1}}(p_d) \leq d_{A_{k_0}}(p_d) < d_{A'}(p_d)$. Similarly $d_{A_{k_0-1}}(p_f) \geq d_{A_{k_0}}(p_f)$ and thus $d_{A_{k_0-1}}(p_d) + 1 < d_{A_{k_0-1}}(p_f)$ and $(p_d, \ldots, v_i, \ldots, v_j, \ldots, p_f)$ is augmenting. If $D(A_{k_0-1}) = D(A_{k_0})$, then we reach a contradiction with the definition of k_0. Otherwise we know that $D(A_{k_0-1}) = d_{A_{k_0-1}}(p_{f_{k_0-1}})$ and can prove that $(p_d, \ldots, v_i, \ldots, v_j, \ldots, p_{f_{k_0-1}})$ is an augmenting path with all the properties we want, thus another contradiction.

Finally, we prove that v_i is after v_j in x_{k_0-1}. Therefore, $y = (p_{d_{k_0}}, \ldots, v_j, \ldots, p_f)$ is an alternating path according to A_{k_0-1}. As previously, $d_{A_{k_0-1}}(p_{d_{k_0}}) < d_{A'}(p_{d_{k_0}}) \leq D(A')$. Thus $d_{A_{k_0-1}}(p_{d_{k_0}}) + 1 \leq D(A')$. Similarly, $D(A') < D(A_{k_0}) = d_{A_{k_0}}(p_f) \leq d_{A_{k_0-1}}(p_f)$. Thus, y is augmenting and if $D(A_{k_0-1}) = D(A_{k_0})$, then we reach a contradiction with the definition of k_0. Otherwise, we consider $y' = (p_d, \ldots, v_i, \ldots, p_{f_{k_0}})$ and reach a similar contradiction.

Hence, we prove that, in any case, $k_0 > 0$ implies the existence of an augmenting path according to A_{k_0-1} with all desired properties. Therefore, $k_0 = 0$ and there is an augmenting path according to $A_0 = A$, what achieves the proof of the theorem.

□

Based on Theorem 4.10, we can build an algorithm to find an assignment with minimum maximum degree as follows.

Algorithm 4: FindAssignment(G)

Input: A bipartite graph $G = (P, T, E)$
Output: An assignment A of G of minimum maximal degree
$A = \emptyset$;
foreach $t_j \in T$ **do**
  Choose a random p_i such that $(p_i, t_j) \in E$ and add this edge to A;
while *there exists an augmenting path according to* A **do**
  Compute an augmenting path x according to A;
  $A \leftarrow A \otimes x$;
return A

To prove the termination of FindAssignment, let us consider $\sum d_A(p_i)^2$. If $x = (p_d, \ldots, p_f)$ is an augmenting path, then

$$\begin{aligned}\sum d_{A\otimes x}(p_i)^2 - \sum d_A(p_i)^2 &= (d_A(p_d)+1)^2 + (d_A(p_f)-1)^2 \\ &\quad - d_A(p_d)^2 - d_A(p_f)^2 \\ &= 2(d_A(p_d) + 1 - d_A(p_f)) < 0\end{aligned}$$

Therefore, $\sum d_A(p_i)^2$ is decreasing during the execution of FindAssignment. Since this value is bounded (trivially by 0), there is an instant where there no longer exists an augmenting path and FindAssignment terminates, returning an assignment with minimum maximum degree.

However, in FindAssignment, the search of augmenting path can be expensive. This is why we propose a faster algorithm, BestAssignment.

Algorithm 5: BestAssignment (G)

Input: A bipartite graph $G = (P, T, E)$
Output: An assignment A of G of minimum maximal degree and minimum total load imbalance

```
A = ∅ ;
while A is not an assignment do
    foreach p_i ∈ P do
        if There is the alternating path according to A from p to t
         unassigned then
            x = the smallest such path ;
            A ← A ⊗ x ;
        else
            Remove p from P
return A
```

Trivially BestAssignment has a worst case of $O(nm)$ (therefore $O(n^2)$ where the number of edges is a function of n, that is the case in our problem) and in practice performs well because more of the alternating path it finds are very short, in particular on the fist step. However its optimality is not immediate.

Theorem 4.14. BestAssignment *returns an Assignment with no augmenting path.*

Proof. Let A_∞ be the final set returned by BestAssignment. If we suppose there is always an edge incident to each task, trivially A_∞ is an assignation. It is constructed only using alternating path and there is always, for each task, a such path from a processor (in particular a simple edge from one of its neighbours). Let us now prove the optimality of this assignation by showing there is no augmenting path according to A_∞.

Let p_1 be the first processor, if any, such that there is no alternating path from p to an unassigned task according to the actual state of A we denote A_1. Let us now define P_1 as the subset of P that contains all the processors that can be reach with an alternating path from p_1 according to A_1. Let T_1 be the neighbourhood of P_1, *i.e.* $T_1 = \{t \in T,\ \exists p \in P_1,\ (p,t) \in E\}$.

We now want to show that for all $t \in T_1$ there exists p in P_1 such that $(p,t) \in A_1$. Let t be in T_1 and $p \in P_1$ be in its neighbourhood. By definition of P_1 there exists an alternating path $x = (p_1, \ldots, p)$ according to A_1. If $(p,t) \in A_1$ we have our result, else $(p_1, \ldots, p, t)$ is a valid alternating path according to A_1. By definition of p_1, t is not unassigned and then there is p' such that $(p',t) \in A_1$ and thus $(x, \ldots, p, t, p')$ is also a valid alternating path and so $p' \in P_1$. Therefore we have for all $t \in T_1$ there exists p in P_1 such that $(p,t) \in A_1$.

Let now A_{1,T_1} (respectively A_{∞,T_1}) be $\{(p,t) \in A_1,\ t \in T_1\}$ (respectively $\{(p,t) \in A_\infty,\ t \in T_1\}$). We also denote similarly A'_{T_1} for all A' that is a partial assignation. Then, for all partial assignation A' after A_1, $A_{1,T_1} = A'_{T_1}$. Else let A' the first one with $A_{1,T_1} \otimes A'_{T_1} \neq \emptyset$. Because all the edges of A_1 from T_1 goes to an element of P_1, we know that there is $(p,t) \in A_{1,T_1} \otimes A'_{T_1}$ such that $p \in P_1$. In addition, during the assignation just before A' (we denote A''), there is an alternating path that contains (p,t). Let $x = (p_d, \ldots, p, t, \ldots, t')$ be this path ($A' = A'' \otimes x$). We know that t' is unassigned in A'' and thus $t' \notin T_1$ and so $x = (p_d, \ldots, p, t, \ldots, p_f, t')$ with $p_f \notin P_1$. Hence there is p' such that p' is the first vertex in $P \setminus P_1$. Without loss of generality we can assume that p' is after t'. By definition $A''_{T_1} = A_{1,T_1}$ and then $y = (p_1, \ldots, p, t, p')$ is a valid alternating path according to A_1 and A'' and therefore $p' \in P_1$ that is a contradiction. Then, for all partial assignation A' after A_1, $A_{1,T_1} = A'_{T_1}$. Similarly we also prove that $d_{A_1}(p) = d_{A'}(p)$ for all p in P_1 and, by construction $|d_{A_1}(p) - d_{A_1}(p')| \leq 1$ for all $p, p \in P_1^2$ (at each step of the algorithm the degree of the departure vertex increase of one).

Now we define p_i as the first processor of $P \setminus \bigcup_{1 \leq j < i} P_j$ if any, such that there is no alternating path from p to an unassigned task, with P_i the subset of $P \setminus \bigcup_{1 \leq j < i} P_j$ that contains all the processors that can be reach with an alternating path from p_1 according to the current partial assignation A_i. We also define T_i its neighbourhood in $T \setminus \bigcup_{1 \leq j < i} T_j$. Finally we also denote $P_\infty = P \setminus \bigcup P_i$ and T_∞ its neighbourhood in $T \setminus \bigcup T_i$ and T_∞.

With the same reasoning than for P_1, T_1 we proved:

- For all $t \in T_i$ there exists p in P_i such that $(p,t) \in A_i$.
- For all partial assignation A' after A_i, $A_{i,T_i} = A'_{T_i}$.
- For all partial assignation A' after A_i, $d_{A_i}(p) = d_{A'}(p)$ for all p in P_i and $|d_{A_i}(p) - d_{A_i}(p')| \leq 1$ for all $p, p \in P_i^2$.

Note that $|d_{A_i}(p) - d_{A_i}(p')| \leq 1$ for all $p, p \in P_i^2$ and for all $t \in T_i$ there exists p in P_i such that $(p,t) \in A_i$ are true for $i = \infty$. Note also that P_∞ can be equal to P.

We now want to prove that there is no augmenting path according to A_∞. First, as $|d_{A_i}(p) - d_{A_i}(p')| \leq 1$ for all $p, p \in P_i^2$ and $d_{A_i}(p) = d_{A_\infty}(p)$ for all p in P_i. Thus there is no augmenting path insides the P_is.

In addition we easily note that $|(max\ d_{A_i}(p)) - (min\ d_{A_j}(p'))| \leq 1$ for $i < j$. Therefore there is no augmenting path from a vertex of P_j to a vertex of P_i if $i < j$. Hence the only possible augmenting path are from P_i to P_j with $i < j$. However, in this case, there is t in T_i such there exists p in $P \setminus P_i$ such that $(p,t) \in A_\infty$ (a necessary condition to a such alternating path). Yet we prove for all $(p,t) \in A_{\infty,T_i} = A_{i,T_i}$, $p \in P_i$ and thus we have our proof of optimality of BestAssignment.

□

5 Performance Analysis of Optimal Assignments

In Section 4.1, we show how the performance of the greedy algorithm can be analyzed using Balls into Bins literature and we propose in Section 4.4 a new algorithms, BestAssignment, that computes efficiently an optimal assignment, given an initial randomized distribution of data chunks similar to the one achieved by HDFS. In this section, we aim at analyzing the performance of BestAssignment algorithm. We denote as *perfect* an assignment that achieves the $\lceil n/p \rceil$ bound, and as *quasi-perfect* an assignment whose maximum degree is $\lceil n/p \rceil + 1$. Note that BestAssignment is a deterministic algorithm and that randomness only comes from the initial chunk distribution only, and that $\lceil n/p \rceil$ is an absolute bound that does not depend of the quality of the distribution.

Our main result (Theorem 5.3) states that BestAssignment finds with high probability (when the number of tasks becomes large) a *quasi-perfect* assignment. This result is proved under the assumption of a minimal replication ratio ($r = 2$) and therefore a fortiori holds true for all possible of $r \geq 2$, as established in Theorem5.11. The analysis technique we propose to establish this result is inspired by the seminal paper of Walkup [27]. We also rely on classical results on bipartite graph's matching from the literature, such as Hall's Theorem.

Theorem 5.1 (Hall's Theorem [14])**.** *Let $G = (P, T, E)$ be a bipartite graph. There exists a perfect matching (of cardinal $min(|P|, |T|)$) of G if and only if for all subset T' of T, its neighborhood P' verifies $|P'| \geq |T'|$.*

and its corollary

Lemma 5.2. *Let $G = (P, T, E)$ be a bipartite graph. There exists an assignment of G of maximum degree m if and only if for all subset T' of T, its neighborhood P' satisfies $m|P'| \geq |T'|$.*

Proof. Let T' be a subset of T and let $v_G(T')$ be its neighborhood in G. By construction of $G^m = (P^m, T, E^m)$, $|v_{G^m}(T')| = m|v_G(T')|$. In addition, thanks to Lemma 4.6, there exists an assignment of G of maximum degree m if and only if there is a perfect matching of G^m, what is equivalent (Theorem 5.1) to $\forall T' \subseteq T$, $|T'| \leq |v_{G^m}(T')| = m|v_G(T')|$. □

5.1 Probability of Existence of a Quasi-Perfect Assignment

In what follows, we rely on Lemma 5.2 to analyze the probability that there exists an assignment of maximum degree m given a random initial distribution of the data chunks. Let t_j be a task of T and let P' be a subset of P of cardinal q. The probability that the entire neighborhood of t_j is included in P' is $\left(\frac{q}{p}\right)^r$ using a draw with replacement (a more pessimistic hypothesis that change only slightly the probabilities).

Let $X_{P'}$ be the random variable that represents the number of tasks whose neighborhood is included in P'. $X_{P'}$ follows a binomial law of parameter n (the number of tasks) and $\left(\frac{q}{p}\right)^r$ (the probability that a task has its neighborhood included in P'), where q denotes the cardinal of P'. Therefore,

$$Pr(X_{P'} = k) = \binom{n}{k} \left(\left(\frac{q}{p}\right)^r\right)^k \left(1 - \left(\frac{q}{p}\right)^r\right)^{n-k}.$$

Let $Y_{q,k}$ denote the number of subsets of P of size q such that exactly k elements of T have their neighborhood included in this subset, *i.e.*

$$Y_{q,k} = \sum_{\substack{P' \subseteq P \\ |P'|=q}} \mathbb{1}_{X_{P'}=k}$$

where $\mathbb{1}_{X_{P'}=k}$ is the random variable equal to 1 when $X_{P'} = k$ and to 0 otherwise. Therefore, $E(\mathbb{1}_{X_{P'}=k}) = Pr(X_{P'} = k)$ and, by linearity of expectation,

$$\begin{aligned}
E(Y_{q,k}) &= E\left(\sum_{\substack{P' \subseteq P \\ |P'|=q}} \mathbb{1}_{X_{P'}=k}\right) \\
&= \sum_{\substack{P' \subseteq P \\ |P'|=q}} E(\mathbb{1}_{X_{P'}=k}) \\
&= \sum_{\substack{P' \subseteq P \\ |P'|=q}} Pr(X_{P'} = k) \\
&= \sum_{\substack{P' \subseteq P \\ |P'|=q}} \binom{n}{k} \left(\left(\frac{q}{p}\right)^r\right)^k \left(1 - \left(\frac{q}{p}\right)^r\right)^{n-k} \\
&= \binom{p}{q} \binom{n}{k} \left(\frac{q}{p}\right)^{rk} \left(1 - \left(\frac{q}{p}\right)^r\right)^{n-k}.
\end{aligned}$$

Lemma 5.2 states that there exists an assignment of maximal degree m if and only if, for all subset T' of T, $m|P'| \geq |T'|$, where P' is the neighborhood of T'. Therefore, there is no such assignment if and only if there exists T' and its neighborhood P' such that $m|P'| < |T'|$ and thus if and only if there exists (q, i), with $q \in [0, p], i \in \mathbb{N}^*$, such that $Y_{q,mq+i} \geq 1$.

Note the condition $(Y_{q,mq+m+i} \geq 1)$ is included in the condition $(Y_{q+1,m(q+1)+i} \geq 1)$. Indeed, if there is a set of q processors that contains the entire neighborhood of $mq + m + i$ tasks, there also exists a set of $q + 1$ processors that contains this neighborhood (obtained by adding any processor not already in

the subset). Thus, $Y_{q,mq+m+i} \geq 1$ implies $Y_{q+1,m(q+1)+i} \geq 1$. On the contrary, $Y_{q+1,m(q+1)+i} = 0$ implies $Y_{q,mq+m+i} = 0$. Therefore, we can focus on the events $(Y_{q,mq+i} \geq 1)$ with $i \in [1, m]$ only. Let us define the random variable $Z_m = \sum_{q=0}^{p} \sum_{i=1}^{m} Y_{q,mq+i}$. If $Z_m = 0$, then $Y_{q,mq+i} = 0$ for all (q, i) and there exists an assignment of maximum degree m. Otherwise, if $Z_m \geq 1$, there exists a (q, i) such that $Y_{q,mq+i} \geq 1$ and then there is no assignment of maximum degree m.

Using to Markov inequality, we obtain

$$\begin{aligned} Pr(Z_m \geq 1) &\leq \frac{E(Z_m)}{1} \\ &\leq \sum_{q=0}^{p} \sum_{i=1}^{m} Y_{q,mq+i} \\ &\leq \sum_{q=0}^{p} \sum_{i=1}^{m} \binom{p}{q} \binom{n}{mq+i} \left(\frac{q}{p}\right)^{r(mq+i)} \left(1 - \left(\frac{q}{p}\right)^{r}\right)^{n-mq-i}. \end{aligned}$$

5.2 Existence of quasi-perfect assignment in the case $p = \alpha n$, $\alpha \in]0, 1]$

The following theorem proves that quasi-perfect assignment exists with high probability when the number of tasks becomes large. We will show through simulations in Section 6 that this result holds true in practice even for very small values of n.

Theorem 5.3. *Let us assume that (i) $r = 2$, (ii) n is a multiple of p, i.e. $p = \alpha n$ where $1/\alpha \in \mathbb{N}^*$ and (iii) $m = \frac{n}{p} + 1$. Then $Pr(Z_m \geq 1) = O\left(\frac{1}{n}\right)$.*

Proof. The proof of Theorem 5.3 is a direct consequence of Theorem 5.4 and Theorem 5.5, that consider respectively the case $\alpha = 1$ and the case $\alpha < 1$. □

Theorem 5.4. *Let us assume that (i) $r = 2$, (ii) $n = p$ and (iii) $m = \frac{n}{p} + 1$. Then $Pr(Z_m \geq 1) = O\left(\frac{1}{n}\right)$.*

Proof. Let us now assume that $n = p$, $m = 2$ and $r = 2$. Then,

$$Pr(Z_m \geq 1) \leq \sum_{q=0}^{n} Y_{q,2q+1} + Y_{q,2q+2}$$

with

$$E(Y_{q,2q+i}) = \binom{n}{q} \binom{n}{2q+i} \left(\frac{q}{n}\right)^{4q+2i} \left(1 - \left(\frac{q}{n}\right)^{2}\right)^{n-2q-i}.$$

Let us remark that $\binom{n}{2q+1} = \binom{n}{2q+2} = 0$ for $q > \frac{n}{2}$. Then, in this case, $Y_{q,2q+1} = 0$ and $Y_{q,2q+2} = 0$. Furthermore, $Y_{0,k} = 0$ and hence,

$$Pr(Z_m \geq 1) \leq \sum_{q=1}^{\frac{n}{2}} E(Y_{q,2q+1}) + E(Y_{q,2q+2}).$$

In addition,

$$\begin{aligned}
E(Y_{q,2q+2}) &= \binom{n}{q}\binom{n}{2q+2}\left(\frac{q}{n}\right)^{4q+4}\left(1-\left(\frac{q}{n}\right)^2\right)^{n-2q-2} \\
&= \binom{n}{q}\binom{n}{2q+1}\frac{n-2q-1}{2q+2}\left(\frac{q}{n}\right)^{4q+2}\left(\frac{q}{n}\right)^2\left(1-\left(\frac{q}{n}\right)^2\right)^{n-2q-1}\left(\frac{n^2-q^2}{n^2}\right)^{-1} \\
&= E(Y_{q,2q+1})\frac{n-2q-1}{2q+2}\left(\frac{q}{n}\right)^2\frac{n^2}{n^2-q^2} \\
&= E(Yq,2q+1)\frac{n-2q-1}{n-q}\frac{q}{2q+2}\frac{q}{n+q}.
\end{aligned}$$

Note that $\frac{q}{2q+2} < \frac{1}{2}$ and, since $n \geq q$, $\frac{q}{n+q} \leq \frac{1}{2}$. In addition, $x \mapsto \frac{n-2x-1}{n-x}$ is a decreasing function on $\left[0, \frac{n}{2}\right]$. Therefore, $\frac{n-2q-1}{n-q} \leq \frac{n-1}{n} < 1$ and

$$E(Y_{q,2q+2}) < \frac{E(Y_{q,2q+1})}{4}.$$

Using this bound on $E(Y_{q,2q+2})$, we obtain

$$\begin{aligned}
Pr(Z_m \geq 1) &= \sum_{q=1}^{\frac{n}{2}} E(Y_{q,2q+1}) + E(Y_{q,2q+2}) \\
&< \frac{5}{4}\sum_{q=1}^{\frac{n}{2}} E(Y_{q,2q+1}) \\
&< \frac{5}{4}\sum_{q=1}^{\frac{n}{2}} \binom{n}{q}\binom{n}{2q+1}\left(\frac{q}{n}\right)^{4q+2}\left(1-\left(\frac{q}{n}\right)^2\right)^{n-2q-1} \\
&< \frac{5}{4}\sum_{q=1}^{\frac{n}{2}} \binom{n}{q}\binom{n}{2q+1}\left(\frac{q}{n}\right)^{4q+2} \\
&< \frac{5}{4}\sum_{q=1}^{\frac{n}{2}} u_q
\end{aligned}$$

where $u_q = \binom{n}{q}\binom{n}{2q+1}\left(\frac{q}{n}\right)^{4q+2}$.

Let us now consider the sequence $(u_q)_{q\in\mathbb{N}^*}$. First,

$$\begin{aligned}
u_{q+1} &= \binom{n}{q+1}\binom{n}{2q+3}\left(\frac{q+1}{n}\right)^{4q+6} \\
&= \binom{n}{q}\frac{n-q}{q+1}\binom{n}{2q+1}\frac{(n-2q-1)(n-2q-2)}{(2q+2)(2q+3)}\left(\frac{q}{n}\right)^{4q+2}\left(\frac{q+1}{q}\right)^{4q+2}\left(\frac{q+1}{n}\right)^4 \\
&= u_q\frac{n-q}{q+1}\frac{(n-2q-1)(n-2q-2)}{(2q+2)(2q+3)}\left(\frac{q+1}{q}\right)^{4q+2}\left(\frac{q+1}{n}\right)^4
\end{aligned}$$

$$
\begin{aligned}
&< u_q \frac{n-q}{q+1} \left(\frac{n-2q}{2q+2}\right)^2 \left(\frac{q+1}{q}\right)^{4q+2} \left(\frac{q+1}{n}\right)^4 \\
&< u_q \frac{n-q}{n}\frac{q+1}{q+1} \left(\frac{n-2q}{n}\right)^2 \left(\frac{q+1}{2q+2}\right)^2 \left(\frac{q+1}{q}\right)^{4q+2} \left(\frac{q+1}{n}\right) \\
&< u_q \left(1-\frac{q}{n}\right)\left(1-2\frac{q}{n}\right)^2 \frac{1}{4}\frac{q}{n} \left(\frac{q+1}{q}\right)^{4q+3}
\end{aligned}
$$

The function $x \mapsto x(1-x)(1-2x)^2$ is smaller than $\frac{1}{16}$ on $[0,1]$. Thus,

$$\frac{u_{q+1}}{u_q} < \frac{1}{64}\left(\frac{q+1}{q}\right)^{4q+3}.$$

Yet, $\lim_{q\to+\infty} \left(\frac{q+1}{q}\right)^{4q+3} = e^4 < 64$. Then, there exists q_0, independent of n, such that u_q is decreasing from q_0. Furthermore,

$$\frac{u_{q+1}}{u_q} < \frac{1}{4}\frac{q}{n} \max_{q\in\mathbb{N}^*}\left(\left(\frac{q+1}{q}\right)^{4q+3}\right) = \frac{1}{4}\frac{q}{n}2^7 = 32\frac{q}{n}.$$

Thus, if $\frac{n}{q} \geq 32$, u_q is a decreasing function. Therefore, as soon as $n \geq 32q_0$, then $\forall q \leq q_0$, $(u_q)_{q\in\mathbb{N}^*}$ is decreasing and

$$
\begin{aligned}
\Pr(Z_m \geq 1) &< \frac{5}{4}\sum_{q=1}^{n/2} u_q \\
&< \frac{5}{4}\sum_{q=1}^{n/2} u_1 \\
&< \frac{5}{4}\sum_{q=1}^{n/2} \binom{n}{1}\binom{n}{3}\left(\frac{1}{n}\right)^6 \\
&< \frac{5}{4}\frac{n}{2}n\frac{n(n-1)(n-2)}{6}\left(\frac{1}{n}\right)^6 \\
&< \frac{5}{4}\frac{n}{2}n\frac{n^3}{6}\left(\frac{1}{n}\right)^6 \\
&< \frac{5}{48n} = O\left(\frac{1}{n}\right)
\end{aligned}
$$

what achieves the proof of Theorem 5.4. □

Theorem 5.5. *Let us assume that (i) $r = 2$, (ii) n is a multiple of p, i.e. $p = \alpha n$ where $\alpha < 1$ and (iii) $m = \frac{n}{p} + 1$. Then $Pr(Z_m \geq 1) = O\left(\frac{1}{n}\right)$.*

Proof. We recall that

$$E(Y_{q,mq+i}) = \binom{p}{q}\binom{n}{mq+i}\left(\frac{q}{p}\right)^{2(mq+i)}\left(1-\left(\frac{q}{p}\right)^2\right)^{n-mq-i}.$$

In order to bound

$$Pr(Z_m \geq 1) \leq \sum_{q=0}^{p}\sum_{i=1}^{m} Y_{q,mq+i},$$

we first use the following lemma in order to eliminate one of the two sums.

Lemma 5.6.

$$Pr(Z_m \geq 1) < 2\sum_{q=1}^{p} E(Y_{q,mq+1}).$$

Proof.

$$\begin{aligned}
E(Y_{q,mq+i+1}) &= \binom{p}{q}\binom{n}{mq+i+1}\left(\frac{q}{p}\right)^{2(mq+i+1)}\left(1-\left(\frac{q}{p}\right)^2\right)^{n-mq-i-1} \\
&= \binom{p}{q}\binom{n}{mq+i}\frac{n-mq-i}{mq+i+1}\left(\frac{q}{p}\right)^{2}\left(\frac{q}{p}\right)^{2(mq+i)}\left(1-\left(\frac{q}{p}\right)^2\right)^{n-mq-i}\left(1-\left(\frac{q}{p}\right)^2\right)^{-1} \\
&= E(Y_{q,mq+i})\frac{n-mq-i}{mq+i+1}\left(\frac{q}{p}\right)^{2}\frac{p^2}{p^2-q^2} \\
&= E(Y_{q,mq+i})\frac{n-mq-i}{p-q}\frac{q}{mq+i+1}\frac{q}{p+q}.
\end{aligned}$$

Note that $\frac{q}{mq+i+1} < \frac{1}{m}$ and, since $p \geq q$, then $\frac{q}{p+q} \leq \frac{1}{2}$. In addition, the function $x \mapsto \frac{n-mx-i}{p-x}$ is a decreasing function on $[1, p[$ (we assume that $q \geq 1$, otherwise $E(Y_{q,mq+i}) = 0$) and therefore

$$\frac{n-mq-i}{p-q} \leq \frac{n-m-i}{p-1} = \frac{m(\frac{n}{m}-1-\frac{i}{m})}{p-1}.$$

Let us recall that $m = \frac{n}{p}+1$. Therefore, $\frac{m}{n} \geq \frac{1}{p}+\frac{1}{n}$ and thus $\frac{n}{m} \leq \frac{np}{n+p} \leq \frac{np}{n} \leq p$. Finally,

$$\frac{n-mq-i}{p-q} \leq \frac{m(\frac{n}{m}-1-\frac{i}{m})}{p-1} \leq \frac{m(p-1-\frac{i}{m})}{p-1} \leq m$$

$$\text{and } E(Y_{q,mq+i+1}) \leq \frac{1}{2}E(Y_{q,mq+i}).$$

Using this bound on $E(Y_{q,mq+i+1})$, we obtain

$$\begin{aligned} Pr(Z_m \geq 1) &\leq \sum_{q=0}^{p}\sum_{i=1}^{m} E(Y_{q,mq+i}) \\ &\leq \sum_{q=1}^{p}\sum_{i=1}^{m} \frac{1}{2^{i-1}} E(Y_{q,mq+1}) \\ &\leq \sum_{q=1}^{p} \frac{1-\frac{1}{2^m}}{1-\frac{1}{2}} E(Y_{q,mq+1}) \\ &\leq 2\sum_{q=1}^{p} E(Y_{q,mq+1}), \end{aligned}$$

what achieves the proof of Lemma 5.6. □

In order to prove that $Pr(Z_m \geq 1) = O\left(\frac{1}{n}\right)$, we therefore need to bound $E(Y_{q,mq+1})$. Let us first recall the following well known upper and lower bounds on the factorial. For any $n \in \mathbb{N}^*$,

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n < n! < \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$

and thus, for any $n > p > 0$,

$$\begin{aligned} \binom{n}{p} &= \frac{n!}{p!(n-p)!} \\ &< \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}}{\sqrt{2\pi p}\left(\frac{p}{e}\right)^p \sqrt{2\pi(n-p)}\left(\frac{n-p}{e}\right)^{n-p}} \\ &< \frac{1}{\sqrt{2\pi}} \frac{n^{n+\frac{1}{2}}}{p^{p+\frac{1}{2}}(n-p)^{n-p+\frac{1}{2}}} e^{\frac{1}{12n}}. \end{aligned}$$

Due to the domain of validity of above inequality, the rest of the proof will be split into two parts. The case $mq+1 = n$ is considered in Lemma 5.7 and the case $mq+1 < n$ is considered in Lemmas 5.8, 5.9, 5.10.

Lemma 5.7. *If $mq+1 = n$, then $E(Y_{q,mq+1}) = O\left(\frac{1}{n}\right)$.*

Proof. If $mq+1 = n$, then $q = \frac{n-1}{m}$.

$$
\begin{aligned}
E\left(Y_{\frac{n-1}{m},n}\right) &= \binom{p}{\frac{n-1}{m}}\binom{n}{n}\left(\frac{n-1}{mp}\right)^{2n}\left(1-\left(\frac{n-1}{mp}\right)^{2}\right)^{n-n} \\
&= \binom{p}{\frac{n-1}{m}}\left(\frac{n-1}{mp}\right)^{2n} \\
&\leq \frac{1}{\sqrt{2\pi}}\frac{p^{p+\frac{1}{2}}}{\left(\frac{n-1}{m}\right)^{\frac{n-1}{m}+\frac{1}{2}}\left(p-\frac{n-1}{m}\right)^{p-\frac{n-1}{m}+\frac{1}{2}}}\left(\frac{n-1}{(1+\alpha)n}\right)^{2n} \\
&\leq \frac{1}{\sqrt{2\pi p}}\frac{1}{\left(\frac{n-1}{mp}\right)^{\frac{n-1}{m}+\frac{1}{2}}\left(1-\frac{n-1}{mp}\right)^{p-\frac{n-1}{m}+\frac{1}{2}}}\left(\frac{1}{(1+\alpha)}\right)^{2n}\left(1-\frac{1}{n}\right)^{2n}.
\end{aligned}
$$

Note that $\left(\frac{1}{(1+\alpha)}\right)^{2n}\left(1-\frac{1}{n}\right)^{2n} = O\left(\frac{1}{(1+\alpha)^{2n}}\right)$. In addition,

$$
\begin{aligned}
\left(\frac{n-1}{mp}\right)^{\frac{n-1}{m}+\frac{1}{2}} &= \left(\frac{1-\frac{1}{n}}{1+\alpha}\right)^{p\frac{n-1}{mp}+\frac{1}{2}} \\
&= \left(\frac{1-\frac{1}{n}}{1+\alpha}\right)^{\alpha n\frac{1-\frac{1}{n}}{1+\alpha}+\frac{1}{2}} \\
&= \left(\frac{1-\frac{1}{n}}{1+\alpha}\right)^{\frac{\alpha n}{1+\alpha}-\frac{\alpha}{1+\alpha}+\frac{1}{2}} \\
&= \left(\frac{1}{1+\alpha}\right)^{\frac{\alpha n}{1+\alpha}}\left(\frac{1}{1+\alpha}\right)^{-\frac{\alpha}{1+\alpha}+\frac{1}{2}}\left(1-\frac{1}{n}\right)^{\frac{\alpha n}{1+\alpha}-\frac{\alpha}{1+\alpha}+\frac{1}{2}} \\
&= O\left(\frac{1}{(1+\alpha)^{\frac{\alpha n}{1+\alpha}}}\right)
\end{aligned}
$$

and

$$
\begin{aligned}
\left(1-\frac{n-1}{mp}\right)^{p-\frac{n-1}{m}+\frac{1}{2}} &= \left(1-\frac{1-\frac{1}{n}}{1+\alpha}\right)^{p\left(1-\frac{n-1}{mp}\right)+\frac{1}{2}} \\
&= \left(\left(1-\frac{1}{1+\alpha}\right)\left(1-\frac{1}{n}\right)\right)^{p\left(1-\frac{1-\frac{1}{n}}{1+\alpha}\right)+\frac{1}{2}} \\
&= \left(\frac{\alpha}{1+\alpha}\right)^{\alpha n\left(1-\frac{1}{1+\alpha}\right)-\frac{\alpha}{\alpha+1}+\frac{1}{2}}\left(1-\frac{1}{n}\right)^{\alpha n\left(1-\frac{1}{1+\alpha}\right)-\frac{\alpha}{\alpha+1}+\frac{1}{2}} \\
&= O\left(\left(\frac{\alpha}{1+\alpha)}\right)^{\alpha n\left(1-\frac{1}{1+\alpha}\right)}\right)
\end{aligned}
$$

Thus,

$$E(Y_{\frac{n-1}{m},n}) \leq O\left(\frac{1}{\sqrt{p}} \times \frac{1}{\left(\frac{\alpha}{1+\alpha}\right)^{\alpha n\left(1-\frac{1}{1+\alpha}\right)}} \times \frac{1}{\frac{1}{(1+\alpha)^{\frac{\alpha n}{1+\alpha}}}} \times \frac{1}{(1+\alpha)^{2n}}\right)$$
$$\leq O\left(\frac{1}{\sqrt{p}} \frac{1}{\alpha^{\frac{\alpha^2 n}{1+\alpha}}(1+\alpha)^{(2-\alpha)n}}\right)$$
$$\leq O\left(\frac{e^{-n\left(\frac{\alpha^2}{1+\alpha}\ln(\alpha)+(2-\alpha)\ln(1+\alpha)\right)}}{\sqrt{\alpha n}}\right)$$

Let $g(y) = \frac{y^2}{1+y}\ln(y)+(2-y)\ln(1+y)$. Then, $g'(y) = \frac{y(y+2)\ln(y)+2(y+1)-(y+1)\ln(1+y)}{(1+y)^2}$ and $g''(y) = -\frac{y+1-2\ln(y)}{(1+y)^3}$. $\ln(y) \leq y-1$ and thus $y+1-2\ln(y) \geq 3-y > 0$ on $]0,1[$. Therefore, $g''(y) < 0$ and $\forall y \in]0,1[$, $g'(y) \geq g'(1) = \frac{2-\ln(2)}{2} > 0$. Hence $\forall y \in]0,1[$, $g(y) > \lim_{y\to 0} g(y) = 0$. Then, if we denote $K'' = g(\alpha)$, $K'' > 0$,

$$E\left(Y_{\frac{n-1}{m},n}\right) \leq O\left(\frac{e^{-nK''}}{\sqrt{\alpha n}}\right) = O\left(\frac{1}{n}\right),$$

what ends the proof of Lemma 5.7. □

Let us now consider the case $mq+1 < n$ and the corresponding sum $\sum_{q=1}^{\frac{n-1}{m}-1} E(Y_{q,mq+1})$. Lemma 5.8 provides a first bound on this sum.

Lemma 5.8. *If $mq+1 < n$,*

$$E(Y_{q,mq+1}) < \frac{e^{\frac{1}{12p}} e^{\frac{1}{12n}}}{2\pi\sqrt{np}} e^{n\left(f_{0,n}(x,\alpha)+\frac{1}{n}f_{1,n}(x,\alpha)\right)}$$

where $x = \frac{q}{p}$,

$$f_{0,n}(x,\alpha) = -\ln(1+\alpha) + (\alpha+2)x\ln(x) + (1-x-\alpha)\ln(1-x) + (1-(1+\alpha)x)\ln(1+x)$$
$$-(\alpha+1)x\ln\left(\frac{1}{n(\alpha+1)}+x\right) - (1+\alpha)\left(\frac{1}{1+\alpha}-x\right)\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)$$

and

$$f_{1,n}(x,\alpha) = -\ln(1+\alpha) + \frac{1}{2}\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right) - \ln(1+x) + \frac{3}{2}\left(\ln(x) - \ln\left(\frac{1}{n(\alpha+1)}+x\right) - \ln(1-x)\right).$$

Proof. Note that $mq+1<n$ implies $q<\frac{n-1}{m}<\frac{n-1}{\frac{n}{p}+1}<p\frac{n-1}{n+p}<p$. Therefore, with the same previous bound on the binomial coefficients, for $mq+1<n$,

$$E(Y_{q,mq+1}) = \binom{p}{q}\binom{n}{mq+1}\left(\frac{q}{p}\right)^{2(mq+1)}\left(1-\left(\frac{q}{p}\right)^2\right)^{n-mq-1}$$
$$< \frac{e^{\frac{1}{12p}}e^{\frac{1}{12n}}}{2\pi}\frac{p^{p+\frac{1}{2}}}{q^{q+\frac{1}{2}}(p-q)^{p-q+\frac{1}{2}}}\frac{n^{n+\frac{1}{2}}}{(mq+1)^{mq+\frac{3}{2}}(n-mq-1)^{n-mq-\frac{1}{2}}}\left(\frac{q}{p}\right)^{2(mq+1)}\left(1-\left(\frac{q}{p}\right)^2\right)^{n-mq-}$$

Let us denote $x=\frac{q}{p}$. In addition $mp=n+p=(\alpha+1)n$ with $p=\alpha n$ and $\alpha\in]0,1[$. Note that $x\in I_n=\left[\frac{1}{\alpha n},\frac{n-1}{(\alpha+1)n}-\frac{1}{\alpha n}\right]$. Using above notations,

$$\begin{aligned}\frac{p^{p+\frac{1}{2}}}{q^{q+\frac{1}{2}}(p-q)^{p-q+\frac{1}{2}}} &= \frac{p^{p+\frac{1}{2}}}{x^{q+\frac{1}{2}}(1-x)^{p-q+\frac{1}{2}}p^{p+1}}\\ &= \frac{1}{x^{q+\frac{1}{2}}(1-x)^{p-q+\frac{1}{2}}\sqrt{p}}\\ &= \frac{1}{\sqrt{p}}\times\frac{1}{x^{xp+\frac{1}{2}}(1-x)^{p(1-x)+\frac{1}{2}}}\\ &= \frac{1}{\sqrt{p}}e^{-\left(\left(xp+\frac{1}{2}\right)\ln(x)+\left(p(1-x)+\frac{1}{2}\right)\ln(1-x)\right)}\\ &= \frac{1}{\sqrt{p}}e^{-n\left(\left(\alpha x+\frac{1}{2n}\right)\ln(x)+\left(\alpha(1-x)+\frac{1}{2n}\right)\ln(1-x)\right)}\end{aligned}$$

and

$$\begin{aligned}\frac{n^{n+\frac{1}{2}}}{(mq+1)^{mq+\frac{3}{2}}(n-mq-1)^{n-mq-\frac{1}{2}}} &= \frac{n^{n+\frac{1}{2}}}{\left(\frac{1}{mp}+x\right)^{mq+\frac{3}{2}}\left(\frac{n-1}{mp}-x\right)^{n-mq-\frac{1}{2}}(mp)^{n+1}}\\ &= \frac{n^{n+\frac{1}{2}}}{\left(\frac{1}{mp}+x\right)^{mpx+\frac{3}{2}}\left(\frac{n-1}{mp}-x\right)^{mp\left(\frac{n}{mp}-x\right)-\frac{1}{2}}(mp)^{n+1}}\\ &= \frac{1}{\sqrt{n}}\frac{\left(\frac{n}{mp}\right)^{n+1}}{\left(\frac{1}{mp}+x\right)^{mpx+\frac{3}{2}}\left(\frac{n-1}{mp}-x\right)^{mp\left(\frac{n}{mp}-x\right)-\frac{1}{2}}}\\ &= \frac{1}{\sqrt{n}}e^{(n+1)\ln\left(\frac{n}{mp}\right)-\left(\left(mpx+\frac{3}{2}\right)\ln\left(\frac{1}{mp}+x\right)+\left(mp\left(\frac{n}{mp}-x\right)-\frac{1}{2}\right)\ln\left(\frac{n-1}{mp}-x\right)\right)}\\ &= \frac{1}{\sqrt{n}}\times\\ & e^{n\left(\left(1+\frac{1}{n}\right)\ln\left(\frac{1}{1+\alpha}\right)-\left(\left((\alpha+1)x+\frac{3}{2n}\right)\ln\left(\frac{1}{n(\alpha+1)}+x\right)+\left((1+\alpha)\left(\frac{1}{1+\alpha}-x\right)-\frac{1}{2n}\right)\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)\right)\right.}\end{aligned}$$

Similarly,

$$\left(\frac{q}{p}\right)^{2mq+2} = x^{2mq+2} = e^{(2mq+2)\ln(x)} = e^{n\left(\left(2(\alpha+1)x+\frac{2}{n}\right)\ln(x)\right)}$$

and

$$\begin{aligned}\left(1-\left(\frac{q}{p}\right)^2\right)^{n-mq-1} &= \left(1-x^2\right)^{mp\left(\frac{n}{mp}-x\right)-1}\\ &= e^{\left(mp\left(\frac{n}{mp}-x\right)-1\right)(\ln(1+x)+\ln(1-x))}\\ &= e^{n\left((1+\alpha)\left(\frac{1}{1+\alpha}-x\right)-\frac{1}{n}\right)(\ln(1+x)+\ln(1-x))}.\end{aligned}$$

Therefore,

$$E(Y_{q,mq+1}) < \frac{e^{\frac{1}{12p}}e^{\frac{1}{12n}}}{2\pi\sqrt{np}}e^{n\left(f_{0,n}(x,\alpha)+\frac{1}{n}f_{1,n}(x,\alpha)\right)}$$

where

$$\begin{aligned}f_{0,n}(x,\alpha) &= \ln\left(\frac{1}{1+\alpha}\right)+2(\alpha+1)x\ln(x)+(1+\alpha)\left(\frac{1}{1+\alpha}-x\right)(\ln(1+x)+\ln(1-x))-\alpha x\ln(x)\\ &\quad-\alpha(1-x)\ln(1-x)-(\alpha+1)x\ln\left(\frac{1}{n(\alpha+1)}+x\right)-(1+\alpha)\left(\frac{1}{1+\alpha}-x\right)\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)\\ f_{0,n}(x,\alpha) &= -\ln(1+\alpha)+(\alpha+2)x\ln(x)+(1-x-\alpha)\ln(1-x)+(1-(1+\alpha)x)\ln(1+x)\\ &\quad-(\alpha+1)x\ln\left(\frac{1}{n(\alpha+1)}+x\right)-(1+\alpha)\left(\frac{1}{1+\alpha}-x\right)\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)\end{aligned}$$

and

$$\begin{aligned}f_{1,n}(x,\alpha) &= \ln\left(\frac{1}{1+\alpha}\right)+\frac{1}{2}\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)+2\ln(x)-\frac{1}{2}(\ln(x)+\ln(1-x))\\ &\quad-\frac{3}{2}\ln\left(\frac{1}{n(\alpha+1)}+x\right)-(\ln(1+x)+\ln(1-x))\\ f_{1,n}(x,\alpha) &= -\ln(1+\alpha)+\frac{1}{2}\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)-\ln(1+x)+\frac{3}{2}\left(\ln(x)-\ln\left(\frac{1}{n(\alpha+1)}+x\right)-\ln(1-x)\right).\end{aligned}$$

what completes the proof of Lemma 5.8. □

The following Lemma 5.9 proves that we can concentrate on the detailed analysis of a single function f (as defined below).

Lemma 5.9. *If* $mq+1<n$,

$$\sum_{q=1}^{\frac{n-1}{m}} E(Y_{q,mq+1}) \leq O\left(\frac{1}{n}\right) \sum_{x=q/p\in I_n} e^{nf(x)}$$

with $I_n = \left]\frac{1}{\alpha n}, \frac{n-1}{(\alpha+1)n} - \frac{1}{\alpha n}\right[$ *and* $f(x) = x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(x+(1-x)^{\frac{1}{x}}(1+x))$.

Proof. Let $f_{0,\infty}$ and $f_{1,\infty}$ be the limit functions of $f_{0,n}$ and $f_{1,n}$ when n goes to infinity. We have

$$\begin{aligned} f_{0,\infty}(x,\alpha) &= -\ln(1+\alpha) + (\alpha+2)x\ln(x) + (1-x-\alpha)\ln(1-x) + (1-(1+\alpha)x)\ln(1+x) \\ &\quad - (\alpha+1)x\ln(x) - (1-(1+\alpha)x)\ln\left(\frac{1}{1+\alpha} - x\right) \\ f_{0,\infty}(x,\alpha) &= -\ln(1+\alpha) + x\ln(x) + (1-x-\alpha)\ln(1-x) + (1-(1+\alpha)x)\ln(1+x) \\ &\quad - (1-(1+\alpha)x)\ln\left(\frac{1}{1+\alpha} - x\right) \end{aligned}$$

and

$$\begin{aligned} f_{1,\infty}(x,\alpha) &= -\ln(1+\alpha) + \frac{1}{2}\ln\left(\frac{1}{1+\alpha} - x\right) - \ln(1+x) + \frac{3}{2}\left(\ln(x) - \ln(x) - \ln(1-x)\right) \\ f_{1,\infty}(x,\alpha) &= -\ln(1+\alpha) + \frac{1}{2}\ln\left(\frac{1}{1+\alpha} - x\right) - \ln(1+x) - \frac{3}{2}\ln(1-x). \end{aligned}$$

In order to bound the different terms $f_{0,n}(x,\alpha)$ and $f_{1,n}(x,\alpha)$, we rely on the following technical claims (Claims 1, 2 and 3).

Claim 1. $\exists K\in\mathbb{R}, \forall\alpha$ *in* $]0,1[$, $\forall x\in I_n$ *and* $\forall n\in\mathbb{N}$,

$$f_{1,n}(x,\alpha) \leq f_{1,\infty}(x,\alpha) \leq K.$$

Proof. Let $g_n(x,\alpha) = f_{1,\infty}(x,\alpha) - f_{1,n}(x,\alpha)$. Then,

$$g_n(x,\alpha) = \frac{1}{2}\left(\ln\left(\frac{1}{1+\alpha} - x\right) - \ln\left(\frac{1-\frac{1}{n}}{1+\alpha} - x\right)\right) - \frac{3}{2}\left(\ln\left(\frac{1}{n(\alpha+1)} + x\right) - \ln(x)\right).$$

As $x\mapsto\ln(x)$ is increasing, $\frac{1}{1+\alpha} \geq \frac{1-\frac{1}{n}}{1+\alpha}$ and $\frac{1}{n(\alpha+1)} \geq 0$, we deduce that $\ln\left(\frac{1}{1+\alpha} - x\right) - \ln\left(\frac{1-\frac{1}{n}}{1+\alpha} - x\right) \geq 0$ and $\ln\left(\frac{1}{n(\alpha+1)} + x\right) - \ln(x) \geq 0$ that leads to $g_n(x,\alpha)\geq 0$ and $f_{1,n}(x,\alpha) \leq f_{1,\infty}(x,\alpha)$.

Note that $f_{1,\infty}$ is continuous on $\left[0, \frac{1}{1+\alpha}\right[$ and $\forall n$, $I_n \subseteq \left[0, \frac{1}{1+\alpha}\right[$. In addition $\lim_{x\to\frac{1}{1+\alpha}} f_{1,\infty}(x,\alpha) = -\infty$. Therefore, there exists $K\in\mathbb{R}$ such that $f_{1,\infty}(x,\alpha) \leq K$, what achieves the proof of Claim 1. □

Claim 2. *$\forall\alpha$ in $]0,1[$, $\forall x \in I_n$ and $\forall n \in \mathbb{N}$,*

$$f_{0,n}(x,\alpha) \leq f_{0,\infty}(x,\alpha) + C_n,$$

where $C_n = O\left(\frac{1}{n}\right)$.

Proof.

$$\begin{aligned} f_{0,n}(x,\alpha) - f_{0,\infty}(x,\alpha) &= (\alpha+2)x\ln(x) - (\alpha+1)x\ln\left(\frac{1}{n(\alpha+1)} + x\right) \\ &\quad - (1-(1+\alpha)x)\ln\left(\frac{1-\frac{1}{n}}{1+\alpha} - x\right) \\ &\quad - x\ln(x) + (1-(1+\alpha)x)\ln\left(\frac{1}{1+\alpha} - x\right) \\ &= (\alpha+1)x\left(\ln(x) - \ln\left(\frac{1}{n(\alpha+1)} + x\right)\right) \\ &\quad + (1-(1+\alpha)x)\left(\ln\left(\frac{1}{1+\alpha} - x\right) - \ln\left(\frac{1-\frac{1}{n}}{1+\alpha} - x\right)\right). \end{aligned}$$

Let us consider the two terms $(\alpha+1)x\left(\ln(x) - \ln\left(\frac{1}{n(\alpha+1)} + x\right)\right)$ and $(1-(1+\alpha)x)\left(\ln\left(\frac{1}{1+\alpha} - x\right) - \ln\left(\frac{1-\frac{1}{n}}{1+\alpha} - x\right)\right)$.

Let us consider the first term $(\alpha+1)x\left(\ln(x) - \ln\left(\frac{1}{n(\alpha+1)} + x\right)\right)$.

$$\begin{aligned} (\alpha+1)x\left(\ln(x) - \ln\left(\frac{1}{n(\alpha+1)} + x\right)\right) &= (\alpha+1)x\left(\ln(x) - \ln\left(x\left(1 + \frac{1}{nx(1+\alpha)}\right)\right)\right) \\ &= -(\alpha+1)x\ln\left(1 + \frac{1}{nx(1+\alpha)}\right) \end{aligned}$$

Let us consider $g_n(x,\alpha) = x\ln\left(1 + \frac{1}{nx(1+\alpha)}\right)$. Then $\frac{\partial g_n}{\partial x}(x,\alpha) = \ln\left(1 + \frac{1}{nx(1+\alpha)}\right) - \frac{1}{nx(1+\alpha)+1}$ and $\frac{\partial^2 g_n}{\partial^2 x}(x,\alpha) = \frac{-1}{x(nx(1+\alpha)+1)^2} < 0$. Therefore, $\frac{\partial g_n}{\partial x}(x,\alpha) \geq \lim_{x\to+\infty} \frac{\partial g_n}{\partial x}(x,\alpha) = 0$. Thus, $g_n(x,\alpha) \geq g_n(\frac{1}{\alpha n},\alpha) = \frac{1}{\alpha n}\ln(1+\frac{\alpha}{1+\alpha})$ for all x in I_n. Finally,

$$\begin{aligned} (\alpha+1)x\left(\ln(x) - \ln(\frac{1}{n(\alpha+1)} + x)\right) &\leq -(\alpha+1)g_n(x,\alpha) \\ &\leq -\frac{\alpha+1}{\alpha n}\ln(1+\frac{\alpha}{1+\alpha}) = O(\frac{1}{n}). \end{aligned}$$

Let us consider the second term $(1-(1+\alpha)x)\left(\ln\left(\frac{1}{1+\alpha} - x\right) - \ln\left(\frac{1-\frac{1}{n}}{1+\alpha} - x\right)\right)$.

$$\begin{aligned}\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right) &= \ln\left(\frac{1-(1+\alpha)x}{1+\alpha}-\frac{1}{n(1+\alpha)}\right)\\ &= \ln\left(\frac{1-(1+\alpha)x}{1+\alpha}\left(1-\frac{1}{n(1-(1+\alpha)x)}\right)\right)\\ &= \ln\left(\frac{1-(1+\alpha)x}{1+\alpha}\right)+\ln\left(1-\frac{1}{n(1-(1+\alpha)x)}\right).\end{aligned}$$

Hence,

$$(1-(1+\alpha)x)\left(\ln\left(\frac{1}{1+\alpha}-x\right)-\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)\right) = -(1-(1+\alpha)x)\ln\left(1-\frac{1}{n(1-(1+\alpha)x)}\right).$$

Let us consider $h_n(x,\alpha) = (1-(1+\alpha)x)\ln\left(1-\frac{1}{n(1-(1+\alpha)x)}\right)$. Then,

$$\frac{\partial h_n}{\partial x}(x,\alpha) = (1+\alpha)\left(\frac{1}{1+n((1+\alpha)x-1)}-\ln\left(1+\frac{1}{n((1+\alpha)x-1)}\right)\right)$$

and $\frac{\partial^2 h_n}{\partial^2 x}(x,\alpha) = \frac{-(1+\alpha)^2}{(1-(1+\alpha)x)(1+n((1+\alpha)x-1))^2} < 0$ (remember that $x < \frac{1}{1+\alpha}$). Therefore h_n is concave on I_n and $h_n(x,\alpha) \geq \min(h_n(\frac{1}{\alpha n},\alpha), h_n(\frac{n-1}{(1+\alpha)n}-\frac{1}{\alpha n},\alpha))$ and thus $-(1-(1+\alpha)x)\ln(1-\frac{1}{n(1-(1+\alpha)x)}) \leq -\min(h_n(\frac{1}{\alpha n},\alpha), h_n(\frac{n-1}{(1+\alpha)n}-\frac{1}{\alpha n},\alpha))$. Then,

$$\begin{aligned}h_n\left(\frac{1}{\alpha n},\alpha\right) &= -\left(1-\frac{1+\alpha}{\alpha n}\right)\ln\left(1-\frac{1}{n-\frac{1+\alpha}{\alpha}}\right)\\ &= -\left(1-\frac{1+\alpha}{\alpha n}\right)\left(-\frac{1}{n-\frac{1+\alpha}{\alpha}}+o\left(\frac{1}{n}\right)\right)\\ &= \frac{1}{n-\frac{1+\alpha}{\alpha}}+o\left(\frac{1}{n}\right)\\ &= O\left(\frac{1}{n}\right)\end{aligned}$$

and, after noticing that $n(1-(1+\alpha)x) = \frac{2\alpha+1}{\alpha}$ when $x = \frac{n-1}{(1+\alpha)n}-\frac{1}{\alpha n}$,

$$\begin{aligned}h_n\left(\frac{n-1}{(1+\alpha)n}-\frac{1}{\alpha n},\alpha\right) &= -\left(1-\frac{n-1}{n}+\frac{\alpha+1}{\alpha n}\right)\ln\left(1-\frac{\alpha}{2\alpha+1}\right)\\ &= -\left(\frac{1}{n}+\frac{\alpha+1}{\alpha n}\right)\ln\left(\frac{2\alpha+1-\alpha}{2\alpha+1}\right)\\ &= -\frac{2\alpha+1}{\alpha n}\ln\left(\frac{\alpha+1}{2\alpha+1}\right)\\ &= O\left(\frac{1}{n}\right).\end{aligned}$$

Then,

$$(1-(1+\alpha)x)\left(\ln\left(\frac{1}{1+\alpha}-x\right)-\ln\left(\frac{1-\frac{1}{n}}{1+\alpha}-x\right)\right) =\leq -\min\left(h_n\left(\frac{1}{\alpha n},\alpha\right),h_n\left(\frac{n-1}{(1+\alpha)n}-\frac{1}{\alpha n},\alpha\right)\right) \leq O\left(\frac{1}{n}\right).$$

Therefore, if $C_n = \frac{\alpha+1}{\alpha n}\ln\left(1+\frac{\alpha}{1+\alpha}\right) - \min\left(h_n\left(\frac{1}{\alpha n},\alpha\right),h_n\left(\frac{n-1}{(1+\alpha)n}-\frac{1}{\alpha n},\alpha\right)\right)$, then $C_n = O\left(\frac{1}{n}\right)$ and $f_{0,n}(x,\alpha) \leq f_{0,\infty}(x,\alpha) + C_n$, what achieves the proof of Lemma 2. □

Using above two claims, we obtain

$$f_{0,n}(x,\alpha) + \frac{1}{n}f_{1,n}(x,\alpha) \leq f_{0,\infty}(x,\alpha) + C_n + \frac{K}{n}.$$

Let us now bound $f_{0,\infty}(x,\alpha)$ and let us denote $f(x) = x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(x+(1-x)^{\frac{1}{x}}(1+x))$.

Claim 3. *$\forall x$ in I_n, $\forall \alpha \in]0,1[$:*

$$f_{0,\infty}(x,\alpha) \leq f(x)$$

Proof. Let us first consider $f_{0,\infty}(x,\alpha)$ as a function of α. Then, $\frac{\partial f_{0,\infty}}{\partial \alpha}(x,\alpha) = -\ln(1-x) - x\ln(1+x) + x\ln(\frac{1}{1+\alpha}-x)$ and $\frac{\partial^2 f_{0,\infty}}{\partial^2 \alpha}(x,\alpha) = \frac{-x}{(1+\alpha)(1-x(1+\alpha)x)} < 0$ (we recall that $x \leq \frac{1}{1+\alpha}$). Therefore, $\frac{\partial f_{0,\infty}}{\partial \alpha}$ is equal to 0 only once for a certain value of α denoted α_0 and such that $f_{0,\infty}(x,\alpha) \leq f_{0,\infty}(x,\alpha_0)$. Note that

$$\begin{aligned} \frac{\partial f_{0,\infty}}{\partial \alpha}(x,\alpha_0) &= 0 \\ -\ln(1-x) - x\ln(1+x) + x\ln\left(\frac{1}{1+\alpha_0}-x\right) &= 0 \\ \ln\left(\frac{1}{1+\alpha_0}-x\right) &= \frac{\ln(1-x)}{x} + \ln(1+x) \end{aligned}$$

and

$$\begin{aligned} \frac{1}{1+\alpha_0} - x &= (1-x)^{\frac{1}{x}}(1+x) \\ \frac{1}{1+\alpha_0} &= x + (1-x)^{\frac{1}{x}}(1+x) \\ -\ln(1+\alpha_0) &= \ln(x+(1-x)^{\frac{1}{x}}(1+x)). \end{aligned}$$

Therefore,

$$\begin{aligned}
f_{0,\infty}(x,\alpha_0) &= -\ln(1+\alpha_0) + x\ln(x) + (1-x-\alpha_0)\ln(1-x) + (1-(1+\alpha_0)x)\ln(1+x) \\
&\quad - (1-(1+\alpha_0)x)\ln\left(\frac{1}{1+\alpha_0} - x\right) \\
&= -\ln(1+\alpha_0) + x\ln(x) - x\ln(1-x) + (1-\alpha_0)\ln(1-x) \\
&\quad + (1-(1+\alpha_0)x)\left(\ln(1+x) - \frac{\ln(1-x)}{x} - ln(1+x)\right) \\
&= -\ln(1+\alpha_0) + x\ln(x) - x\ln(1-x) + (1-\alpha_0)\ln(1-x) \\
&\quad - (1-(1+\alpha_0)x)\frac{\ln(1-x)}{x} \\
&= -\ln(1+\alpha_0) + x\ln(x) - x\ln(1-x) + (1-\alpha_0)\ln(1-x) \\
&\quad - \frac{\ln(1-x)}{x} + (1+\alpha_0)\ln(1-x) \\
&= -\ln(1+\alpha_0) + x\ln(x) - x\ln(1-x) + 2\ln(1-x) - \frac{\ln(1-x)}{x} \\
&= -\ln(1+\alpha_0) + x\ln(x) + (2-x-\frac{1}{x})\ln(1-x) \\
&= -\ln(1+\alpha_0) + x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) \\
&= x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(x+(1-x)^{\frac{1}{x}}(1+x)) \\
&= f(x)
\end{aligned}$$

what achieves the proof of Lemma 3. □

Together, Claims 1, 2 and 3 prove that

$$E(Y_{q,mq+1}) < \frac{e^{\frac{1}{12p}}e^{\frac{1}{12n}}}{2\pi\sqrt{np}}e^{nf(x)+nC_n+K}$$

for $x \in I_n$.

Thus,

$$\begin{aligned}
\sum_{q=1}^{\frac{n-1}{m}} E(Y_{q,mq+1}) &\leq \sum_{x=q/p\in I_n} \frac{e^{\frac{1}{12p}}e^{\frac{1}{12n}}}{2\pi\sqrt{np}}e^{nf(x)+nC_n+K} \\
&\leq \frac{e^{\frac{1}{12p}}e^{\frac{1}{12n}}}{2\pi n\sqrt{\alpha}}e^{nC_n+K}\sum_{x=q/p\in I_n} e^{nf(x)} \\
&\leq O\left(\frac{1}{n}\right)\sum_{x=q/p\in I_n} e^{nf(x)}.
\end{aligned}$$

what achieves the proof of Lemma 5.9. □

To complete the study of $\sum_{q=1}^{\frac{n-1}{m}} E(Y_{q,mq+1})$, we rely on the following lemma.

Lemma 5.10.

$$\sum_{x=q/p \in I_n} e^{nf(x)} = O(1)$$

Proof. In order to prove claimed result, we rely on two intermediate claims (Claim 4 and 5) that respectively prove that f is a negative function on $]0,1[$ and that f is asymptotically close to $x\ln(x)$ when x is closed of 0.

Claim 4. $\forall x \in]0,1[,\ f(x) < 0$.

Proof. To establish the result, we consider three different cases. The first two ones are obtained by upper bounding $f(x)$ on the left $x \in]0, \frac{1}{5}]$ and right $x \in [\frac{17}{20}, 1[$ parts of the interval. For the middle part, $x \in [\frac{1}{5}, \frac{17}{20}]$, we will rely on interval arithmetic to establish the result.

case (1): $x \in]0, \frac{1}{5}]$

$$\begin{aligned} f(x) &= x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(x + (1-x)^{\frac{1}{x}}(1+x)) \\ &= x\ln(x) - \frac{1}{x}\ln(1-x) + (2-x)\ln(1-x) + \ln(x + (1-x)^{\frac{1}{x}}(1+x)) \\ &= x\ln(x) - \frac{1}{x}\ln(1-x) + \ln\left((1-x)^{2-x}\left(x + (1-x)^{\frac{1}{x}}(1+x)\right)\right). \end{aligned}$$

Since $e^{-x} \leq 1 - x + \frac{x^2}{2}$ and $\ln(1-x) \leq -x - \frac{x^2}{2}$ on $[0,1]$ (a simple study of $x \mapsto 1 - x + \frac{x^2}{2} - e^{-x}$ and $x \mapsto -x - \frac{x^2}{2} - \ln(1-x)$ leads to both results), then

$$(1-x)^{2-x} = e^{(2-x)\ln(1-x)} \leq e^{-(2-x)x} \leq 1 - (2-x)x + \frac{((2-x)x)^2}{2}.$$

Similarly,

$$(1-x)^{\frac{1}{x}} = e^{\frac{1}{x}\ln(1-x)} \leq e^{\frac{1}{x}(-x-\frac{x^2}{2})} \leq e^{-1-\frac{x}{2}} \leq e^{-1}\left(1 - \frac{x}{2} + \frac{x^2}{8}\right)$$

and thus

$$\begin{aligned} x + (1-x)^{\frac{1}{x}}(1+x) &\leq x + e^{-1}\left(1 - \frac{x}{2} + \frac{x^2}{8}\right)(1+x) \\ &\leq x + e^{-1}\left(1 - \frac{x}{2} + \frac{x^2}{8} + x - \frac{x^2}{2} + \frac{x^3}{8}\right) \\ &\leq e^{-1}\left(1 + \left(e + \frac{1}{2}\right)x - \frac{3}{8}x^2 + \frac{x^3}{8}\right). \end{aligned}$$

Therefore,

$$(1-x)^{2-x}\left(x+(1-x)^{\frac{1}{x}}(1+x)\right) \leq e^{-1}\left(1-(2-x)x+\frac{((2-x)x)^2}{2}\right)\left(1+\left(e+\frac{1}{2}\right)x-\frac{3}{8}x^2+\frac{x^3}{8}\right)$$
$$\leq e^{-1}\left(1+\left(e-\frac{3}{2}\right)x-\left(2e-\frac{13}{8}\right)x^2+\left(3e+\frac{3}{8}\right)x^3-\left(2e+\frac{15}{8}\right)x^4 + \left(\frac{e}{2}+\frac{11}{8}\right)x^5-\frac{7}{16}x^6+\frac{1}{16}x^7\right).$$

Note that $-\left(2e+\frac{15}{8}\right)x^4+\left(\frac{e}{2}+\frac{11}{8}\right)x^5 \leq -\left(\frac{5}{8}+\frac{11}{8}\right)x^4 < 0$ and $-\frac{7}{16}x^6+\frac{1}{16}x^7 \leq -\frac{3}{8}x^6 < 0$ on $]0,\frac{1}{5}]$. Thus,

$$(1-x)^{2-x}\left(x+(1-x)^{\frac{1}{x}}(1+x)\right) < e^{-1}\left(1+\left(e-\frac{3}{2}\right)x-\left(2e-\frac{13}{8}\right)x^2+\left(3e+\frac{3}{8}\right)x^3\right)$$

and

$$\ln\left((1-x)^{2-x}\left(x+(1-x)^{\frac{1}{x}}(1+x)\right)\right) < -1+\left(e-\frac{3}{2}\right)x-\left(2e-\frac{13}{8}\right)x^2+\left(3e+\frac{3}{8}\right)x^3.$$

In addition,

$$-\ln(1-x) \leq x+\frac{1}{2}x^2+\frac{1}{2}x^3,$$

since the derivative of $x \mapsto -\ln(1-x)-\left(x+\frac{1}{2}x^2+\frac{1}{2}x^3\right)$ is $\frac{x^2(3x-1)}{2(1-x)} \leq 0$ on $]0,\frac{1}{5}]$ and thus $\ln(1-x)-\left(x+\frac{1}{2}x^2+\frac{1}{2}x^3\right) \leq 0$. Finally,

$$f(x) = x\ln(x)-\frac{1}{x}\ln(1-x)+\ln\left((1-x)^{2-x}\left(x+(1-x)^{\frac{1}{x}}(1+x)\right)\right)$$
$$< x\ln(x)+1+\frac{1}{2}x+\frac{1}{2}x^2-1+\left(e-\frac{3}{2}\right)x-\left(2e-\frac{13}{8}\right)x^2+\left(3e+\frac{3}{8}\right)x^3$$
$$< x\ln(x)+(e-1)x-\left(2e-\frac{17}{8}\right)x^2+\left(3e+\frac{3}{8}\right)x^3 = g(x).$$

$g'(x) = \ln(x)+1+(e-1)-\left(4e-\frac{17}{4}\right)x+\left(9e+\frac{9}{8}\right)x^2$ and $g''(x) = \frac{1}{x}-\left(4e-\frac{17}{4}\right)+\left(18e+\frac{9}{4}\right)x$. Hence, on $]0,\frac{1}{5}]$, g'' has the same sign than $1-\left(4e-\frac{17}{4}\right)x+\left(18e+\frac{9}{4}\right)x^2$ whose discriminant is $\Delta = 16e^2+\frac{253}{4}-98e < 0$. Therefore $g''(x) > 0$ on $]0,\frac{1}{5}]$ and g is convex. In addition, $\lim_{x\to 0} g(x) = 0$ and $g(\frac{1}{5}) < 0$ so that, on $]0,\frac{1}{5}]$, $f(x) < 0$.

case (2): $x \in [\frac{17}{20}, 1[$

$$f(x) = x\ln(x)-\frac{(1-x)^2}{x}\ln(1-x)+\ln(x+(1-x)^{\frac{1}{x}}(1+x))$$
$$= (x+1)\ln(x)-\frac{(1-x)^2}{x}\ln(1-x)+\ln\left(1+(1-x)^{\frac{1}{x}}\left(1+\frac{1}{x}\right)\right)$$
$$< 2(x-1)-\frac{(1-x)^2}{x}\ln(1-x)+(1-x)^{\frac{1}{x}}\left(1+\frac{1}{x}\right).$$

Indeed, for $y > 0$, $ln(1+y) < y$ (and $(1-x)^{\frac{1}{x}}\left(1+\frac{1}{x}\right) > 0$ if $x \neq 1$). Similarly $(x+1)\ln(x) \leq 2(x-1)$ (the derivative of the function $x \mapsto \ln(x) - \frac{2(x-1)}{x+1}$ is $\frac{(x-1)^2}{x(x+1)^2} \geq 0$ and therefore $\ln(x) - \frac{2(x-1)}{x+1} \geq 0$). Thus,

$$\begin{aligned}
&2(x-1) - \frac{(1-x)^2}{x}\ln(1-x) + (1-x)^{\frac{1}{x}}\left(1+\frac{1}{x}\right) \leq 0 \\
\iff &-2 - \frac{(1-x)}{x}\ln(1-x) + (1-x)^{\frac{1}{x}-1}\left(1+\frac{1}{x}\right) \leq 0 \\
\iff &-2x - (1-x)\ln(1-x) + (1-x)^{\frac{1}{x}-1}(1+x) \leq 0
\end{aligned}$$

Let us now prove that $(1-x)^{\frac{1}{x}-1} \leq 1 + \frac{3}{4}(1-x)\ln(1-x)$. First, $e^y \leq 1 + \frac{3}{4}y$ for $y \in [-e^{-1}, 0]$. Indeed, the derivative of the function $g(y) = e^y - 1 - \frac{3}{4}y$ is $g'(y) = e^y - \frac{3}{4}$ and $g''(y) = e^y > 0$. Thus $g(y) \leq \max(g(0), g(e^{-1}) \leq 0$. Let us use $y = (\frac{1}{x} - 1)\ln(1-x)$ in above result. For that, observe that for $x \in [\frac{17}{20}, 1[$, $e^{-1} \leq (\frac{1}{x} - 1)\ln(1-x) \leq 0$ using the function $h(x) = (\frac{1}{x} - 1)\ln(1-x)$. Indeed, $h'(x) = \frac{-1}{x^2}(x + \ln(1-x)) \leq \frac{-1}{x^2}(x-x) \leq 0$ on $[\frac{17}{20}, 1[$. Thus h is increasing and $e^{-1} \leq h(\frac{17}{20}) \leq h(x) \leq h(1) \leq 0$. Finally, the result holds true in the interval $(1-x)^{\frac{1}{x}-1} \leq 1 + \frac{3}{4}(1-x)\ln(1-x)$.

Therefore,

$$-2x - (1-x)\ln(1-x) + (1-x)^{\frac{1}{x}-1}(1+x) \leq -2x - (1-x)\ln(1-x) + (1+x)\left(1 + \frac{3}{4}(1-x)\ln(1-x)\right)$$

and

$$\begin{aligned}
&-2x - (1-x)\ln(1-x) + (1+x)\left(1 + \frac{3}{4}(1-x)\ln(1-x)\right) \leq 0 \\
\iff &(-2x + x + 1) + (1-x)\ln(1-x)\left(-1 + \frac{3(x+1)}{4x}\right) \leq 0 \\
\iff &(1-x) + (1-x)\ln(1-x)\frac{3-x}{4x} \leq 0 \\
\iff &1 + \frac{3-x}{4x}\ln(1-x) \leq 0 \\
\iff &\frac{4x}{3-x} + \ln(1-x) \leq 0.
\end{aligned}$$

In order to conclude, consider $g(x) = \frac{4x}{3-x} + \ln(1-x)$. Then, $g'(x) = \frac{-x^2-6x+3}{(3-x)^2(1-x)}$ and the roots of $-x^2 - 6x + 3$ are $-3 - 2\sqrt{3}$ and $-3 + 2\sqrt{3}$, both smaller than $\frac{17}{20}$. Therefore $g'(x) \leq 0$ on $[\frac{17}{20}, 1[$ and thus $g(x) \leq g(\frac{17}{20}) \leq 0$. Finally $\frac{4x}{3-x} + \ln(1-x) \leq 0$ and $f(x) < 0$ on $[\frac{17}{20}, 1[$.

case (3): $x \in [\frac{1}{5}, \frac{17}{20}]$

Remember that

$$\begin{aligned}
f(x) &= x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(x + (1-x)^{\frac{1}{x}}(1+x)) \\
&= (x+1)\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(1 + (1-x)^{\frac{1}{x}}(1+\frac{1}{x})).
\end{aligned}$$

Note that $x \mapsto (x+1)\ln(x)$ and $x \mapsto -\frac{(1-x)^2}{x}\ln(1-x)$ are increasing functions on $]0,1[$, and $x \mapsto \ln(1+(1-x)^{\frac{1}{x}}(1+\frac{1}{x}))$ is a decreasing function on $]0,1[$. These properties are not direct but a double derivation and a the evaluation of the derivative on the extremal points of the interval prove the result. Therefore, if $x \in [x_1, x_2]$:

$$f(x) \leq (x_2+1)\ln(x_2) - \frac{(1-x_2)^2}{x_2}\ln(1-x_2) + \ln(1+(1-x_1)^{\frac{1}{x_1}}(1+\frac{1}{x_1})).$$

To finish the proof we apply this bound on small closed intervals of $[\frac{1}{5}, \frac{17}{20}]$ and prove that this value is negative. To establish this result, we use $[x_1, x_2]$ with $x_2 - x_1 = \frac{1}{100}$. In order to have a reliable evaluation of this bound we use the MPFI (version 1.5.1) library for C [24] that offers a C implementation of interval arithmetic and thus an upper bound of the actual value of $(x_2+1)\ln(x_2) - \frac{(1-x_2)^2}{x_2}\ln(1-x_2) + \ln(1+(1-x_1)^{\frac{1}{x_1}}(1+\frac{1}{x_1}))$, using the code given below.

```
#include "mpfi.h"
#include "mpfi_io.h"
#include "assert.h"

                                    void f2(mpfi_t res,mpfi_t x){
                                     //computation of -(1-x)^2/x ln(1-x)
   void f1(mpfi_t res,mpfi_t x){      mpfi_t y,z ;
    //computation of (x+1)*ln(x)      mpfi_init(y) ; mpfi_init(z) ;
     mpfi_t y,z ;                     mpfi_si_sub(y,1,x) ;
     mpfi_init(y) ; mpfi_init(z) ;    mpfi_mul(y,y,y) ;
     mpfi_add_si(y,x,1) ;             mpfi_div(y,y,x) ;
     mpfi_log(z,x) ;                  mpfi_si_sub(z,1,x) ;
     mpfi_mul(res,y,z) ;              mpfi_log(z,z) ;
     mpfi_clear(y) ; mpfi_clear(z) ;  mpfi_si_sub(z,0,z) ;
   }                                  mpfi_mul(res,y,z) ;
                                      mpfi_clear(y) ; mpfi_clear(z) ;
                                    }

void f3(mpfi_t res,mpfi_t x){ //computation of ln(1+(1-x)^(1/x)(1+1/x))
  mpfi_t y,z ;
  mpfi_init(y) ; mpfi_init(z) ;
  mpfi_si_sub(y,1,x) ;
  mpfi_log(y,y) ;
  mpfi_div(y,y,x) ;
  mpfi_exp(y,y) ;
  mpfi_si_div(z,1,x) ;
  mpfi_add_si(z,z,1) ;
  mpfi_mul(y,y,z) ;
  mpfi_add_si(y,y,1) ;
  mpfi_log(res,y) ;
```

```
  mpfi_clear(y) ; mpfi_clear(z) ;
}


int main(){
  mpfi_t x ;
  mpfi_t res,res1,res2,res3 ;
  //initialisation of the intervals
  mpfi_init(x) ; mpfi_init(res) ; mpfi_init(res1) ; mpfi_init(res2) ; mpfi_init(res3) ;
  double x1,x2 ;
  double gap = 0.01 ; // gap between x1 and x2
  for(x1 = 0.2;x1 < 0.85;x1 += gap){
    x2 = x1+gap ;
    mpfi_set_d(x,x2) ;
    f1(res1,x) ; //computation of (x2+1)*ln(x2)
    f2(res2,x) ; //computation of -(1-x2)^2/x2 ln(1-x2)
    mpfi_set_d(x,x1) ;
    f3(res3,x) ; //computation of ln(1+(1-x1)^(1/x1)(1+1/x1))
    //computation of the interval that represents the upper bound of f(x) for x in [x1,x2]
    mpfi_add(res,res1,res2) ;
    mpfi_add(res,res,res3) ;
    //stop the program if the right bound of the interval representing the upper bound is no
    assert(mpfi_is_strictly_neg(res)==1) ;
  }
  //clear memory
  mpfi_clear(x) ; mpfi_clear(res) ; mpfi_clear(res1) ; mpfi_clear(res3) ; mpfi_clear(res2) ;
  return 0;
}
```

Combining the results of all 3 cases, we achieve the proof of Claim 4. □

Claim 5. *For all* $\epsilon \in]0, 1-\alpha[$*, there exists* x_0 *such that* $x \leq x_0$ *implies* $f(x) \leq (1-\epsilon)x\ln(x)$.

Proof. Remember that $f(x) = x\ln(x) - \frac{(1-x)^2}{x}\ln(1-x) + \ln(x + (1-x)^{\frac{1}{x}}(1+x))$.
First,

$$\begin{aligned} -\frac{(1-x)^2}{x}\ln(1-x) &= -\frac{(1-x)^2}{x}(-x + O(x^2)) \\ &= (1-x)^2(1+O(x)) \\ &= 1 + O(x). \end{aligned}$$

and

$$\begin{aligned}
\ln(x+(1-x)^{\frac{1}{x}}(1+x)) &= \ln\left(x+e^{\frac{\ln(1-x)}{x}}(1+x)\right)\\
&= \ln\left(x+e^{-1+O(x)}(1+x)\right)\\
&= \ln\left(x+e^{-1}(1+O(x))(1+x)\right)\\
&= \ln\left(x+e^{-1}(1+O(x)\right)\\
&= \ln\left(e^{-1}(ex+1+O(x))\right)\\
&= -1+\ln(1+O(x))\\
&= -1+O(x).
\end{aligned}$$

Finally, $f(x) = x\ln(x)+1+O(x)-1+O(x) = x\ln(x)+O(x)$.

Let ϵ be in $]0,1-\alpha[$. As $f(x)\sim x\ln(x)$ we know that there exists x_0 such that $x \leq x_0$ implies (we recall that $x\ln(x)<0$):

$$\begin{aligned}
\left|1-\frac{f(x)}{x\ln(x)}\right| &\leq \epsilon\\
1-\frac{f(x)}{x\ln(x)} &\leq \epsilon\\
\frac{f(x)}{x\ln(x)} &\geq 1-\epsilon\\
f(x) &\leq (1-\epsilon)x\ln x
\end{aligned}$$

that ends the proof of Claim 5. □

Without loss of generality we can assume that the x_0 from Claim 5 is smaller than e^{-1}. As $x \mapsto x\ln(x)$ is decreasing on $]0,e^{-1}]$, $\forall x \in \left[\frac{1}{\alpha n}, x_0\right]$, $f(x) \leq -(1-\epsilon)\frac{1}{\alpha n}\ln(\alpha n)$. In addition, f is continuous on $\left[x_0, \frac{1}{1+\alpha}\right]$ and thus,there exists $x_1 \in \left[x_0, \frac{1}{1+\alpha}\right]$ such that $f(x) \leq f(x_1)$. Thanks to Lemma 4, $f(x_1) < 0$ and $\exists K' < 0$ such that $\forall x \in \left[x_0, \frac{1}{1+\alpha}\right]$, $f(x) \leq K'$. Thus,

$$\begin{aligned}
\sum_{x=q/p\in I_n} e^{nf(x)} &= \sum_{x=q/p\in\left[\frac{1}{\alpha n},x_0\right]} e^{nf(x)} + \sum_{x=q/p\in\left[x_0,\frac{1}{1+\alpha}\right]} e^{nf(x)}\\
&= \sum_{x=q/p\in\left[\frac{1}{\alpha n},x_0\right]} e^{-\frac{1-\epsilon}{\alpha}ln(\alpha n)} + \sum_{x=q/p\in\left[x_0,\frac{1}{1+\alpha}\right]} e^{nK'}\\
&= \frac{1}{(\alpha n)^{\frac{1-\epsilon}{\alpha}}}\left(\sum_{x=q/p\in\left[\frac{1}{\alpha n},x_0\right]} 1\right) + e^{nK'}\left(\sum_{x=q/p\in\left[x_0,\frac{1}{1+\alpha}\right]} 1\right)\\
&= \frac{1}{(\alpha n)^{\frac{1-\epsilon}{\alpha}}}\times n + e^{nK'}\times n
\end{aligned}$$

$$= O(n^{\frac{\alpha-1+\epsilon}{\alpha}}) + O(ne^{nK'})$$
$$= O(1)$$

because $\alpha - 1 + \epsilon \leq 0$, what ends the proof of Lemma □

With Lemmas 5.7, 5.9 and 5.10 we obtain

$$\begin{aligned} Pr(Z_m \geq 1) &\leq 2\sum_{q=1}^{\frac{n-1}{m}} E\left(Y_{q,mq+1}\right) \\ &\leq 2\sum_{q=1}^{\frac{n-1}{m}-1} E\left(Y_{q,mq+1}\right) + 2E\left(Y_{\frac{n-1}{m},n}\right) \\ &\leq O\left(\frac{1}{n}\right) \sum_{x=q/p\in I_n} e^{nf(x)} + O\left(\frac{1}{n}\right) \\ &\leq O\left(\frac{1}{n}\right) \end{aligned}$$

what achieves the proof of Theorem 5.5, and therefore the proof of Theorem 5.3. □

Theorem 5.3 proves that there exists a quasi-perfect assignment if $r = 2$ and n is a multiple of p ($p = \alpha n$ where $1/\alpha \in \mathbb{N}^*$) with high probability when n becomes large. In fact, the assumptions stating that n is a multiple of p and $p \leq n$ can be removed, as stated in Theorem 5.11.

Theorem 5.11. *Let $G = (P, T, E)$ a bipartite graph such that for every $t_i \in T$, the degree of t_i is at least r. Let $n = |T|$, $p = |P| = \alpha n$, and $\alpha \in]0, 1]$. Then, as soon as $r \geq 2$,*

$$P(\exists \text{ a quasi-perfect assignment}) = 1 - O\left(\frac{1}{n}\right).$$

Proof. First, we can consider the case $r = 2$ only. Indeed, if $r > 2$, extra edges can be removed in order to obtain a subgraph where $r = 2$ and if there exists an assignment of maximum degree m for this graph, this a fortiori holds true for the original one.

Furthermore, if p divide n we can use Theorem 5.5 to bound the probability that there is no such assignment. Otherwise, let k be an integer such that $k < \frac{n}{p} < k + 1$. Let $n' = (k + 1)p > n$ and let us remark that $m = \lceil \frac{n}{p} \rceil + 1 = k + 2 = \lceil \frac{n'}{p} \rceil + 1$. Let $G' = (P, T', E')$ with $T \subset T'$, $E \subset E'$ and $|T'| = n'$. Then, if there exists an assignment of maximum degree m in G', there exists one with equal or lower maximum degree in G. Therefore, the probability of existence of such an assignment in G is larger than the one of in G' and can be bounded using Theorem 5.5. This achieves the proof of Theorem 5.11. □

6 Simulation Results

In previous sections, we have studied the theoretical behavior of the following two assignment strategies

- Greedy: A greedy assignment strategy where idle processors execute some random unprocessed task for which they own the corresponding data chunk. We prove in Section 4.1 that the expected makespan of this strategy with homogeneous processing times is $\left\lceil \frac{n}{p} \right\rceil + O(\log \log n)$.
- BestAssignment: A strategy using a matching-based approach, presented in Section 4.3. We prove in Theorem 5.11 that it is quasi-perfect, *i.e.* that its makespan is at most $\lceil n/p \rceil + 1$ with high probability when n is large. Remember that $\lceil n/p \rceil$ is a trivial lower bound on the achievable makespan.

We present simulations to illustrate additional properties that are harder to reach through theoretical analysis. More specifically, we first study the case for small values of n, as above theoretical results are established when n is arbitrarily large. Then, we get back to the case when chunks are communicated from heavy loaded nodes to idle nodes at the end of the process to balance the load and experimentally assess the performance of both strategies.

6.1 Behavior for Small Number of Processors and Tasks

In Section 5.2 we study the asymptotic behavior of the BestAssignment strategy. We complete these results with simulations that include small number of processors (from 2 to 100). For each value of p in this interval and for different n/p ratios, we perform 250 tests and report the average overhead (the difference between the obtained makespan and the perfect makespan n/p) on Figure 4. For this set of experiments, we set $r = 2$, the smallest value for which Theorem 5.11 holds true. Note that an increase of r decreases the average overhead as expressed in Theorem 5.11, but does not impact the general behavior.

We first notice that in all cases, Greedy quickly reaches an average overhead larger than 1, usually as soon as p is equal to 5. Its average makespan increases with p (and thus with n, as n/p is constant for each curve). The n/p ratio has little influence. This is consistent with the expected overhead $O(\ln \ln n)$.

The overhead of the BestAssignment strategy mainly depends on the n/p ratio, and is bounded above by 1. For $n/p = 50$, the average overhead is always 0, meaning the assignment found by BestAssignment is always perfect. Unlike Greedy, BestAssignment benefits from the increase of n for a given n/p ratio.

Note that on this set of simulations (more than 10^5 runs in total), we notice a single overhead larger than 1 for BestAssignment, what shows that the near-perfect makespan behavior outlined by Theorem 5.11 also extends in practice to small values of n.

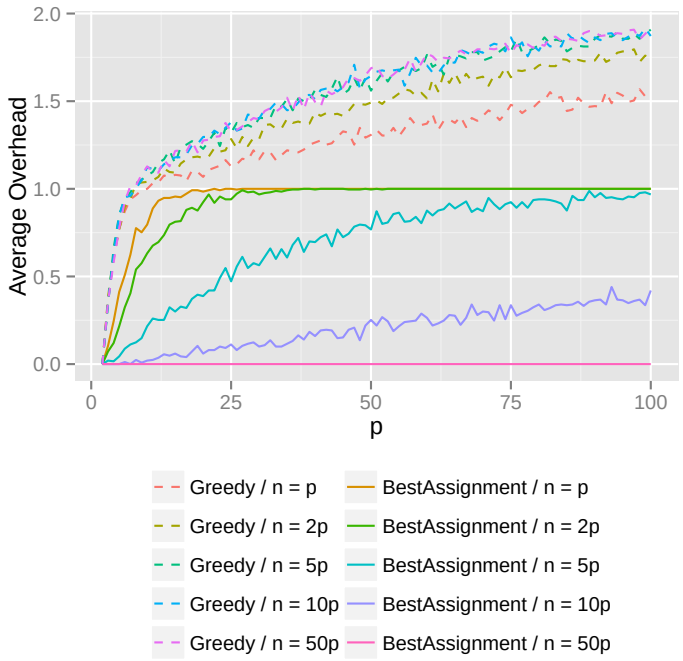


Figure 4: Average Overhead (Makespan - n/p) for small values of p and $r = 2$.

6.2 Load Balancing and Communications

We move to the study of non-local schedulers. Indeed, in some cases when perfect makespan n/p must be achieved, communicating data chunks through the network at runtime is required. The assignments computed by both previous algorithms may easily be transformed to take into account this additional constraint. The original assignment is used as long as they exist local tasks to schedule. When a processor is idle and has no more local tasks while there are still unprocessed tasks in the system, a random non-local task is assigned to the idle processor, and the corresponding data chunk is sent. We now evaluate the amount of communicated chunks *i.e.*, the number of non-local tasks, for both algorithms.

To evaluate these communications we consider processors whose load is strictly larger than perfect makespan. To achieve perfect makespan, tasks need to be moved from under-loaded processors. Therefore the metric we propose is the *total load imbalance*, *i.e.* $\sum_{p_i \in P} \max(0, |\{t_j \text{ assigned to } p_i\}| - \lceil n/p \rceil)$. This represents a lower bound on communications, as each processor with a load larger than n/p has to send extra tasks.

In Section 4.1, we relate the Greedy scheduler with initial data replication

to the balls into bins process with multiple choice. In this context, Berenbrink et al. [3] introduce the number of "holes" in an assignment, which corresponds to $\sum_{p_i \in P} \max(0, \lceil n/p \rceil - |\{t_j \text{ assigned to } p_i\}|)$. For integers n/p ratios, this is equal to the total load imbalance introduced above. In [3], it is proved that in the case of balls into bins process with multiple choices, above metric is bounded by a linear function of p.

Note that a perfect assignment, *i.e.* an assignment with makespan $\lceil n/p \rceil$, has a total load imbalance of 0, while a near-perfect assignment (with makespan $\lceil n/p \rceil + 1$) has a total load imbalance bounded by $p-1$ (at most $p-1$ processors have an extra task).

In the following set of experiments, r takes values from 1 to 5, while n/p lies in the interval $[1, 50]$. We set p to 50 but results are similar for other values of p. For each couple (r, p), we run 250 simulations for each algorithm, for each r from 1 to 5 and for different ratio n/p. Results are plotted on Figure 5.

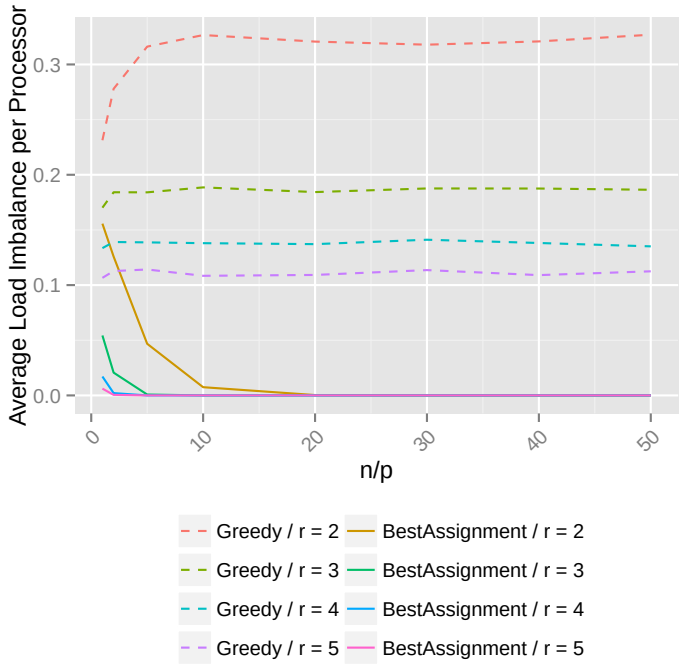


Figure 5: Average load imbalance per processor for $p = 50$, for different values of n/p, for different values of r and for the algorithms Greedy and BestAssignment.

We normalize the total load imbalance by the number of processors, so as to compare to the bound by Berenbrink et al. [3] mentioned above. Thus, Figure 5 shows a lower bound on the number of data chunks that each processor has

to send to other processors in order to achieve perfect makespan. One of the advantages of this presentation is that we observe that n/p ratio has very limited impact on the total load imbalance for GREEDY on n/p, and its total load imbalance per processors is very stable for a fixed r, what experimentally corroborates the result of Berenbrink et al. We also notice that BESTASSIGNMENT achieves very good results: for $r \geq 3$, the average load imbalance per processor is below 0.05 and is quickly decreases to 0 when n/p increases. For $r = 2$, the decrease is faster for larger values of r but the load imbalance per processor for $n = p$ is close to 0.15, which is much better than the results of GREEDY for similar values of r. Finally, BESTASSIGNMENT with only 2 replicas of each data chunk is in general more efficient than GREEDY with 5 replications, as it uses data replication and large number of tasks more efficiently. Finally, note that if there is a clear improvement from 2 to 3 replicas, having a larger number of replicas does not provide a significantly better load balance of the task for BESTASSIGNMENT. Therefore it is not worth paying the large additional storage cost of increasing r, as far as communications are concerned.

Furthermore, the good results of BESTASSIGNMENT on load imbalance can be theoretically validated. Indeed given a initial distribution of the data chunks, an assignment with no augmenting path has a minimum total load imbalance (in addition of the optimal maximum degree), as shown in Theorem 6.1 (the formal definition with graph notation of total imbalance is $\mathrm{Imb}(A) = \sum_{\substack{p_i \in P \\ d_A(p_i) > \left\lceil \frac{|T|}{|P|} \right\rceil}} d_A(p_i) - \left\lceil \frac{|T|}{|P|} \right\rceil$). Note that this property is satisfied by assignments produced by FINDASSIGNMENT and BESTASSIGNMENT but not by assignments produced by NAIVEASSIGNMENT.

Theorem 6.1. *Let $G = (P, T, E)$ be a bipartite graph, let A be an assignment of G. If there is no augmenting path according to A then A has a minimum total load imbalance.*

Lemma 6.2. *Let $G = (P, T, E)$ be a bipartite graph and A be an assignment of G. Let $x = (p_d, \ldots, p_f)$ be an alternating path according to A. Then,*

- *if $d_A(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_A(p_f)$ hence $Imb(A \otimes x) = Imb(A) - 1$,*
- *if $d_A(p_d) > \left\lceil \frac{|T|}{|P|} \right\rceil > d_A(p_f)$ hence $Imb(A \otimes x) = Imb(A) + 1$,*
- *otherwise $Imb(A \otimes x) = Imb(A)$.*

Lemma 6.2 can be proved using Lemma 4.8. Note that an alternating path $x = (p_d, \ldots, p_f)$ such that $d_A(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_A(p_f)$ is an augmenting path.

Let us now prove Theorem 6.1. Let $G = (P, T, E)$ be a bipartite graph and let A be an assignment of G. Let us suppose that A has not an optimal total load imbalance. In this case, there exists an assignment A' such that $\mathrm{Imb}(A) > \mathrm{Imb}(A')$. Thanks to Lemma 4.13, we can build two finite sequences $(A_k)_{k \leq l}$ and $(x_k)_{k \leq l-1}$ such that

- $A_0 = A$,
- $A_l = A'$,
- $\forall p_i \in P,\ d_{A_l}(p_i) = d_{A'}(p_i)$,
- $\forall k < m$, x_k is alternating and $A_{k+1} = A_k \otimes x_k$.

In particular $\mathrm{Imb}(A_0) > \mathrm{Imb}(A_l)$. Therefore, there exists k_0 such that $\mathrm{Imb}(A_{k_0}) > \mathrm{Imb}(A_{k_0+1}) = \mathrm{Imb}(A_{k_0} \otimes x_{k_0})$. Let $x_{k_0} = (p_{d_{k_0}}, \ldots, p_{f_{k_0}})$, then, thanks to Lemma 6.2, $d_{A_{k_0}}(p_{d_{k_0}}) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_{A_{k_0}}(p_{f_{k_0}})$. Moreover, by construction of the sequences, $d_{A_{k_0}}(p_{f_{k_0}}) > d_{A'}(p_{f_{k_0}})$.

Thus, we have proven that there exists k_0 such that $x = (p_d, \ldots, p_f)$ is an alternating path according to A_{k_0} and such that $d_{A_{k_0}}(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_{A_{k_0}}(p_f)$ and $d_{A_{k_0}}(p_f) > d_{A'}(p_f)$. Let k_0 be the smallest k such that there exists such a path according to A_k (not necessarily x_k). In order to prove that $k_0 = 0$, let us assume by contradiction that $k_0 > 0$.

Let $x = (p_d, \ldots, p_f)$ be an alternating path according to A_{k_0} such that $d_A(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_A(p_f)$ and $d_{A_{k_0}}(p_f) > d_{A'}(p_f)$.

Let us suppose that x and x_{k_0-1} are disjoint. In this case, x is a valid alternating path according to A_{k_0-1}, $d_{A_{k_0-1}}(p_d) = d_{A_{k_0}}(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil$ and $d_{A_{k_0-1}}(p_f) = d_{A_{k_0}}(p_f) > \left\lceil \frac{|T|}{|P|} \right\rceil$. Thus, we reach a contradiction with the definition of k_0.

Let us assume that x and x_{k_0-1} are not disjoint. In this case, let us define v_i and v_j such that

- v_i is the first vertex in x to be in x_{k_0-1},
- v_j is the last vertex in x to be in x_{k_0-1}.

If v_i is before v_j in x_{k_0-1}, then $(p_d, \ldots, v_i, \ldots, v_j, \ldots p_f)$ is a valid alternating path according to A_{k_0-1}. Furthermore, if $v_i \neq p_d$, p_d is not in x_{k_0-1} and $d_{A_{k_0-1}}(p_d) = d_{A_{k_0}}(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil$. Otherwise, $d_{A_{k_0-1}}(p_d) = d_{A_{k_0}}(p_d) - 1 < \left\lceil \frac{|T|}{|P|} \right\rceil$. Thus, in all cases $d_{A_{k_0-1}}(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil$ and similarly, we prove that $\left\lceil \frac{|T|}{|P|} \right\rceil < d_{A_{k_0-1}}(p_f)$ and reach a contradiction with the definition of k_0.

Therefore, v_j is before v_i in x_{k_0-1}. Let us consider the path $y = (p_d, \ldots, v_i, \ldots, p_{f_{k_0-1}})$, that is a valid alternating path according to A_{k_0-1}. As previously, $d_{A_{k_0-1}}(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil$. In addition, $d_{A'}(p_f) < d_{A_{k_0}}(p_f) \leq d_{A_{k_0-1}}(p_f)$ by assumption and because $(|d_{A_k}(p_i) - d_{A'}(p_i)|)_{k \leq l}$ is decreasing. Thus, by construction of the p_{f_k}'s, $d_{A_{k_0-1}}(p_f) \leq d_{A_{k_0-1}}(p_{f_{k_0-1}})$. Therefore, $\left\lceil \frac{|T|}{|P|} \right\rceil < d_{A_{k_0-1}}(p_{f_{k_0-1}})$ and y is an alternating path with all desired properties. Therefore, we reach a contradiction with the definition of k_0.

Hence, we prove by contradiction that $k_0 = 0$ and thus that there exists an alternating path $x = (p_d, \ldots, p_f)$ according to A such that $d_A(p_d) <$

$\left\lceil \frac{|T|}{|P|} \right\rceil < d_A(p_f)$. Therefore, there exists an augmenting path according to A, what achieves the proof of Theorem 6.1 by contraposition.

□

7 Conclusion

In this paper, we consider the problem of data locality for map tasks in the MapReduce scheduler, and more generally for all problems of scheduling tasks with replicated input files. To do this, we do not modify the replication mechanism provided by distributed filesystems, but rather propose a new scheduling strategy, named BestAssignment that takes advantage of data locality in order to reduce both the completion time and the amount of communications needed at runtime. We focus on the minimization of the total completion time (makepan) under the constraint that only local tasks are allowed, thus forbidding communication of data chunks. We prove that the dynamic scheduler of MapReduce has an overhead of $O(\log \log n)$ compared to the lower bound using analogies with multiple-choice balls-into-bins problems. Then, we prove that BestAssignment, based on bipartite matchings, achieves a quasi-perfect assignment (*i.e.* optimal up to an additional increase of 1 step) with high probability when the number of tasks is arbitrarily large. In addition, we provide simulation results to show the superiority of BestAssignment even in the non-asymptotic case. We also show through simulations that in the case where communications are imposed to achieve perfect load-balancing, BestAssignment reduces significantly their amount compared to default Greedy MapReduce scheduler.

Perspectives of this work include adapting the analysis for non-homogeneous task running times, that could be done both theoretically and through simulations, as well as the theoretical analysis of the communication volume of both algorithms in the case when perfect load balancing is imposed.

References

[1] O. Beaumont, T. Lambert, L. Marchal, and B. Thomas. Matching-Based Assignment Strategies for Improving Data Locality of Map Tasks in MapReduce. available online `https://hal.inria.fr/hal-01386539/`, 2016.

[2] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking. Balanced allocations: the heavily loaded case. In *Proceedings of the 22nd annual ACM symposium on Theory of computing*, pages 745–754. ACM, 2000.

[3] P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin. On weighted balls-into-bins games. *Theoretical Computer Science*, 409(3):511–520, 2008.

[4] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957.

[5] D. Borthakur. Hdfs architecture guide. *HADOOP http://hadoop. apache. org/common/docs/current/hdfs design. pdf*, page 39, 2008.

[6] R. E. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems, Revised Reprint*. Siam, 2009.

[7] M. Chowdhury and I. Stoica. Coflow: a networking abstraction for cluster applications. In *11th ACM Workshop on Hot Topics in Networks, HotNets-XI, Redmond, WA, USA - October 29 - 30, 2012*, pages 31–36, 2012.

[8] M. Chowdhury and I. Stoica. Efficient coflow scheduling without prior knowledge. *Computer Communication Review*, 45(5):393–406, 2015.

[9] F. Dufossé, K. Kaya, and B. Uçar. Two approximation algorithms for bipartite matching on multicore architectures. *Journal of Parallel and Distributed Computing*, 85:62 – 78, 2015.

[10] P. Erdös and A. Rényi. On random matrices. *Studia Sci. Math. Hungar.*, 8:455–461, 1964.

[11] L. R. Ford Jr and D. R. Fulkerson. *Flows in networks*. Princeton university press, 2015.

[12] A. Goel, M. Kapralov, and S. Khanna. Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. *SIAM Journal on Computing*, 42(3):1392–1404, 2013.

[13] Z. Guo, G. C. Fox, and M. Zhou. Investigation of data locality in mapreduce. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 419–426, 2012.

[14] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935.

[15] M. Hammoud and M. F. Sakr. Locality-aware reduce task scheduling for mapreduce. In *IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, pages 570–576, 2011.

[16] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[17] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu. Maestro: Replica-aware map scheduling for mapreduce. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 435–442, 2012.

[18] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan. An analysis of traces from a production mapreduce cluster. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 94–103, 2010.

[19] J. Langguth, F. Manne, and P. Sanders. Heuristic initialization for bipartite matching problems. *J. Exp. Algorithmics*, 15:1.3:1.1–1.3:1.22, Mar. 2010.

[20] M. Mitzenmacher. The power of two choices in randomized load balancing. *Parallel and Distributed Systems, IEEE Transactions on*, 12(10):1094–1104, 2001.

[21] Y. Peres, K. Talwar, and U. Wieder. The (1+ β)-choice process and weighted balls-into-bins. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1613–1619. Society for Industrial and Applied Mathematics, 2010.

[22] Z. Qiu, C. Stein, and Y. Zhong. Minimizing the total weighted completion time of coflows in datacenter networks. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '15, pages 294–303, New York, NY, USA, 2015. ACM.

[23] M. Raab and A. Steger. Balls into bins: A simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer, 1998.

[24] N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the mpfi library. *Reliable computing*, 11(4):275–290, 2005.

[25] A. W. Richa, M. Mitzenmacher, and R. Sitaraman. The power of two random choices: A survey of techniques and results. *Combinatorial Optimization*, 9:255–304, 2001.

[26] J. Tan, S. Meng, X. Meng, and L. Zhang. Improving reducetask data locality for sequential mapreduce jobs. In *Proceedings of the IEEE INFOCOM*, pages 1627–1635, 2013.

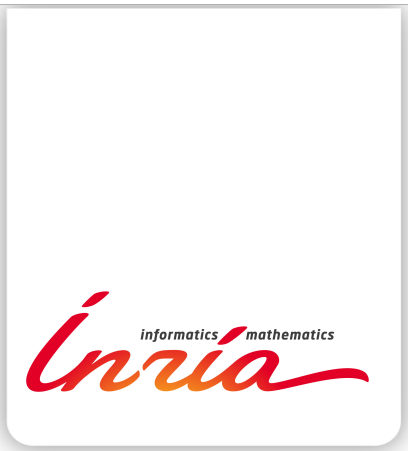
[27] D. W. Walkup. Matchings in random regular bipartite digraphs. *Discrete Mathematics*, 31(1):59–64, 1980.

[28] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang. Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality. In *Proceedings of the IEEE INFOCOM*, pages 1609–1617, 2013.

[29] T. White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[30] Q. Xie and Y. Lu. Degree-guided map-reduce task assignment with data locality constraint. In *Proceedings of the 2012 IEEE International Symposium on Information Theory (ISIT)*, pages 985–989, 2012.

[31] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems (EuroSys)*, pages 265–278. ACM, 2010.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399