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

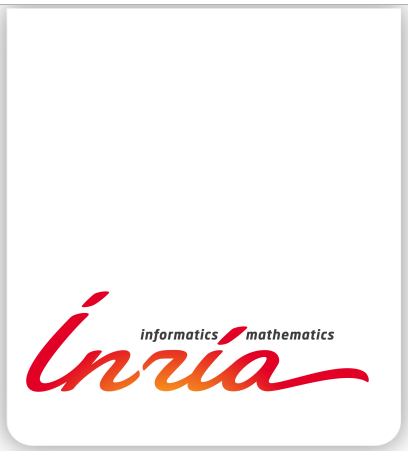
RESEARCH REPORT

N° 8968

October 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 — October 2017 — 59 pages

Abstract: We consider scheduling problems that arise in the context of the Map phase of MapReduce, when input data chunks are replicated over the platform using a distributed file system like HDFS. Replication is used by standard Hadoop schedulers in order to improve locality, by favoring local tasks at runtime. In this paper, we both provide a rigorous analysis on these dynamic runtime strategies using balls-into-bins with power of 2 choices and graph orientation problems, and we propose optimal offline algorithms based of matchings in bipartite graphs. Then, we compare online and offline strategies through a set of simulations based on a trace containing the record of jobs during 10 months on a Hadoop cluster, in order to assess the advantages and drawbacks of static and dynamic scheduling strategies in this context.

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

* Inria & University of Bordeaux, France
† University of Manchester, United Kingdom
‡ 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 relions ce problème à deux problèmes classiques de la littérature, l'orientation de graphe et "balls-into-bins", afin d'obtenir une évaluation probabiliste des performances attendues lorsque l'on interdit les tâches non-locales et nous proposons aussi deux algorithmes optimaux dans le cas où l'on cherche cette fois à optimiser le temps de calcul en limitant les communications nécessaires. Ces résultats théoriques sont appuyés par des simulations basées sur des traces provenant d'un cluster Hadoop. Nous montrons ainsi que si la distribution de la durée des tâches est de variance faible ou moyenne nous obtenons une diminution notable du pourcentage de tâches non-locales.

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

1 Introduction

MapReduce is a well-known framework for distributing data-processing computations onto parallel clusters that has been introduced by Google and has been popularized by implementations like Hadoop [1]. In MapReduce, a large computation is broken into small tasks that run concurrently 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. Let us consider a textbook example: letter count, computing the number of occurrences of each letter in a word.

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 by Kavulya et al. [2]. 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, see Guo et al. [3].

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 [4]. By default, HDFS replicates randomly data chunks several times onto the nodes (usually 3 times). 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 job priorities 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. In order to avoid delays, Zaharia et al. have proposed in [5] a delay scheduler to achieve locality while maintaining fairness by letting resources wait for a small delay in the highest priority job does not have local tasks to process in this node. In general, intuitively, having more replicas provides more opportunities for a given chunk of being processed locally.

It has been shown in [6] that depending on the size of the job, 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.

In this paper we focus on the Map phase of MapReduce, the distribution of

independent and presumed-homogeneous tasks between processors.

2 Mathematical Definition of the Problem and Notations

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 (*i.e.* tasks with the same duration), 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 a processor idle if it does not own unprocessed chunks anymore).

Both of these metrics will be considered, always under the assumption that the other one is optimized. More precisely, if we consider the makespan metric, we assume that only local tasks are performed and if we consider the communication metric, we assume that if a processor is idle it will begin an unprocessed task.

Throughout this paper, we rely on the following model. We represent the initial placement of data chunks by a bipartite graph $G = (T, P, E)$ with a set T of n task nodes and a set P of m processor nodes. An edge $e \in E$ between task node t_j and processor node p_i indicates the presence of (copy of) a chunk of task t_j on processor p_i. Let σ be a function from T to P (tasks allocation). With the assumption of homogeneity, our two metrics are equivalent to the following two problems.

Problem 1 (MAKESPAN-MAPREDUCE)**.** *Given a bipartite graph $G = (T, P, E)$ find a function $\sigma : T \to P$ such that*

- *For every $t_j \in T$, $\{\sigma(t_j), t_j\} \in E$.*
- $\max_{p_i \in P} |\{t_j,\ \sigma(t_j) = p_i\}|$ *is minimized.*

Problem 2 (COMMUNICATION-MAPREDUCE)**.** *Given a bipartite graph $G = (T, P, E)$ find a function $\sigma : T \to P$ such that:*

- $\max_{p_i \in P} |\{t_j,\ \sigma(t_j) = p_i\}| \leq \left\lceil \frac{|T|}{|P|} \right\rceil$.
- $|\{t_j,\ \{\sigma(t_j), t_j\} \notin E\}|$ *is minimized.*

In MAKESPAN-MAPREDUCE the first condition ensures that all tasks are local and $\max_{p_i \in P} |\{t_j,\ \sigma(t_j) = p_i\}|$ denotes the makespan of the allocation (as all tasks have the same processing time, the most loaded processor will be the last to end). In COMMUNICATION-MAPREDUCE, the first condition ensures the optimality of the makespan (if all processors perform no more than $\left\lceil \frac{|T|}{|P|} \right\rceil$ tasks, load balancing is perfect) and $|\{t_i,\ \{\sigma(t_j), t_j\} \notin E\}|$ denotes the number of non-local tasks. To build the schedule σ, we will rely on assignments (see Definition 1). An assignment is thus a subset of the edges such that each

task node has a unique edge in this subset. Therefore every assignment is a valid schedule. In particular this schedule respects the locality forced in MAKESPAN-MAPREDUCE, and the quality of an assignment can be measured using its maximum degree and its total load imbalance (see Definition 2).

Definition 1 (Assignment)**.** *Let $G = (P, T, E)$ be a bipartite graph. An* assignment *of G is a subset A of E such that:*

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

Definition 2 (Degree in an assignment, maximum degree and total load imbalance)**.** *Let A be an assignment of $G = (P, T, E)$, then*
(i) 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,*
(ii) the maximum degree *$d(A)$ of A is defined as $D(A) = \max_{p_i \in P} d_A(p_i)$ and*
(iii) the total load imbalance *$Imb(A)$ of A is given by*

$$Imb(A) = \sum_{\substack{p_i \in P \\ d_A(p_i) > \left\lceil \frac{|T|}{|P|} \right\rceil}} \left(d_A(p_i) - \left\lceil \frac{|T|}{|P|} \right\rceil \right).$$

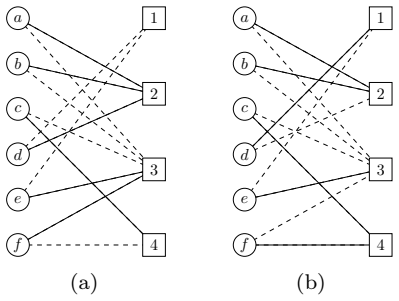


Fig. 1: Two examples of assignments for the same input graph.

These notions are illustrated on Figure 1, where tasks are on the left hand side and processors on the right. Solid edges represent an assignment while 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) and the total load imbalance is 1. On Figure 1(b), the maximum degree is 2 (reached on 2 and 4) and the total load imbalance is 0.

Every schedule that is a solution of MAKESPAN-MAPREDUCE is a valid assignment (and every assignment is a schedule that is a solution of MAKESPAN-MAPREDUCE) and if $D(A)$ is minimized, then $\max_{p_i \in P} |\{t_j,\ A(t_j) = p_i\}|$ is minimized. Therefore, to solve MAKESPAN-MAPREDUCE we can focus on assignments with minimal maximal degree only.

For COMMUNICATION-MAPREDUCE we proceed as follows.

- If A is an assignment, then for every processor p_i such that $d_A(p_i) > \left\lceil \frac{|T|}{|P|} \right\rceil$ we unassign $d_A(p_j) - \left\lceil \frac{|T|}{|P|} \right\rceil$ tasks and attribute them to processors $p_{i'}$s such that $d_A(p_{i'}) < \left\lceil \frac{|T|}{|P|} \right\rceil$. At the end, we obtain a schedule σ such that $\max_{p_i \in P} |\{t_j,\ \sigma(t_j) = p_i\}| \leq \left\lceil \frac{|T|}{|P|} \right\rceil$ and $|\{t_j,\ \{\sigma(t_j), t_i\} \notin E\}| \leq Imb(A)$ as non-local tasks may only be tasks whose assignment has been modified by the previous transformation.
- If σ is a schedule such that $\max_{p_i \in P} |\{t_j,\ \sigma(t_j) = p_i\}| \leq \left\lceil \frac{|T|}{|P|} \right\rceil$, then for every non-local task, we assign this task to an arbitrary processor such that there is an edge between them. With this transformation, we obtain a valid assignment A and $Imb(A) \leq |\{t_j,\ \{\sigma(t_j), t_j\} \notin E\}|$. Indeed, set $\{p_i,\ d_A(p_i) > \left\lceil \frac{|T|}{|P|} \right\rceil\}$ is empty before the transformation (as $\{t_j,\ \sigma(t_j) = p_i\}| \leq \left\lceil \frac{|T|}{|P|} \right\rceil$) and thus during the transformation, $Imb(A)$ only increases by at most 1 at each new assignment of the t_j.

Thus, from every solution of COMMUNICATION-MAPREDUCE with C communications we can produce an assignment A such that $\text{Imb}(A) \leq C$ and from every assignment A, we can produce a solution of COMMUNICATION-MAPREDUCE with less than $Imb(A)$ communications. Thus, to solve COMMUNICATION-MAPREDUCE, we focus on assignment with minimum total load imbalance.

3 Related Work

3.1 Locality in Map-Reduce

A number of papers 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, see Chowdhury and Stoica [7], [8] or Qiu et al. [9]. Some papers have also proposed to place Reduce tasks close to their input to reduce the communications of the shuffle phase, see Hammoud and Sakr [10] or Tan et al. [11].

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. [5] 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. [6] 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 and Lu [12] 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 that 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, [13] propose a new queuing algorithm to simultaneously maximize throughput and minimize delay in heavily loaded conditions.

Guo et al. [3] 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, their 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.

Isard et al. [14] propose a flow-based scheduler: Quincy. More precisely they consider the case of concurrent jobs and want to ensure that if a job is submitted to the platform its computation time would be smaller than Jt second where t is its computation time with exclusive access to the platform and J the number of jobs running on the platform. To respect this deadline, they propose a sophisticated model with many parameters (size of the input files, bandwidth, data transfer, possible pre-emption of tasks,...) and transform this problem into a flow problem (see Ford and Fulkerson [15]). Recently this algorithm's computation time has been significantly improved by Gog et al. [16] in order to have sub-second scheduling time at each submission of a new job.

3.2 Matchings in Bipartite Graphs

A first research direction (more precision in Section 5) 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 ErdÃűs and RÃľnyi [17] on random bipartite graphs proves that there exists a perfect matching of a bipartite graph of $2n$ vertices with asymptotic probability $e^{-2e^{-c}}$ as soon as the number of edges is $n \ln n + cn + o(n)$. Walkup [18], 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, see Ford and Fulkerson [15]. 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 [19], with a complexity of $O(m\sqrt{n})$, where n denotes the number of vertices and m the number of edges, or the one proposed by Goel et al. [20] for regular bipartite graphs, whose expected complexity is $O(n \log n)$. There also exist approximation algorithms with better computation time that no longer guarantee the computation of a perfect matching, see Lanffuth et al. [21] or DufossÃĺ et al. [22].

3.3 Balls-into-Bins

In the context of MAKESPAN-MAPREDUCE, processors are only allowed 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-In-Bins problems, see Raab and Steger [23]. More specifically, we prove in Section 4 that it is possible to simulate the greedy algorithm for assigning Map tasks with a variant of a Balls-In-Bins game. In this randomized process, n balls are placed randomly into m 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 by Berenbrink et al. [24]. It is shown 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, Raab and Steger [23] provide value on the expected maximum load, depending on the ratio $\frac{m}{n}$. For example, in the case $m = n$, the expected maximum load is $\frac{\log n}{\log \log n}(1 + o(1))$.

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 Mitzenmacher [25] and Richa et al. [26] for surveys that illustrate the power of two choices. Typically, combining previous results with those of Berenbrink et al. [27], it can be proved that whatever the expected number of balls per bin, the expected maximum load is of order $n/m + O(\log \log m)$ even in the case $r = 2$, what represents a very strong improvement over the single choice case. Peres et al. [28] study cases with a non-integer r. In this

case, with a given parameter β, for each ball, with probability β the assignation is made after choosing between two bins or, with probability $(1-\beta)$, the assignation is made like for a regular Balls-In-Bins process. In this case, for $0 < \beta < 1$, the expected maximum load is $\frac{n}{m} + O(\frac{\log m}{\beta})$. Thus, the exact $\beta = 1$ (*i.e.* $r = 2$) is needed to reach the $O(\log \log m)$ regular Balls-In-Bins gap. The combination of multiple choice games with weighted balls has also been considered by Peres et al. [28]. 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. Both the results for $(1+\beta)$ and for $r = 2$ have been extended.

4 Greedy Approach

Let us consider here a simple dynamic scheduler to solve MAKESPAN-MAPREDUCE, called Greedy and inspired by the Hadoop default scheduler. If there exist 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 the processor stops its execution. Note that this produces a 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, as in [24] or [28], that this heterogeneity in task durations and processor availability times allows us to consider that no ties have to be broken. Note that in Greedy, the initial chunk distribution is given by n random sets of r choices: $\mathcal{C}_1, \ldots, \mathcal{C}_n$. Together with the initial processor availability times and the task durations, these random choices entirely define the scheduler behavior.

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. The process of distributing n balls of sizes $w_1, \ldots, w_n$ into m bins whose initial loads are given by $\epsilon_1, \ldots, \epsilon_m$ 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 3. *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,*

$$\textsc{MaxLoad}(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_m) = C_{\max}(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_m).$$

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

Lemma 4. *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 $(\mathcal{C}_1, \ldots, \mathcal{C}_n, \epsilon_1, \ldots, \epsilon_m)$ 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-In-Bins processes derived in the literature, as related in the previous section. 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 $n/m + O(\log \log m)$ (with high probability).

5 Matching-Based Approach

In this section we are interested in optimal solutions of MAKESPAN-MAPREDUCE and COMMUNICATION-MAPREDUCE. First, in Section 5.1, we link MAKESPAN-MAPREDUCE to an already studied problem GRAPH-ORIENTIABILITY and rely of a few pre-existing results, notably the existence, with high probability, of an assignment with makespan smaller or equal than $\left\lceil \frac{|T|}{|P|} \right\rceil + 1$. Finally, in Section 5.2 and 5.3, we focus on COMMUNICATION-MAPREDUCE and present two algorithms to solve it in polynomial time. Note that these two algorithms are also proven optimal for MAKESPAN-MAPREDUCE, therefore, on every bipartite graph, there exists an assignment that is optimal for both MAKESPAN-MAPREDUCE and COMMUNICATION-MAPREDUCE.

5.1 Results for Makepan Metric

Let us first introduce the GRAPH-ORIENTIABILITY problem. We distinguish (undirected) edges, denoted $\{u, v\}$ from (directed) arcs, denoted (u, v).

Problem 3 (GRAPH-ORIENTIABILITY)**.** *Given an undirected graph* $G = (V, E)$*, find a directed version* $G' = (V, E')$*, with* $\{u, v\} \in E \Leftrightarrow (u, v) \in E'$ *or* $(v, u) \in E'$*, such that* $\max_{v \in V} |\{u \in V,\ (u, v) \in E'\}|$ *is minimized.*

GRAPH-ORIENTIABILITY can be used to solve MAKESPAN-MAPREDUCE in the case where the degree of each task is 2. Let $G = (P, T, E)$ be such an instance of MAKESPAN-MAPREDUCE. Let $G' = (P, E')$ be an undirected graph such that $\{p_i, p_{i'}\} \in E'$ if and only if $\{p_i, t_j\}$ and $\{p_{i'}, t_j\}$ are in E (we allow multiple edges). Note that $|E'| = |T|$. If we consider a directed version $G'' = (P, E'')$ of G' then A, defined such that $\{p_i, t_j\} \in A$ if and only if $(p_{i'}, p_i) \in E''$, it is a valid assignation and its maximal degree is the maximal number of incident arc to a vector in G'' (the orientation of an edge represents a choice between two vertices), see Figure 2. Note that GRAPH-ORIENTIABILITY can be extended to hypergraphs to model versions of MAKESPAN-MAPREDUCE with arbitrary r (where r is the number of replicas of each task).

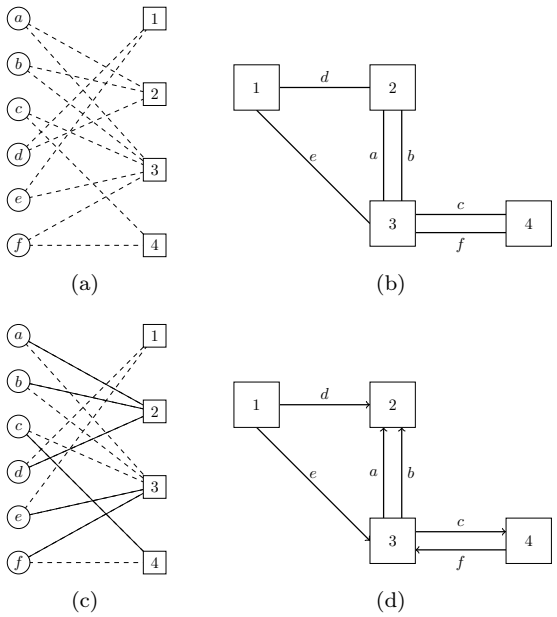


Fig. 2: Figure 2(a) represents an input of MAKESPAN-MAPREDUCE which equivalent version of an input of GRAPH-ORIENTIABILITY is on Figure 2(b). The assignment on Figure 2(c) is equivalent to the orientation of Figure 2(d).

One of the main contributions to this problem comes from Sanders et al. [29],

where the authors provide polynomial algorithms for GRAPH-ORIENTIABILITY and its hypergraph version and a probabilistic evaluation on the expected value of the optimal value of an input of GRAPH-ORIENTIABILITY. More precisely, with high probability, the optimal solution is at least *near-perfect* (*i.e.* $\max_{v \in V} |\{u \in V,\ (u,v) \in E'\}| \leq 1 + \left\lceil \frac{|E'|}{|P|} \right\rceil$).

Theorem 5 (Sanders et al. [29])**.** *Let $G = (V, E)$ be an undirected graph. With high probability there exists a directed version of G such that the vertex with the most in-incident arcs has an in-degree inferior or equal to $1 + \left\lceil \frac{|E|}{|V|} \right\rceil$.*

Therefore, under the assumption that each task has at least two replicas (otherwise we are in a case close to the Balls-In-Bins model without power of r choices), Theorem 5 says that there exists a near-perfect assignment for the MAKESPAN-MAPREDUCE problem, as stated in the following corollary.

Corollary 6. *Let $G = (P, T, E)$ be a bipartite graph such that the degree of every element of T is at least 2. Then, with high probability there exists an assignation with maximal degree smaller or equal to $1 + \left\lceil \frac{|T|}{|P|} \right\rceil$.*

Proof. By randomly removing excess edges, let us consider the case where for all $t_j \in T$, the degree of t_j is 2. Then, we can transform G into an undirected graph $G' = (P, E')$ as described above (see Figure 2). Thanks to Theorem 5, we know that there is a directed version of G' such that the vertex with the most in-incident arcs has an in-degree inferior or equal to $1 + \left\lceil \frac{|E'|}{|P|} \right\rceil = 1 + \left\lceil \frac{|T|}{|P|} \right\rceil$. This oriented version of G' has an induced assignment for G with the transformation also described in Figure 2, and this assignment has the same maximal degree as the maximal in-degree in G'', what achieves proof of Corollary 6. □

Note that we propose in the Annexe a proof of Corollary 6 using the biparite graph modeling and results on matching.

Furthermore, Theorem 5 has been extended by Czumaj et al. [30] by proving that for a small number of tasks the optimal solution of GRAPH-ORIENTIABILITY is exactly $1 + \left\lceil \frac{|T|}{|P|} \right\rceil$ and for large numbers it is $\left\lceil \frac{|T|}{|P|} \right\rceil$.

Theorem 7 (Czumaj et al. [30])**.** *Let $G = (V, E)$ be an undirected graph. There exists two positive constants λ, c such that:*

- *If $|E| \geq c|V| \log |V|$ then with high probability there exists a directed version of G such that the vertex with the most in-incident arcs has an in-degree equal to $\left\lceil \frac{|E|}{|V|} \right\rceil$.*
- *If $|E| \leq \lambda |V| \log |V|$ then with high probability there exists a directed version of G such that the vertex with the most in-incident arcs has an in-degree equal to $1 + \left\lceil \frac{|E|}{|V|} \right\rceil$.*

With this theorem, the probabilistic study of the expected optimal maximal degree of an assignment is closed. Very efficient algorithms have been proposed to polynomially solve GRAPH-ORIENTIABILITY, see Sanders [31] or Cain et al. [32], with very good complexity (near-linear in the case of graphs, *i.e.* $r = 2$), and we will therefore concentrate in what follows on communication optimal algorithms.

5.2 A First Communication-Optimal Algorithm

In this section we focus on COMMUNICATION-MAPREDUCE problem. In what follows, we propose a polynomial algorithm for COMMUNICATION-MAPREDUCE, based on the literature on matchings and flow problems (see Ford and Fulkerson [15] for instance).

5.2.1 Alternating Paths

Let us first adapt the notion of alternating path.

Definition 8. *Let $G = (P, T, E)$ be a bipartite graph and let A be a subset of E.*

- *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 unused edges. On the right hand side, the proposed path (to improve the clarity of the scheme, edges that are not in the path have been removed) is an alternating one according to the previous assignment.

Alternating paths can be used to improve an existing assignment. Indeed, Lemma 9 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 while increasing by one the degree of the first one.

Lemma 9. *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.*
- *$d_{A \otimes x}(p_{i_1}) = d_A(p_{i_1}) + 1$*

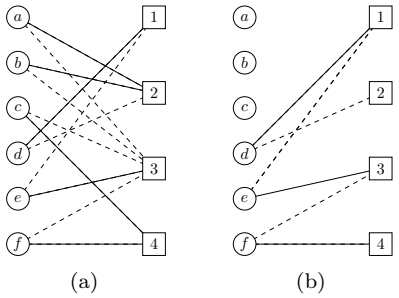


Fig. 3: Example of alternating path.

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

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 neighbourhood in $A \otimes x$,
- if $p_i = p_{i_k}$, then $(t_{j_{k-1}}, p_{i_k})$ is removed from its neighbourhood 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 neighbourhood and the edge $(t_{j_{l-1}}, p_{i_l})$ is removed from its neighbourhood in $A \otimes x$,
- otherwise there is no modification of its neighbourhood from A to $A \otimes x$.

Therefore,

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

□

From now on, let us focus on finding an alternating path that improves the whole assignment. For this purpose we define improving path.

Definition 10. *Let $G = (P, T, E)$ be a bipartite graph and A be an assignment of G. An* improving 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 9 implies that an improving 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 $d_A(p_{i_k}) = D(A)$. Similarly, an improving path can also be used to build an assignment that decreases the total load imbalance if $d_A(p_{i,k}) > \left\lceil \frac{|T|}{|P|} \right\rceil$ and $d_A(p_{i,1}) < \left\lceil \frac{|T|}{|P|} \right\rceil$.

5.2.2 Optimality

In fact, a stronger property holds true. If there is no improving path, then the assignment has a minimum total load imbalance. This result is formalized in Theorem 11.

Theorem 11. *Let $G = (P, T, E)$ be a bipartite graph, let A be an assignment of G. If there is no improving path according to A then A has a minimum total load imbalance.*

Proof. As a first step, we establish the relationship between assignments and matchings. We recall that a matching of a bipartite graph $G = (P, T, E)$ is a subset M 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 set of integer $(l_1, \ldots, l_m)$, we build an auxiliary bipartite graph $G^{(l_1,\ldots,l_m)}$ from the bipartite graph representation of our problem, by replicating l_i times each p_i, see Figure 4. We show in the following lemma that the existence of an assignment of local degree l_i for all p_i in G is directly related to the existence of a matching of size $|T|$ in $G^{(l_1,\ldots,l_m)}$.

Definition 12. *Let $G = (P, T, E)$ be a bipartite graph and $(l_1, \ldots, l_m)$ be a set of integers be an integer. We define $G^{(l_1,\ldots,l_m)} = (P^{(l_1,\ldots,l_m)}, T, E^{(l_1,\ldots,l_m)})$ with:*

- $P^{(l_1,\ldots,l_m)} = \bigcup_{p_i \in P} \{p_i^1, \ldots, p_i^{l_i}\}$.
- $(p_i^k, t_j) \in E^{(l_1,\ldots,l_m)}$ *if and only if* $(p_i, t_j) \in E$.

Lemma 13. *Let $G = (P, T, E)$ be a bipartite graph with $|T| = n$, $|P| = m$ and $(l_1, \ldots, l_m)$ be a set of integers. Then, there exists an assignment A with, for all $p_i \in P$, $d_A(p_i) \leq l_i$ if and only if there exists a matching of size n in $G^{(l_1,\ldots,l_m)}$.*

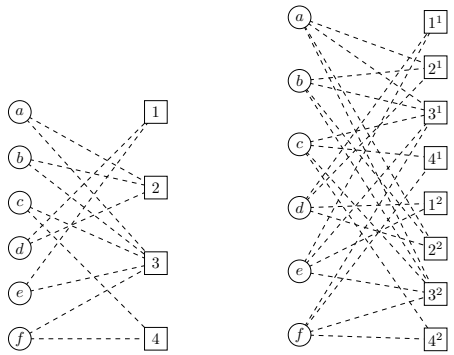


Fig. 4: An example of replication of a bipartite graph. On the left G, on the right $G^{(2,2,2,2,2,2)}$.

Proof. Let us first suppose that there exists an assignment A such that for all $p_i \in P$, $d_A(p_i) \leq l_i$. For $p_i \in P$ let $v_A(p_i)$ denotes its neighbourhood in the subgraph 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 $E^{(l_1,\dots,l_m)}$ with $M = \bigcup_{p_i \in P} M_{p_i}$ with

$$M_{p_i} = \big\{(p_i^1, t_1), \dots, (p_i^{d_A(p_i)}, t_{d_A(p_i)}), \\ \{t_1, \dots, t_{d_A(p_i)}\} = v_A(p_i)\big\}.$$

Since $\forall p_i \in P,\ d_A(p_i) \leq l_i$, then $\forall p_i,\ \{p_i^1, \dots, p_i^{d_A(p_i)}\}$ is a valid subset of $P^{(l_1,\dots,l_m)}$. 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 $G^{(l_1,\dots,l_m)}$.

Let us now assume that there exists a matching M of $G^{(l_1,\dots,l_m)}$ 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 and for all $p_i \in P$, $d_A(p_i) \leq l_i$ for all $p_i \in P$. □

Now, let us introduce a second lemma to state the different cases that may happen if we use an alternating path to change an assignment, with regards on load imbalance.

Lemma 14. *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 $d_A(p_d) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_A(p_f)$ then $Imb(A \otimes x) = Imb(A) - 1$,*
- *if $d_A(p_d) > \left\lceil \frac{|T|}{|P|} \right\rceil > d_A(p_f)$ then $Imb(A \otimes x) = Imb(A) + 1$,*
- *otherwise $Imb(A \otimes x) = Imb(A)$.*

Proof. By direct application of Lemma 13. □

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 improving path.

We will rely on the three following lemmas to prove Theorem 11.

Lemma 15. *(Berge [33]) 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 16. *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 in A.*

Proof. Let now A and A' be two assignments of a bipartite graph G such that $\exists p_i$ that fulfils $d_A(p_i) > d_{A'}(p_i)$. Let p_f be the vertex satisfying $d_A(p_f) > d_{A'}(p_f)$ with the largest degree in A. Let us consider the graph $G^{(l_1,\ldots,l_m)}$ with $l_f = d_A(p_f) - 1$ times and for $i \neq f$, $l_i = \max(d_A(p_i), d_{A'}(p_i))$. Let $t \in T$ be such that $(p_f, t) \in A$. Thanks to the modified version of Lemma 13, we can define a matching M' of $G^{(l_1,\ldots,l_m)}$ of size $|T|$ and a matching M of size $|T| - 1$ 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 15, 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 fulfil $x = (p_d^{k_d}, \ldots, t)$. In addition, to have a free replicate in $G^{(l_1,\ldots,l_m)}$ according to M, p_d must fulfil $d_A(p_d) < \max(d_A(p_d), d_{A'}(p_d))$ 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 we have the claimed result. □

Lemma 17. *Let $G = (P, T, E)$ be a bipartite graph and let A, A' be two assignments of G. Therefore there are 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_k} *is the* p_i *with larger degree in* A_k *such that* $d_{A_k}(p_i) > d_{A'}(p_i)$.
- $\forall p_i \in P$, *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. Otherwise, we recall that $\sum_{p_i \in P} d_{A_k}(p_i) = |T| = \sum_{p_i \in P} d_{A'}(p_i)$. Thus if $\exists p_i$ such that $d_{A_k}(p_i) \neq d_{A'}(p_i)$, then $d_{A_k}(p_i) > d_{A'}(p_i)$ or $\exists p'_i$ is such that $d_{A_k}(p'_i) > d_{A'}(p'_i)$. Therefore, according to Lemma 16, 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_k} is the vertex with the larger degree in A_k 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. □

Let us now finish the proof of Theorem 11. 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 17, 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 14, $d_{A_{k_0}}(p_{d_{k_0}}) < \left\lceil \frac{|T|}{|P|} \right\rceil < d_{A_{k_0}}(p_{f_{k_0}})$. 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 improving path according to A, what achieves the proof of Theorem 11.

□

Therefore, by searching for an improving path until there is none left, COMMUNICATION-MAPREDUCE can be optimally solved. Theorem 11 therefore provides a first algorithm, FindAssignment (see Algorithm 1).

Algorithm 1: FindAssignment(G)

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

$A = \emptyset$;
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 improving path according to* A **do**
  Compute an improving path x according to A;
  $A \leftarrow A \otimes x$;
return A

In addition, Theorem 11 can be adapted to MAKESPAN-MAPREDUCE. Indeed, if there is no improving path according to an assignment A, then A has a minimum maximum degree, as stated in Theorem 18. Therefore FindAssignment solves MAKESPAN-MAPREDUCE and COMMUNICATION-MAPREDUCE simultaneously and can be used in both cases.

Theorem 18. *Let $G = (P, T, E)$ be a bipartite graph, let A be an assignment of G. If there is no improving path according to A, then A has a minimum maximum degree.*

Proof. This proof is very similar to the proof of Theorem 11 and relies on the same lemmas, in particular Lemma 17.

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 17, 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 $p_{f_{k_0}}$ is the only processor 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 improving, $D(A_{k_0}) = d_{A_{k_0}}(p_{f_{k_0}})$ 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 improving 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 improving path $x = (p_d, \ldots, p_f)$ according to A_k (and that improving path could differ from x_k) such that $d_{A_k}(p_f) = D(A_k)$ and $d_{A_{k_0}}(p_d) < d_{A'}(p_d)$. In order to prove that $k_0 = 0$ by contradiction, let us assume that $k_0 > 0$.

Let $x = (p_d, \ldots, p_f)$ be such an improving 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 improving path and $d_{A_{k_0-1}}(p_f) = D(A_{k_0-1})$. Otherwise, $D(A_{k_0-1}) = d_{A_{k_0-1}}(p_{f_{k_0-1}})$, indeed $D(A_{k_0-1}) > D(A_{k_0}) > D(A')$, 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 improving 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 improving. 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 improving 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 improving 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 improving path according to A_{k_0-1} with all desired properties. Therefore, $k_0 = 0$ and there is an improving path according to $A_0 = A$, what achieves the proof of the theorem.

□

5.2.3 Correctness and Complexity

To prove the termination of FindAssignment, let us prove that the number of improving paths goes to 0. Let us consider $\sum d_A(p_i)^2$. If $x = (p_d, \ldots, p_f)$ is an improving 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 a decreasing function during the execution of FindAssignment. Since this value is bounded (trivially by 0), then there is an instant where there is no longer an improving path and FindAssignment terminates, returning an assignment with minimum total load imbalance and minimum maximum degree.

As for Ford-Fulkerson Algorithm [15], the search for an improving path can be done using breadth-first-search in $O(|E|)$. Furthermore $0 \leq \sum d_A(p_i)^2 \leq \left(\sum d_A(p_i)\right)^2 = |T|^2$ and $\sum d_A(p_i)^2$ strictly decreases at each step. Thus, there are at most $|T|^2$ steps and each step requires $|E|$ operations. Therefore, the worst case complexity of FindAssignment is $O(|E||T|^2)$. Note that in the application problem $|E| = r|T|$ where r is the number of replications of each chunk. Then the worst case complexity is $O(|T|^3)$.

Theorem 19. FindAssignment *terminates in at most* $O(|E||T|^2)$ *operations.*

5.3 A Faster Communication-Optimal Algorithm

However, in FindAssignment, the search of an improving path can be expensive, whereas the problem of building fast scheduling algorithms has recently been highlighted by Goel et al. in [16]. We therefore propose a faster algorithm, BestAssignment.

5.3.1 Presentation

Instead of going from any assignment and improving it step by step, like FindAssignment, BestAssignment goes a processor after another, searching at each time for the smallest alternating path from this processor to a non-assigned task (if any) and applying it to the current partial assignment, see Algorithm 2.

Algorithm 2: 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 an alternating path according to A from p_i to an
        unassigned t_j then
            x = the smallest such path ;
            A ← A ⊗ x ;
        else
            Remove p_i from P ;
return A
```

5.3.2 Correctness and Complexity

As for FindAssignment, the search for alternating path can be done in $O(|E|)$. Furthermore at each step, either a task is assigned or a processor is removed, thus the number of steps is $O(|T| + |P|)$. Thus BestAssignment has a worst case complexity of $O((|T| + |P|)|E|)$. Furthermore, we prove the optimality of BestAssignment by showing there is no improving path in the returned assignment.

Theorem 20. BestAssignment *returns an Assignment with no improving path.*

Proof. Let A_∞ be the final set returned by BestAssignment. If we suppose that there is always an edge incident to each task (else there is no existing assignment anyway), trivially A_∞ is an assignment. 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 assignment by showing there is no improving path according to A_∞.

Let p_{i_1} be the first processor, if any, such that there is no alternating path from p_{i_1} to an unassigned task according to the current state of A that we denote A_1. Let us now define P_1 as the subset of P that contains all the processors

that can be reached with an alternating path from p_{i_1} according to A_1. Let T_1 be the neighbourhood of P_1, *i.e.* $T_1 = \{t_j \in T,\ \exists p_i \in P_1,\ (p_i, t_j) \in E\}$.

We now want to show that for all $t_j \in T_1$ there exists p_i in P_1 such that $(p_i, t_j) \in A_1$. Let t_j be in T_1 and $p_i \in P_1$ be in its neighbourhood. By definition of P_1 there is an alternating path $(p_{i_1}, \ldots, p_i)$ according to A_1. If $(p_i, t_j) \in A_1$ we have our result, else $(p_{i_1}, \ldots, p_i, t_j)$ is a valid alternating path according to A_1. By definition of p_{i_1}, t_j is assigned and then there is $p_{i'}$ such that $(p_{i'}, t_j) \in A_1$ and thus $x(p_{i_1}, \ldots, p_i, t_j, p_{i'})$ is also a valid alternating path and $p_{i'} \in P_1$. Therefore we have for all $t_j \in T_1$ there exists p_i in P_1 such that $(p_i, t_j) \in A_1$.

Let now A_{1,T_1} (respectively A_{∞,T_1}) be $\{(p_i, t_j) \in A_1,\ t_j \in T_1\}$ (respectively $\{(p_i, t_j) \in A_\infty,\ t_j \in T_1\}$). We also denote similarly A'_{T_1} for any partial assignment A'. Then, for every partial assignment A' after A_1, we prove $A_{1,T_1} = A'_{T_1}$. Let us suppose otherwise and search for a contradiction. Let A' be the first partial assignment after A_1 such that $A_{1,T_1} \otimes A'_{T_1} \neq \emptyset$. Because all the edges of A_1 from T_1 goes to an element of P_1, we know that there is $(p_i, t_j) \in A_{1,T_1} \otimes A'_{T_1}$ such that $p_i \in P_1$. In addition, in the assignment just before A' (we denote it A''), there is an alternating path that contains (p_i, t_j). Let $x = (p_d, \ldots, p_i, t_j, \ldots, t_{j'})$ be this path ($A' = A'' \otimes x$). We know that $t_{j'}$ is unassigned in A'' and thus $t_{j'} \notin T_1$ and so $x = (p_d, \ldots, p_i, t_j, \ldots, p_f, t_{j'})$ with $p_f \notin P_1$. Hence there is $p_{i'}$ such that $p_{i'}$ is the first vertex in $P \setminus P_1$ after p_i in x. Without loss of generality we can assume that $p_{i'}$ is after t_j. By definition $A''_{T_1} = A_{1,T_1}$ and then $(p_{i_1}, \ldots, p_i, t_j, p_{i'})$ is a valid alternating path according to A_1 and A'' and therefore $p_{i'} \in P_1$ that is a contradiction. Then, for every partial assignation A' after A_1, $A_{1,T_1} = A'_{T_1}$. More precisely, after A_1, all processors in P_1 will not be in any alternating path (this is why they are removed from P in BestAssignment) and all tasks in T_1 will stay assigned to processors from P_1.

Similarly we also prove that $d_{A_1}(p_i) = d_{A'}(p_i)$ for all p_i in P_1 and, by construction $|d_{A_1}(p_i) - d_{A_1}(p_{i'})| \leq 1$ for all $p_i, p_{i'} \in P_1^2$ (at each step of the algorithm the degree of the departure vertex increase by one).

Now we define p_{i_k} as the first processor of $P \setminus \bigcup_{1 \leq l < k} P_l$ if any, such that there is no alternating path from p_{i_k} to an unassigned task. Let P_k the subset of $P \setminus \bigcup_{1 \leq l < k} P_k$ that contains all the processors that can be reached with an alternating path from p_{i_k} according to the current partial assignation, denoted A_k. We also define T_k its neighbourhood in $T \setminus \bigcup_{1 \leq l < k} T_k$. Finally we also denote $P_\infty = P \setminus \bigcup P_k$ and T_∞ its neighbourhood in $T \setminus \bigcup T_k$.

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

- For all $t_j \in T_k$ there is p_i in P_k such that $(p_i, t_j) \in A_k$.
- For every partial assignation A' after A_k, $A_{k,T_k} = A'_{T_k}$.
- For every partial assignation A' after A_k, $d_{A_k}(p_i) = d_{A'}(p_i)$ for all p_i in P_k and $|d_{A_k}(p_i) - d_{A_k}(p_{i'})| \leq 1$ for all $p_i, p_{i'} \in P_k^2$.

Note that $|d_{A_k}(p_i) - d_{A_k}(p_{i'})| \leq 1$ for all $p_i, p_{i'} \in P_k^2$ and for all $t_j \in T_k$ there

is p_i in P_k such that $(p_i, t_j) \in A_k$ holds true for $k = \infty$. Note also that P_∞ can be equal to P in the case where an alternating path is found at each step.

We now want to prove that there is no improving path according to A_∞. First, as $|d_{A_k}(p_i) - d_{A_k}(p_{i'})| \leq 1$ for all $p_i, p_{i'} \in P_k^2$ and $d_{A_k}(p_i) = d_{A_\infty}(p_i)$ for all p_i in P_k. Thus there is no improving path inside the P_ks.

In addition we easily note that $|(\max\ d_{A_k}(p_i)) - (\min\ d_{A_{k'}}(p_{i'}))| \leq 1$ for $k < k'$. Therefore there is no improving path from a vertex of $P_{k'}$ to a vertex of P_k if $k < k'$. Hence the only possible improving path are from P_k to $P_{k'}$ with $k < k'$. However, in this case, there is t_j in T_k such there exists p_i in $P \setminus P_k$ such that $(p_i, t_j) \in A_\infty$ (a necessary condition to a such alternating path). Yet we prove for all $(p_i, t_j) \in A_{\infty,T_k} = A_{k,T_k}$, $p_i \in P_k$ and thus we have our proof of optimality of BestAssignment. □

With Theorem 20 and the previous remark, we prove that BestAssignment solves COMMUNICATION-MAPREDUCE and MAKESPAN-MAPREDUCE with a worst case complexity of $O((|T| + |P|)|E|)$ ($O(|T|^2 + |T||P|)$ in the case $|E| = r|T|$).

In practice, we wrote a C implementation for simulations of Section 6 and we measured its computation time. The results are depicted in Figure 5. The average computation time is below $1s$ (as targeted in [16], but the difference with their model brings a need for further investigations to do a fair comparison), except for very high values of $|T|$, *i.e.* when there are more than 250000 tasks, what is 3 times the number of tasks of the largest job in [2]. Anyway, even in these cases, the average computation time is, at most, around $3s$. Note that a cluster of 10000 machines is not unrealistic, a the full Google cluster is composed of 12500 machines according to [16].

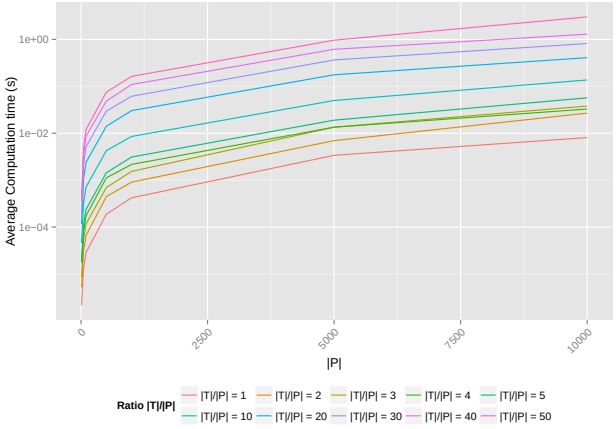


Fig. 5: Average computation time in second of BestAssignment for different values of $\frac{|T|}{|P|}$ and $|P|$.

6 Simulations

In this section, we consider both MAKESPAN-MAPREDUCE and COMMUNICATION-MAPREDUCE problems, that can both be solved in polynomial time. We first consider a setting where all tasks have the same computation time (homogeneous case, Section 6.2), *i.e.* a setting where BestAssignment is optimal. In this case, we aim at comparing it against pre-existing strategies in order to evaluate the gain of using an optimal strategy. Second, we consider on-line heterogeneous tasks, *i.e.* tasks for which the duration may defer from one tasks to another (heterogeneous case ,Section 6.3) and is not known at the time when the scheduling decision is made. In this case, we aim at evaluating the resilience of BestAssignment-based strategies.

6.1 Settings

In this section we present the settings we use later for our simulations.

6.1.1 Strategies

We concentrate on 3 classes of scheduling strategies, *i.e.* Greedy-based, BestAssignment-based and Maestro-based. The first two ones are based on Greedy and BestAssignment defined in above sections. The last one uses a strategy proposed by Ibrahim et al. in [6]. Maestro is a greedy two-waves scheduler. During the first phase, Maestro assigns to the processor with the highest risk to process non-local tasks the local task it hosts that has the highest risk of not being processed locally. During the second phase, that is purely dynamic, each time a processor is idle, the local task with the highest chunk weight is assigned to this processor. Let us now define the different strategies. First, if non-local tasks are forbidden (makespan metric),

- Greedy-Makespan uses Greedy algorithm to dynamically perform the assignment.
- Static-BestAssignment-Makespan uses the assignment statically computed by BestAssignment. Note that on uniform settings, any optimal-certified algorithm will achieve the same efficiency.
- BestAssignment-Makespan uses the assignment statically computed by BestAssignment. If a processor is idle with no pre-assigned tasks available, it randomly chooses an unprocessed task among those for which it owns a chunk locally. This strategy is only used in Section 6.3 to adapt BestAssignment in the heterogeneous case.
- Maestro-Makespan uses the two waves of Maestro to statically and then dynamically assign tasks. Note that for the second wave, non-local-tasks are forbidden and thus processors remain idle when all the tasks for which they own chunks locally have been processed.

Then, if non-local tasks are forbidden (communication metric),

- GREEDY-Communications uses GREEDY algorithm to dynamically perform the assignment. If a processor is idle without local chunks, it randomly picks a task among the ones on the processor with the largest number of chunks at this instant.
- BESTASSIGNMENT-Communications uses the assignment (statically) computed by BESTASSIGNMENT. If a processor is idle with no pre-assigned tasks, it first tries to process one of its local tasks. If there is no longer unprocessed local tasks, then the processor steals a task among the local tasks of the processor with the largest number of unprocessed assigned tasks at this instant. This stealing procedure is close to the transformation described in Section 2. An idle processor is idle corresponds to the case where its degree is below $\left\lceil \frac{|T|}{|P|} \right\rceil$. As the stolen processor has the highest degree, its degree is above $\left\lceil \frac{|T|}{|P|} \right\rceil$. If the stolen task is local, the stolen processor degree may be below $\left\lceil \frac{|T|}{|P|} \right\rceil$ but in this case this is equivalent to use an alternating path without changing the load imbalance.
- MAESTRO-Communications uses the two waves from MAESTRO to statically and then dynamically compute the assignment.

Note that in both cases, the MAESTRO strategies slightly differ from [6] as we do not launch speculative tasks to deal with failures. Therefore, the results of MAESTRO in our simulations are slightly better for the communications metric.

6.1.2 Traces

In order to produce realistic test cases for the heterogeneous settings, we use the traces from [2]. These traces correspond to the record of jobs during 10 months on a Hadoop cluster. In order to emulate the on-line aspect, *i.e.* the fact that task durations are unkknown when they are submitted, the computation time of each task is randomly picked at the beginning of its simulated execution. In the following, we focus on jobs with at most 10000 tasks (more than 95% of the jobs in [2]). In addition, we classify the jobs according to their Normalized Standard Deviation (NSD), *i.e.* the standard deviation of the computation times divided by the mean of the computation times, see Table 1.

6.2 Homogeneous Settings

In this section, we focus on the case where all tasks have the same computation time (thus not based on traces). For each strategy, we run 250 simulations for $|P| = 50$, for $\frac{|T|}{|P|} \in \{1, 2, 3, 4, 5, 10, 20, 30, 40, 50\}$ and for $r \in \{2, 3, 4, 5\}$ where r is the number of replications of each chunk ($r = 3$ in default HDFS, [4]).

	$NSD < 0.05$	$NSD \in [0.05, 0.1[$	$NSD \in [0.1, 0.25[$	$NSD \in [0.25, 0.5[$	$NSD \in [0.5, 1[$	$NSD \geq 1$	Total
$\|T\| < 50$	49 (2, 41%)	109 (5, 35%)	123 (6, 04%)	60 (2, 95%)	39 (1, 92%)	26 (1, 28%)	406 (19, 94%)
$\|T\| \in [50, 100[$	18 (0, 89%)	50 (2, 46%)	93 (4, 57%)	61 (3, 00%)	34 (1, 67%)	23 (1, 13%)	279 (13, 70%)
$\|T\| \in [100, 250[$	11 (0, 54%)	75 (3, 68%)	205 (10, 07%)	110 (5, 40%)	78 (3, 83%)	25 (1, 23%)	504 (24, 75%)
$\|T\| \in [250, 500[$	10 (0, 49%)	55 (2, 70%)	105 (5, 16%)	68 (3, 34%)	50 (2, 46%)	17 (0, 83%)	305 (14, 98%)
$\|T\| \in [500, 1000[$	1 (0, 05%)	5 (0, 25%)	21 (1, 03%)	44 (2, 16%)	33 (1, 62%)	18 (0, 88%)	122 (5, 99%)
$\|T\| \in [1000, 5000[$	1 (0, 05%)	10 (0, 49%)	43 (2, 11%)	32 (1, 57%)	33 (1, 62%)	76 (3, 73%)	195 (9, 58%)
$\|T\| \in [5000, 10000[$	0 (0, 00%)	16 (0, 79%)	57 (2, 80%)	33 (1, 62%)	16 (0, 79%)	10 (0, 49%)	132 (6, 48%)
$\|T\| \geq 10000$	0 (0, 00%)	9 (0, 44%)	4 (0, 20%)	14 (0, 69%)	39 (1, 91%)	27 (1, 33%)	93 (4, 57%)
Total	90 (4, 42%)	329 (16, 16%)	651 (31, 97%)	422 (20, 73%)	322 (15, 82%)	222 (10, 90%)	2036 (100%)

Tab. 1: Repartition in function of the number of tasks ($|T|$) and the normalized standard deviation (NSD) of the jobs from [2].

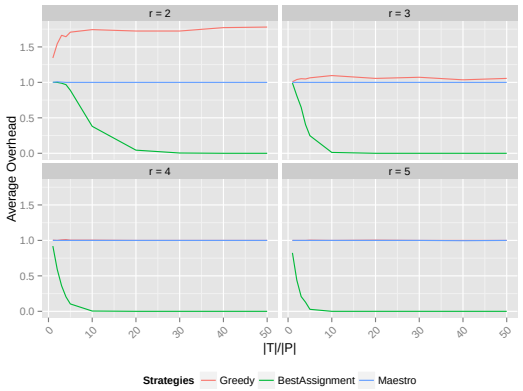


Fig. 6: Results for the makespan metric in the homogeneous settings when $|P| = 50$.

6.2.1 Makespan

Let us focus on the makespan metric, *i.e.* the case where non-local tasks are forbidden. As stated in Section 5.1, the asymptotically expected optimal makespan is $\frac{|T|}{|P|} + 1$ (for a small number of tasks per processor) or $\frac{|T|}{|P|}$ (for a large number of tasks per processor) and, as proved in Sections 5.2 and 5.3 BestAssignment is optimal.

The results of the simulations are depicted on Figure 6, where we display the average overhead, *i.e.* the makespan minus $\frac{|T|}{|P|}$ (the perfect load-balancing). We can notice the validity of the asymptotic theoretical results even for relative small values. Indeed, even in the case of small $|P|$ or $|T|$, the makespan is consistently below $\frac{|T|}{|P|} + 1$ (on more than 10^5 runs, the optimal makespan was $\frac{|T|}{|P|} + 2$ in only two cases). Moreover, we can notice a three phases behavior of the optimal value BestAssignment. First, for small $|T|$ (*i.e.* small $\frac{|T|}{|P|}$) the optimal is $\frac{|T|}{|P|} + 1$. Then, during a transition phase, the average of this optimal value is between $\frac{|T|}{|P|}$ and $\frac{|T|}{|P|} + 1$ and finally, for large values of $|T|$ the

optimal is $\frac{|T|}{|P|}$. At last, we can observe the behavior predicted by Theorem 7, with $|T| \leq \lambda |P| \log |P|$ at first and $|T| \geq c|P| \log |P|$ at the end. Note that according to our results, constants c and λ depend on the value of r. As for the results of GREEDY and MAESTRO, we can notice note that both are, on average, above $\frac{|T|}{|P|} + 1$ and, except sometimes for $|T| = |P|$, never optimal. In general, there is a clear gain when using an optimal static strategy in place of a dynamic one (MAESTRO strongly relies on its second wave, that is purely dynamic, when $|T| > |P|$).

6.2.2 Communications

Let us now focus on the communication metric, *i.e.* the case where non-local tasks are allowed. Unlike for the makespan metric, there are very few theoretical studies that we can relate to this problem. On the other hand, we proved the optimality of BESTASSIGNMENT in Sections 5.2 and 5.3. Second, Berenbrink et al. [24] introduce the number of "holes" in an assignment, that corresponds to $\sum_{p_i \in P} \max(0, \left\lceil \frac{|T|}{|P|} \right\rceil - |\{t_j \text{ assigned to } p_i\}|)$. When $\frac{|T|}{|P|}$ is an integer, this is equal to the load imbalance. Berenbrink et al. prove that in the case of BALLS-IN-BINS processes with multiple choices (or, as proven in Section 4, in the case of GREEDY), the above metric is bounded by a linear function in $|P|$. At last, let us notice that an assignment of maximal degree $\frac{|T|}{|P|}$ has a load imbalance of 0. Thus, we know that there exists a constant c such that, with high probability, there exists a solution with no communications if $|T| \geq c|P| \log |P|$.

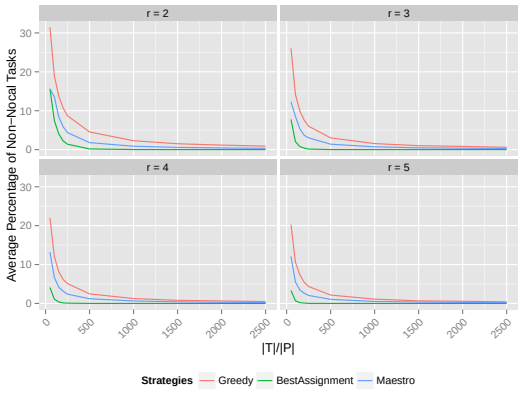


Fig. 7: Results for the communication metric and homogeneous settings with $|P| = 50$.

The results of our simulations are depicted on Figure 7. Like for the makespan metric there are several phases for the BESTASSIGNMENT strategy. In a first phase, there is not always a solution without communications, *i.e.* there is not

always a solution of MAKESPAN-MAPREDUCE with maximal degree equal to $\frac{|T|}{|P|}$ In a second phase, BESTASSIGNMENT manages to completely avoid communications if there are enough tasks with respect to the number of processors. Note that even in the first phase, the number of non-local tasks is below 15% in the worst case and in general below 5% as soon as $r \geq 2$.

Like for the makespan metric, there is clear hierarchy between GREEDY and MAESTRO. MAESTRO performs slightly better, in particular for small number of tasks per processor whereas GREEDY can achieve more than 30% of non-local tasks. However, if MAESTRO and GREEDY are relatively close to BESTASSIGNMENT when the number of tasks per processor is large, the fraction of non-local tasks is always positive. This can be seen on Figure 8 and on Figure 6 In addition, $\frac{|T|}{|P|}$ has little impact on the performance of MAESTRO and GREEDY (except for small values) and their performance seems rather to be related to $|P|$ and r, what is consistent with the claims from Berenbrink et al. [24].

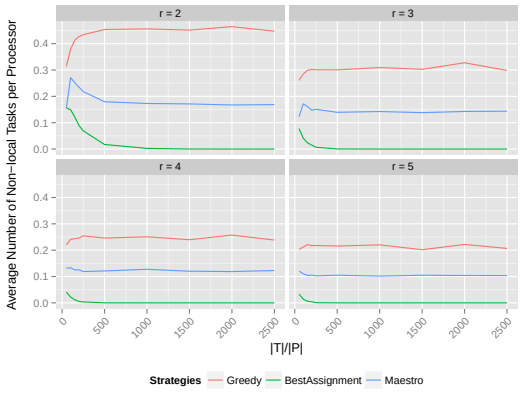


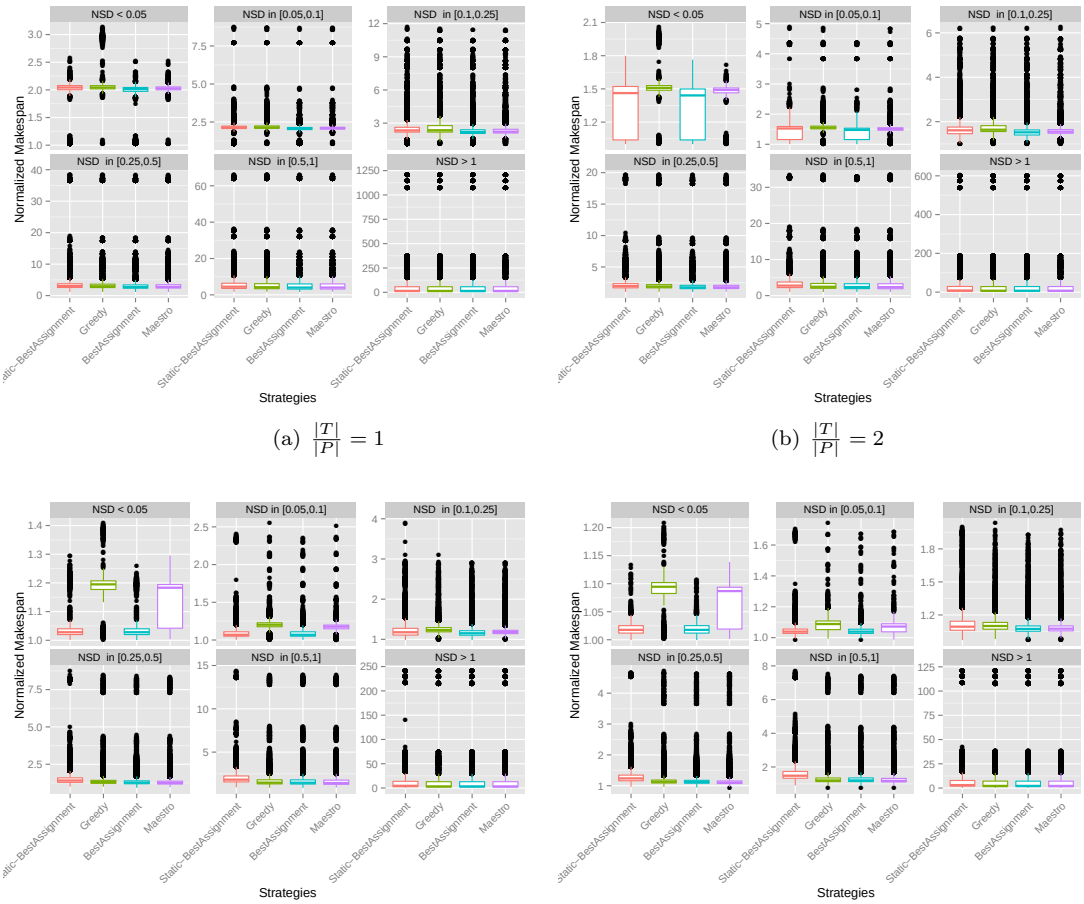
Fig. 8: Non-local tasks per processor for homogeneous settings and $|P| = 50$.

In a more general way, for both metrics, increasing r slightly improves the results and mainly benefits to BESTASSIGNMENT and the value $r = 3$, the default value using HDFS, seems to be a good trade-off between replication cost and communication-avoiding parallel execution.

6.3 Heterogeneous Settings

Let us now consider simulations on the traces presented in Section 6.1. As already mentioned, scheduling algorithms do not make use of the information on the execution time in order to emulate the online aspect of MapReduce scheduling. The replication ratio r is set to 3. Furthermore, for each job, we perform simulations for different values of $\frac{|T|}{|P|}$, $|T|$ being defined in the traces. At last, for each job, each $\frac{|T|}{|P|} \in \{1, 2, 5, 10\}$ and each strategy we perform 50 simulations.

6.3.1 Makespan



(a) $\frac{|T|}{|P|} = 1$ (b) $\frac{|T|}{|P|} = 2$

(c) $\frac{|T|}{|P|} = 5$ (d) $\frac{|T|}{|P|} = 10$

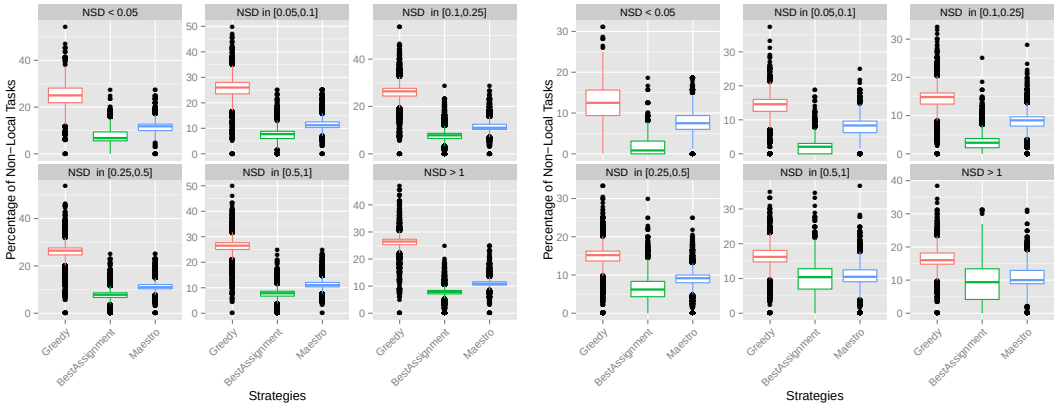
Fig. 9: Results for the makespan metric and heterogeneous settings

In order to provide a fair comparison between jobs with different average computational times, we use the normalized makespan, *i.e.* the actual makespan divided by the ideal one (the sequential makespan divided by the number of processors) as a metric. The results are depicted on Figures 9(a), 9(b), 9(c) and 9(d).

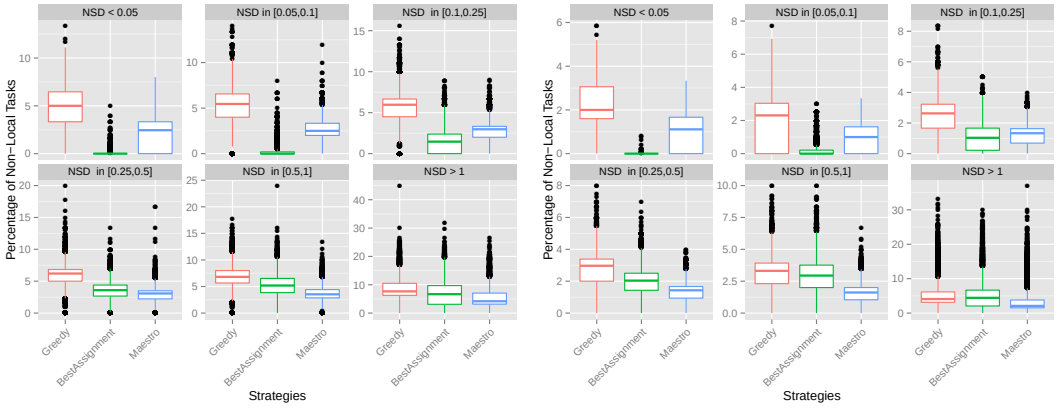
The behavior does not heavily depends on $\frac{|T|}{|P|}$. For small variance ($NSD \leq 0.1$), results are close to the ones in the homogeneous settings. Thus BestAssignment and Static-BestAssignment are more efficient than Greedy and Maestro. Otherwise, all three strategies with dynamic part have similar effi-

ciency and Static-BestAssignment suffers from the homogeneity assumption. Therefore, in any case, BestAssignment augmented with a dynamic part, appears to be the best strategy for small and medium variances and it performs as well as the other dynamic strategies for large variances.

6.3.2 Communications



(a) $\frac{|T|}{|P|} = 1$ (b) $\frac{|T|}{|P|} = 2$



(c) $\frac{|T|}{|P|} = 5$ (d) $\frac{|T|}{|P|} = 10$

Fig. 10: Results for the communication metric and heterogeneous settings.

As for the homogeneous settings we use the percentage of non-local tasks as a metric. Results can be found on Figure 10(a), 10(b), 10(c) and 10(d). Let us consider the case $|T| = |P|$ (Figure 10(a)). For all strategies, no processor can be idle as long as there exists one unprocessed task. Therefore, each processor

processes exactly one task (if the assignment gives no tasks to a processor, this one will automatically steal one to another processor). Thus, the heterogeneity of computation times has no effect here on the communication cost. Thus, for all standard deviation, BestAssignment performs well with about 7.5% of non-local tasks against 11.4% for Maestro and 25.8% for Greedy. Note that the small differences that appear between the results of two classes with different standard deviations might probably come from the repartition of the value $|T|$ (the number of tasks of a job, which, according to the results with homogeneous settings, impacts the communication ratio, even with $|T| = |P|$) inside each class, which differs from a class to another.

When $\frac{|T|}{|P|} > 1$, the results are not as one-sided. First, as expected, lower variances favors better BestAssignment. The two most dynamic strategies, Greedy and Maestro, results are less impacted by heterogeneity, so that the gap between static and dynamic strategies decreases when heterogeneity increases. For instance, in the case $\frac{|T|}{|P|} = 5$ (Figure 10(c)), for $NSD < 0.1$, BestAssignment results are close to those achieved in the homogeneous setting, with on average 0.13% of non-local-tasks whereas the fraction of non-local tasks reaches 2.61% with Maestro and 4.75% with Greedy. However the gap decreases for $NSD \in [0.1, 0.25[$ (1.51% against 2.67%) and finally, for greater NSD, Maestro performs better (5.13% for BestAssignment against 3.96% for Maestro). Note that jobs with NSD below 0.25 represent slightly more than half of the jobs.

The other important factor on the efficiency of the different strategies is the ratio $\frac{|T|}{|P|}$.Clearly, strategies produce less communications when the ratio $\frac{|T|}{|P|}$ increases as already noticed in the homogeneous case. Therefore, as expected, when the number of tasks per processor is large and when the heterogeneity is important, the interest of relying on a static efficient strategy like BestAssignment decreases. Indeed, during the first phase, BestAssignment tries to assign to each processor exactly $\frac{|T|}{|P|}$ tasks, *i.e.* the expected number of tasks to complete, whereas with Maestro or Greedy, tasks are assigned at runtime based on the state of the platform. Therefore, a high number of tasks and a high heterogeneity favor dynamic strategies such as Maestro or Greedy.

For example, for $\frac{|T|}{|P|} = 1$ and $\frac{|T|}{|P|} = 2$, BestAssignment is the most effective strategy even for large standard deviation. However, as stated above, for $\frac{|T|}{|P|} = 5$ there are cases where BestAssignment is less efficient than Maestro and for $\frac{|T|}{|P|} = 10$ BestAssignment is even challenged by Greedy in the case $NSD > 1$. In order to see what to this evolution keeps going, we propose on Figure 11 an illustration of the case $\frac{|T|}{|P|} = 50$. In this case, the only case where BestAssignment is the indisputably better strategy is when $NSD < 0.05$, that represents few cases. However, even in the case where BestAssignment is no the best strategy, its performance is not so bad. For example, in the case $\frac{|T|}{|P|} = 50$, BestAssignment is on average below 2% of non local tasks. More generally, a percentage of non-local tasks below 10% is a good expectation (the only time the average is above 10% is for $\frac{|T|}{|P|} = 2$ and $NSD > 0.5$, but in this

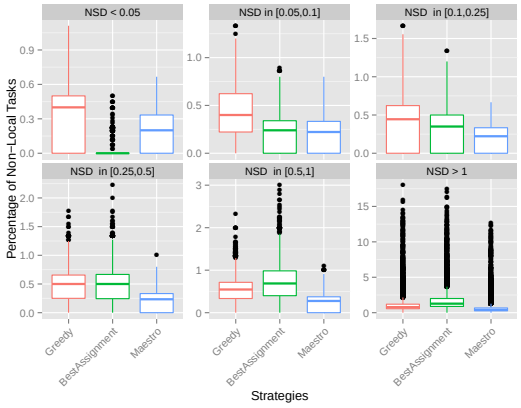


Fig. 11: Results for the communication metric for $\frac{|T|}{|P|} = 50$.

case MAESTRO is even worse). For example, for $\frac{|T|}{|P|} = 1$, BESTASSIGNMENT is the most effective strategy even for large standard deviation. However, as stated above, for $\frac{|T|}{|P|} = 5$ there are cases where BESTASSIGNMENT is less efficient than MAESTRO.

7 Conclusion and Perspectives

In this paper, we investigate the use of sophisticated strategies to allocate tasks to resources in the context of Map tasks. We prove that if all map tasks of the same job have the same duration, then it is possible to derive an optimal strategy BESTASSIGNMENT that can be efficiently implemented and whose complexity is acceptable in practice. To assess the performance of BESTASSIGNMENT in the more general case of tasks with heterogeneous durations, we rely on simulations based on an actual trace of MapReduce jobs in a production cluster and we compare it to best dynamic strategies of the literature, such as MAESTRO. We prove that except for extremely large heterogeneity and number of tasks per processor, BESTASSIGNMENT outperforms MAESTRO, what shows the interest of designing efficient static scheduling strategies, even in a dynamic setting. This work opens many perspectives in this direction. Indeed, predicting the performance of individual tasks and communications becomes more and more difficult, both for HPC and BigData applications, what lead to the design and massive use of dynamic schedulers, that take their decision at runtime, based on a correct state of the platform but a poor knowledge of the rest of the work to be processed, what sometimes lead to poor performance. The work presented here shows that in the context of Map tasks, injecting static knowledge on data distribution can help to make better decisions at runtime.

Annexe

In this section we propose an alternative proof of Corollary 6.

Theorem (Recall of Corollary 6)**.** *Let $G = (P, T, E)$ be a bipartite graph such that the degree of every element of T is at least 2. Then, with high probability there exists an assignation with maximal degree smaller or equal to $1 + \left\lceil \frac{|T|}{|P|} \right\rceil$.*

The analysis technique we propose to establish this result is inspired by the seminal paper of Walkup [18]. We also rely on classical results on bipartite graph's matching from the literature, such as Hall's Theorem.

Theorem 21 (Hall's Theorem [34])**.** *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 22. *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 13, 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 21) to $\forall T' \subseteq T$, $|T'| \leq |v_{G^m}(T')| = m|v_G(T')|$. □

Probability of Existence of a Quasi-Perfect Assignment

In what follows, we rely on Lemma 22 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 22 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

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

$$\leq \sum_{q=0}^{p}\sum_{i=1}^{m}\binom{p}{q}\binom{n}{mq+i}\left(\frac{q}{p}\right)^{r(mq+i)}\left(1-\left(\frac{q}{p}\right)^{r}\right)^{n-mq-i}.$$

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.

Theorem 23. *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 23 is a direct consequence of Theorem 24 and Theorem 25, that consider respectively the case $\alpha = 1$ and the case $\alpha < 1$. □

Theorem 24. *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,

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

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 24. □

Theorem 25. *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 26.

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

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 26. □

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 27 and the case $mq + 1 < n$ is considered in Lemmas 28, 29, 33.

Lemma 27. *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,

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

$$= \left(\frac{1}{1+\alpha}\right)^{\frac{\alpha n}{1+\alpha}} \left(\frac{1}{1+\alpha}\right)^{-\frac{\alpha}{1+\alpha}+\frac{1}{2}} \left(1-\frac{1}{n}\right)^{\frac{\alpha n}{1+\alpha}-\frac{\alpha}{1+\alpha}+\frac{1}{2}}$$
$$= O\left(\frac{1}{(1+\alpha)^{\frac{\alpha n}{1+\alpha}}}\right)$$

and

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

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 27. □

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 28 provides a first bound on this sum.

Lemma 28. *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 $mq+1<n$,

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

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

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

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

and

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

what completes the proof of Lemma 28. □

The following Lemma 29 proves that we can concentrate on the detailed analysis of a single function f (as defined below).

Lemma 29. *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

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

and

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

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 30, 31 and 32).

Claim 30. $\exists K \in \mathbb{R}, \forall \alpha$ *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 30. □

Claim 31. $\forall \alpha$ *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 $(\alpha+1)x\left(\ln(x)-\ln\left(\frac{1}{n(\alpha+1)}+x\right)\right)$.

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

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,

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

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

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

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,

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

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 31. □

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 32. *$\forall x$ 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 32. □

Together, Claims 30, 31 and 32 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 29. □

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

Lemma 33.

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

Proof. In order to prove claimed result, we rely on two intermediate claims (Claim 34 and 35) 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 34. $\forall x \in]0,1[,\ f(x) < 0$.

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,

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

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,

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

$g'(x) = \ln(x) + 1 + (e-1) - \left(4e - \frac{17}{4}\right)x + \left(9e + \frac{9}{8}\right)x^2$ 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): $x \in [\frac{17}{20}, 1[$

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

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

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

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 [35] 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 f1(mpfi_t res,mpfi_t x){
 //computation of (x+1)*ln(x)
  mpfi_t y,z ;
  mpfi_init(y) ; mpfi_init(z) ;
  mpfi_add_si(y,x,1) ;
  mpfi_log(z,x) ;
  mpfi_mul(res,y,z) ;
  mpfi_clear(y) ; mpfi_clear(z) ;
}
```

```
void f2(mpfi_t res,mpfi_t x){
 //computation of -(1-x)^2/x ln(1-x)
  mpfi_t y,z ;
  mpfi_init(y) ; mpfi_init(z) ;
  mpfi_si_sub(y,1,x) ;
  mpfi_mul(y,y,y) ;
  mpfi_div(y,y,x) ;
  mpfi_si_sub(z,1,x) ;
  mpfi_log(z,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
    //non negative
    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 34. □

Claim 35. *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 35. □

Without loss of generality we can assume that the x_0 from Claim 35 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 34, $f(x_1) < 0$ and $\exists K' < 0$ such that $\forall x \in \left[x_0, \frac{1}{1+\alpha}\right]$, $f(x) \leq K'$. Thus,

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

because $\alpha - 1 + \epsilon \leq 0$, what ends the proof of Lemma 33 □

With Lemmas 27, 29 and 33 we obtain

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

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

what achieves the proof of Theorem 25, and therefore the proof of Theorem 23. □

Theorem 23 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 36.

Theorem 36. *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 25 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 25. This achieves the proof of Theorem 36 and thus Corollary 6. □

References

[1] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, 2012.

[2] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An Analysis of Traces from a Production MapReduce Cluster," in *CCGrid'10*. IEEE/ACM, 2010, pp. 94–103.

[3] Z. Guo, G. C. Fox, and M. Zhou, "Investigation of Data Locality in MapReduce," in *International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE/ACM, 2012, pp. 419–426.

[4] D. Borthakur, "HDFS Architecture Guide," *HADOOP http://hadoop. apache. org/common/docs/current/hdfs design. pdf*, p. 39, 2008.

[5] 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 *European Conference on Computer Systems (EuroSys)*. ACM, 2010, pp. 265–278.

[6] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu, "Maestro: Replica-Aware Map Scheduling for MapReduce," in *CCGrid'12*. IEEE/ACM, 2012, pp. 435–442.

[7] M. Chowdhury and I. Stoica, "Coflow: A Networking Abstraction for Cluster Applications," in *Workshop on Hot Topics in Networks (HotNets)*, 2012, pp. 31–36.

[8] ——, "Efficient Coflow Scheduling Without Prior Knowledge," *Computer Communication Review (CCR)*, vol. 45, no. 5, pp. 393–406, 2015.

[9] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the Total Weighted Completion Time of Coflows in Datacenter Networks," in *Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, 2015, pp. 294–303.

[10] M. Hammoud and M. F. Sakr, "Locality-Aware Reduce Task Scheduling for MapReduce," in *International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2011, pp. 570–576.

[11] J. Tan, S. Meng, X. Meng, and L. Zhang, "Improving ReduceTask data locality for sequential MapReduce jobs," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 1627–1635.

[12] Q. Xie and Y. Lu, "Degree-Guided Map-Reduce Task Assignment with Data Locality Constraint," in *International Symposium on Information Theory (ISIT)*. IEEE, 2012, pp. 985–989.

[13] 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 *International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 1609–1617.

[14] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters," in *Symposium on Operating Systems Principles (SOSP)*. ACM, 2009, pp. 261–276.

[15] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 2015.

[16] I. Gog, M. Schwarzkopf, A. Gleave, R. N. Watson, and S. Hand, "Firmament: Fast, Centralized Cluster Scheduling at Scale," in *Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX, 2016, pp. 99–115.

[17] P. ErdÃűs and A. RÃľnyi, "On Random Matrices," *Studia Scientiarum Mathematicarum Hungarica*, vol. 8, pp. 455–461, 1964.

[18] D. W. Walkup, "Matchings in Random Regular Bipartite Digraphs," *Discrete Mathematics*, vol. 31, no. 1, pp. 59–64, 1980.

[19] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs," *SIAM Journal on Computing (SICOMP)*, vol. 2, no. 4, pp. 225–231, 1973.

[20] A. Goel, M. Kapralov, and S. Khanna, "Perfect Matchings in $O(n \log n)$ Time in Regular Bipartite Graphs," *SIAM Journal on Computing (SICOMP)*, vol. 42, no. 3, pp. 1392–1404, 2013.

[21] J. Langguth, F. Manne, and P. Sanders, "Heuristic Initialization for Bipartite Matching Problems," *Journal of Experimental Algorithmics (JEA)*, vol. 15, pp. 1.3:1.1–1.3:1.22, Mar. 2010.

[22] F. DufossÃľ, K. Kaya, and B. UÃğar, "Two Approximation Algorithms for Bipartite Matching on Multicore Architectures," *Journal of Parallel and Distributed Computing*, vol. 85, pp. 62 – 78, 2015.

[23] M. Raab and A. Steger, "Balls into Bins: A Simple and Tight Analysis," in *Randomization and Approximation Techniques in Computer Science (RANDOM)*. Springer, 1998, pp. 159–170.

[24] P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin, "On weighted Balls-into-Bins Games," *Theoretical Computer Science (TCS)*, vol. 409, no. 3, pp. 511–520, 2008.

[25] M. Mitzenmacher, "The Power of Two Choices in Randomized Load Balancing," *Transactions on Parallel and Distributed Systems (TPDS)*, vol. 12, no. 10, pp. 1094–1104, 2001.

[26] A. W. Richa, M. Mitzenmacher, and R. Sitaraman, "The Power of Two Random Choices: A Survey of Techniques and Results," *Combinatorial Optimization*, vol. 9, pp. 255–304, 2001.

[27] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking, "Balanced Allocations: The Heavily Loaded Case," in *Symposium on Theory of Computing (STOC)*. ACM, 2000, pp. 745–754.

[28] Y. Peres, K. Talwar, and U. Wieder, "The (1+ β)-Choice Process and Weighted Balls-into-Bins," in *Symposium on Discrete Algorithms (SODA)*. SIAM, 2010, pp. 1613–1619.

[29] P. Sanders, S. Egner, and J. Korst, "Fast Concurrent Access to Parallel Disks," *Algorithmica*, vol. 35, no. 1, pp. 21–55, 2003.

[30] A. Czumaj, C. Riley, and C. Scheideler, "Perfectly Balanced Allocation," *Approximation, Randomization, and Combinatorial Optimization (APPROX)*, pp. 240–251, 2003.

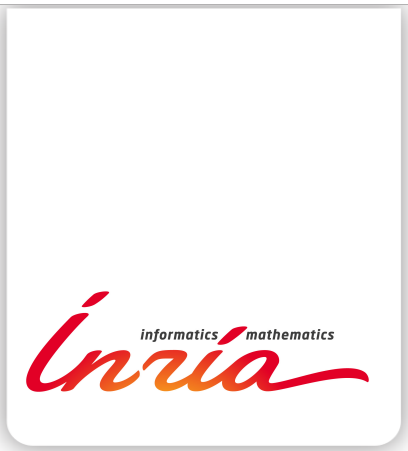
[31] P. Sanders, "Algorithms for Scalable Storage Servers," in *International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*. Springer, 2004, pp. 82–101.

[32] J. A. Cain, P. Sanders, and N. Wormald, "The random graph threshold for k-orientiability and a fast algorithm for optimal multiple-choice allocation," in *Symposium on Discrete Algorithms (SODA)*. SIAM, 2007, pp. 469–476.

[33] C. Berge, "Two theorems in Graph Theory," *Proceedings of the National Academy of Sciences*, vol. 43, no. 9, pp. 842–844, 1957.

[34] P. Hall, "On representatives of subsets," *Journal of the London Mathematical Society*, vol. s1-10, no. 1, pp. 26–30, 1935.

[35] N. Revol and F. Rouillier, "Motivations for an arbitrary precision interval arithmetic and the mpfi library," *Reliable computing*, vol. 11, no. 4, pp. 275–290, 2005.



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