Distributed β-assignment on graphs*

Gianluca De Marco[1], Mauro Leoncini[2], Lucia Mazzali[2], and Manuela Montangero[2] **

[1] Dipartimento di Informatica, Università di Salerno, Italy
gianluca.demarco@dia.unisa.it
[2] Dipartimento di Scienze Fisiche, Informatiche e Matematiche, Università di Modena e Reggio Emilia, Italy
leoncini@unimore.it, 168267@studenti.unimore.it, manuela.montangero@unimore.it

Abstract. 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 arbitrary graph. Each agent initially holds a subset of the items. We analyze the problem of distributively assigning colors to agents in such a way that (a) each color is assigned to one agent only and (b) the number of different colors assigned to each agent is minimum (c) the number of items initially assigned to agents that are not consistent with the assignment of colors is minimum.

This problem, known in the centralized setting as a matching problem, has been introduced in the distributed setting in [3] for the simple ring topology.

Here, we generalize the problem to arbitrary graphs, by solving it with a distributed algorithm which is efficient both in terms of time complexity and message complexity for a large class of graphs. Namely, our results can be outlined as follows. We prove a lower bound on the message complexity of $\Omega(m/n \cdot D^2)$, where D is the diameter of the graph.

We give a distributed deterministic algorithm with time complexity $O(\max\{n^2, D \log q\})$ and message complexity $O\big(\frac{n}{\log n} \cdot \big(\log q + m \log m\big)\big)$, where q is the maximum number of items of a given color held by an agent. It can be noticed that for a large class of instances of practical interest, namely for $m \in O(n^c)$, for some constant c, and $q \in O(m^m)$, our algorithm exhibits a message complexity of $O(m \cdot n)$, which turns out to be optimal, in view of our lower bound, for graphs with diameter linear in the number of nodes.

We finally show that the cost of our solution for arbitrary graphs is at most three times the optimal cost (found by a centralized algorithm).

Keywords: algorithms, distributed computing, assignment problems.

---

* A preliminary version of this work appeared in Lucia Mazzali's Master Thesis.

** Work partially supported by the SUCCESS Project H2020-EU.3.4., G.A. 633338

1 Introduction

In this paper we consider the *Balanced Color Assignment Problem* and we propose a distributed algorithm to find an approximated solution for arbitrary communication graphs. This problem was introduced in [3] for the ring topology.

1.1 The problem

Consider a set of items and a set of m colors, where each item is associated to one color. There are n agents connected by a communication graph, and each agent holds a subset of the colored items. A *color assignment* assigns each color to exactly one agent, while a *balanced color assignment* is a color assignment that minimizes the maximum, over all agents, of the number of different colors assigned to the same agent.

Let's say that the inital assignment of an item to an agent is *invalid* for a certain color assignment if the color of the item is not assigned to the agent by the color assignment.

Our goal is to search for a balanced color assignment that minimizes the number of invalid initial assignments.

Fig. 1 shows an example of (a) an initial item distribution and (b-c) two possible color assignments.

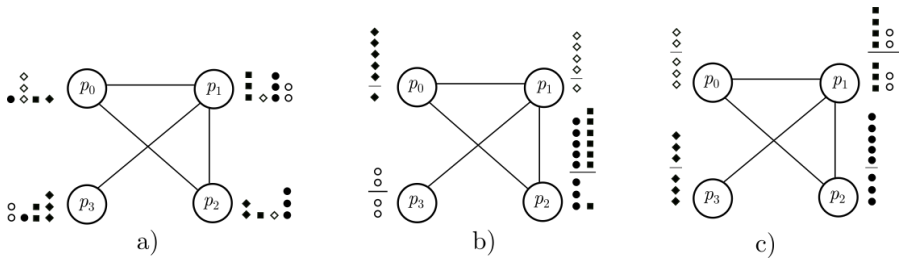

Fig. 1. Example for $n = 4$ agents and $m = 5$ colors. Items above the lines are invalid initial assignments, their total number gives the cost of the assignment. (a) Initial item distribution, (b) balanced color assignments of cost 22 (c) optimal assignment of cost 16.

The problem was fist introduced in [3], where an optimal message 3-approximation distributed algorithm was given for the case of the ring topology. Authors showed that the centralized version of the problem is equivalent to a maximum weight perfect matching problem on complete bipartite graphs when $m = n$, while it is equivalent to the weighted β-assignment [2] problem on weighted bipartite graphs when $m > n$.

Definition 1 (Balanced color assignment [3]). *Let* $\mathcal{A} = \{a_0, a_1, ..., a_{n-1}\}$ *be a set of* n *agents connected by a communication graph and let* $\mathcal{C} =$

$\{c_0, c_1, ..., c_{m-1}\}$ *be a set of* m *colors. Let* $Q_{j,i} \geqslant 0$ *be the number of items of color* c_j *initially held by the agent* a_i*, for* $j = 0, 1, ..., m-1$ *and* $i = 0, 1, ..., n-1$*. A Balanced Coloring is an assignment*

$$\pi : \{0, 1, ...m-1\} \to \{0, 1, ...n-1\}$$

of the m *colors to the* n *agents in such a way that:*

i) for every color c_j *there is exactly one agent* a_i *such that* $\pi(j) = i$*;*
ii) for every agent a_i *we have*

$$\left\lfloor \frac{m}{n} \right\rfloor \leqslant |\{c_j : \pi(j) = i\}| \leqslant \left\lceil \frac{m}{n} \right\rceil$$

i.e.*, the number of colors assigned to agents is balanced. In particular* $\lfloor \frac{m}{n} \rfloor$ *colors are assigned to* $[(\lfloor \frac{m}{n} \rfloor + 1)n - m]$ *agents and* $\lfloor \frac{m}{n} \rfloor + 1$ *colors to the remaining* $(m - \lfloor \frac{m}{n} \rfloor n)$ *agents.*

Definition 2 (Distributed balanced color assignment problem [3]). *The Distributed Balanced Color Assignment Problem aims at finding a balanced coloring of minimum cost in a distributed way, where the cost of a balanced coloring* $\pi : \{0, 1, ...m-1\} \to \{0, 1, ...n-1\}$ *is defined as:*

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

Intuitively, the cost of the assignment is the total number of invalid initial assignments.

1.2 The model

We assume that the agents in $\mathcal{A} = \{a_0, \ldots, a_{n-1}\}$ are connected by a generic graph and each agent can communicate only with its neighbors. We assume that each agent knows n (the number of agents), and $\mathcal{C}$ (the set of colors).

We measure message complexity in the standard way (see for example [5]), *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; that is, one basic message can carry at most a constant number of agent ID's. Non basic messages of length L are allowed, and we charge a cost $c' \lceil L / \log n \rceil$ for their transmission, for some constant c'.

Brute force approach. Let $q = \max_i \max_j Q_{j,i}$ be the maximum number of items of a given color initially assigned to agents. In a brute force algorithm, all agents send all their color information to a designated leader agent which solves the problem locally using the optimal algorithm for the maximum weight perfect matching (if $n = m$), or for the weighted β-assignment problem (if $m > n$). The leader then sends the solution back to all the other agents. With such an approach, each agent sends at most $O(m)$ non basic messages, each one accounting for $O(\frac{\log q}{\log n})$ basic messages. Each message passes through $O(D)$ channels, where D is the diameter of the graph and, thus, the overall the message complexity is $O(mnD\frac{\log q}{\log n})$.

2 Lower bound on message complexity

To derive a lower bound we proceed in the following way: we describe two instances of the problem, we pair agents and show that these two instances are indistinguishable by one agent in the pair unless some communication is involved.

We start with the following lemma.

Lemma 1. *Given a connected graph G with diameter D, we have $\Omega(D)$ disjoint pairs of nodes at distance $\Omega(D)$.*

Proof. Let P be a shortest path connecting two nodes x and y at distance D. Arbitrarily number the nodes in P according to their distance from x. Then the $\lceil D/2 \rceil$ pairs $(i, i + \lceil \frac{D+1}{2} \rceil)$ are disjoint and the distance between the node components is $\lceil \frac{D+1}{2} \rceil$, for $i = 0, \ldots, \lfloor \frac{D+1}{2} \rfloor - 1$. □

W.l.o.g., we consider instances of the problem in which the number agents n is even and the number of colors m satisfies $m = nt/2$, for some even integer t.

By Lemma 1, we can form $P \in \Omega(D)$ disjoint pairs of nodes at distance $\Omega(D)$. Since $P \leq n/2$, we pair the remaining $n - 2P$ agents arbitrarily. .

We then partition the set of colors $\mathcal{C}$ into $n/2$ subsets of cardinality t each; *i.e.*, $\{\mathcal{C}_0, \mathcal{C}_1, ...\mathcal{C}_{\frac{n}{2}-1}\}$ is a partition of the total set of colors $\mathcal{C}$ such that $|\mathcal{C}_j| = t$ for all $j = 0, 1, ..., \frac{n}{2} - 1$.

We consider the set of instances I in which a pair $(a_i, a_{i'})$ of agents is initially assigned only items whose colors are in the subset $\mathcal{C}_i = \{c_{i_1}, ..., c_{i_t}\}$; *i.e.*, no other agent is initially assigned items of colors in $\mathcal{C}_i$.

Now we define two specific instances $I_1, I_2 \in I$. Fix an integer $u > 1$, we have

Instance I_1 :

agent a_i

$Q_{j,i} = u$ for $j \in \mathcal{C}_i$

agent $a_{i'}$

$Q_{j,i'} = u$ for $j = i_1, \ldots i_{\frac{t}{2}}$

$Q_{j,i'} = u + 1$ for $j = i_{\frac{t}{2}+1}, \ldots i_t$

Instance I_2 :

agent a_i

$Q_{j,i} = u$ for $j \in \mathcal{C}_i$

agent $a_{i'}$

$Q_{j,i'} = u$ for $j = i_1, \ldots i_{\frac{t}{2}}$

$Q_{j,i'} = u - 1$ for $j = i_{\frac{t}{2}+1}, \ldots i_t$

Observe that, from the point of view of a_i the two instances are identical, as the initial color assignment in the two instances are identical. On the other hand, the optimal solutions for instances I_1 and I_2 are dual: in I_1 agent a_i has to keep the colors with indices $i_1, \ldots i_{\frac{t}{2}}$, while in I_2 agent a_i has to keep the colors with indices $i_{\frac{t}{2}+1}, \ldots i_t$ (an example can be found in Figure 2). Formal proofs omitted here for lack of space can be found in an extended version of this work.

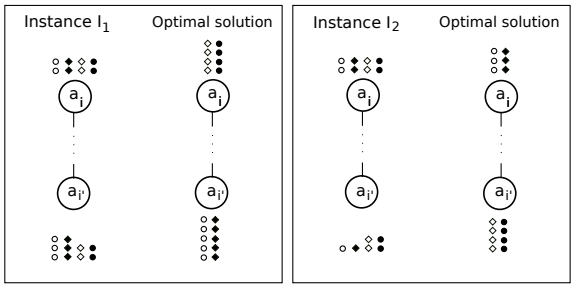

Fig. 2. Example: Instances I_1, I_2, with $t = 4$ and $u = 2$. Agent a_i has the same initial item assignment, while $a_{i'}$ has exactly $2 = t/2$ colors for which the initial assignment differs. The optimal solutions for instance I_1 and I_2 assign distinct set of colors to the two agents.

Lemma 2. *Given a pair of agents $(a_i, a_{i'})$ and instances I_1 and I_2, agent a_i can distinguish instance I_1 form instance I_2 and compute the optimal solution only if a_i knows at least $\frac{t}{2}$ colors initially held by $a_{i'}$.*

Intuitively, as instances I_1 and I_2 are indistinguishable from a_i's point of view, the only way for a_i to break this symmetry is to communicate with $a_{i'}$, and such communication must involve all the $t/2$ colors that might be assigned to a_i or $a_{i'}$, according to the initial item distribution.

Lemma 3. *Given a pair of agents $(a_i, a_{i'})$ and instances I_1 and I_2, in order to compute the optimal solution, agents in the pair must exchange at least $\Omega(\frac{m}{n})$ basic messages.*

Proof. By Lemma 2, a_i has to collect information about at least $\frac{t}{2} = \frac{m}{n}$ different colors held by $a_{i'}$. Using Shannon's Entropy, the minimum number B of bits needed to pin down m/n out of m colors is:

$$B = \log \binom{m}{m/n} = \log \left(\frac{m!}{\frac{m}{n}!(m-\frac{m}{n})!} \right)$$

Stirling's approximation gives us the following bound to the quantity $\log n!$:

$$n \log n - \frac{n-1}{\ln 2} \leq \log n! \leq n \log n - \frac{n-1}{\ln 2} + \log n. \tag{1}$$

Using Equation (1), some elementary algebra and the fact that $m \geq n$, we derive the following lower bound on B:

$$B = \log \binom{m}{m/n} \geq -\frac{1}{\ln 2} + \frac{m}{n} \log n - 2 \log m + \log n \in \Omega \left(\frac{m}{n} \log n \right).$$

A basic message contains $\log n$ bits, therefore the pair needs to exchange at least $\Omega(\frac{m}{n})$ basic messages.

□

Theorem 1. *The distributed balance color assignment problem has* $\Omega\left(\frac{m}{n} \cdot D^2\right)$ *message complexity.*

Proof. At the end of the execution of any distributed algorithm running on instance I_1 or I_2, each process has to determine its own assignment of colors.

We now recall that in both instances there are $P \in \Omega(D)$ disjoint pairs at a distance $\Omega(D)$, and, by Lemma 3 any algorithm must use, for each such pair, $\Omega(m/n)$ basic messages to compute the optimal solution. Hence, we have the following lower bound on message complexity:

$$\Omega(D) \cdot \Omega(D) \cdot \Omega\Big(\frac{m}{n}\Big) = \Omega\left(\frac{m}{n} \cdot D^2\right).$$

Corollary 1. *Given a graph* G *with diameter* D *linear in the number of nodes (*i.e.*,* $D \in \Omega(n)$*), then, the message complexity lower bound for the balanced color assignment problem is* $\Omega(m \cdot n)$*.*

Corollary 2. *Given a graph* G *with constant diameter* D *(*i.e.*,* $D \in \Theta(1)$*), then, the message complexity lower bound for the balanced color assignment problem is* $\Omega(m)$*.*

Proof. If the graph is connected, there are $\Theta(n)$ disjoint pairs of agents at distance $\Theta(1)$, hence we have:

$$\Omega(n) \cdot \Omega(1) \cdot \Omega\Big(\frac{m}{n}\Big) = \Omega\left(m\right).$$

□

3 Algorithm `Breadth-Balance`

In this section we present the algorithm `Breadth-Balance` to find an approximate solution to the balanced color assignment on graph topology.

The algorithm works in three phases: (1) the algorithm elects a leader agent, (2) it builds a breadth-first spanning tree rooted at the leader, and (3) it assigns colors to agents in successive rounds, each round consisting of a convergecast and a broadcast along the spanning tree.

Table 3 reports notation used in the description of the algorithm.

Algorithm `Breadth-Balance`.
Phase 1: To elect the leader, the algorithm uses the *Mega-Merger* [4] that works in a decentralized way for arbitrary network and uses $O(n \log n + |E|)$ messages on a graph with n nodes and $|E|$ edges.

Phase 2: The algorithm builds a breadth-first spanning tree using the BFST algorithm with centralized control [6], where the centralized control is done by the leader agent elected in Phase 1. The message complexity of the algorithm is $O(n^2)$.

$\mathcal{C}$	set of colors to be assigned
$Q_{j,i}$	number of items of color c_j initially assigned to agent a_i
K_i	total number of colors to be assigned to agent a_i
E_i	number of colors already assigned to agent a_i
$\mathcal{C}_r$	set of colors assigned by the end of round r
I_r	interval of weights (number of items of the same color) in round r
$\mathcal{L}_{i,r}$	list of colors requested by agent a_i in round r
$L_{i,r}$	list of colors requested by agents in the subtree rooted at a_i in round r
L_{I_r}	flagged list of colors requested in round r

At the end of this phase each agent a_i knows the identifier of its parent (denoted with $parent_i$) and the identifiers of its children (stored in the list $children_i$), for $i = 1, \ldots, n$.

Phase 3: The algorithm computes an approximate solution to the balanced color assignment problem on graphs. The phase is in turn divided into two stages.

3.1 Agents compute the value $q = \max_i \max_j Q_{j,i}$ (i.e. the maximum number of items of the same color initially held by any agent), with a convergecast followed by a broadcast:
- *Convergecast:* each agent a_i computes $q_i = \max_j Q_{j,i}$ and collects all the values $q_{i_0}, q_{i_1}, \ldots, q_{i_{k_i-1}}$ from its $k_i \geq 0$ children. Then it sends $\max\{q_i, q_{i_0}, q_{i_1}, \ldots, q_{i_{k_i-1}}\}$ to its parent.
- *Broadcast:* the leader computes the exact value q and broadcasts it back down the tree.

Phase 3.1 is accomplished with $O(n \cdot \frac{\log q}{\log n})$ basic messages.

3.2 Agents actually assigns colors to themselves. After an initialization stage, the second stage proceeds in rounds.

Initialization: Each agent a_i computes $g = (\lfloor \frac{m}{n} \rfloor)n - m$ and stores in the variable K_i the number of colors it will assign to itself according to the following rule:

$$K_i = \begin{cases} \lfloor \frac{m}{n} \rfloor & \text{if } i < g \\ \lfloor \frac{m}{n} \rfloor + 1 & \text{otherwise} \end{cases}$$

A counter variable E_i is used by a_i to store the number of colors that agent a_i has already assigned to itself: it is initially set to zero and it is incremented after each assignment.

Round r: The actual color assignment is done in $\lceil \log q \rceil + 1$ rounds.[3]In each round a (possibly empty) subset of colors is assigned to agents (each color to exactly one agent). At the beginning of a generic round r, for $r = 0, 1, ..., \lceil \log q \rceil$, $\mathcal{C}_{r-1}$ is the set of colors that has already been assigned in previous rounds and is known by all agents, Initially $\mathcal{C}_{-1}$ is the empty set, while at the end $\mathcal{C}_{\log q} = \mathcal{C}$, the set of all colors.

[3] The way in which rounds are defined is analogous to the one in [3]

In round r, $r = 0, \ldots, \lceil \log q \rceil$, agents consider only colors whose weight $Q_{i,j}$ falls in the interval $I_r = [l_r, u_r[$ defined as:

$$\begin{cases} I_0 = \left[\frac{q}{2}, +\infty\right[\\ I_r = \left[\frac{q}{2^{r+1}}, \frac{q}{2^r}\right[\quad \text{for } 1 \leq r \leq \lceil \log q \rceil \text{ -1} \\ I_{\lceil \log q \rceil} = [0, 1[\end{cases}$$

Observe that each agent is able to compute I_r, for each r, independently and consistently.

Each agent a_i computes $\mathcal{L}_{i,r} = \{c_j \in \mathcal{C} : c_j \notin \mathcal{C}_{r-1} \text{ and } Q_{i,j} \in I_r\}$, i.e. the set of not yet assigned colors whose weight falls in the interval I_r. If the cardinality of $\mathcal{L}_{i,r}$ is larger than $K_i - E_i$ (i.e., the maximum number of colors still assignable to the agent) then agents a_i selects the $K_i - E_i$ colors in $\mathcal{L}_{i,r}$ with greatest weight.

Each round $r \leq \lceil \log q \rceil$ consists of a convergecast followed by a broadcast:

- *Convergecast:* Leaves send a message containing their $\mathcal{L}_{i,r}$s to their parents. Internal node a_i waits for messages from all its children, then it composes the message $L_{i,r}$ containing the union of the sets of requested colors in the sub-tree rooted in a_i and sends this message to its parent. Finally, the root (leader) computes the list L_{I_r} of colors to be assigned in the current round.
- *Broadcast:* The leader a_ℓ assigns to itself the colors in $\mathcal{L}_{\ell,r}$ (i.e., the colors requested in the current round), updates the list of assigned colors ($\mathcal{C}_r = \mathcal{C}_{r-1} \cup L_{I_r}$) and prepares the messages to be delivered to its children. Each such message contains the complete list of requested colors L_{I_r} together with a binary flag associated to each color. Flag value 1 means that the color is available, while 0 means that the color has been assigned to some agent in the current round.

  For each child a_j, the leader set flags according to the following rules:
  (1) all flags of colors in $\mathcal{L}_{\ell,r}$ are set to 0;
  (2) the flag for color $c_z \in L_{I_r}$ is set to 0 if $c_z \notin L_{z,r}$; i.e., if the color c_z has not been requested by any agent in the subtree rooted in a_j.
  (3) the flag for color $c_z \in L_{I_r}$ is set to 1 only if $c_z \in L_{z,r}$ (i.e., if the color c_z has been requested by some agent in the subtree rooted in a_j). If the color appears in the request lists of two or more children, then the flag of color c_z is set to 1 in exactly one message (arbitrarily chosen) and to 0 in all the others.

  Upon receiving the message from their parents, all other agents act as the leader, starting from the received message.

Lemma 4. *Stage 3.2 of Algorithm* `Breadth-Balance` *can be completed using* $O(nm \cdot \frac{\log m}{\log n})$ *basic messages.*

Proof. Assume that A_r colors are assigned in round $r = 0, 1, ..., \lceil \log q \rceil$. In a generic round r we have:

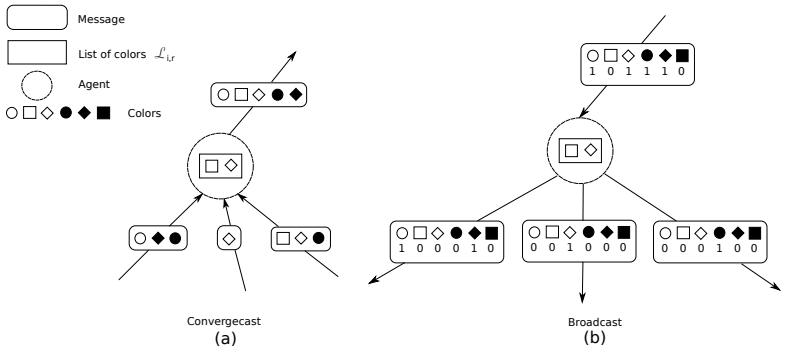

Fig. 3. Example of round r in stage 3.2. (a) Convergecast: each leaf agent sends to its parent the list of desired colors for the current round. Internal nodes merge their list of desired colors with those received by their children. (b) Broadcast: each node receives from its parent the list of all colors required in the round. Flags tell the agent if a color is available for choosing. Each agent prepares messages for its children in such a way that the flag for one color is set to one only in one message.

- $n-1$ non basic messages are sent to perform the *convergecast*. Each message contains at most A_r color identifiers, for a maximum of $A_r \log m$ bits per message. Hence, convergecast uses $O(n \cdot A_r \log m / \log n)$ basic messages.
- $n-1$ non basic messages are sent to perform the *broadcast*. Each message contains A_r color identifiers plus A_r bit for flags. Hence, also broadcast uses $O(n \cdot A_r \log m / \log n)$ basic messages.

Summing up for all rounds, as we have $\sum_{r=0}^{\log q} A_r = m$, we get:

$$O\left(\sum_{r=0}^{\lceil \log q \rceil} n \cdot \frac{A_r \log m}{\log n}\right) = O\left(n \cdot \frac{\log m}{\log n} \cdot \sum_{r=0}^{\lceil \log q \rceil} A_r\right) = O\left(n \cdot \frac{\log m}{\log n} \cdot m\right),$$

□

Theorem 2. *The* `Breadth-Balance` *algorithm finds a feasible solution to the balanced color assignment problem using* $O\big(\frac{n}{\log n} \cdot \big(\log q + m \log m\big)\big)$ *basic messages.*

Proof. First we prove that the algorithm works correctly, i.e. (a) one color is assigned to exactly one agent, (b) each color is assigned to at least one agent, and (c) the assignment is balanced.

a) Two agents cannot assign themselves the same color. Assume two agents requested the same (still available) color c_z in the same round r. Then, if one is ancestor of the other in the spanning tree, then the ancestor is the first to have the possibility to assign itself the color (if the flag associated to the color is 1). When it does, then it will set the flag to 0, and the descendant will not be able to select the color. Otherwise, if the two agents belong to different

subtrees, then a common ancestor will decide in which of the subtrees the color can be assigned and the flag is set to 1 in the message going down one (and only one) subtree.

b) In the last round ($r = \lceil \log q \rceil$), colors whose $Q_{j,i}$s fall in the interval $I_{\lceil \log q \rceil} = [0, 1[$ are assigned. Thus, it is possible that an agent is assigned a color even it initially held no item of that color. Hence, If a color is not taken before, it is eventually assigned in round $\log q$.

c) All colors are assigned, and each color is assigned to exactly one agent. Since each agent a_i assigns itself no more that K_i colors, by a counting argument, the assignment must be balanced.

Now we concentrate on the message complexity of the algorithm `Breadth-Balance`. The first phase requires $O(n \log n + |E|)$ basic messages, but $|E| \in O(n^2)$, hence $O(n \log n + |E|) \subset O(n^2)$. The second phase requires $O(n^2)$ basic messages. Phase 3.1 requires $O(n \frac{\log q}{\log n})$ basic message and phase 3.2 requires $O(nm \frac{\log m}{\log n})$ basic messages, by Lemma 4. Summing up upper bounds for single phases, and remembering that $m \geq n$, we get:

$$O(n^2) + O\Big(n \frac{\log q}{\log n}\Big) + O\Big(nm \frac{\log m}{\log n}\Big) = O\Big(\frac{n}{\log n} \cdot \Big(\log q + m \log m\Big)\Big).$$

□

Corollary 3. *Assuming $m \in O(n^c)$, for some constant c, and $q \in O(m^m)$, `Breadth-Balance` algorithm finds a solution to the balanced color assignment problem using $\Theta(m \cdot n)$ message complexity.*

We conclude this analysis with two observations. Under the hypothesis of Corollary 3, we have that:

1. if the diameter of the graph is linear in the number of nodes (*i.e.*, $D \in \Omega(n)$), then the algorithm `Breadth-Balance` has optimal message complexity $\Theta(m \cdot n)$;
2. the upper bound of the brute force approach becomes $O(mnD \frac{\log q}{\log n}) = O(m^2 n^2)$, the square of the lower bound.

Synchronous case. The algorithm can be used also for the synchronous case and we have the following results:

Theorem 3. *The `Breadth-Balance` algorithm finds a feasible solution to the balanced color assignment problem in time $O(\max\{n^2, D \log q\})$.*

Proof. The first phase, that uses the Mega-Merger algorithm, requires $O(n)$ time units [1]. The second phase, that uses the BFST algorithm with centralized control, requires $O(n^2)$ time units [6]. Phase 3.1 requires $O(D)$ time units for each broadcast and each convergecast. Phase 3.2 requires $O(D \log q)$ time units, as it performs one convergecast and one broadcast in each of the $\lceil \log q \rceil + 1$ rounds. Summing up upper bounds for single phases we get:

$$O(n) + O(n^2) + O(D) + O(D \log q) = O(\max\{n^2, D \log q\}).$$

3.1 Approximation factor

The analysis of [3] can be applied to the `Breadth- Balance` algorithm, giving the following result:

Theorem 4. *`Breadth-Balance` is a 3-approximation algorithm for the distributed balanced color assignment problem.*

Proof. For lack of space, the proof is omitted here and given in an extended version of this work.

We now show that the above results is tight by defining a particular family of input instances whose approximation ratios get arbitrarily close to 3. To this end, it is sufficient to assume $m = n$.

Fix $q > 30$ and let T be the spanning tree computed in Phase 2 of the algorithm. We form as many disjoint pairs (a_i, a_k) as possible, with the constraint that a_i is an ancestor of a_k in T, and let A denote the set of remaining nodes, i.e., ones that are not involved in any pair. We now consider the specific initial assignment of items to agents described below.
(i) Each agent $a_j \in A$ has q items colored with c_j and none of other colors. No other agent has items colored with c_j, initially.
(ii) Let $x = x(r) = \left\lfloor \frac{q}{2^{r+1}} \right\rfloor - 1$, for an arbitrary $r \in [2..\lceil \log q \rceil - 1]$; then, the paired agents (a_i, a_k), have the following initial item assignment:

agent a_i :	$Q_{i,i} = x + 3$	*i.e.,* $x + 3$ items of color c_i
	$Q_{k,i} = x$	*i.e.,* x items of color c_k
	$Q_{j,i} = 0 \ \ \forall j \neq i, k$	*i.e.,* 0 items of any other color
agent a_k :	$Q_{i,k} = 2x - 3$	*i.e.,* $2x - 3$ items of color c_i
	$Q_{k,k} = 0$	*i.e.,* 0 items of color c_k
	$Q_{j,k} = 0 \ \ \forall j \neq i, k$	*i.e.,* 0 items of any other color

Since $m = n$, each agent is assigned exactly one color.
Let us now consider the characteristics of any optimal solution.

(i) For each $a_j \in A$, optimal solutions assign color c_j to agent a_j, since it is the only one with that color. There are no invalid assignments for these items, hence the contribution of c_j to the cost of the optimal assignment is 0.
(ii) Consider now a pair (a_i, a_k): agents a_i and a_k are the only ones holding items of color c_i initially; on the other hand, a_i is the only agent with items of color c_k. Hence, an optimal solution may assign color c_i to agent a_k and color c_k to a_i. In this way, the number of invalid assignments for this pair is $x + 3$.

It turns out the cost of an optimal solution is then equal to $(x+3)$ times the number of pairs formed, which is $(n - |A|)/2$.

Let us now consider a possible solution computed by algorithm `Breadth-Balance`. Agent $a_j \in A$ will be the only one to ask for color c_j, in a proper round of stage 3.2, and will succeed in self assigning the color. As before, this assignment will give a zero contribution to the total cost of the solution.

Consider the pair (a_i, a_k), by the choice of x we have that

$$x < \frac{q}{2^{r+1}} \leq x + 3 \leq 2x - 3 \leq \frac{q}{2^r},$$

hence, both a_i and a_k will ask for color c_i in round r of stage 3.2. As a_i is ancestor of a_k, a_i it will self assign color c_i, while a_k will self assign one of the colors for which it has zero items and some other agent x items. For the sake of presentation, let us assume that this color is c_k.

The total number of invalid assignments for the pair is $(2x - 3)$ (due to the assignment of c_i to a_i) plus x (due to the assignment of c_k to a_k). Hence, the cost of the solution found by algorithm `Breadth-Balance` is equal to $\frac{n-|A|}{2} \cdot (3x - 3)$.

It follows that the ratio between the cost of the approximated solution and the cost of optimal solution is $\frac{3x-3}{x+3}$ which can be made arbitrarily close to 3.

4 Conclusions and further work

In this paper we studied the distributed balanced color assignment problem [3] in general graph topologies. We presented a lower bound on the message complexity of the problem, described a distributed algorithm and showed that the algorithm has optimal message complexity for graphs with diameter linear in the number of nodes. Finally, we show that the algorithm finds an approximated solution to the problem, with tight approximation factor equal to three.

Future research will involve experiments to investigate, in practice, how far the computed solution is from the optimal one, and the study of the specific case of graphs with constant diameter.

References

1. B. Awerbuch, *Optimal Distributed Algorithm for Minimum Weight Spanning Tree, Counting, Leader Election and Other Problems*, SIAM Journal on Computing, 1987.
2. C. J. Chang and P. H. Ho, *The β-assignment problems*, European Journal of Operational Research 104, 1998.
3. Gianluca De Marco, Mauro Leoncini and Manuela Montangero, *Distributed algorithm for a color assignment on asynchronous rings*, Proc. 20th International Parallel and Distributed Processing Symposium (IPDPS06), 2006.
4. R. G. Gallager , P. A. Humblet and P. M. Spira, *A Distributed Algorithm for Minimum-Weight Spanning Trees*, ACM Transactions on Programming Languages and Systems (TOPLAS), v.5 n.1, p.66-77, Jan. 1983.
5. D. Peleg, Distributed Computing: A Locality-Sensitive Approach, *SIAM*, Philadelphia, PA, 2000.
6. Y. Zhu and T.Y. Cheung, *A new distributed breadth-first search algorithm.* Information Processing Letters, 25: 239-333, 1987.