Distributed Algorithm for a Color Assignment on Asynchronous Rings

Gianluca De Marco[1], Mauro Leoncini[2,3], Manuela Montangero[2,3]

[1]Dipartimento di Informatica e Applicazioni
Università di Salerno
Via Salvador Allende 1
84081 Baronissi (SA), Italy
demarco@dia.unisa.it

[2]Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905/b
41100 Modena, Italy
leoncini@acm.org
montangero.manuela@unimo.it

[3]Istituto di Informatica e Telematica - CNR
Via Moruzzi 1
56124 Pisa, Italy

Abstract

*We study a version of the β-assignment problem [3] on asynchronous rings: consider a set of items and a set of m colors, where each item is associated to one color. Consider also n computational agents connected by an asynchronous ring. Each agent holds a subset of the items, where initially different agents might hold items associated to the same color. We analyze the problem of distributively assigning colors to agents in such a way that (a) each color is assigned to one agent and (b) the number of different colors assigned to each agent is minimum. Since any color assignment requires that the items be distributed according to it (*e.g. *all items of the same color are to be held by only one agent), we define the cost of a color assignment as the amount of items that need to be moved, given an initial allocation. We first show that any distributed algorithm for this problem on the ring requires a communication complexity of $\Omega(n \cdot m)$ and then we exhibit a polynomial time distributed algorithm with message complexity matching the bound, that determines a color assignment with cost at most $(2 + \epsilon)$ times the optimal cost, for any $0 < \epsilon < 1$.*

Work supported in part by the European RTN Project under contract HPRN-CT-2002-00278, COMBSTRU.

1. Introduction

We consider the following problem. Let $\mathcal{A} = \{a_0, \ldots a_{n-1}\}$ be a set of n agents connected on an asynchronous ring and let $\mathcal{C} = \{c_0, \ldots, c_{m-1}\}$ be a set of m colors. Colors are used to color items held by agents (each item can be colored only with one color), so let $Q_{j,i} \geq 0$ be the number of items with color c_j initially held by agent a_i, for every $j = 0, \ldots, m-1$, and for every $i = 0, \ldots, n-1$.

Definition 1 (Balanced Coloring) *A Balanced Coloring is an assignment $\pi : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, n-1\}$ of the m colors to the n agents in such a way that:*

- *for every color c_j, there is exactly one agent a_i such that $\pi(j) = i$;*
- *for every agent a_i, $\lfloor \frac{m}{n} \rfloor \leq |\{c_j \mid \pi(j) = i\}| \leq \lceil \frac{m}{n} \rceil$.*

Observe that any Balanced Coloring assigns almost the same number of colors to each agent. When m is a multiple of n, then each agent receives exactly the same number of colors.

Definition 2 (Distributed Balanced Color Assignment Problem) *The cost of a Balanced Coloring $\pi : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, n-1\}$ is defined as*

$$Cost(\pi) = \sum_{j=0}^{m-1} \sum_{\substack{i=0,\\ i\neq\pi(j)}}^{n-1} Q_{j,i}. \quad (1)$$

The Distributed Balanced Color Assignment Problem consist of designing a distributed algorithm that finds a Balanced Coloring of minimum cost.

The cost of the optimal assignment will be denoted by $Cost_{OPT}$. The *approximation ratio* of a suboptimal algorithm A is the quantity $\frac{Cost_{\mathtt{A}}}{Cost_{OPT}}$, where $Cost_{\mathtt{A}}$ is the cost of the solution computed by A.

Informally we can restate the problem in the following way: we are given a set of computational agents connected by an asynchronous ring, and a set of items, each associated to one color from a given set. Initially each agent holds a set of items and items with the same color may be held by different agents (*e.g.* see Fig 1.(a)). We wish the agents to agree on an assignment of colors to agents in such a way that each color is assigned to one agent only, balancing the number of colors that are assigned to agents. That is why we call the assignment *balanced assignment* (in Fig 1.(b) and Fig 1.(c) two examples are shown). Among all such assignments, we seek the one that minimizes the total number of items that agents have to collect from other agents in order to satisfy the constraint; *i.e.*, each agent collects all the items colored with the colors that are assigned to it. For example, agent a_0 in Fig 1.(b) is assigned colors ∇ and $\spadesuit$, and therefore needs just to collect four items colored ∇, since no other agent has items colored $\spadesuit$.

The scenario defined above may arise in many practical situations in which a set of agents (distributed crawlers, sensor networks, etc...) independently search a common space and then have to reorganize the retrieved data (items) according to a given classification by topics (colors). In these cases, determining a distributed balanced color assignment guarantees specialization and balanced computational load of agents. The choice of the ring topology is motivated (beside the fact that it is a very popular architecture in distributed computing research) by the observation that in a number of different network settings (*e.g.*, peer-to-peer networks [16] and sensor networks [4]) nodes organize themselves in a virtual ring on top of an unstructured topology.

The model. We assume that the agents in $\mathcal{A} = \{a_0, \ldots, a_{n-1}\}$ are connected by an asynchronous ring; *i.e.*, for each $i = 0, \ldots, n-1$, agent a_i can communicate only with its two neighbors $a_{(i+1) \bmod n}$ and $a_{(i-1) \bmod n}$. We also assume that each agent knows n (the number of agents), $\mathcal{C}$ (the set of colors), and either the parameter $p = \max_{0\leq j\leq m-1} \sum_{i=0}^{n-1} Q_{j,i}$ (the maximum number of items having the same color), or an upper bound to p.

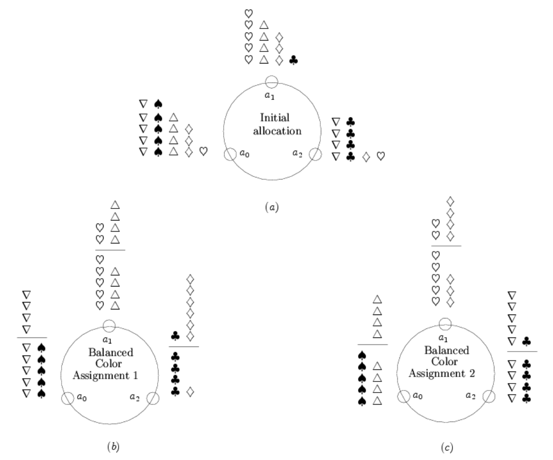

Figure 1. Three agents: a_0, a_1, a_2, and six colors: $\nabla, \diamondsuit, \heartsuit, \triangle, \spadesuit, \clubsuit$. (a) is the initial allocation, while (b) and (c) are two possible balanced color assignments. Items above the line are those that the agent collects from the others. Therefore their total number is the cost of the assignment. The assignment in (b) costs $(4\times\nabla)+(2\times\heartsuit+4\times\triangle)+(1\times\clubsuit+6\times\diamondsuit) = 17$ items, while the assignment in (c) costs $(4\times\triangle)+(2\times\heartsuit+4\times\diamondsuit)+(5\times\nabla+1\times\clubsuit) = 16$ items.

We measure message complexity in the standard way (e.g. [11, 13]), i.e., we assume that messages of bit length at most $c \log n$, for some constant c, (called *basic messages*) can be transmitted at unit cost. Message can carry at most a constant number of agent IDs. Non basic messages of length L are allowed, and we charge a cost $c_1 \lceil L/\log n\rceil$ for their transmission, for some constant c_1.

Our results. The goal of this paper is to analyze the efficiency under which we can solve the Distributed Balanced Color Assignment problem. In Section 2 we show the equivalence of the problem with the weighted β-assignment problem in a centralized setting [3]. In Section 3 we give a $\Omega(mn)$ lower bound on the message complexity to determine a solution to the problem. In Section 4 we present an algorithm that finds a feasible solution to the problem that matches the bound on message complexity and in Section 5 we show that the proposed algorithm computes a feasible solution whose cost is only a factor of $(2+\epsilon)$ off the optimal one, for any $0 < \epsilon < 1$. Finally, we show that we can find a feasible solution with approximation ratio two at the expenses of a slight increase of message complexity.

2. Preliminaries

It is easy to see that when $m = n$, our problem is equivalent to a maximum weight perfect matching problem on a bipartite graph. On the other hand, when $m \geq n$, our problem reduces to the weighted β-assignment problem. The class of β-assignment problems has been introduced by Chang and Lee [3], in the context of the problems of assigning jobs to workers, in order to incorporate the problem of balancing the work load that is given to each worker. In the weighted β-assignment problem one aims at minimizing the maximum number of jobs assigned to each worker. The interested reader can find useful references on these problems, their complexity, and related approximation issues in [1, 2, 10, 14, 15].

We associate to agents and colors the complete bipartite graph on $n+m$ vertices, which we denote by $G = (\mathcal{C}, \mathcal{A}, \mathcal{C} \times \mathcal{A})$. We add weights to G as follows: the weight of the edge joining agent a_i and color c_j is $Q_{j,i}$.

Case $m = n$. Given a graph (V, E), a perfect matching is a subset M of edges in E such that no two edges in M share a common vertex and each vertex of V is incident to some edge in M. When edges of the graph have an associated weight, then a maximum weight perfect matching is a perfect matching such that the sum of the weights of the edges in the matching is maximum.

Lemma 3 *A maximum weight perfect matching on G is a minimum cost solution to the balanced color assignment problem.*

Proof. Given a perfect matching $E \subseteq \mathcal{E} = \mathcal{C} \times \mathcal{A}$ on G, for every $(c_j, a_i) \in E$ we assign color c_j to agent a_i. As G is complete and E is a perfect matching on G, every color is assigned to one and only one agent and vice-versa. Moreover, the cost of a balanced color assignment, given any matching E, can be written as $\sum_{e \in \mathcal{E} \setminus E} w(e)$, and this expression achieves its minimum when E is a maximum weight perfect matching. ■

Finding matchings in graphs is one of the most deeply investigated problems in Computer Science and Operations Research (see [12] for a comprehensive description of the different variants, theoretical properties, and corresponding algorithms). The best known algorithm to find a perfect matching in a bipartite graph is due to Hopcroft and Karp [8], and runs in $O\left(|E|\sqrt{|V|}\right)$ time, where V and E denote the vertex and edge sets, respectively. In our case this bound reduces to $O(n^{5/2})$. The best known algorithm for finding a maximum weight perfect matching is the *Hungarian method*, due to Kuhn [9], which runs in time $O(n^3)$.

Case $m \geq n$. The β-assignment problem is defined on a bipartite graph $G = (S, T, E)$ where (S, T) is the bipartition of the vertex set. A β-assignment of S in G is a subset of the edges $X \subseteq E$ such that, given the graph $G' = (S, T, X)$, the degree of every vertex in S is exactly one. Let $\beta(X)$ be the maximum degree, in G', of vertices in T and let $\beta(G)$ be the minimum value of $\beta(X)$ among all possible β-assignments X. The weighted β-assignment problem consists of finding a β-assignment X with $\beta(X) = \beta(G)$ which maximizes the total weight of the edges in X. The following lemma is straightforward.

Lemma 4 *The balanced color assignment problem is a weighted β-assignment of $\mathcal{C}$ in the complete bipartite graph $G = (\mathcal{C}, \mathcal{A}, \mathcal{C} \times \mathcal{A})$, with $\beta(G) = m/n$.*

The fastest known algorithm to solve the weighted β-assignment problem is due to Chang and Ho [2] and runs in $O(\max\{|S|^2|T|, |S||T|^2\})$ time, which in our case gives the bound $O(m^2 n)$. While the maximum perfect matching problem (with its variants) has been widely investigated in the distributed setting (see [6, 7]), no distributed results are known for the weighted β-assignment problem.

A brute force approach. A brute force distributed solution to the problem can be obtained by asking all the agents to send their color information to one specific agent (a priori chosen or elected as the leader of the ring); such an agent will then solve the problem sequentially and send the solution back to all the other agents. The factor dominating the message complexity of the algorithm above is the information collecting stage, even if leader election is considered. Indeed, each agent sends $O(m)$ non-basic messages, each corresponding to $O(\log p / \log n)$ basic messages, through $O(n)$ links, on the average. This results in a message complexity of $O(mn^2 \log p / \log n)$. On the other hand, we might think of an algorithm in which each agent selects the correct number of colors basing its choice just on local information (*e.g.*, its label). This requires no communication at all, but, even if we are able to prove that the agents agree correctly on a balanced coloring, we have no guarantee on how good the solution is. Can we do better? In the next section we show a $\Omega(mn)$ lower bound on the message complexity of the problem, and in Section 4 we describe a $O(n \log p)$ time distributed algorithm matching the lower bound on message complexity whose cost is at most $(2+\epsilon)$ times the optimal, for any $0 < \epsilon < 1$.

For the sake of presentation and without loss of generality, we will assume that m is a multiple of n, so that exactly $\frac{m}{n}$ colors are assigned to each agent. In general, $\lfloor \frac{m}{n} \rfloor$ colors will be assigned to $\left(\lfloor \frac{m}{n} \rfloor + 1\right) n - m$ agents, and $\lfloor \frac{m}{n} \rfloor + 1$ color to $m - \lfloor \frac{m}{n} \rfloor n$ agents. Our results can be easily extended to handle the general case.

3. Lower bound on message complexity

Before formally proving the lower bound we make some observations to clarify the nature of the problem. Assume agents can agree on a balanced coloring without communicating, *i.e.*, each agent is able to select the colors that are assigned to it autonomously without "making errors". In such a situation, the choice of agent a_i must be independent of the particular values of the $Q_{j,i}$'s, for otherwise, the agent cannot be sure that no other agent choose the same colors (suppose the two have exactly the same initial distribution of items). On the other hand, an assignment that is done without taking the initial distribution of items to agents into consideration can not give any guarantee on the cost. Consider, for example, the case of an initial distribution such that each agent has items of all colors except the ones that it selects. This leads to a cost that is equal to the total amount of items. Any assignment in which at least one agent is assigned one of the colors it initially holds has a smaller cost. We conclude that any solution in which the agents' selection is done according to the initial distribution of items requires some communication among agents. In the rest of this section we will prove that agents need to exchange $\Omega(mn)$ messages to solve the problem.

Assume w.l.o.g. that n is even and let $m = (nt)/2$, for some integer t. For any agent a_i, let $a_{i'}$ denote the agent at maximum distance from a_i in the ring.

Let $\mathcal{P}$ be the set of all possible partitions of the set of colors $\mathcal{C} = \{1, \ldots, m\}$ into $n/2$ subsets of the same cardinality $t = 2m/n$. Consider the set of initial allocations $\mathcal{I}$ in which, for any $i = 0, \ldots, n/2-1$ and some partition $\{\mathcal{C}_0, \ldots, \mathcal{C}_{n/2-1}\} \in \mathcal{P}$, agents a_i and $a_{i'}$ hold only items of the colors in $\mathcal{C}_i = \{i_1, \ldots, i_t\}$, where $i_1 < i_2 < \ldots < i_t$. It is easy to see that any solution in which any a_i is assigned a color that it does not initially hold is worse than any solution in which a_i and $a_{i'}$ arbitrarily choose among the colors in $\mathcal{C}_i$. Suppose that there is at least one a_i that is assigned a color $j \notin \mathcal{C}_i$, then there must be a_k, with $k \neq i, i'$, and $h \in \mathcal{C}_i$ (hence $h \notin \mathcal{C}_k$) such that h is assigned to a_k. Let us calculate the cost of this solution:

$$\begin{aligned} Cost_1 &= \Gamma + \sum_{\substack{w=0,\\ w\neq i}}^{n-1} Q_{j,w} + \sum_{\substack{w=0,\\ w\neq k}}^{n-1} Q_{h,w} \\ &= \Gamma + \sum_{\substack{w=0,\\ w\neq i}}^{n-1} Q_{j,w} + Q_{h,i} + Q_{h,i'}, \end{aligned}$$

where Γ is the contribution to the cost given by all colors different from j and h.

Consider a solution that differs from the previous one only by the fact that h is assigned to a_i and j to a_k. Obviously, the contribution given by all other colors remains the same and the cost of this solution is:

$$Cost_2 = \Gamma + \sum_{\substack{w=0,\\ w\neq k}}^{n-1} Q_{j,w} + \sum_{\substack{w=0,\\ w\neq i}}^{n-1} Q_{h,w} = \Gamma + \sum_{\substack{w=0,\\ w\neq k}}^{n-1} Q_{j,w} + Q_{h,i'},$$

and

$$Cost_1 - Cost_2 = Q_{j,k} - Q_{j,i} + Q_{h,i} = Q_{j,k} + Q_{h,i} \geq 0$$

as $Q_{j,i} = 0$. Hence, for any instance in $\mathcal{I}$ we can always assume that an optimal solution assigns colors in $\mathcal{C}_i$ to the pair $(a_i, a_{i'})$, for all i. We now concentrate on one such a pair.

Partition set $\mathcal{C}_i = \{i_1, \ldots, i_t\}$ into two sets $\mathcal{C}'$ and $\mathcal{C}''$ of cardinality $t/2$ each (observe that, as we assume m is a multiple of n and since $t/2 = m/n$, then t has to be even). Consider the following instance for the pair $(a_i, a_{i'})$:

$\mathcal{I}_1$:

$Q_{j,i} = 2^j$		for each $j \in \mathcal{C}_i$
$Q_{j,i'} = Q_{j,i}$		if $j \in \mathcal{C}'$
$Q_{j,i'} = Q_{j',i}$	where $j' \in \mathcal{C}' \cup \{i_t\}$ is the smallest index s.t. $j \leq j'$	if $j \in \mathcal{C}''$

That is: a_i has a different quantity of each color, which depends on the color index. Concerning agent $a_{i'}$: for each $j \in \mathcal{C}'$, then $Q_{j,i'}$ is exactly $Q_{j,i}$, *i.e.*, the same amount that agent a_i has of that color. For each index $j \in \mathcal{C}''$, consider the smallest index $j' \in \mathcal{C}'$ that

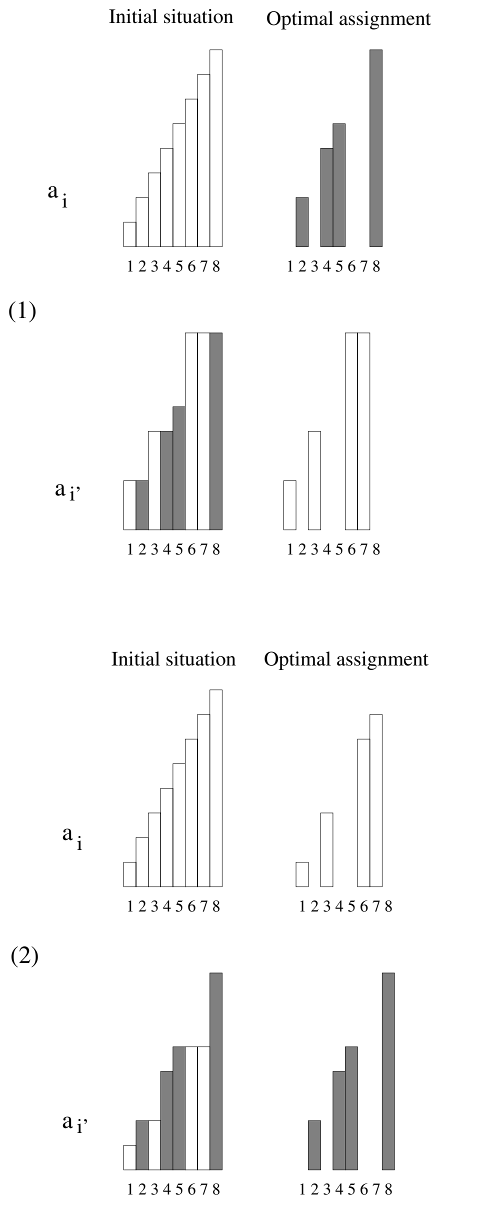

Figure 2. Example: $\mathcal{C}' = \{2, 4, 5, 8\}, \mathcal{C}'' = \{1, 3, 6, 7\}$ **and instances $\mathcal{I}_1$ and $\mathcal{I}_2$ (resp. in (1) and (2)); agent a_i has knowledge of $a_{i'}$ colors whose indices are in $\mathcal{C}'$ only (in gray). Even if a_i knows the exact position of these colors in the ordering of $a_{i'}$ colors it is not able to distinguish between instances $\mathcal{I}_1$ and $\mathcal{I}_2$. Observe that these two instances have complementary optimal solutions.**

is greater than j (if this does not exist, then $j' = i_t$). Then $Q_{j,i'}$ is exactly $Q_{j',i}$ (see Fig 2.1 - initial situation - for an example). Observe that, by construction, for any $j \in \mathcal{C}''$ we have that $Q_{j,i'} > Q_{j,i}$. The following lemma holds (see Fig 2.1 - optimal assignment - for an example).

Lemma 5 *There is only one optimal solution for instance $\mathcal{I}_1$: assign colors in $\mathcal{C}'$ to a_i and colors in $\mathcal{C}''$ to $a_{i'}$.*

Proof. Let us first compute the cost of the solution that assigns colors in $\mathcal{C}'$ to a_i and colors in $\mathcal{C}''$ to $a_{i'}$:

$$\begin{aligned} Cost &= \sum_{j \in \mathcal{C}'} Q_{j,i'} + \sum_{j \in \mathcal{C}''} Q_{j,i} \\ &= \sum_{j \in \mathcal{C}'} Q_{j,i} + \sum_{j \in \mathcal{C}''} Q_{j,i} \\ &= \sum_{j \in \mathcal{C}_i} Q_{j,i}. \end{aligned}$$

Consider any other partition of $\mathcal{C}_i$ into two sets, $\overline{\mathcal{C}'}$ and $\overline{\mathcal{C}''}$. Consider another solution that assigns $\overline{\mathcal{C}'}$ to a_i and $\overline{\mathcal{C}''}$ to $a_{i'}$ and let us compute the cost of this new solution:

$$\begin{aligned} \overline{Cost} &= \sum_{j \in \overline{\mathcal{C}'} \cap \mathcal{C}'} Q_{j,i'} + \sum_{j \in \overline{\mathcal{C}'} \cap \mathcal{C}''} Q_{j,i'} + \sum_{j \in \overline{\mathcal{C}''}} Q_{j,i} \\ &= \sum_{j \in \overline{\mathcal{C}'} \cap \mathcal{C}'} Q_{j,i} + \sum_{j \in \overline{\mathcal{C}'} \cap \mathcal{C}''} Q_{j,i'} + \sum_{j \in \overline{\mathcal{C}''}} Q_{j,i} \\ &= \sum_{j \in \mathcal{C}_i \setminus (\overline{\mathcal{C}'} \cap \mathcal{C}'')} Q_{j,i} + \sum_{j \in \overline{\mathcal{C}'} \cap \mathcal{C}''} Q_{j,i'} \\ &> \sum_{j \in \mathcal{C}_i \setminus (\overline{\mathcal{C}'} \cap \mathcal{C}'')} Q_{j,i} + \sum_{j \in \overline{\mathcal{C}'} \cap \mathcal{C}''} Q_{j,i} = Cost \end{aligned}$$

Since $\overline{\mathcal{C}'} \cap \mathcal{C}'' \neq \emptyset$ (otherwise the two partitions would coincide) and for any $j \in \mathcal{C}''$ we have $Q_{j,i'} > Q_{j,i}$. ■

Consider now a dual instance

$$\begin{array}{lll} \mathcal{I}_2 : & & \\ Q_{j,i} = 2^j & & \text{for each } j \in \mathcal{C}_i \\ Q_{j,i'} = Q_{j,i} & & \text{if } j \in \mathcal{C}' \\ Q_{j,i'} = Q_{j',i} & \text{where } j' \in \mathcal{C}' \cup \{i_1\} & \\ & \text{is the largest index} & \\ & \text{s.t. } j' \leq j & \text{if } j \in \mathcal{C}'' \end{array}$$

This instance differs from $\mathcal{I}_1$ for the values $Q_{j,i'}$ when $j \in \mathcal{C}''$. In this case consider the greater index $j' \in \mathcal{C}'$ that is smaller than j (if this does not exist, then $j' = i_1$). Then $Q_{j,i'}$ is exactly $Q_{j',i}$ (see Fig 2.2

- initial situation - for an example). Observe that by construction, for any $j \in \mathcal{C}''$ we have that $Q_{j,i'} < Q_{j,i}$. Analogously to the previous lemma we can prove the following (see Fig 2.2 - optimal assignment - for an example):

Lemma 6 *There is only one optimal solution for instance $\mathcal{I}_2$: assign colors in $\mathcal{C}'$ to $a_{i'}$ and colors in $\mathcal{C}''$ to a_i .*

Corollary 7 *If agent a_i knows the $Q_{j,i}$s and up to $t/2$ of the $Q_{j,i'}$s, it can not compute an optimal solution (even if a_i knows the exact position of the colors in $a_{i'}$ ordering).*

Proof. Construct a partition of $\mathcal{C}_i$ in the following way: place index j in $\mathcal{C}'$ if a_i has knowledge of $Q_{j,i'}$. If the cardinality of $\mathcal{C}'$ is smaller than $t/2$, randomly add indices to make the cardinality exactly $t/2$. Let all the other indices in $\mathcal{C}''$. Agent a_i can not distinguish between instances $\mathcal{I}_1$ and $\mathcal{I}_2$ build according to this partition of $\mathcal{C}_i$ and, hence, can not decide if it has to be assigned colors whose indices are in $\mathcal{C}'$ or in $\mathcal{C}''$. Finally, observe that in both instances indices in $\mathcal{C}'$ are exactly in the same position in the ordering of the colors held by $a_{i'}$, thus the knowledge of these positions does not help. ■

Theorem 8 *The message complexity of the distributed balanced color assignment problem on ring is $\Omega(mn)$.*

Proof. Let `A` be any distributed algorithm for the problem. We analyze two cases separately.

Assume first that during the execution of `A`, in at least $\Omega(n)$ pairs $(a_i, a_{i'})$, agent a_i gets to know at least $m/n + 1$ of the colors a'_i has. We use Shannon's Entropy to calculate the minimum number of bit B to be exchanged between each pair a_i and a'_i so that that information is known by a_i. We have:

$$B = \log \binom{m}{\frac{m}{n}+1}.$$

Using Stirling's approximation, and as $m \geq n$, we get

$$\begin{aligned} B &\approx \left(\frac{m}{n}+1\right) \cdot \log \frac{m \cdot n}{m+n} \\ &\geq \left(\frac{m}{n}+1\right) \cdot \log \frac{n}{2} \in \Omega\left(\frac{m}{n} \log n\right). \end{aligned}$$

A basic message contains $\log n$ bits, hence we need at least $\Omega(m/n)$ basic messages to exchange B bits. Finally, as the distance between a_i and a'_i is $\Omega(n/2)$ we have that at least $\Omega(n/2) \cdot (m/n + 1) \cdot \Omega(n) \in \Omega(m \cdot n)$ messages are exchanged during the execution of the algorithm.

Assume now that the hypothesis above about a_is knowledge does not hold, then there must be at least one pair $(a_i, a_{i'})$ for which at most m/n colors are communicated. We show that there is an adversary that puts a_i in the situation described in Corollary 7. We assume that a_i asks the adversary to know $Q_{h,k}$ (*i.e.*, imagine the adversary simulates communication between agents). The adversary chooses a partition in $\mathcal{P}$ and one between the corresponding $\mathcal{I}_1$ and $\mathcal{I}_2$, answering accordingly and placing the first m/n requests of $Q_{j,i'}$ in $\mathcal{C}'$. Whenever a_i asks for some $Q_{j,h}$ with $h \neq i, i'$, then the adversary answers with a fixed constant K if $j \notin \mathcal{C}_i$, with zero otherwise. By Corollary 7, it is clear that the pair $(a_i, a_{i'})$ can not compute an optimal solution. ■

4. An approximation algorithm

Let us now describe a greedy algorithm that matches the lower bound on the message complexity, and is guaranteed to compute approximations that are within a factor three from the optimal solutions. In the next section we will show how the same algorithm can be used to find a $(2 + \epsilon)$-approximation, for any $\epsilon < 1$.

The algorithm works in $\lceil \log p \rceil + 3$ rounds (unless differently stated, we assume logarithms are in base two). The first round is for electing a leader a_L among the set of agents. In the second round, every agent a_i is assigned a label $l(a_i)$ which represents its left-distance from the leader. By *left-distance* of an agent a_i from another agent a_j, we mean the length of the path from a_j to a_i when traveling the ring clockwise. In this way the leader has label $l(a_L) = 0$, its left neighbor has label 1, and so on. The right neighbor of the leader has label $n-1$. It is well known that leader election can be done in $O(n)$ time with $O(n \log n)$ message complexity on a non anonymous unidirectional or bidirectional asynchronous ring of n nodes, even when the nodes are not aware of the size n of the ring (*e.g.*, [5]). Therefore, in the following we shall focus on the last $\lceil \log p \rceil + 1$ rounds. Every such round consists of $2n-1$ steps numbered 0 through $2n-2$. During the first n steps of each round, the variables ξ_i, which list the set of colors assigned to individual agents, are updated. The purpose of the remaining $n - 1$ steps is the computation (at each agent) of the variable $\mathcal{C}_r = \cup_i \xi_i$, which stores the set of colors assigned up to round r. At the end of the last round $\mathcal{C}_{\lceil \log p \rceil}$ is equal to the set of all colors. The colors assigned during round r of the algorithm are those whose weights fall in the interval $[l_r, u_r)$, where

$l_r = \left\{\frac{p}{2^{r+1}}\right\}$, $u_r = \left\{\frac{p}{2^r}\right\}$, and

$$\left\{\frac{a}{b}\right\} = \begin{cases} \lceil \frac{a}{b} \rceil & \text{if } \frac{a}{b} > \frac{1}{2}; \\ 0 & \text{otherwise.} \end{cases}$$

We now formally describe the algorithm performed by any agent a_i at any round r. Message $\mathcal{M}$ is received from the right neighbor and contains a list of the colors already assigned to agents closer to the leader at the present round. $\mathcal{M}'$ is the list of candidate colors to be assigned to agent a_i: colors whose weights fall in the interval relative to the round r, that have not been already assigned to agents closer to the leader in round r or to any other agent in previous rounds (*i.e.*, $\mathcal{M}' \cap (\mathcal{M} \cup \mathcal{C}_{r-1}) = \emptyset$). Finally, $\mathcal{M}^* \subseteq \mathcal{M}'$ contains the maximum number of colors still assignable to the agent (*i.e.* $|\mathcal{M}^*| \leq \frac{m}{n} - |\xi_i|$).

ALGORITHM `Balance` (performed by agent a_i)

```
1. $\mathcal{C}_{-1} := \xi_i := \emptyset$;
   **For** $r := 0$ **to** $\lceil \log p \rceil$ **do**
      2. Wait for message $\mathcal{M}$ from the right;   /* the
         leader starts the round by setting $\mathcal{M} := \emptyset$; */
      3. $\mathcal{M}' := \{c_j | \ c_j \notin \mathcal{M} \cup \mathcal{C}_{r-1}$ and $Q_{j,i} \in [l_r, u_r)\}$;
         /* if $r = 0$,  $Q_{j,i} \in [l_r, u_r]$ */
      4. Let $\mathcal{M}^*$ be the set with the $\min\{\frac{m}{n} - |\xi_i|, |\mathcal{M}'|\}$
         colors of highest weight in $\mathcal{M}'$;
      5. $\xi_i := \xi_i \cup \mathcal{M}^*$;    /* $a_i$ updates its own set of
         assigned colors */
      6. Send message $\mathcal{M} \cup \mathcal{M}^*$ to the left;
      7. Wait for message $\mathcal{M}_r$ from the right;
      8. $\mathcal{C}_r := \mathcal{C}_{r-1} \cup \mathcal{M}_r$;    /* Update $\mathcal{C}_r$ with the colors
         assigned in round $r$ */
      9. Send message $\mathcal{M}_r$ to the left;
      10. If $\mathcal{C}_r = \mathcal{C}$ then **stop**;
   **nextfor**
```

Theorem 9 *Algorithm* `Balance` *finds a feasible solution to the distributed balanced color assignment problem in time $O(n \log p)$ using $O(mn)$ messages.*

Proof. To prove correctness, we show that any assignment of colors to agents computed by algorithm `Balance` satisfies the two following conditions:

(i) $i \neq j \Rightarrow \xi_i \cap \xi_j = \emptyset$;

(ii) $\bigcup_{i=1}^{n} \xi_i = \{c_1, \ldots, c_m\}$.

(i) Starting from the empty set, the algorithm assigns new colors to agent a_i (by "storing" them in the variable ξ_i) only in statement 5. A color c_j is assigned to a_i only if $c_j \notin \mathcal{M} \cup \mathcal{C}_{r-1}$ (statement 3.), where $\mathcal{M} \cup \mathcal{C}_{r-1}$ represents the set of currently assigned colors. Since balanced color assignment is done sequentially (starting from the leader and following the ring clockwise $\lceil \log p \rceil + 1$ times) no color can be assigned to two different agents. In this way `Balance` prevents the assignment of the same color to two different agents.

(ii) If an available color c_j of weight $Q_{j,i} \in [l_r, u_r)$ is not taken by a_i during round r, it is only because a_i has enough colors already (statement 4.). However, this circumstance may not occur at all agents during the same round (for this would imply that there were more than m colors). Thus, either the color is taken by a higher labeled agent in round r, or is "left free" for agents for which the weight of c_j is less than l_r. By iterating the reasoning, we may conclude that, if not taken before, the color must be eventually assigned in round $\lceil \log p \rceil + 1$, where agents are allowed to pick colors for which their weight is zero.

As for the time complexity, algorithm `Balance` requires $O(\log p)$ rounds, and every round lasts $2n$ time units. Note, however, that the $(r+1)$th round can start as soon as the leader has sent the message $\mathcal{M}_r$ to its left neighbor, so that the first half of round $r+1$ and the second half of round r overlap. This saves a factor 2 in the time complexity. In addition there are two preliminary rounds, taking $O(n)$ time.

As far as the message complexity is concerned, it is easy to see that the number of messages exchanged during round r is $O(n|\mathcal{M}_r|)$, where $\mathcal{M}_r$ is the set of colors assigned during round r. Therefore the total number of messages exchanged during the whole algorithm is $O(\sum_{r=0}^{\log p} n|\mathcal{M}_r|) = O(n \sum_{r=0}^{\log p} |\mathcal{M}_r|) = O(nm)$. ■

Under certain circumstances, it may be possible to reduce the actual time bound by avoiding rounds during which no colors are possibly assigned to agents. To this end, we endow algorithm `Balance` with the following simple preliminary computation, just after the leader election stage. First, each agent a_i constructs a string s_i of $\log p$ bits such that the r^{th} bit of s_i is '1' *if and only if* there is at least one color c_j for which $Q_{j,i}$ falls in the interval $[l_r, u_r)$. Second, every agent computes the string $S = \bigvee_{i=0}^{n-1} s_i$ in the following way: the leader (say, a_0) starts by just sending s_0 to its left neighbor; when an agent a_i receives the string $S_{i-1} = \bigvee_{k=1}^{i-1} s_k$ from its right neighbor, it computes $S_i = S_{i-1} \vee s_i$ and forwards it to its left neighbor. A final round is required to broadcast S to all the agents.

At this point, we can slightly modify algorithm `Balance` by simply *skipping* any round r such that the r^{th} bit of S is 0. It is easy to observe that the additional stage requires $O(n \log p)$ messages and $O(n)$ time units. This does not affect the asymptotic complexity of the whole algorithm, provided that $\log p / \log n \in O(m)$.

5. Approximation Factor of Algorithm `Balance`

In this section, for the sake of presentation, we will first prove that the cost of the solution computed by algorithm `Balance` is at most three times the cost of the optimal solution, and that the analysis is tight. We will then show how the algorithm can be modified to achieve a $(2+\epsilon)$-approximation ratio for any $\epsilon < 1$, without affecting time and message complexities. Finally, we will show how to modify the algorithm to get a 2-approximation ratio with a little increase of message complexity.

Lemma 10 *Let color c_j be assigned to agent a_i in round r by algorithm* `Balance`*. Let a_k be a different agent with $Q_{j,k} \in [l_r, u_r)$. Then $Q_{j,i} \le 2Q_{j,k}$.*

Proof. If $r = \log p$, then $Q_{j,i} = Q_{j,k} = 0$, and we are done. Otherwise, as c_j is assigned to agent a_i in round r then $Q_{j,i} \in [l_r, u_r)$ and the thesis easily follows from

$$\frac{p}{2^{r+1}} \le Q_{j,i}, Q_{j,k} < \frac{p}{2^r}.$$

■

Let `B` be the assignment of colors to agents determined by algorithm `Balance`, and let OPT be an optimal assignment. Define a partition of the set of colors based on their indices, as follows:

- $C' = \{j \mid \mathtt{B}(j) = OPT(j)\}$; *i.e.*, indices of the colors for which the assignment made by algorithm `Balance` coincides with (that of) the optimal solution.
- $C'' = \{0, \ldots, m-1\} \setminus C'$.

Lemma 11 *Let $j \in C''$, and let $k \ne j$ be any other index in C'' such that $\mathtt{B}(k) = OPT(j)$. Then*

$$Q_{j,OPT(j)} \le \max\{2 \cdot Q_{j,\mathtt{B}(j)}, Q_{k,\mathtt{B}(k)}\}.$$

Proof. First notice that, by a simple cardinality argument, at least one such k must exist.

Given that $j, k \in C''$ and that $\mathtt{B}(k) = OPT(j)$, we easily get $\mathtt{B}(j) \ne \mathtt{B}(k)$. For simplicity, suppose $a_1 = \mathtt{B}(j)$ and $a_2 = \mathtt{B}(k)$. What we have to prove is then $Q_{j,2} \le \max\{2Q_{j,1}, Q_{k,2}\}$. Suppose that $Q_{j,2} > Q_{k,2}$. Clearly this happens if and only if agent a_1 gets color c_j before a_2 can process it, for otherwise a_2 would get c_j instead. Now, the "worst" case for a_2 is to try to get c_j in the same round (though after) a_1. By Lemma 10, this implies $Q_{j,2} \le 2Q_{j,1}$. ■

Theorem 12 *Algorithm* `Balance` *is a 3-approximation algorithm for the distributed balanced color assignment problem.*

Proof. Let $Cost_{\mathtt{B}}$ and $Cost_{OPT}$ be the cost of the solutions given by algorithm `Balance` and OPT, respectively. We can express these costs in the following way (for an easy reading, we omit index i's range, that is always $[0, n-1]$):

$$\begin{aligned} & Cost_{\mathtt{B}} \\ = & \sum_{j=0}^{m-1} \sum_{i \ne \mathtt{B}(j)} Q_{j,i} \\ = & \sum_{j \in C'} \sum_{i \ne \mathtt{B}(j)} Q_{j,i} + \sum_{j \in C''} \sum_{i \ne \mathtt{B}(j)} Q_{j,i} \\ = & \sum_{j \in C'} \sum_{i \ne \mathtt{B}(j)} Q_{j,i} + \sum_{j \in C''} \left(Q_{j,OPT(j)} + \sum_{\substack{i \ne OPT(j), \\ i \ne \mathtt{B}(j)}} Q_{j,i} \right). \end{aligned}$$

Analogously,

$$\begin{aligned} & Cost_{OPT} \\ = & \sum_{j \in C'} \sum_{i \ne OPT(j)} Q_{j,i} + \sum_{j \in C''} \left(Q_{j,\mathtt{B}(j)} + \sum_{\substack{i \ne OPT(j), \\ i \ne \mathtt{B}(j)}} Q_{j,i} \right). \end{aligned}$$

By definition, $\mathtt{B}(j) = OPT(j)$, for $j \in C'$, and thus $\sum_{j \in C'} \sum_{i \ne \mathtt{B}(j)} Q_{j,i} = \sum_{j \in C'} \sum_{i \ne OPT(j)} Q_{j,i}$, *i.e.*, the cost associated with color $c_j \in C'$ is exactly the same for `Balance` and OPT. Notice also that the term $\sum_{j \in C''} \sum_{\substack{i \ne OPT(j) \\ i \ne \mathtt{B}(j)}} Q_{j,i}$ appears in both cost expressions. Hence, to prove that $Cost_{\mathtt{B}}/Cost_{OPT} \le 3$, it is sufficient to show that

$$\sum_{j \in C''} Q_{j,OPT(j)} \le 3 \sum_{j \in C''} Q_{j,\mathtt{B}(j)}.$$

To this end, we build a partition of the set C'' according to the following procedure (which corresponds to decomposing a permutation into the unique product of cycles). We start from any j_1 in C'' and find another index j_2 such that $OPT(j_1) = \mathtt{B}(j_2)$. As already pointed out in Lemma 11, such an index j_2 must exist. We repeat until, for some t, we get $j_t = j_1$. We then set

$$\mathcal{C}_1 = \{j_1, j_2, \ldots, j_t\}.$$

If $\mathcal{C}'' = \mathcal{C}_1$, we stop; otherwise, we pick another index $l_1 \notin \mathcal{C}_1$ and repeat the same procedure to define a second set $\mathcal{C}_2$. We continue until we obtain the partition

$$C'' = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \ldots \cup \mathcal{C}_s.$$

Now, for simplicity, let us regard the set $\mathcal{C}_i$ also as the ith cycle of the permutation corresponding to C'', with $(j, j') \in C''$ meaning that (j, j') is an inversion of C''. Then, using Lemma 11,

$$\begin{aligned}
& \sum_{j \in C''} Q_{j,OPT(j)} \\
= & \sum_{\mathcal{C}_i} \sum_{(j,j') \in \mathcal{C}_i} Q_{j,OPT(j)} \\
\leq & \sum_{\mathcal{C}_i} \sum_{(j,j') \in \mathcal{C}_i} \max\{2 \cdot Q_{j,\mathtt{B}(j)}, Q_{j',\mathtt{B}(j')}\} \\
\leq & \sum_{\mathcal{C}_i} \sum_{(j,j') \in \mathcal{C}_i} \left(2 \cdot Q_{j,\mathtt{B}(j)} + Q_{j',\mathtt{B}(j')}\right) \\
= & \sum_{j \in C''} 3Q_{j,\mathtt{B}(j)}
\end{aligned}$$

■

Theorem 13 *For any $0 < \epsilon < 1$, there exist instances of the distributed balanced color assignment problem such that $COST_B$ is a factor $3 - 4\epsilon/(4\delta + \epsilon)$ larger than the optimal cost, for some $0 < \delta < 1$.*

Proof. Consider the following instance of the problem. For the sake of presentation, our example is given for the case $m = n$ even, but it is straightforward to extend it to the general case. Fix any rational $0 < \epsilon < 1$, let $q, \delta > 0$ be such that $q\epsilon/4$ is integer and $q\delta = \lceil q \rceil$. Assume the colors are distributed as follows:

$$\left\{\begin{array}{lcl} Q_{2i,2i} & = & q(\delta + \epsilon/4) \\ Q_{2i+1,2i} & = & q\delta \\ Q_{2i,2i+1} & = & q(2\delta - \epsilon/4) \\ Q_{2i+1,2i+1} & = & 0. \end{array}\right. \qquad i = 0, 1, \ldots, \frac{n}{2} - 1$$

Suppose that a_0 is the leader elected in the first round of algorithm `Balance`, and that the label assigned to agent a_i is i, for $i = 1, \ldots, n-1$. Consider agents a_{2i} and a_{2i+1}, for any $0 \leq i \leq \frac{n}{2} - 1$. We can always assume that q is such that $\frac{p}{2^{r+1}} \leq q\delta < q(\delta + \epsilon/4) < q(2\delta - \epsilon/4) < \frac{p}{2^r}$, for some r. Hence, the weights of color c_{2i} for agents a_{2i} and a_{2i+1} belong to the same interval $\left[p/2^{r+1}, p/2^r\right)$.

It is easy to see that the optimal assignment gives c_{2i+1} to a_{2i} and c_{2i} to a_{2i+1}. The corresponding cost is $Cost_{OPT} = \frac{n}{2}q(\delta + \epsilon/4)$. On the other hand, algorithm `Balance` assigns c_{2i} to a_{2i} and c_{2i+1} to a_{2i+1}, with a corresponding cost $Cost_{\mathtt{B}} = \frac{n}{2}q(3\delta - \epsilon/4)$. As for the approximation factor, we get

$$\frac{Cost_{\mathtt{B}}}{Cost_{OPT}} = \frac{3\delta - \frac{\epsilon}{4}}{\delta + \frac{\epsilon}{4}} = \frac{3\left(\delta + \frac{\epsilon}{4}\right)}{\delta + \frac{\epsilon}{4}} - \frac{\frac{3\epsilon}{4} + \frac{\epsilon}{4}}{\delta + \frac{\epsilon}{4}} = 3 - \frac{4\epsilon}{4\delta + \epsilon}.$$

■

Theorem 13 proves that the analysis produced for algorithm `Balance` is tight. However, if we are willing to pay something in message complexity, we can get a 2-approximation algorithm.

Corollary 14 *Algorithm `Balance` can be transformed into a 2-approximation algorithm, by paying an additional multiplicative $O(\log p)$ factor in message complexity.*

Proof. Given round r and for every color c_j in $\mathcal{M}_r$, let $a_{j'}$ be the agent with maximum $Q_{j,j'}$ between all agents a such that $p/2^{r+1} \leq Q_{j,j'} < p/2^r$.

We modify Algorithm `Balance` to have color c_j assigned to agent $a_{j'}$ even if $a_{j'}$ is not the agent closest to the leader. Any agent selecting color c_j adds the information concerning Q_j to the outgoing message. This requires $O(\log p)$ extra bits per color and each message is now a set of pairs (c_j, q), such that c_j is a color and there is an agent a for which $Q_{j,j'} = q$. The cost analysis for the modified algorithm is straightforward. ■

The algorithm and the proof of the approximation factor (and tightness of approximation) can be modified to work for any base of the logarithm, giving us the following result.

Theorem 15 *For every $0 < \epsilon < 1$, there is a $(2+\epsilon)$-approximation algorithm for the distributed balanced color assignment with running time $O(n \log_{1+\epsilon} n)$ and message complexity $O(mn)$.*

Proof. Use Algorithm `Balance` with the following changes: $l_r = \left\{\frac{p}{(1+\epsilon)^{r+1}}\right\}$, $u_r = \left\{\frac{p}{(1+\epsilon)^r}\right\}$, and

$$\left\{\frac{a}{b}\right\} = \left\{\begin{array}{ll} \lceil \frac{a}{b} \rceil & \text{if } \frac{a}{b} > \frac{1}{1+\epsilon}; \\ 0 & \text{otherwise.} \end{array}\right.$$

Statements of Lemma 11 becomes

$$Q_{j,OPT(j)} \leq \max\{(1+\epsilon) \cdot Q_{j,\mathtt{B}(j)}, Q_{k,\mathtt{B}(k)}\}$$

and following the same reasoning as in Theorem 12 we get the $(2+\epsilon)$ approximation factor. ■

6. Conclusion

In this paper we have considered the Distributed Balanced Color Assignment problem, which models problems where distributed agents search a common space and need to rearrange or organize the retrieved data. The question addressed here is intimately related to different matching problems.

Our results indicate that these kinds of problems can be solved quite efficiently in a distributed setting, and that the loss incurred by the lack of centralized control is not significant. We have focussed our attention to distributed solutions tailored for a ring of agents. A natural extension would be to consider more general topologies, and analyze how our techniques and ideas have to be modified in order to give efficient algorithms in these more general settings.

Acknowledgements. The authors wish to thank Bruno Codenotti for many helpful comments and discussions.

References

[1] Y. Amir, B. Awerbuch, A. Barak, R.S. Borgstrom, A. Keren: An Opportunity Cost Approach for Job Assignment and Reassignment in a Scalable Computing Cluster. *IEEE Trans. on Parallel and Distributed Systems* 11:760-768, 2000.

[2] G. J. Chang, P. H. Ho: The β-assignment Problems. *European J. Oper. Research* 104:593-600, 1998.

[3] R. S. Chang, R. C. T. Lee: On a scheduling problem where a job can be executed only by a limited number of processors. *Computers and Operation Research* 15:471-478, 1988.

[4] C. Cramer, T. Fuhrmann, K Kutzner: Scalable source routing - protocol specification, version 1.0. *Tech. Rep. 2005-4*, Faculty of Informatics, University of Karlsruhe, 2005.

[5] D. Dolev, M. Klawe, M. Rodeh, An $O(n \log n)$ Unidirectional Distributed Algorithm for Extrema Finding in a Circle, *J. of Algorithms* 3:245-260, 1982.

[6] M. Hanckoviak, M. Karonski, A. Panconesi: On the distributed complexity of computing maximal matchings. *Proc. of SODA 98, the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 219-225, 1998.

[7] M. Hanckoviak, M. Karonski, A. Panconesi: A faster distributed algorithm for computing maximal matchings deterministically. *Proc. of PODC 99, the Eighteenth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 219-228, 1999.

[8] J. E. Hopcroft, R. M. Karp: An $n^{\frac{5}{2}}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.* 2:225-231, 1973.

[9] H. W. Kuhn: The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly* 2:83-97, 1955.

[10] J. Könemann, R. Ravi: A Matter of Degree: Improved Approximation Algorithms for Degree-Bounded Minimum Spanning Trees. *Proc. of Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 537-546, 2000.

[11] N. Lynch, Distributed Algorithms. Morgan Kaufmann Publishers, San Mateo, CA, 1996.

[12] L. Lovasz, M. D. Plummer: Matching Theory. Annals of Discrete Mathematics 29, North-Holland Mathematics Studies 121, 1986.

[13] D. Peleg: Distributed Computing: A Locality-Sensitive Approach. *SIAM Monographs on Discrete Math. Appl.*, Philadelphia, PA, 2000.

[14] B. Schieber and S. Moran: Parallel Algorithms for Maximum Bipartite Matching and Maximum 0-1 Flows. *J. of Parallel and Distributed Computing* 6:20-38, 1989.

[15] J. Šilk, B. Robič: Processor Allocation Based on Weighted Bipartite Matching Scheduling. *Technical Report CSD-97-1*, Computer System Department, Jožef Stefan Institut, Ljubljana, Slovenia.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149-160, 2001.