Equitable $d$-degenerate choosability of graphs*

Ewa Drgas-Burchardt$^1$, Hanna Furmańczyk$^2$, and Elżbieta Sidorowicz$^1$

$^1$ Faculty of Mathematics, Computer Science and Econometrics, University of Zielona Góra, Prof. Z. Szafrana 4a, 65-516 Zielona Góra, Poland
$\{E.Drgas-Burchardt,E.Sidorowicz\}@wmie.uz.zgora.pl
$^2$ Institute of Informatics, Faculty of Mathematics, Physics and Informatics, University of Gdańsk, 80-309 Gdańsk, Poland
hanna.furmanczyk@ug.edu.pl

Abstract. Let $\mathcal{D}_d$ be the class of $d$-degenerate graphs and let $L$ be a list assignment for a graph $G$. A colouring of $G$ such that every vertex receives a colour from its list and the subgraph induced by vertices coloured with one color is a $d$-degenerate graph is called the $(L,\mathcal{D}_d)$-colouring of $G$. For a $k$-uniform list assignment $L$ and $d \in \mathbb{N}_0$, a graph $G$ is equitably $(L,\mathcal{D}_d)$-colorable if there is an $(L,\mathcal{D}_d)$-colouring of $G$ such that the size of any colour class does not exceed $\lceil V(G)/k \rceil$. An equitable $(L,\mathcal{D}_d)$-colouring is a generalization of an equitable list coloring, introduced by Kostochka at al., and an equitable list arboricity presented by Zhang. Such a model can be useful in the network decomposition where some structural properties on subnets are imposed.

In this paper we give a polynomial-time algorithm that for a given $(k,d)$-partition of $G$ with a $t$-uniform list assignment $L$ and $t \geq k$, returns its equitable $(L,\mathcal{D}_{d-1})$-colouring. In addition, we show that 3-dimensional grids are equitably $(L,\mathcal{D}_1)$-colorable for any $t$-uniform list assignment $L$ where $t \geq 3$.

Keywords: equitable choosability · hereditary classes · $d$-degenerate graph

1 Motivation and preliminaries

In last decades, a social network graphs, describing relationship in real life, started to be very popular and present everywhere. Understanding key structural properties of large-scale data networks started to be crucial for analyzing and optimizing their performance, as well as improving their security. This topic has been attracting attention of many researches, recently (see [1,5,6,10]). We consider one of problems connected with the decomposition of networks into smaller pieces fulfilling some structural properties. For example, we may desire that, for some security reason, the pieces are acyclic or even independent. This is because of in such a piece we can easily and effectively identify a node failure since the local structure around such a node in this piece is so clear that it

* The short version of the paper was accepted to be published in proceedings of IWOCA2020
can be easily tested using some classic algorithmic tools [10]. Sometimes, it is also desirable that the sizes of pieces are balanced. It helps us to maintain the whole communication network effectively. Such a problem can be modeled by minimization problems in graph theory, called an equitable vertex arboricity or an equitable vertex colourability of graphs. Sometimes we have some additional requirements on vertices/nodes that can be modeled by a list of available colours. So, we are interested in the list version, introduced by Kostochka, Pelsmajer and West [3] (an independent case), and by Zhang [9] (an acyclic case).

In colourability and arboricity models the properties of a network can be described in the language of the upper bound on the minimum degree, i.e. each colour class induces a graph whose each induced subgraph has the minimum degree bounded from above by zero or one, respectively. In the paper we consider the generalization of these models in which each colour class induces a graph whose each induced subgraph has the minimum degree bounded from above by some natural constant.

Let $N_0 = \mathbb{N} \cup \{0\}$. For $d \in N_0$, the graph $G$ is $d$-degenerate if $\delta(H) \leq d$ for any subgraph $H$ of $G$, where $\delta(H)$ denotes the minimum degree of $H$. The class of all $d$-degenerate graphs is denoted by $D_d$. In particular, $D_0$ is the class of all edgeless graphs and $D_1$ is the class of all forests. A $D_d$-coloring is a mapping $c : V(G) \to \mathbb{N}$ such that for each $i \in \mathbb{N}$ the set of vertices coloured with $i$ induces a $d$-degenerate graph. A list assignment $L$, for a graph $G$, is a mapping that assigns a nonempty subset of $\mathbb{N}$ to each vertex $v \in V(G)$. Given $k \in \mathbb{N}$, a list assignment is called a $(k, \mathbb{N})$-choosable if for each vertex $v \in V(G)$, there is such an ordering $\{x_i^j\}$ of vertices of $S_j$ that

$$|N_G(x_i^j) \cap (S_1 \cup \cdots \cup S_{j-1})| \leq di - 1,$$

for every $i \in \{1, \ldots, k\}$. Observe that if $S_1 \cup \cdots \cup S_{\eta+1}$ is a $(k, d)$-partition of $G$, then $\eta + 1 = \lceil |V(G)|/k \rceil$. Moreover, immediately by the definition, each $(k, d)$-partition of $G$ is also its $(k, d+1)$-partition. Surprisingly, the monotonicity of the $(k, d)$-partition with respect to the parameter $k$ is not so easy to analyze.
We illustrate this fact by Example 1. Note that for integers $k, d$ the complexity of deciding whether $G$ has a $(k, d)$-partition is unknown. The main result of this paper is as follows.

**Theorem 1.** Let $k, d, t ∈ \mathbb{N}$ and $t ≥ k$. If a graph $G$ has a $(k, d)$-partition, then it is equitably $(t, \mathcal{D}_{d-1})$-choosable. Moreover, there is a polynomial-time algorithm that for any graph with a given $(k, d)$-partition and for any $t$-uniform list assignment $L$ returns an equitable $(L, \mathcal{D}_{d-1})$-colouring of $G$. 

The first statement of Theorem 1 generalizes the result obtained in [3] for $d = 1, 2$. In this paper we present an algorithm that confirms both, the first and second statements of Theorem 1 for all possible $d$. The algorithm, given in Section 2, for a given $(k, d)$-partition of $G$ with $t$-uniform list assignment $L$ returns its equitable $(L, \mathcal{D}_{d-1})$-colouring. Moreover, in Section 3 we give a polynomial-time algorithm that for a given 3-dimensional grid finds its $(3, 2)$-partition, what, in consequence, implies $(t, \mathcal{D}_1)$-choosability of 3-dimensional grids for every $t ≥ 3$.

## 2 The proof of Theorem 1

### 2.1 Background

For $S ⊆ V(G)$ by $G − S$ we denote a subgraph of $G$ induced by $V(G) \setminus S$. We start with a generalization of some results given in [3, 4, 9] for classes $\mathcal{D}_0$ and $\mathcal{D}_1$.

**Proposition 1.** Let $k, d ∈ \mathbb{N}$ and let $S$ be a set of distinct vertices $x_1, \ldots, x_k$ of a graph $G$. If $G − S$ is equitably $(k, \mathcal{D}_{d-1})$-choosable and

$$|N_G(x_i) \setminus S| ≤ di − 1$$

holds for every $i ∈ \{1, \ldots, k\}$, then $G$ is equitably $(k, \mathcal{D}_{d-1})$-choosable.

**Proof.** Let $L$ be a $k$-uniform list assignment for $G$ and let $c$ be an equitable $(L|_{V(G) \setminus S}, \mathcal{D}_{d-1})$-colouring of $G − S$. Thus each colour class in $c$ has the cardinality at most $\lceil |V(G)| / k \rceil$ and induces in $G − S$, and consequently in $G$, a graph from $\mathcal{D}_{d-1}$. We extend $c$ to $(V(G) \setminus S) \cup \{x_k\}$ by assigning to $x_k$ a colour from $L(x_k)$ that is used on vertices in $N_G(x_k) \setminus S$ at most $d − 1$ times. Such a colour always exists because $|N_G(x_k) \setminus S| ≤ dk − 1$ and $|L(x_k)| = k$. Next, we colour vertices $x_{k−1}, \ldots, x_1$, sequentially, assigning to $x_i$ a colour from its list that is different from colours of all vertices $x_{i+1}, \ldots, x_k$ and that is used at most $d − 1$ times in $N_G(x_i) \setminus S$. Observe that there are at least $i$ colours in $L(x_i)$ that are different from $c(x_{i+1}), \ldots, c(x_k)$, and, since $|N_G(x_i) \setminus S| ≤ di − 1$, $i ∈ \{1, \ldots, k−1\}$, then such a choice of $c(x_i)$ is always possible. Next, the colouring procedure forces that the cardinality of every colour class in the extended colouring $c$ is at most $\lceil |V(G)| / k \rceil$. Let $G_i = G[(V(G) \setminus S) \cup \{x_1, \ldots, x_i\}]$. It is easy to see that for each $i ∈ \{1, \ldots, k\}$ each colour class in $c_{V(G_i)}$ induces a graph belonging to $\mathcal{D}_{d-1}$. In particular this condition is satisfied for $G_1$, i.e. for $G$. Hence $c$ is an equitable $(L, \mathcal{D}_{d-1})$-colouring of $G$ and $G$ is equitably $(k, \mathcal{D}_{d-1})$-choosable. \qed
Note that if a graph \( G \) has a \((k, d)\)-partition, then one can prove that \( G \) is equitably \((k, D_{d-1})\)-choosable by applying Proposition 1 several times. In general, the equitable \((k, D_{d-1})\)-choosability of \( G \) does not imply the equitable \((t, D_{d-1})\)-choosability of \( G \) for \( t \geq k \). Unfortunately, if \( G \) has a \((k, d)\)-partition, then \( G \) may have neither a \((k + 1, d)\)-partition nor a \((k - 1, d)\)-partition. The infinite family of graphs defined in Example 1 confirms the last fact.

**Example 1.** Let \( q \in \mathbb{N} \) and let \( G_1, \ldots, G_{2q+1} \) be vertex-disjoint copies of \( K_6 \) such that \( V(G_i) = \{v_1^i, \ldots, v_6^i\} \) for \( i \in \{1, \ldots, 2q + 1\} \). Let \( G(q) \) (cf. Fig. 1) be the graph resulted by adding to \( G_1, \ldots, G_{2q+1} \) edges that join vertices of \( G_i \) with vertices of \( G_{i-1} \), \( i \in \{2, \ldots, 2q + 1\} \), in the following way:

- For \( i \) even:
  - \( N_{G_{i-1}}(v_1^i) = \emptyset \)
  - \( N_{G_{i-1}}(v_2^i) = \{v_1^{i-1}\} \)
  - \( N_{G_{i-1}}(v_3^i) = \{v_2^{i-1}, v_3^{i-1}\} \)
  - \( N_{G_{i-1}}(v_4^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}\} \)
  - \( N_{G_{i-1}}(v_5^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}, v_4^{i-1}, v_5^{i-1}, v_6^{i-1}\} \)
  - \( N_{G_{i-1}}(v_6^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}, v_4^{i-1}, v_5^{i-1}\} \)

- For \( i \) odd:
  - \( N_{G_{i-1}}(v_1^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}, v_4^{i-1}, v_5^{i-1}, v_6^{i-1}\} \)
  - \( N_{G_{i-1}}(v_2^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}, v_4^{i-1}, v_5^{i-1}, v_6^{i-1}\} \)
  - \( N_{G_{i-1}}(v_3^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}\} \)
  - \( N_{G_{i-1}}(v_4^i) = \{v_1^{i-1}, v_2^{i-1}, v_3^{i-1}\} \)
  - \( N_{G_{i-1}}(v_5^i) = \{v_1^{i-1}\} \)
  - \( N_{G_{i-1}}(v_6^i) = \emptyset \)

**Fig. 1.** A draft of graph \( G(2) \) from Example 1

The construction of \( G(q) \) immediately implies that for every \( q \in \mathbb{N} \) the graph \( G(q) \) has a \((6, 1)\)-partition. Also, observe that \( \deg_{G(q)}(v) \geq 5 \) for each vertex \( v \) of \( G(q) \). Suppose that \( G(q) \) has a \((5, 1)\)-partition \( S_1 \cup \cdots \cup S_{q+1} \) with \( S_{q+1} = \{x_1^{q+1}, \ldots, x_5^{q+1}\} \) such that \( |N_{G(q)}(x_1^{q+1}) \cap (S_1 \cup \cdots \cup S_{q})| \leq i - 1 \). Thus \( |N_{G(q)}(x_1^{q+1}) \cap (S_1 \cup \cdots \cup S_{q})| = 0 \) and consequently \( \deg_{G(q)}(x_1^{q+1}) \leq 4 \), contradicting our previous observation. Hence, \( G(q) \) has no \((5, 1)\)-partition. Next, we will show that \( G(q) \) has no \((7, 1)\)-partition for \( q \geq 2 \).

**Proposition 2.** For every integer \( q \) satisfying \( q \geq 2 \) the graph \( G(q) \) constructed in Example 1 has no \((7, 1)\)-partition.
Proof. Let $G = G(q)$ and suppose that $S'_1 \cup \cdots \cup S'_{\eta+1}$ is a $(7, 1)$-partition of $G$. Let $x_1^{\eta+1}, x_2^{\eta+1}, \ldots, x_{\eta+1}^{\eta+1}$ be an ordering of vertices of $S'_{\eta+1}$ such that $|N_G(x_i^{\eta+1}) \cap (S'_{1} \cup \cdots \cup S'_{\eta})| \leq i - 1$. Thus $\deg_G(x_i^{\eta+1}) \leq 6$. Since for every vertex $v \in V(G) \setminus \{v_{\eta+1}^{2\eta+1}, x_6^{2\eta+1}\}$ it holds that $\deg_G(v) \geq 7$, we have $x_1^{\eta+1} = v_6^{2\eta+1}$ or $x_1^{\eta+1} = v_6^{2\eta+1}$ and furthermore $N_G(x_1^{\eta+1}) \subseteq S'_{\eta+1}$. So, $\{v_1^{2\eta+1}, \ldots, v_6^{2\eta+1}\} \subseteq S'_{\eta+1}$ and we must recognize the last vertex of $S'_{\eta+1}$. Since every vertex in $S'_{\eta+1}$ has at most 6 neighbors outside $S'_{\eta+1}$, it follows that either $v_1^{2\eta} \in S'_{\eta+1}$ or $v_2^{2\eta} \in S'_{\eta+1}$. Let $G' = G - S'_{\eta+1}$, $S'_{\eta} = \{x_1^{\eta}, x_2^{\eta}, \ldots, x_{\eta}^{\eta}\}$, and $|N_{G'}(x_i^{\eta}) \cap (S'_{1} \cup \cdots \cup S'_{\eta-1})| \leq i-1$.

Case 1. $v_1^{2\eta} \in S'_{\eta+1}$

Since $\deg_G(x_1^{\eta}) \leq 6$, we have that either $x_1^{\eta} = v_2^{2\eta}$ or $x_1^{\eta} = v_3^{2\eta}$. In the first case, $\{v_1^{2\eta-1}, v_2^{2\eta}, \ldots, v_6^{2\eta}\} \subseteq S'_{\eta}$ and we must recognize the last vertex of $S'_{\eta}$. However, $v_2^{2\eta-1}$ has 10 neighbors outside $\{v_1^{2\eta-1}, v_2^{2\eta}, \ldots, v_6^{2\eta}\}$. It contradicts the condition $v_1^{\eta-1} \in S'_{\eta}$, since every vertex in $S'_{\eta}$ can have at most 6 neighbors outside. In the second case, $\{v_2^{2\eta-2}, v_3^{2\eta-1}, v_2^{2\eta}, \ldots, v_6^{2\eta}\} \subseteq S'_{\eta}$. Similarly, we can observe that $v_2^{2\eta-1}$ has 8 neighbors outside $S'_{\eta}$, which contradicts the fact $v_2^{2\eta-1} \in S'_{\eta}$.

Case 2. $v_2^{2\eta} \in S'_{\eta+1}$

Since $\deg_G(x_2^{\eta}) \leq 6$, we have that either $x_2^{\eta} = v_2^{2\eta}$ or $x_2^{\eta} = v_3^{2\eta}$. By the same argument as in Case 1 we can observe that $x_2^{\eta} \neq v_3^{2\eta}$, and so we may assume that $x_2^{\eta} = v_1^{2\eta}$ and $\{v_1^{2\eta}, v_3^{2\eta}, \ldots, v_6^{2\eta}\} \subseteq S'_{\eta}$. Thus we must recognize two last vertices of $S'_{\eta}$. Since each of $v_1^{2\eta-1}, v_2^{2\eta-1}$ has at least 8 neighbors outside $\{v_1^{2\eta}, v_3^{2\eta}, \ldots, v_6^{2\eta}\}$, we have $\{v_1^{2\eta-1}, v_2^{2\eta-1}, v_3^{2\eta-1}\} \cap S'_{\eta} = \emptyset$. Hence we conclude that $v_3^{2\eta}, v_4^{2\eta}, v_5^{2\eta}, v_6^{2\eta}$ have at least two neighbors outside $S'_{\eta}$, and so none of them is $x_2^{\eta}$, because $x_2^{\eta}$ has at most one neighbor outside $S'_{\eta}$. Furthermore, every vertex in $V(G') \setminus \{v_1^{2\eta}, v_3^{2\eta}, \ldots, v_6^{2\eta}\}$ has at least 5 neighbors outside $\{v_1^{2\eta}, v_3^{2\eta}, \ldots, v_6^{2\eta}\}$, thus also none of them is $x_2^{\eta}$. Since $x_2^{\eta}$ does not exist, the $(7, 1)$-partition of $G$ does not exist either.

2.2 Algorithm

Now we are ready to present the algorithm that confirms both statements of Theorem 1. Note that Proposition 1 and the induction procedure could be used to prove the first statement of Theorem 1 for $t = k$ but this approach seems to be useless for $t > k$, as we have observed in Example 1. Mudrock et al. [7] proved the lack of monotonicity for the equitable $(k, D_0)$-choosability with respect to the parameter $k$. It motivates our approach for solving the problem. To simplify understanding we give the main idea of the algorithm presented in the further part. It can be expressed in a few steps (see also Fig. 2):

- on the base of the given $(k, d)$-partition $S_1 \cup \cdots \cup S_{\eta+1}$ of $G$, we create the list $S$, consisting of the elements from $V(G)$, whose order corresponds to the order in which the colouring is expanded to successive vertices in each $S_j$ (cf. the proof of Proposition 1).
– let $|V(G)| = \beta t + r_2$, $1 \leq r_2 \leq t$; we colour $r_2$ vertices from the beginning of $S$ taking into account the lists of available colours; we delete the colour assigned to $v$ from the lists of available colours for vertices from $N_{G}^{col}(d, v)$;
– let $|V(G)| = \beta(\gamma k + r) + r_2 = \beta\gamma k + (\rho k + x) + r_2$; observe that $r \equiv t \pmod{k}$ and $\rho k + x \equiv 0 \pmod{r}$; we colour $\rho k + x$ vertices taking into account the lists of available colours in such a way that every sublist of length $k$ is formed by vertices coloured differently (consequently, every sublist of length $r$ is coloured differently); we delete the colour assigned here into $\beta$ sets each one of cardinality $r$; we delete $c(v)$ from the lists of vertices from $N_{G}^{col}(d, v)$;
– we extend the list colouring into the uncoloured $\beta\gamma k$ vertices by colouring $\beta$ groups of $\gamma k$ vertices; first, we associate each group of $\gamma k$ vertices with a set of $r$ vertices coloured in the previous step (for different groups these sets are disjoint); next, we color the vertices of each of the group using $\gamma k$ different colors that are also different from the colors of $r$ vertices of the set associated with this group;
– our final equitable list colouring is the consequence of a partition of $V(G)$ into $\beta + 1$ coloured sets, each one of size at most $t$ and each one formed by vertices coloured differently.

Now we illustrate Equitable $(L, D_{d-1})$-colouring using a graph constructed in Example 2.

**Example 2.** Let $G_1, G_2$ be two vertex-disjoint copies of $K_5$ and $V(G_i) = \{v_1^i, \ldots, v_5^i\}$ for $i \in \{1, 2\}$. We join every vertex $v_j^i$ to $v_j^1, v_{j+1}^1, \ldots, v_5^1$ for $j \in \{1, 2, 3, 4, 5\}$. Next, we add a vertex $w_j^i$ and join it with $v_j^i$ for $i \in \{1, 2\}, j \in \{1, 2, 3, 4, 5\}$. In addition, we join $w_j^i$ to arbitrary two vertices in $\{v_p^q : (q < i) \lor (q = i \land p < j)\} \cup \{w_j^q : (q < i) \lor (q = i \land p < j)\}$, $i \in \{2, 3, 4, 5\}, j \in \{1, 2\}$. Let $G$ be a resulted graph. Observe that $|V(G)| = 20$ and the partition $S_1 \cup S_2 \cup \ldots \cup S_{10}$ of $V(G)$ such that $S_{p+1} = \{v_{r+1}^p, w_{s+1}^p\}$ for $p \in \{0, \ldots, 9\}$, where $s = \left[\frac{p}{5}\right], r \equiv p \pmod{5}$ is a $(2, 3)$-partition of $G$. 

\[\begin{array}{c}
S_1 \quad \cdots \quad S_{\beta\gamma+1} \quad S_{\beta\gamma+2} \quad \cdots \quad S_{\eta+1}
\end{array}\]

**Fig. 2.** An exemplary illustration of the input of the Equitable $(L, D_{d-1})$-colouring algorithm.
Algorithm 1: Equitable \((L, D_{d-1})\)-colouring(G)

**Input**: Graph \(G\) on \(n\) vertices; \(L\) - \(t\)-uniform list assignment; a \((k, d)\)-partition \(S_1 \cup \cdots \cup S_{\eta+1}\) of \(G\), given by lists \(S_1 = (x_1^1, \ldots, x_r^1)\) and \(S_j = (x_1^j, \ldots, x_k^j)\) for \(j \in \{2, \ldots, \eta+1\}\).

**Output**: Equitable \((L, D_{d-1})\)-colouring of \(G\).

1. **Initialization**;
2. \(S := \text{empty}; L_R := \text{empty}; L_X := \text{empty};\)
3. for \(j := 1\) to \(\eta + 1\) do
   4. add \(\text{reverse}(S_j)\) to \(S\); //\text{reverse} is the procedure for reversing lists
5. end
6. \(\beta := \lceil n/t \rceil - 1;\)
7. if \(n \equiv 0 \pmod{t}\) then
   8. \(r_2 := t\)
   9. else
   10. \(r_2 := n \pmod{t};\)
11. end
12. \(\gamma := t \div k; r := t \pmod{k}; \rho := \beta r \div k; x := \beta r \pmod{k};\)
13. take and delete \(r_2\) elements from the beginning of \(S\), and add them, vertex by vertex, to list \(L_R;\)
14. \(\text{COLOUR\_LIST}(L_R, r_2);\)
15. take and delete \(x\) elements from the beginning of \(S\), and add them, vertex by vertex, to list \(L_X;\)
16. \(\text{COLOUR\_LIST}(L_X, x);\)
17. \(S_{\text{col}} := L_X;\)
18. for \(j = 1\) to \(\rho\) do
   19. take and delete \(k\) elements from the beginning of \(S\), and add them, vertex by vertex, to list \(S';\)
20. \(\text{COLOUR\_LIST}(S', k);\)
21. \(S_{\text{col}} := S_{\text{col}} + S';\)
22. end
23. \(\text{REORDER}(S_{\text{col}});\)
24. \(\overline{S} := S;\) //an auxiliary list
25. \(\text{MODIFY\_COLOUR\_LISTS}(S_{\text{col}}, \overline{S});\)
26. \(\text{COLOUR\_LIST}(S, \gamma k);\)

\textbf{Fig. 3.} An exemplary graph \(G\) depicted in Example 2 with an exemplary colouring returned by Equitable \((L, D_{d-1})\)\)-colouring(G).
Procedure 2: colour_list\( \left( S', p \right) \)

**Input**: List \( S' \) of vertices; integer \( p \).//the length of \( S' \) is multiple of \( p \)

**Output**: \( L \)-colouring of the vertices from \( S' \).

//The procedure also modifies a global variable of the list assignment \( L \).

1. initialization;
2. while \( S' \neq \) empty do
3. let \( S'' \) be the list of the \( p \) first elements of \( S' \);
4. \( C := \emptyset \); //set \( C \) is reserved for the colours being assigned to vertices of \( S'' \)
5. while \( S'' \neq \) empty do
6. let \( v \) be the first element of \( S'' \);
7. \( L(v) := L(v) \setminus C \);
8. \( c(v) := \text{colour} \_\text{vertex}(v) \);
9. delete vertex \( v \) from \( S' \) and \( S'' \); \( C := C \cup \{ c(v) \} \);
10. end
11. end

Procedure 3: \text{colour}\_\text{vertex}(v)

**Input**: Vertex \( v \) of the graph \( G \).

**Output**: \( L \)-colouring of the vertex \( v \).

//The procedure modifies also a global variable of the list assignment \( L \).

1. initialization;
2. \( c(v) := \) any colour from \( L(v) \);
3. delete \( c(v) \) from \( L(w) \) for all \( w \in N^\text{col}_G(d, v) \); //\( d \) is a global variable
4. return \( c(v) \);

Procedure 4: reorder\((S')\)

**Input**: List \( S' \) of coloured vertices of \( G \).

**Output**: List \( S' \) - reordered in such a way that every its sublist of length \( k \) is formed by vertices being coloured with different colours.

1. initialization;
2. \( S_{aux} := \emptyset \); //an auxiliary list
3. take and delete \( r_1 \) elements from \( S' \), and add them, vertex by vertex, to \( S_{aux} \);
4. for \( j = 1 \) to \( \eta - \beta \gamma \) do
5. \( P := \emptyset \);
6. take and delete first \( k \) elements from \( S' \), and add them to set \( P \);
7. for \( i = 1 \) to \( k \) do
8. let \( v \) be a vertex from \( P \) such that \( c(v) \) is different from colours of the last \( k - 1 \) vertices of \( S_{aux} \); add \( v \) to the end of \( S_{aux} \);
9. end
10. end
11. \( S' := S_{aux} \);
Procedure 5: MODIFY_COLOURLIST($L_1, L_2$)

**Input**: List $L_1$ of $\beta r$ coloured vertices and list $L_2$ of $\beta \gamma k$ uncoloured vertices.

**Output**: Modified colour list assignment $L$ for vertices of $L_2$. 

\[ //L\text{ is a global variable} \]

1. initialization;
2. $C := \emptyset$; \hspace{1cm} // $C$ is a set of colours of vertices from the depicted part of $L_1$
3. for $i = 1$ to $\beta$
4. \hspace{1cm} take and delete first $r$ vertices from $L_1$;
5. \hspace{1cm} let $C$ be the set of colours assigned to them;
6. \hspace{1cm} for $j = 1$ to $\gamma k$
7. \hspace{1cm} \hspace{1cm} let $v$ be the first vertex from $L_2$;
8. \hspace{1cm} \hspace{1cm} $L(v) := L(v) \setminus C$; delete $v$ from $L_2$;
9. \hspace{1cm} end
10. end

For the purpose of Example 2 we assume the following 3-uniform list assignment for the graph from Figure 3

\[ L(v^j_i) = \{1, 2, 3\}; \text{ for } i \in \{1, 2, 3, 4\}, \]

\[ L(w^j_i) = \{2, 3, 4\}, \text{ and } L(u^j_i) = \{1, 2, 4\}, \text{ for } j \in \{1, 2, 3, 4, 5\}, \text{ while given } (2, 3)-\text{partition of } G \text{ is: } S_p = \{w^{s+1}_{r+1}, v^{s+1}_{r+1}\}, \text{ where } s = \lceil \frac{p}{5} \rceil, r \equiv \eta \pmod{5}, p \in [10]. \]

Thus EQUIABLE ($L, D_{d-1}$)-COLOURING returns equitable ($L, D_2$)-coloring of $G$. Note, that $20 = |V(G)| = \eta \cdot k + r_1 = 9 \cdot 2 + 2$. While on the other hand, we have $20 = |V(G)| = \beta \cdot t + r_2 = 6 \cdot 3 + 2$. Observe that $x = 0$. When we colour a vertex, we always choose the first colour on its list. The list $S$ determined in lines 3-5 of EQUIABLE ($L, D_{d-1}$)-COLOURING and the colours assigned to first part of vertices of $S$ (lines 18-25) are as follows:

\[ S = (v^1_1, w^1_1, v^4_2, w^2_2, v^1_3, w^1_3, v^1_4, w^1_4, v^4_5, w^2_5, v^1_6, w^2_6, \ldots, v^2_9, w^2_9) \]

\[ \text{colours: } 1 \hspace{1cm} 2 \hspace{1cm} 1 \hspace{0.5cm} 2 \hspace{0.5cm} 1^* \hspace{0.5cm} 2 \hspace{0.5cm} 2 \hspace{0.5cm} 3 \]

\[ \beta \hspace{0.5cm} \gamma \hspace{0.5cm} k \]

\[^*: \text{after colouring } v^3_3 \text{ with } 1, L(v^3_3) = \{2, 3\} - \text{the result of line 8} \text{in the COLOUR_VERTEX procedure.} \]

List $S_{col}$ after REORDER($S_{col}$): $(v^1_2, w^1_2, v^3_3, w^3_3, v^1_4, v^1_4)$ with corresponding colours: $(1, 2, 1, 2, 3, 2)$.

\[ \overline{S} = (v^1_5, w^1_5, v^1_5, w^1_7, v^1_5, w^1_5, v^3_3, v^3_3, v^3_3, v^3_3, v^3_3, v^3_3, \ldots, v^3_9, w^3_9) \]

\[ \text{lists after procedure} \hspace{1cm} 2 \hspace{1cm} 2 \hspace{1cm} 3 \hspace{1cm} 1 \hspace{1cm} 2 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1 \hspace{1cm} 1 \]

\[ \text{MODIFY_COLOURLIST} \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \hspace{1cm} 4 \]

\[ \text{final } c(v) \hspace{1cm} 2 \hspace{1cm} 3 \hspace{1cm} 3 \hspace{1cm} 1 \hspace{1cm} 2 \hspace{1cm} 4 \hspace{1cm} 1 \hspace{1cm} 4 \hspace{1cm} 1 \hspace{1cm} 2 \hspace{1cm} 1 \hspace{1cm} 4 \]

To prove the correctness of the EQUIABLE ($L, D_{d-1}$)-COLOURING algorithm, we give some observations and lemmas.
Observation 2 The colour function $c$ returned by Equitable $(L, D_{d-1})$-colouring is constructed step by step. In each step, $c(v)$ is a result of Colour_Vertex($v$) and this value is not changed further.

Observation 3 The list assignment $L$, as a part of the input of Equitable $(L, D_{d-1})$-colouring, is modified for a vertex $v$ by Colour_List or by Modify_ColourLists.

Lemma 1. Every time when Equitable $(L, D_{d-1})$-colouring($G$) calls Colour_Vertex($v$), the list $L(v)$ for $v$ is non-empty, i.e. Colour_Vertex($v$) is always executable.

Proof. Note that Colour_Vertex is called by Colour_List. Let

$R = \{ v \in V(G) : \text{colour}_\text{Vertex}(v) \text{ is called when Colour_List}(L_R, r_2) \}$

in line 15 of Equitable $(L, D_{d-1})$-colouring is executed\},

$X = \{ v \in V(G) : \text{colour}_\text{Vertex}(v) \text{ is called when Colour_List}(L_X, x) \}$

in line 18 of Equitable $(L, D_{d-1})$-colouring is executed\}.

Let $V_1 := S_1 \cup \cdots \cup S_{\gamma+1} - \beta \gamma$, $V_2 := V(G) \setminus V_1 = S_{\gamma+1} - (\beta \gamma - 1) \cup \cdots \cup S_{\eta+1}$. Note that $V_1 \setminus (R \cup X) = \{ v \in V(G) : \text{colour}_\text{Vertex}(v) \text{ is called when Colour_List}(S', k) \}$

in line 22 of Equitable $(L, D_{d-1})$-colouring is executed\}.

Observe that $|R| = r_2, |X| = x, |V_1 \setminus (R \cup X)| = \rho k, |V_2| = \beta \gamma k$.

Case 1. $v \in R$.

In this case, the vertex $v$ is coloured by Colour_List$(L_R, r_2)$ in line 15 of Equitable $(L, D_{d-1})$-colouring. Since $|R| = r_2$, the while loop in line 2 of Colour_List is executed only once. The while loop in line 3 of Colour_List is executed $r_2$ times. Suppose, $v$ is a vertex such that Colour_Vertex($v$) is called in the $i$-th execution of the while loop in line 3 of Colour_List. By Observation 3, the fact that Equitable $(L, D_{d-1})$-colouring has not called Modify_ColourLists so far, and because it is the first time when Colour_List works, we have $|C| = i - 1$, and $L(v) \setminus C$ is the current list of $v$. Since $t \geq r_2$, the list of $v$ is non-empty.

Case 2. $v \in X$.

This time, the vertex $v$ is coloured by Colour_List$(L_X, x)$ called in line 18 of Equitable $(L, D_{d-1})$-colouring. Similarly as in Case 1, the while loop in line 2 of Colour_List is executed only once and the while loop in line 3 of Colour_List is executed $x$ times. Suppose that $v$ is a vertex such that Colour_Vertex($v$) is called in the $i$-th iteration of the while loop in line 3 of Colour_List. Observe that properties of the $(k, d)$-partition $S_1 \cup \cdots \cup S_{\eta+1}$ of $G$ (given as the input of Equitable $(L, D_{d-1})$-colouring) and the Reverse procedure from line 4 of Equitable $(L, D_{d-1})$-colouring imply that $v$ has at most $(x - i + 1)d - 1 + (k - x)$ neighbours $w$ for which Colour_Vertex($w, d$) was executed earlier than Colour_List$(L_X, x)$. More precisely, by the definition of the $(k, d)$-partition, $v$ has at most $(x - i + 1)d - 1$ neighbours in $R \setminus Y$, or by Modifying ColourLists.
where $Y$ consists of the last $k - x$ vertices $w$ for which \texttt{COLOUR\_VERTEX}(w) was executed (being called by \texttt{COLOUR\_LIST}(L_R, r_2)). Thus, at most $k - i$ colours were deleted from $L(v)$ before \texttt{COLOUR\_LIST}(L_X, x) began. If the \texttt{while} loop in line 3 of \texttt{COLOUR\_LIST} is called for the $i$-th time, then $|C| = i - 1$ and so, from the current list $L(v)$ at most $i - 1$ elements were deleted. Furthermore, \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} has not called \texttt{MODIFY\_COLOUR\_LISTS} so far. Thus the current size of $L(v)$ is at least $t - k + 1$, by Observation 3. Since $t \geq k$, the list of $v$ is non-empty.

**Case 3.** $v \in V_1 \setminus (R \cup X)$.

In this case, the vertex $v$ is coloured during the execution of the \texttt{for} loop in line 20 of \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING}. Observe that this loop is executed $\rho$ times, and in each of the executions the \texttt{while} loop in line 2 of \texttt{COLOUR\_LIST} is executed only once, while the \texttt{while} loop in line 5 of \texttt{COLOUR\_LIST} is executed $k$ times.

Suppose that $v$ is a vertex such that \texttt{COLOUR\_VERTEX}(v) is called in the $j$-th execution of the \texttt{for} loop in line 20 of \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} and the $i$-th execution of the \texttt{while} loop in line 5 of \texttt{COLOUR\_LIST}. Observe that properties of the $(k, d)$-partition $S_1 \cup \cdots \cup S_{n+1}$ of $G$ (given as input of \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING}) and the \texttt{REVERSE} procedure from line 3 of \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} imply that $v$ has at most $(k - i + 1)d - 1$ neighbors $w$ for which \texttt{COLOUR\_VERTEX}(w) was executed before the $j$-th execution of the \texttt{for} loop in line 2 of \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} started. Thus from the list of $v$ at most $k - i$ colours were deleted before the $j$-th execution of the \texttt{for} loop in line 2 of \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} started. If the \texttt{while} loop in line 5 of \texttt{COLOUR\_LIST} is called for the $i$-th time, $|C| = i - 1$ and so, from the current list of $v$ at most $i - 1$ elements were deleted. Furthermore, \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} has not called \texttt{MODIFY\_COLOUR\_LISTS} so far. Thus the current size of the list $L(v)$ is at least $t - k + 1$, by Observation 3. Since $t \geq k$, the list of $v$ is non-empty.

**Case 4.** $v \in V_2$.

In this case, the vertex $v$ is coloured by the \texttt{COLOUR\_LIST}(S, $\gamma k$) procedure, called by \texttt{EQUITABLE}($L, D_{d-1}$)-\texttt{COLOURING} in line 20. Since $|V_2| = \beta \gamma k$, the \texttt{while} loop in line 2 of \texttt{COLOUR\_LIST} is executed $\beta$ times and the \texttt{while} loop in line 5 of \texttt{COLOUR\_LIST} is executed $\gamma k$ times. Suppose that $v$ is a vertex for which \texttt{COLOUR\_VERTEX}(v, d) is called in the $j$-th execution of the \texttt{while} loop in line 2 of \texttt{COLOUR\_LIST} and in the $z$-th execution of the \texttt{while} loop in line 5 of \texttt{COLOUR\_LIST}. Let $z = \gamma k + i$, where $0 \leq y \leq \gamma - 1$, $0 \leq i \leq k$.

Similarly as before, properties of the $(k, d)$-partition $S_1 \cup \cdots \cup S_{n+1}$ of $G$ and the \texttt{REVERSE} procedure imply that $v$ has at most $(k - i + 1)d - 1$ neighbors $w$ for which \texttt{COLOUR\_VERTEX}(w) was executed before the $j$-th execution of the \texttt{while} loop in line 2 of \texttt{COLOUR\_LIST} began. Thus from the list of $v$ at most $k - i$ elements were deleted before the $j$-th execution of the \texttt{while} loop in line 2 of \texttt{COLOUR\_LIST} started. Furthermore, after the execution of \texttt{MODIFY\_COLOUR\_LISTS} in line 24 from the list of every vertex in $V_2$ at most $r$ elements were removed. If the \texttt{while} loop in line 5 of \texttt{COLOUR\_LIST} is called for
the \( z \)-th time, then \(|C| = z - 1 \) and so, from the initial list \( L(v) \) at most \( z - 1 \) elements were deleted. Thus, when \( \text{colour}_\text{Vertex}(v) \) is called, then the size of the current list \( L(v) \) is at least \( t - r - (k - i) - (z - 1) = t - r - k - yk + 1 \). Since \( y \leq \gamma - 1 \) and \( t = \gamma k + r \), the list of \( v \) is non-empty. \( \square \)

**Lemma 2.** An output of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring}(G) \) is an \((L, D_{d-1})\text{-colouring}\) of \( G \).

**Proof.** We will show that if \( \text{colour}_\text{Vertex}(v) \) is executed, then an output \( c(v) \) has always the following property. For each subgraph \( H \) of \( G \) induced by vertices \( x \) for which \( \text{colour}_\text{Vertex}(x, d) \) was executed so far with the output \( c(x) = c(v) \), the condition \( \delta(H) \leq d - 1 \) holds. By Observation 2 and Lemma 4 it will imply that an output \( c \) of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring} \) is an \((L, D_{d-1})\text{-colouring}\) of \( G \). Note that it is enough to show this fact for \( H \) satisfying \( v \in V(H) \).

By a contradiction, let \( v \) be a vertex for which the output \( c(v) \) does not satisfy the condition, i.e. \( v \) has at least \( d \) neighbors in the set of vertices for which \( \text{colour}_\text{Vertex} \) was already executed with the output \( c(v) \). But it is not possible because \( c(v) \) was removed from \( L(v) \) when the last (in the sense of the algorithm steps) of the neighbors of \( v \), say \( x \), obtained the colour \( c(v) \) \( (\text{colour}_\text{Vertex}(x) \) removed \( c(x) \) from \( L(v) \) since \( v \in N_{G}^{\text{col}}(d, x) \) in this step). \( \square \)

**Lemma 3.** An output colour function \( c \) of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring}(G) \) satisfies \(|C_i| \leq |V(G)|/t| \), where \( t \) is the part of the input of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring}(G) \) and \( C_i = \{ v \in V(G) : c(v) = i \} \).

**Proof.** Recall that \(|V(G)|/t| = \beta + 1 \). We will show that there exists a partition of \( V(G) \) into \( \beta + 1 \) sets, say \( W_1 \cup \cdots \cup W_{\beta+1} \), such that for each \( i \in \{ 1, \ldots, \beta + 1 \} \) any two vertices \( x, y \) in \( W_i \) satisfy \( c(x) \neq c(y) \). It will imply that the cardinality of every colour class in \( c \) is at most \( \beta + 1 \), giving the assertion.

Note that after the last, \( \rho \)-th execution of the for loop in line 20 of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring} \) the list \( S_{\text{col}} \) consists of the coloured vertices of the set \( V_1 \setminus R \) (observe that \(|V_1 \setminus R| = \beta r \)). The elements of \( S_{\text{col}} \) are ordered in such a way that the first \( x \) ones have different colours and for every \( i \in \{ 1, \ldots, \rho \} \) the \( i \)-th next \( k \) elements have different colours. Now the \( \text{Reorder}(S_{\text{col}}) \) procedure in line 20 of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring} \) changes the ordering of elements of \( S_{\text{col}} \) in such a way that every \( k \) consecutive elements have different colours. Since \( r \in \{ 0, \ldots, k - 1 \} \), it follows that also every \( r \) consecutive elements of this list have different colours. The execution of \( \text{Reorder}(S_{\text{col}}) \) is always possible because of the previous assumptions on \( S_{\text{col}} \).

For \( i \in \{ 1, \ldots, \beta \} \) let \( H_i = S_{\gamma i+1-((\beta-i)+1)\gamma-1} \cup S_{\gamma i+1-((\beta-i)+1)\gamma-2} \cup \cdots \cup S_{\gamma i-1-((\beta-i)+1)\gamma-1} \). Thus \( H_i \cup \cdots \cup H_\beta \) is a partition of \( V_2 \) into \( \beta \) sets, each of the cardinality \( \gamma k \). Note that the vertices of \( H_i \) are coloured when \( \text{colour}_\text{List}(S, \gamma k) \) in line 29 of \( \text{Equitable} \ (L, D_{d-1})\text{-colouring} \) is executed. More precisely, it is during the \( i \)-th execution of the while loop in line 2 of \( \text{colour}_\text{List} \). It guarantees that the vertices of \( H_i \) obtain pairwise different colours. Moreover,
in line 28 of Equitable \((L, D_{d-1})\)-colouring the lists of vertices of \(H_i\) were modified in such a way that the colours of \(i\)-th \(r\) elements from the current list \(S_{col}\) are removed from the list of each element in \(H_i\). Hence, after the execution of \(\text{COLOUR}\_\text{LIST}(S, \gamma k)\) in line 29 of Equitable \((L, D_{d-1})\)-colouring the elements in \(H_i\) obtain colours that are pairwise different and also different from all the colours of \(i\)-th \(r\) elements from the list \(S_{col}\) (recall that \(S_{col}\) consists of the ordered vertices of \(V_1 \setminus R\)). Hence, for every \(i \in \{1, \ldots, \beta\}\) the elements of \(H_i\) and the \(i\)-th \(r\) elements of \(S_{col}\) have pairwise different colours in \(e\) and can constitute \(W_i\). Moreover, the elements of \(R\) constitute \(W_{\beta+1}\). Thus \(|W_{\beta+1}| = r_2\), which finishes the proof.

**Theorem 4.** For a given graph \(G\) on \(n\) vertices, a \(t\)-uniform list assignment \(L\), a \((k, d)\)-partition of \(G\) the Equitable \((L, D_{d-1})\)-colouring \((G)\) algorithm returns \((L, D_{d-1})\)-colouring of \(G\) in polynomial time.

**Proof.** The correctness of the algorithm follows immediately from Lemmas 2–3. To determine the complexity of the entire Equitable \((L, D_{d-1})\)-colouring \((G)\), let us analyze first its consecutive instructions:

- creating the list \(S\) - the for loop in line 4 can be done in \(O(n)\) time
- \(\text{COLOUR}\_\text{LIST}(L_R, r_2)\) in line 15
  This procedure recalls \(\text{COLOUR}\_\text{VERTEX}(v, d)\) in which the most costly operation is the one depicted in line 8 of \(\text{COLOUR}\_\text{VERTEX}\). Since, we colour \(r_2\) initial vertices we do not need to check \(N_G^{col}(d, v)\) for first \(d\) vertices. If \(r_2 \leq d\), then the cost of \(\text{COLOUR}\_\text{LIST}(L_R, r_2)\) is \(O(r_2)\), otherwise \(O(r_2\Delta^2(G))\).
- \(\text{COLOUR}\_\text{LIST}(L_X, x)\) in line 18
  Similarly as in the previous case, the complexity of this part can be bounded by \(O(x\Delta^2(G))\).
- the for loop in line 20
  This loop is executed \(\eta\) times, \(\rho \leq \eta \leq \lceil n/k \rceil\). The complexity of the internal \(\text{COLOUR}\_\text{LIST}(S', k)\) procedure is \(O(k\Delta^2(G))\). So, we have \(O(n\Delta^2(G))\) in total.
- \(\text{REORDER}(S_{col})\) in line 20
  Since \((\eta - \beta \gamma)k \leq n\), then we get \(O(n)\).
- \(\text{MODIFY}\_\text{COLOUR}\_\text{LIST}(S_{col}, S)\)
  The double for loop implies the complexity \(O(\beta \gamma k) = O(n)\).
- \(\text{COLOUR}\_\text{LIST}(S, \gamma k)\)
  The list \(S\) is of length \(\beta \gamma k\), so the while loop in line 2 of \(\text{COLOUR}\_\text{LIST}\) is executed \(\beta\) times, while the internal while loop is executed \(\gamma k\) times. Taking into account the complexity of \(\text{COLOUR}\_\text{VERTEX}\), we get \(O(n\Delta^2(G))\).

We get \(O(n\Delta^2(G))\) as the complexity of the entire Equitable \((L, D_{d-1})\)-colouring \((G)\). \(\square\)

### 3 Grids

Given two graphs \(G_1\) and \(G_2\), the *Cartesian product* of \(G_1\) and \(G_2\), \(G_1 \square G_2\), is defined to be a graph whose vertex set is \(V(G_1) \times V(G_2)\) and the edge set...
consists of all edges joining vertices \((x_1, y_1)\) and \((x_2, y_2)\) when either \(x_1 = x_2\) and \(y_1, y_2 \in E(G_2)\) or \(y_1 = y_2\) and \(x_1, x_2 \in E(G_1)\). Note that the Cartesian product is commutative and associative. Hence the graph \(G_1 \square \cdots \square G_d\) is unambiguously defined for any \(d \in \mathbb{N}\). Let \(P_n\) denote a path on \(n\) vertices. If each factor \(G_i\) is a path on at least two vertices then \(G_1 \square \cdots \square G_d\) is a \(d\)-dimensional grid (cf. Fig. 4). Note that the \(d\)-dimensional grid \(P_{n_1} \square \cdots \square P_{n_d}\), \(d \geq 3\), may be considered as \(n_1\) layers and each layer is the \((d-1)\)-dimensional grid \(P_{n_2} \square \cdots \square P_{n_d}\). We assume \(n_1 \geq \cdots \geq n_d\).

\[
\begin{array}{cccc}
(1,1,1) & (1,2,1) & (1,3,1) & (1,1,2) \\
(1,2,2) & (1,3,2) & (1,2,2) & (1,2,2) \\
(1,1,3) & (1,2,3) & (1,3,3) & (1,1,3) \\
(1,2,4) & (1,3,4) & (1,2,4) & (1,2,4) \\
(1,3,5) & (1,2,5) & (1,3,5) & (1,2,5) \\
(1,1,6) & (1,2,6) & (1,3,6) & (1,1,6) \\
\end{array}
\]

Fig. 4. The 3-dimensional grid \(P_5 \square P_3 \square P_2\).

Let \(P_{n_1} \square \cdots \square P_{n_d}\) denote an incomplete \(d\)-dimensional grid, i.e. a connected graph being a subgraph of \(P_{n_1} \square \cdots \square P_{n_d}\) such that its some initial layers may be empty, the first non-empty layer may be incomplete, while any next layer is complete (cf. Fig. 5). Note that every grid is particular incomplete grid.

\[
\begin{array}{cccc}
(1,1,1) & (1,2,1) & (1,3,1) & (1,1,2) \\
(1,2,2) & (1,3,2) & (1,2,2) & (1,2,2) \\
(1,1,3) & (1,2,3) & (1,3,3) & (1,1,3) \\
(1,2,4) & (1,3,4) & (1,2,4) & (1,2,4) \\
(1,3,5) & (1,2,5) & (1,3,5) & (1,2,5) \\
(1,1,6) & (1,2,6) & (1,3,6) & (1,1,6) \\
\end{array}
\]

Fig. 5. An incomplete grid \(P_{n_1} \square P_{n_2} \square P_2, n \geq 5\).

In this subsection we construct a polynomial-time algorithm that for each 3-dimensional grid finds its (3, 2)-partition (\(\text{PARTITION}^{3d}(G)\)). Application of Theorem 1 implies the main result of this subsection. Note that using a com-
completely different method, the first statement of Theorem 5 has already been proven in [3].

**Theorem 5.** Let \( t \geq 3 \) be an integer. Every 3-dimensional grid is equitably \((t, D_1)\)-choosable. Moreover, there is a polynomial-time algorithm that for every \( t \)-uniform list assignment \( L \) of the 3-dimensional grid \( G \) returns an equitable \((L, D_1)\)-colouring of \( G \).

**Procedure 6: Corner(G)**

1. **Input**: Incomplete non-empty \( d \)-dimensional grid \( G = P_{a_1} \sqcup \ldots \sqcup P_{a_d}, \quad d \geq 2 \).
2. **Output**: Vertex \( y = (a_1, \ldots, a_d) \in V(G) \) such that \( \deg_G(y) \leq d \).
3. **Initialization**;
4. For \( i = 2 \) to \( d - 1 \) do
5. \( a_i = \min \{ x_i : \exists x_{i+1}, \ldots, x_d (a_1, \ldots, a_{i-1}, x_i, \ldots, x_d) \in V(G) \} \).
6. **End**
7. \( a_d = \min \{ x_d : (a_1, \ldots, a_{d-1}, x_d) \in V(G) \} \).
8. **Return** \((a_1, \ldots, a_d)\);

**Theorem 6.** For a given 3-dimensional grid \( G \) the Partition3d\((G)\) algorithm returns a \((3, 2)\)-partition of \( G \) in polynomial-time.

**Proof.** First, observe that thanks to the condition of the if instruction in line 3 the for loop in line 4 is correctly defined and its instruction are performed every time for a graph with at least three vertices. Thus, the Corner procedure in both line 5 and line 7 is called for a non-empty graph, so the returned vertices \( y_1 \) and \( y_2 \) are correctly defined. There is no doubt that the remaining instruction are also correctly defined, and, in consequence, we get sets \( S_1, \ldots, S_{\alpha+1} \) fulfilling conditions: \(|S_1| \leq 3\) and \(|S_j| = 3, \quad j \in \{2, \ldots, \alpha + 1\} \).

Hence, all we need is to prove that the partition \( S_1 \cup \ldots \cup S_{\alpha+1} \) fulfills also the condition (1) in the definition of the \((k, s)\)-partition of \( G \). Indeed, let us consider set \( S_j = \{ y_1, y_2, y_3 \} \). If the condition from line 6 of Partition3d\((G)\) is true, then \( \deg_{G-S_j}(y_1) = 1, \deg_{G-S_j}(y_2) \leq 3, \) while \( \deg_{G-S_j}(y_3) \leq 5 \). The same inequalities hold whenever the condition in line 14 holds. If the condition in line 18 is true, then \( \deg_{G-S_j}(y_1) \leq 1, \deg_{G-S_j}(y_2) \leq 3, \) while \( \deg_{G-S_j}(y_3) \leq 4 \).

It is easy to see that the complexity of Partition3d\((G)\) is linear due to the number of vertices of \( G \).

As a consequence of the above theorem and Theorem 1 we get the statement of Theorem 5.

### 4 Concluding remarks

In Subsection 2.2 we have proposed the polynomial-time algorithm that finds an equitable \((L, D_{d-1})\)-colouring of a given graph \( G \) assuming that we know a \((k, d)\)-partition of \( G \) \((L \) is a \( t \)-uniform list assignment for \( G, \ t \geq k \)). In this context
Algorithm 7: Partition3d\((G)\)

\begin{algorithm}
\begin{algorithmic}[1]
\Input 3-dimensional grid \(G = P_{n_1} \square P_{n_2} \square P_{n_3}\).
\Output A \((3, 2)\)-partition \(S_1 \cup \cdots \cup S_{\alpha+1}\) of \(G\).
\State initialization;
\State \(\alpha := \lceil \frac{n_1 n_2 n_3}{3} \rceil - 1;\)
\If \(\alpha \geq 1\)
\For \(j := \alpha + 1 \text{ downto } 2\)
\State \(y_j^1 := \text{Corner}(G);\)
\If \(\text{deg}(y_j^1) = 1\)
\State \(y_j^2 := \text{Corner}(G - y_j^1);\)
\If \(y_j^1\) is the only vertex on layer \(a_1\)
\State let \(y_j^3\) be any vertex on layer \(a_1 + 1\) such that \(y_j^3 \neq y_j^2\)
\Else
\State let \(y_j^3\) be any vertex on layer \(a_1\) such that \(y_j^3 \neq y_j^1\) and \(y_j^3 \neq y_j^2\), if exists, otherwise \(y_j^3\) is any vertex on layer \(a_1 + 1\)
\EndIf
\EndIf
\EndFor
\EndIf
\If \(\text{deg}(y_j^1) = 2\)
\State let \(y_j^2\) be the neighbour of \(y_j^1\) lying on the same layer as \(y_j^1\);
\State let \(y_j^3\) be any vertex on layer \(a_1\), if exists, otherwise, \(y_j^3\) is any vertex on layer \(a_1 + 1\)
\EndIf
\If \(\text{deg}(y_j^1) = 3\)
\State \(y_j^2 := (a_1, a_2, a_3 + 1); y_j^3 := (a_1, a_2 + 1, a_3);\)
\EndIf
\State \(S_j := \{y_j^1, y_j^2, y_j^3\}; G := G - S_j;\)
\EndFor
\State \(S_1 := V(G);\)
\end{algorithmic}
\end{algorithm}

the following open question seems to be interesting: What is the complexity of recognition of graphs having a \((k, d)\)-partition?

\textbf{Acknowledgment.} The authors thank their colleague Janusz Dybizbaski for making several useful suggestions improving the presentation.

\section*{References}

1. Abu-Ata, M., Dragan, F. F.: Metric tree-like structures in real-world networks: an empirical study. Networks \textbf{67}(1), 49–68 (2016).
2. Diestel, R.: Graph Theory. 2nd edn. Graduate Texts in Mathematics \textbf{173}, Springer-Verlag, New York (2000).
3. Drgas-Burchardt, E., Dybizbański, J., Furmańczyk, H., Sidorowicz, E.: Equitable List Vertex Colourability and Arboricity of Grids. Filomat \textbf{32}(18), 6353–6374 (2018).
4. Kostochka, A. V., Pelsmajer, M. J., West, D. B.: A List Analogue of Equitable colouring. J. Graph Theory 44(3), 166–177 (2003).
5. Lei, H., Li, T., Ma, Y., Wang, H.: Analyzing lattice networks through substructures, Appl. Math. Comput. 329, 297–314 (2018).
6. Miao, T., Chen, A., Xu, Y.: Optimal structure of damaged tree-like branching networks for the equivalent thermal conductivity. Int. J. Therm. Sci. 102, 89–99 (2016).
7. Mudrock, J. A., Chase, M., Thornburgh, E., Kadera, I., Wagstrom, T.: A note on the equitable choosability of complete bipartite graphs, DMGT https://doi.org/10.7151/dmgt.2232 (2019).
8. Pelsmajer, M. J.: Equitable list-colouring for graphs of maximum degree 3, J. Graph Theory 47(1), 1–8 (2004).
9. Zhang, X.: Equitable list point arboricity of graphs. Filomat 30(2), 373–378 (2016).
10. Zhang, X., Niu, B., Li, Y., Li, B.: Equitable vertex arboricity of $d$-degenerate graphs, [arxiv: 1908.05066v1] (2019).