Journal of Ubiquitous Systems & Pervasive Networks
Volume 17, No. 2 (2022) pp. 91-98

Some Results on Colored Network Contraction

Flavio Lombardi [a, b, c], **Elia Onofri** [b, a, c*]

[a] *Istituto per le Applicazioni del Calcolo, Consiglio Nazionale delle Ricerche (IAC-CNR), Rome (00185), Italy*
[b] *Dipartimento di Matematica e Fisica, Roma Tre University, L. S. Murialdo, 1, Rome (00146), Italy*
[c] *Member of the INdAM-GNCS research group*

Abstract

Networks are pervasive in computer science and in real world applications. It is often useful to leverage distinctive node features to regroup such data in clusters, by making use of a single representative node per cluster. Such contracted graphs can help identify features of the original networks that were not visible before. As an example, we can identify contiguous nodes having the same discrete property in a social network. Contracting a graph allows a more scalable analysis of the interactions and structure of the network nodes. This paper delves into the problem of contracting possibly large colored networks into smaller and more easily manageable representatives. It also describes a simple but effective algorithm to perform this task. Extended performance plots are given for a range of graphs and results are detailed and discussed with the aim of providing useful use cases and application scenarios for the approach.

Keywords: *Colored Networks, Graph Contraction, Greedy Algorithm, Graph Analysis*

1. Introduction

Networks are pervasive in computer science and in real world applications [1]. Collecting, navigating, and extracting insights from such data and from the underlying graph structure is often challenging [2]. A widespread technique is to organize data in clusters by means of some characteristics and extract a representative element from each cluster [3]. The graph made of these representative objects is usually much smaller than the original one while it preserves many useful characteristics.

A common way to describe clusters is to provide a categorical classification of the vertices by means of some coloring function, *i.e.* a mapping of the vertices into some fixed representation. Coloring functions can be generated as the outcome of clustering techniques like, *e.g.*, the renown k-means. Oftentimes, however, this kind of information is natural with the graph – like the language spoken by users in a Social Network – or can be injected as expert-knowledge – as the additional information a geologist can provide on a specific terrain map. Indeed, also continuous variables, like *e.g.* air pressure in weather forecasting maps, can be sketched as discrete if sampled in fixed ranges.

Networks can be quite large in nature, thus having an efficient way to contract them provides a useful approach to ease analysis and detection of issues. In this sense, stakeholders can be identified as the analysts that need to extract information, *e.g.* for urban planning, where city maps are contracted on neighborhoods sharing the same thematic area. Other stakeholders are those ones analyzing networks representing Unspent Transaction Output or account-based cryptocurrencies that might search, *e.g.*, for accumulation nodes or highly connected small clusters of vertices.

In the above scenarios, the vertices sharing the same categorical information can be *merged* via (vertex-) contraction, *i.e.* a novel vertex is generated in place of the previous ones, preserving the adjacency of the substituted vertices. Application of contraction generates novel graphs that share a number of properties with the original graph (*e.g.* connectivity), but whose size is usually much smaller (in both vertices and edges).

Graph contraction problems are an interesting set of problems on their own being a class of NP-Hard problems, *e.g.* determining whether one can obtain a tree by contracting at most k edges from a given connected graph to obtain a tree (*i.e.* Tree Contraction) is a possible example [4].

*Corresponding author. Tel.: +39-06-5733-8230
Fax: +39-06-5733-8080; E-mail: elia.onofri@uniroma3.it

DOI: 10.5383/JUSPN.07.02.006

1.1. Paper Contribution

In this paper, we discuss and apply the algorithm for graph color contraction first presented in [5]. The algorithm is applied in different scenarios and results are shown and discussed. The algorithm is serial albeit designed to be naturally parallelizable, taking care of the most important concepts of parallel computing system design ([6] offers a good review on parallel graph algorithms). For this reason, instead of leveraging global visit algorithms that are known to be hard to parallelize, the algorithm leverages local properties of the vertices. In particular, the underlying data structure is optimized, yielding a computational cost that, on the average case, is proportional to the number of edges of the original graph (just like in a normal visit) while avoiding global assumptions on the vertices (that usually force parallel versions to use barriers).

This paper improves and extends over previously published paper [5] in that it:

- updates and extends the related work section;
- better introduces the contraction problem and the algorithmic solution by providing novel examples and a more in-depth mathematical formalization;
- offers a more detailed description of the graph contraction algorithm by means of a structured representation of the process and a greedy local definition of the problem;
- extends experimental data range including larger graphs (up to 50000 nodes) offering a richer and more comprehensive assessment of the proposed algorithm;
- applies the algorithm to a Social Network graph and studies the outcome.

The rest of this paper is organized as follows: Section 2. provides some information on related work. A brief introduction to graph theory technicalities is given in Section 3., while Section 4. provides a clearer description of the graph contraction algorithm introduced in [5]. Section 5. shows and discusses novel results obtained applying our algorithm to benchmark Erdös-Rényi graphs (Section 5.1.) and to a real-world graph of Facebook pages (Section 5.2.). Finally, Section 6. summarizes the contributions and discusses future developments.

2. Related Work

Graph contraction is useful in many graph-related problems since it tackles the problem of reducing input data into a more manageable size. Solutions to many well-known graph problems have been formulated by means of procedural reduction of the original graph; it is the case, *e.g.*, of finding the shortest path [7, 8], of computing chromatic polynomials [9], evaluating DP coloring [10] or evaluating the number of spanning trees [11] of a simple graph.

Parallel computing, where data must be mapped to processing units via some not a priori known logic, is an interesting use case for graph contraction as well. Ponnusamy *et al.* [12] introduced an iterative parallel graph contraction algorithm to pairwise contract vertices, hence reducing the problem size before mapping data. Meyerhenke *et al.* apply a similar idea in [13], extending it in a multi-leveled way: namely, the graph is iteratively reduced while applying label propagation to obtain graphs of manageable size. An interesting review on these topics can be found in [14].

In both colored and colorless versions, contraction provides many insights about graph connectivity-related features (see *e.g.* [15] for a fast connectivity algorithm based on graph contraction). Studying graphs and color clusters connectivity is useful in a wide variety of applications [16], including when the graphs underlie beneath more complex processes, *e.g.* the connectivity of the basis of Markov process reveals important information about its Ergodicity (see *e.g.* [17] for an Ergodic colored graph-based Markov process).

Despite its usefulness in a wide variety of applications, to the best of our knowledge, there is not a rich research effort on color-based graph contraction. Nevertheless, some research work can be found with applications in distinct areas.

D'Autilia and Spada [18] used color-based graph contraction to retrieve the relationship between pedestrian, vehicular and hybrid areas in city maps to compare different mobility plans; in this context, the authors enforced colors on street junction-based graphs leveraging on Space Syntax approach [19] where a city is analyzed by means of its different areas. Onofri and Corbetta [20] described a procedural classification method of cascaded localizers derived by color contraction over museums graph representations where colors are injected by expert knowledge as architectural and conceptual constraints. Crypto-currency offers some interesting use cases as well – transactions and users graphs, whose analysis can be complex [21, 22] can benefit from a feature-preserving size reduction of the original network. Finally, as previously mentioned, other applications include complex networks such as Social Network and web graphs, with some novel result shown and discussed below.

Nevertheless, to the best of our knowledge, an efficient parallel algorithm for colored graph contraction is not available at present. Some early work by Miller and Reif [23] and by Philips [24], is limited to –colorless– graphs. More recent work on label propagation by Meyerhenke *et al.* [25] aims at building color clusters instead of contracting them. This motivates our work, whose final goal is to provide a contraction algorithm effectively capable of leveraging parallel computing.

3. Some Definitions

In the following, we will use Greek letters for mappings, Gothic letters for procedures, typewriter font for algorithm-related variables, capital Latin letters for sets, and lower-case Latin letters for objects. The same sets of letters will be used to identify the same sets of objects throughout the paper, *e.g.* u, v, w are always adopted to identify vertices. We adopt big-$\mathcal{O}$ and big-Ω function class notation for denoting asymptotic bounds, namely $f(x) \in \mathcal{O}(g(x))$ [resp. $\Omega(g(x))$] if exists $c \in \mathbb{R}$ such that $f(x) < c \cdot g(x)$ [resp. $f(x) > c \cdot g(x)$], for all $x > x_0$, for some $x_0 < \infty$.

We recall from basics of graph theory the definition of a (simple, undirected) *graph* $G = (V, E)$ as a set of vertices V equipped with a set of edges E insisting on V, *i.e.* $E \subseteq V \times V$, where edges are not oriented, *i.e.* $(u, v) \in E \iff (v, u) \in E$. Two vertices $u, v \in V$ are *adjacent* – denoted by $u \sim_G v$ – if there exists an edge $e \in E$ between them. The *neighbourhood* $N(v)$ of a vertex $v \in V$ is defined as the set of vertices adjacent to v, *i.e.* $N_G(v) = \{u \in V \mid u \sim_G v\}$. A (simple) *path* $\Gamma_{u,v}$ between two vertices u, v is a sequence of distinct vertices $w_0, w_1, \dots w_\ell$, where $w_0 = u$, $w_\ell = v$ and, for $0 \leq i < \ell$,

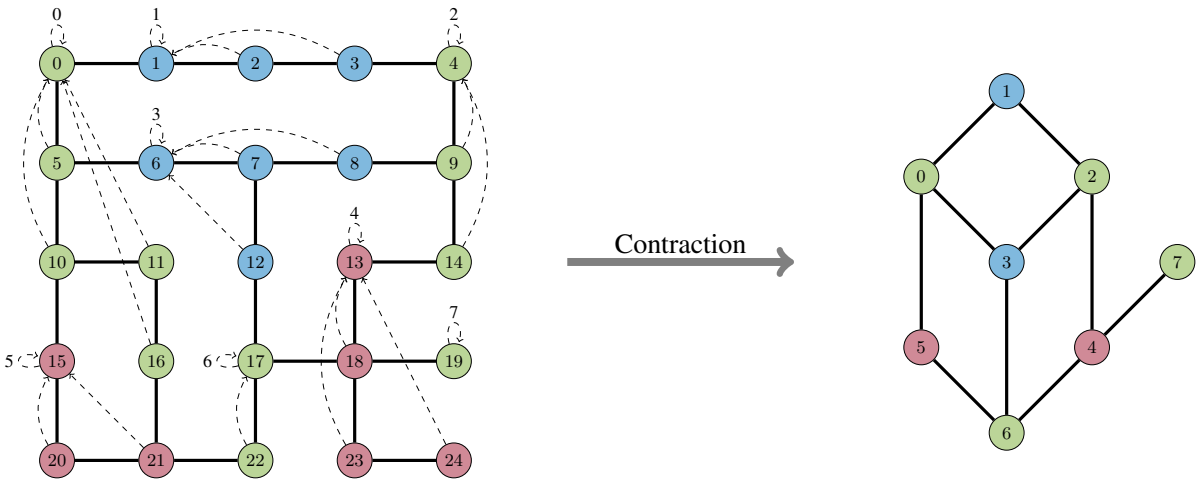

Fig. 1. Example Contraction of the graph G (on the left) to the graph G' (on the right). The map β is represented via dashed lines. Hence, the elements of V' (*i.e.* the codomain of β) are identified by nodes with self loops, where the novel indices are also reported.

we have $(w_i, w_{i+1}) \in E$. Two nodes are *connected* if a path can be constructed between them while a graph is *connected* if each couple of nodes is connected. In the following, we drop the subscripts when they can be inferred from the context.

Vertices or/and edges of a graph can be, in general, equipped with any sort of data. In the rest of the paper we restrict these kinds of data to categorical information – data sampled from a numerable finite set – over vertices, often called as (vertex) colors. More formally, given a set of colors $C \subsetneq \mathbb{N}$, $|C| < \infty$, a (vertex-) *coloring* is a map $\gamma : V \to C$ that associates to every vertex a color. We also denote with $N_\gamma(v)$ the set of nodes adjacent to v that shares the same color with it, *i.e.* $N_\gamma(v) = \{w \in N(v) \mid \gamma(w) = \gamma(v)\}$.

Given two adjacent vertices $u \sim v$, the *contraction* of u and v is a procedure $\mathfrak{m}(u, v) \equiv \mathfrak{m}(v, u)$ that deletes u and v from V and creates a novel blended vertex w such that $N(w) = N(u) \cup N(v)$[1]. A γ-*contraction* – more formally a color contraction of G on the color set C provided by the assignment γ – is a procedure that applies the contraction $\mathfrak{m}$ to each couple of vertices $u \sim v$ such that $\gamma(u) = \gamma(v)$, building a novel vertex $w = \mathfrak{m}(u, v)$ such that $\gamma(u) = \gamma(w) = \gamma(v)$. Do note that the graph-wise γ-contraction is well-defined even if the vertices contraction is defined vertex-pairwise since vertices contraction is commutative, namely $\mathfrak{m}(u, \mathfrak{m}(v, w)) = \mathfrak{m}(\mathfrak{m}(u, v), w)$; in this sense, the procedure $\mathfrak{m}$ is a *variadic function*, *i.e.* it can be applied to any number of arguments as $\mathfrak{m}(u_0, \dots, u_k) = \mathfrak{m}(\dots(\mathfrak{m}(u_0, u_1)\dots), u_k)$.

4. The Graph Contraction Algorithm Explained

In the following, we review the color-contraction algorithm presented in [5]; the algorithm computes the contracted graph $G' = (V', E')$, where $n' = |V'|$ and $m' = |E'|$, having as input a graph $G = (V, E)$, where $n = |V|$ and $m = |E|$, and a color mapping $\gamma : V \to C$, where $|C| = c$. From a technical point of view, the graph data structure is stored as follows:

- `V` the vertex list is stored as an array `V` of `vertex` type; each `vertex` is characterized by its index within the array itself, sampled in $[0, \dots, n-1]$ (we will use $v \in V$ to identify mutually the vertex and its index).
- `Nv` the degree of each vertex is stored as a look-up table in a n-length array `Nv`.
- `E` the edge list is stored as adjacency matrix `E`, *i.e.* a n-length array of arrays of indices (of `vertex`); one `Nv`$[v]$-length array per each vertex $v \in V$.
- `C` the colors are stored as a look-up table in a n-length array `C`.

The algorithm builds upon the variadic form of $\mathfrak{m}$, capturing and contracting the local *color clusters*, *i.e.* sets of the form $S \subset V$ such that for each couple $u, v \in S$, there exists a single-color path $\Gamma_{u,v}$, namely $\gamma(w) = \gamma(u)$ for all $w \in \Gamma_{u,v}$. It is important to notice that the definition of a color cluster is in some sense weak, in fact, it could happen that if $S = S_1 \cup S_2$ is a color cluster, then S_1 and S_2 are identified as two different color clusters instead (see Figure 3). The loose definition of color cluster we provide is however mandatory in order to guarantee the *local* property that the algorithm makes use of.

In order to work locally, it is in fact mandatory to define a vertex total ordering; to keep the on-machine representation compact, we use the one enforced by the vertex indexing in $[0, \dots, n-1]$. Algorithmically, local color clusters are identified by building a mapping $\beta : V \to V$ from each vertex u to the endpoint of the max-decreasing color path rooted in u, *i.e.* the path that, at each step, chooses the minimum amongst the same-color neighbors; hence, we can build such path by performing a greedy search of local minima. More formally, $\beta(u) = v$ if $\Gamma_{u,v} = (w_0 = u, w_1, \dots, w_\ell = v)$ is the path built according to the property

$$w_{i+1} = \min\Big(N_\gamma(w_i)\Big), \qquad \forall i > 0 \tag{1}$$

with stopping conditions being either $N_\gamma(w_i) = \emptyset$ or $w_{i+1} > w_i$.

[1] More in general, we can distinguish the contraction of u in v from the contraction of v in u, depending on how additional data are generated for w. However, this topic is out of the scope of this paper since we only consider contractions where vertex colors (our unique kind of categorical data) are preserved rather than modified. Namely, we only adopt merging procedures $\mathfrak{m}$ where $\mathfrak{m}(u, v) = w = \mathfrak{m}(v, u)$, and $\gamma(w)$ is defined as $\gamma(u) = \gamma(v)$.

Graph color contraction procedure
- **`contractionMapping` evaluation**
  - **`becomes` pre-evaluation**
    - **`becomes` greedy local initialisation**
      - **forall** $(v \in V)$
        - $\texttt{becomes}[v] \leftarrow \min\left(\{v\} \cup N_\gamma(v)\right)$
    - **`becomes` update to end of Γ,**
      - **for** $(i = 0;\ i < n;\ i\texttt{++})$
        - $\texttt{becomes}[i] \leftarrow \texttt{becomes}[\texttt{becomes}[i]]$
  - **`cSize` evaluation**
    - **forall** $(v \in V)$
      - $\texttt{cSize}[\texttt{becomes}[v]]\texttt{++}$
  - **`revBecomes` evaluation**
    - **`revBecomes` pre-evaluation**
      - **let** `I` be a n-length zero-initialised array storing the first available position in $\texttt{revBecomes}[v]$
      - **forall** $(v \in V)$
        - $w \leftarrow \texttt{becomes}[v]$
        - $\texttt{revBecomes}[w][\texttt{I}[w]\texttt{++}] \leftarrow v$
    - **`revBecomes` compacting**
      - $i \leftarrow 0$
      - **foreach** $(v \in V)$
        - $\texttt{revBecomes}[i\texttt{++}] \leftarrow \texttt{revBecomes}[v]$
  - **`becomes` compacting**
    - **forall** $(v' \in V')$
      - **for** $(i = 0;\ i < \texttt{cSize}[v'];\ i\texttt{++})$
        - $\texttt{becomes}[\texttt{revBecomes}[v'][i]] = v'$
- **`contractionMapping` application**
  - **Edges contraction**
    - **Edges destination update**
      - **forall** $(v \in V)$
        - **for** $(i = 0;\ i < \texttt{Nv}[v];\ i\texttt{++})$
          - $\texttt{E}[v][i] \leftarrow \texttt{becomes}[\texttt{E}[v][i]]$
    - **Colour cluster edge merge**
      - **let** `a` be a n'-length bit-array repr. edges endpoints
      - **for** $(v' \in V')$
        - **reinitialise** `a` to `false`
        - **for** $(i = 0;\ i < \texttt{cSize}[v'];\ i\texttt{++})$
          - $w \leftarrow \texttt{E}[\texttt{revBecomes}[v'][i]]$
          - **for** $(j = 0;\ j < \texttt{Nv}[w];\ j\texttt{++})$
            - $\texttt{a}[\texttt{E}[w][j]] \leftarrow \texttt{true}$
        - $\texttt{E}[v'] \leftarrow$ **convert** `a` in adjacency list
  - **Vertices contraction**
    - **Re-build** `V` according `revBecomes`

Fig. 2. Description of the contraction algorithm as a function tree; each leaf summarizes a different phase by means of a simplified pseudocode.

From the implementation point of view, the mapping β is stored as an array-based look-up table `becomes`, ready to guide the application of the contraction methodologies. In particular the codomain of β represents V' itself, so, since in a general case $n' \ll n$, it is important to compact $V' \subset V$ on the set $[0, \ldots, n'-1]$. The inverse mapping β^{-1} is an important tool as well since it speeds-up the following contraction phase[2]. The

[2] It is in particular useful in a parallel environment to schedule the vertex assignment to the different computing units.

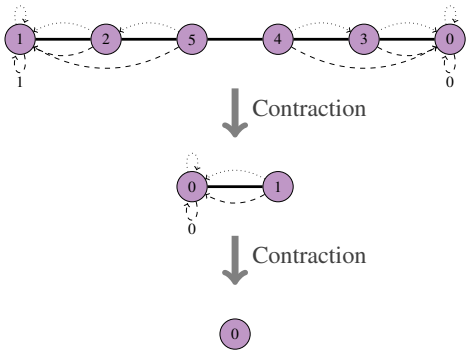

Fig. 3. Example contraction that requires two iterations of the algorithm since two non-maximal color clusters are identified during the first "`contractionMapping` evaluation" phase.

mapping β^{-1} is stored as a look-up table `revBecomes` similarly to `becomes`. However injective mapping β is not surjective, hence `revBecomes` must be stored as a n'-length array of arrays; the dimension of these arrays can be a-priori evaluated in linear time from the sole `becomes` by constructing a look-up table `cSize`.

The set of `becomes`, `revBecomes`, and `cSize` are semantically grouped together in a `contractionMapping` object, the main building block of the algorithm. Building it correctly is in fact the hard part of the work since, once its structure has been defined, it can be trivially used to perform $\mathfrak{m}(\texttt{revBecomes}[v'])$, actually providing the entire set of vertices and edges to be contracted in a single contraction step.

Figure 1 depicts the contraction of a sample graph while Figure 2 summarizes the procedure in the form of a function tree.

The computational time of the algorithm is upper bounded by $\mathcal{O}(n'^2 + m)$ as discussed in [5], however, as we pointed out above, the greedy approach does not guarantee that the color clusters detected are the optimal ones (see Figure 3 for an example). Two possible solutions (possibly used in an adaptive way) are (i) execute the complete algorithm more than once until a convergence is reached or (ii) repeat the "`becomes` pre-evaluation" phase multiple times, actually initializing it as

$$\texttt{becomes}[v] \leftarrow \min\left(\texttt{becomes}[w] \;\middle|\; w \in \{v \cup E_\gamma[v]\}\right). \qquad (2)$$

However (i) is upper bounded by $\log(n)$ iteration applied on a graph that is progressively smaller – since the set of contractible vertices is at least halved after each iteration – while (ii) can take up to $\log(n)$ steps applied to the same sized-graph – it is the case, *e.g.*, of a line graph. For these reasons, pathological graphs (like graphs that asymptotically do not reduce n even when the (i) is applied) may yield unsatisfactory execution times up to $\mathcal{O}(n^2 \log n)$. It is the case, *e.g.*, of a graph where $n/2$ vertices converge in a single one in $\log(n)$ iteration, while the other nodes create a star graph around it. Nevertheless, particular structures like those are rare and can be spotted a priori by simple graph measures so that different approaches can be employed. In random graphs, for example, it is not unusual that the algorithm takes up to $\log(n)$ iteration to stop, however, the sizes involved always sink fast,

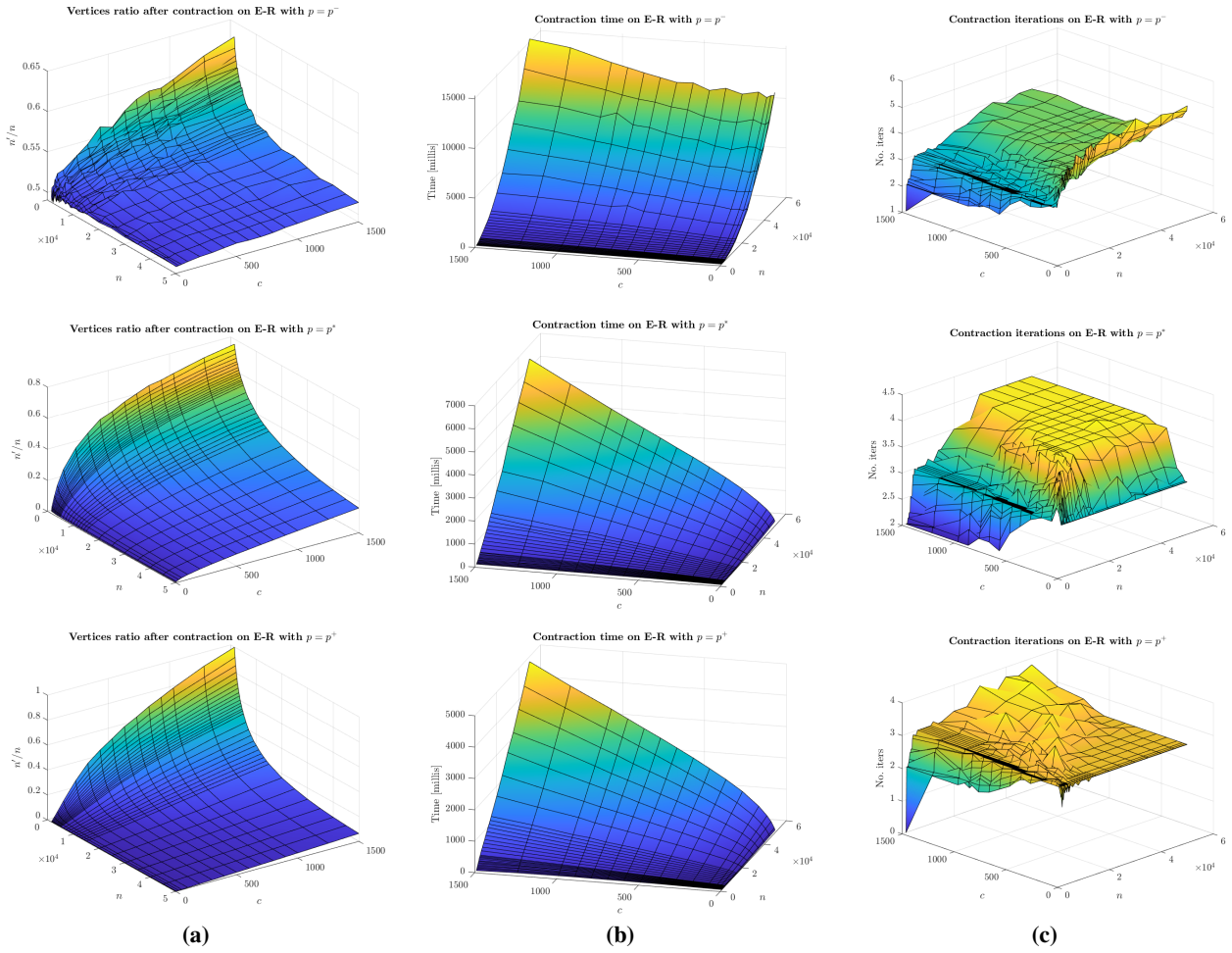

Fig. 4. Averages of contractions results over c-colored $G^{\mathrm{ER}}(n, p)$ wit $c \in [1, 1500]$ and $n \in [1500, 50000]$ (x- and y-axis). Left to right, plots report the average results of (a) contracted vertices ratio n'/n, (b) time of execution, and (c) number of iterations of the algorithm. Top to bottom the edge probability p is set to p^-, p^* and p^+ respectively (do note that z-axis scales are different).

making the dimensions of n and m decrease by an order of two or more magnitudes since the first iteration, actually making the computational time of the upcoming phases negligible.

5. Discussion

The description of the algorithm in Figure 2 is quite high-level to support implementations using any language of choice.

The results presented in this section are gathered from our `C` implementation running over an AMD Ryzen 5 5600X 3.7 GHz 6/12 with 32GB DDR-4 3600 RAM. `C` language is one of the fastest languages available that also guarantees fine-grained memory management – a relevant mark of our algorithm: the space complexity is in fact optimal, allowing contraction in-place over the original graph with an $\mathcal{O}(n)$ overhead – namely the negligible space of the n'-length bit array a and $3n$ pointers and n integers (occupied by the `contractionMapping`).

5.1. Validating and Extending Previous Results

The novel campaign allowed us to validate the past results reported in [5] and check whether we had captured or not the relevant phenomena and behavior of the proposed algorithm.

In fact, leveraging more capable hardware, we managed to enlarge the graph size. The novel experimental campaign here presented considers a maximum graph size of $n = 5 \cdot 10^4$ nodes and $m \sim 1.3 \cdot 10^8$ edges, studied with colors ranging from $c = 1$ (benchmark) up to $c = 1500$[3].

Similarly to the previous campaign, we employed graphs $G^{\mathrm{ER}}(n, p)$ generated by the Erdös-Rényi (E-R) model [26]. The E-R model is a well-known procedure to generate random graphs by providing as input the number of vertices n and the probability p that $u \sim v$ holds for each couple of vertices $u, v \in V$.

The behavior of E-R graphs (for $n \to \infty$) at the varying of p is a well studied topic [27] and, in particular, two sharp thresholds are relevant for the graph structure: $p_1 = {}^1/_n$ and $p_2 = {}^{\log(n)}/_n$. In fact it holds that

[3] The graph with the largest m is the one generated with $n = 50000$ and $c = 1500$, where the probability for an edge to be present (see later) is $p = p^+ = 0.1052$. This yields, on average, $m = p \cdot n^2/2 = 131496250$.

- if $p < p_1$, then $G^{\text{ER}}(n,p)$ have no connected components of size larger than $\mathcal{O}(\log(n))$ with high probability
- if $p = p_1$, then the largest component of $G^{\text{ER}}(n,p)$ has size $\mathcal{O}(n^{2/3})$ with high probability
- if $p > p_1$, then $G^{\text{ER}}(n,p)$ has, with high probability, a (unique) giant component of size $\Omega(n)$.
- if $p < p_2$, then $G^{\text{ER}}(n,p)$ contains isolated vertices with high probability.
- if $p > p_2$, then $G^{\text{ER}}(n,p)$ is disconnected with negligible probability.

It is worthwhile to point out that values $p \in [p_1, p_2]$ are the most significant for what concerns our analysis since $p \geq p_1$ provide graphs that contracts on a small number of nodes n' (*i.e.* $n' \ll n$) while $p \leq p_2$ generates graph with a non-trivial number of connected components ($n' > 1$)

For what concerns colors, vertex colorings are randomly assigned too, where each vertex is given a color uniformly at random from the color set. However, if each vertex has probability $1/c$ to be of a given color, then only n/c vertices are expected to share the same color, hence we properly re-scaled the thresholds on this value, obtaining the two new thresholds

$$p^- = \frac{c}{n} \quad \text{and} \quad p^+ = \frac{\log(n) - \log(c)}{n} \cdot c \tag{3}$$

respectively in place of p_1 and p_2. This is a natural way to extend the E-R model to colors: the expectation of the result of $G^{\text{ER}}(n,p^-)$ with an assignment of c uniformly random colors is in fact to have c single-color sub-graphs of the form $G^{\text{ER}}(n/c, p_1)$ randomly connected between them (analogous relation holds for p^+ and p_2).

Averaged results in terms of **(a)** inverse ratio of contraction n'/n, **(b)** computational execution time, and **(c)** number of iterations before convergence are reported in Figure 4 for $n \in [1500, 50000]$, $c \in [1, 1500]$ and $p \in \{p^-, p^+, p^* = \frac{p^- + p^+}{2}\}$.

Do note that $c > n$ makes little sense for the analysis since the expected number of vertices per color would be less than one (also p^+ would be negative, hence generating disconnected vertices only). This is the reason why we kept c and n running over two quasi-disjoint consecutive ranges, being $n = 1500 = c$ the only intersection (yielding, as expected, 0 iterations on p^+).

The thresholds in the E-R model are sharp when $n \to \infty$, hence it makes sense that the most interesting conditions are when $n/c \gg 1$; when $n/c \sim 1$, in fact, the graph hardly contracts, actually causing little or no iterations. When n grows, contraction is much more effective, actually leading to much fewer color clusters (as can be seen in Figure 4**(a)**); this behavior is less significant in the case $p = p^-$ since we expect to have a big number of small connected components, compared to $p = p^+$, where we expect to have only a few of them – actually $n' \sim c$. Following this behavior, Figure 4**(b)** shows that the more the graph is connected (according to the choice of p), the less it takes to contract (despite m being larger). Such a conclusion seems to be counter-intuitive, however, it totally makes sense since the less the graph is connected, (i) the higher it will be n', causing each further iteration to be more complex and (ii) the higher the chances the local minima are not the global ones (Figure 3), hence requiring more iterations to correctly contract, as can be seen in Figure 4**(c)**. In fact, having a larger number of iterations over larger amounts of vertices renders the algorithm more computationally expensive.

5.2. A real-world use-case: Facebook

We have applied our algorithm to a real-world graph from the SNAP data-sets [28]: the Facebook web-pages network $G_{\text{FB}} = (V_{\text{FB}}, E_{\text{FB}})$ collected in 2017 by [29].

V_{FB} represents 22470 Facebook official pages organized in 4 categories (*i.e.* colors), namely: (i) politicians, (ii) governmental organizations, (iii) television shows and (iv) companies. V_{FB} nodes are linked via 170823 undirected edges E_{FB} representing mutual likes between the pages. It is worth noticing that the numbers n, m, and c are in line with the E-R graphs we have used as benchmarks. In fact, considering an E-R graph with $n = 22470$ and $c = 4$, the expected number of edges ranges between $m^- = n^2/2 \cdot p^- = n \cdot c/2 = 44540$ and $m^+ = n^2/2 \cdot p^+ = m^- \cdot (\ln(n) - \ln(c)) = 387996$.

The contraction converges in four iterations (three effective plus one to confirm the convergence), and the running time is approximately $61ms$ on the test machine. The original graph and the three contraction iterations are reported in Figure 5 (a)–(d), where the graph is rendered using Wolfram Mathematica [30].

The intense computational effort lies in the first contraction step that takes $\sim 95\%$ of the total time, actually contracting vertices with ratio $\sim 5:1$ and edges with ratio $\sim 1:3.5$. Conversely, the second step is the one contracting more, with a ratio of $\sim 1:10$ and $\sim 1:35$ for vertices and edges respectively; despite this fact, it takes significantly less time than the first step given that the size of the graph has already been reduced. The third and last contraction step contracts the last few local minima identified during the previous contraction, actually reducing vertices and edges with a negligible ratio of $\sim 1:1.3$ and $\sim 1:1.8$ respectively.

By observing the contracted graph (Figure 5**(d)**) we can identify one cluster per color but "Politician" pages having two of them. "Company" pages first and "Television Show" pages second, represent the main glue among the central clusters, while "Politician" and "Governmental" pages are either part of the central clusters or nearly-isolated vertices. Furthermore, the "Politician" pages appear to be the only connection to many isolated "Governmental" pages; a similar behavior can be identified for a big number of nearly-isolated "Company" pages that are connected to the "Television show" cluster only.

Overall, the above results show how useful network contraction is in trying to evince meaningful characteristics and behavior from large datasets. A field expert studying a possibly very large network can benefit from the compact representation of the structure under observation offered by the contracted graph.

6. Conclusions

In this paper we provided relevant details of an effective generic graph contraction algorithm limited to colored graphs, presenting an extended performance measurement campaign and providing real-world applications of its implementation, which is presently serial. Results are detailed and discussed, providing useful use-cases and heterogeneous application scenarios.

We believe such results show the usefulness and feasibility of colored graph contraction in visualizing and detecting issues and features related to large colored networks that are pervasive in today's world.

Future work will provide and evaluate a parallel implementation of the methodology, leveraging the formal definition of the

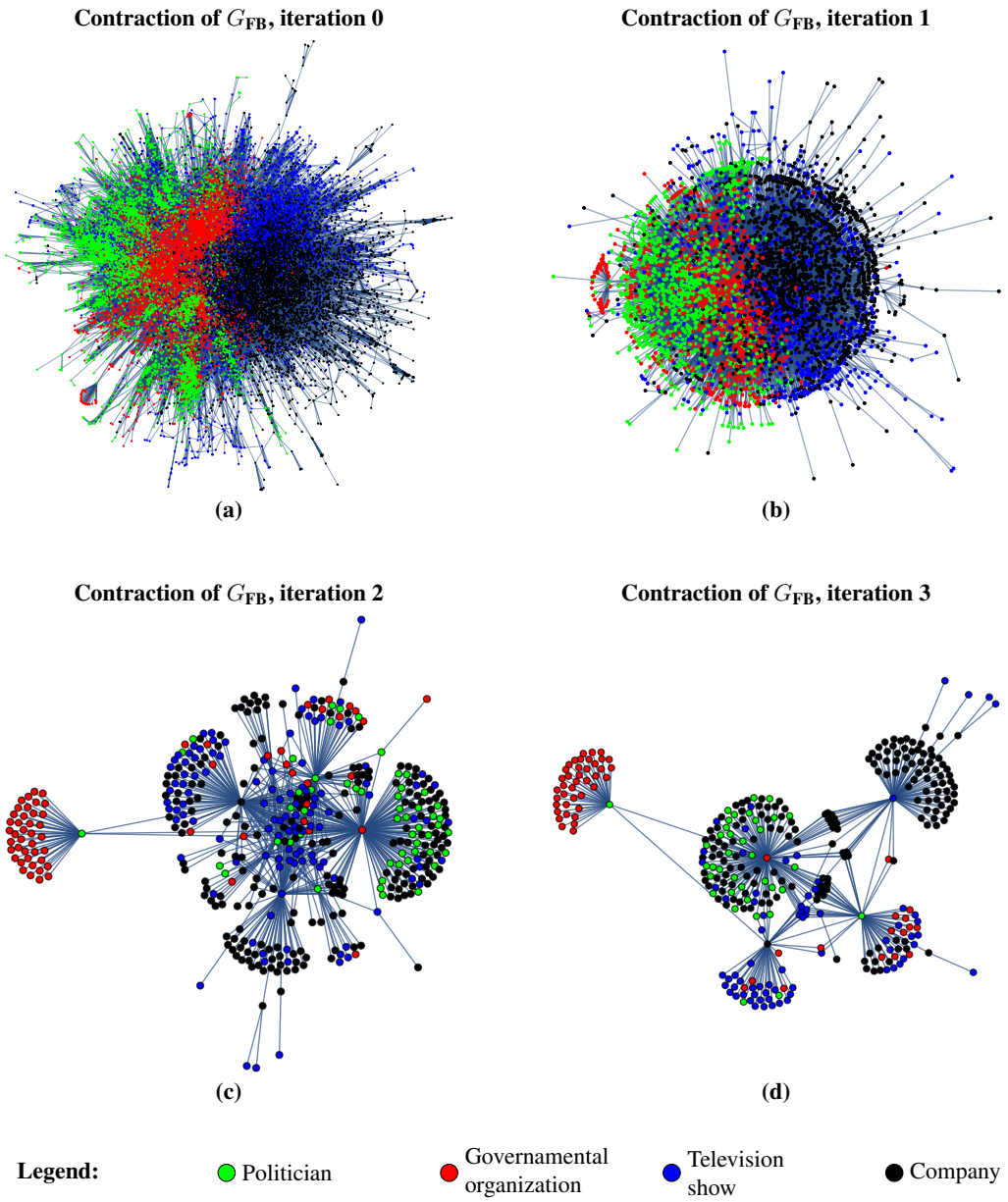

Fig. 5. (a) The original page-page Facebook graph $G_{\mathbf{FB}}$ with self loops elided; $n = 22470, m = 170823$. (b) The first contraction iteration took 59ms, yielding $n = 4441, m = 48178$. (c) The second contraction iteration took 1ms, yielding $n = 437, m = 1380$. (d) The third and last iteration took less than 1ms, yielding $n = 334, m = 782$.

problem here presented. Further practical scenarios and more extended performance campaigns over benchmark networks will also help assessing pros and cons of the proposed approach.

References

[1] Vishv Patel, Devansh Shah, and Nishant Doshi. Emerging Technologies and Applications for Smart Cities. *Journal of Ubiquitous Systems and Pervasive Networks*, 15(02):19–24, March 2021. [DOI:10.5383/juspn.15.02.003].

[2] Massimo Bernaschi, Alessandro Celestini, Marco Cianfriglia, Stefano Guarino, Flavio Lombardi, and Enrico Mastrostefano. Onion under microscope: An in-depth analysis of the tor web. *World Wide Web*, 25(3):1287–1313, May 2022. ISSN 1573-1413. [DOI:10.1007/s11280-022-01044-z].

[3] Absalom E. Ezugwu, Abiodun M. Ikotun, Olaide O. Oyelade, Laith Abualigah, Jeffery O. Agushaka, Christopher I. Eke, and Andronicus A. Akinyelue. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022. ISSN 0952-1976. [DOI:10.1016/j.engappai.2022.104743].

[4] Takao Asano and Tomio Hirata. Edge-contraction problems. *Journal of Computer and System Sciences*, 26(2): 197–208, 1983. [DOI:10.1016/0022-0000(83)90012-0].

[5] Flavio Lombardi and Elia Onofri. Graph contraction on attribute-based coloring. *Procedia Computer Science for 13th Intl Conf on Ambient Systems, Networks and Technologies (ANT)*, 201:429–436, 2022. ISSN 1877-0509. [DOI:10.1016/j.procs.2022.03.056].

[6] Kayhan Erciyes. *Parallel Graph Algorithms*, chapter 3, pages 77–115. Springer Intl. Pub., Cham, 2018. ISBN 978-3-319-73235-0. [DOI:10.1007/978-3-319-73235-0_4].

[7] Daniel Delling, Andrew V. Goldberg, Andreas Nowatzyk, and Renato F. Werneck. Phast: Hardware-accelerated shortest path trees. *Journal of Parallel and Distributed Computing*, 73(7):940–952, 2013. ISSN 0743-7315. [DOI:10.1016/j.jpdc.2012.02.007].

[8] Stephen Guattery and Gary L Miller. A contraction procedure for planar directed graphs. In *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*, SPAA '92, pages 431–441, New York, NY, USA, 1992. ACM. ISBN 089791483X. [DOI:10.1145/140901.141935].

[9] Douglas Brent West. *Introduction to graph theory*, volume 2. Prentice Hall, 2001. ISBN 978-0130144003.

[10] Jeffrey A Mudrock. A deletion-contraction relation for the dp color function. *arXiv preprint*, 2021. [arXiv:2107.08154].

[11] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018. ISBN 978-0429757099.

[12] Ravi Ponnusamy, Nashat Mansour, A Choudhary, and Geoffrey Charles Fox. Graph contraction for mapping data on parallel computers: A quality–cost tradeoff. *Scientific Programming*, 3(1):73–82, 1994. [DOI:10.1155/1994/715918].

[13] Henning Meyerhenke, Peter Sanders, and Christian Schulz. Partitioning complex networks via size-constrained clustering. In *Experimental Algorithms*, pages 351–363, 02 2014. [DOI:10.1007/978-3-319-07959-2_30].

[14] Alan Valejo, Vinícius Ferreira, Renato Fabbri, Maria Cristina Ferreira de Oliveira, and Alneu de Andrade Lopes. A critical survey of the multilevel method in complex networks. *ACM Comput. Surv.*, 53(2), April 2020. ISSN 0360-0300. [DOI:10.1145/3379347].

[15] Jyothish Soman, Kothapalli Kishore, and PJ Narayanan. A fast GPU algorithm for graph connectivity. In *2010 IEEE Intl. Symp. on Par. & Distr. Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010. [DOI:10.1109/IPDPSW.2010.5470817].

[16] Guy Blelloch and Margaret Reid-Miller. Graph contraction and connectivity. *Lecture notes in Parallel and Sequential Data Structures and Algorithms (15-210)*, 03 2022. School of Computer Science of Carnegie Mellon University [PDF].

[17] Filippo Castiglione, Christine Nardini, Elia Onofri, Marco Pedicini, and Paolo Tieri. Explainable drug repurposing approach from biased random walks. *IEEE/ACM Trans. on computational biology and bioinformatics*, 2022. [DOI:10.1109/TCBB.2022.3191392].

[18] Roberto D'Autilia and Marco Spada. Extension of space syntax methods to generic urban variables. *Urban Science*, 2(3), 2018. ISSN 2413-8851. [DOI:10.3390/urbansci2030082].

[19] Bill Hillier and Julienne Hanson. *The Social Logic of Space*. Cambridge University Press, 1984. [DOI:10.1017/CBO9780511597237].

[20] Elia Onofri and Alessandro Corbetta. RSSi-based visitor tracking in museums via cascaded AI classifiers and coloured graph representations. *Collective Dynamics*, 6:1–17, 2022. ISSN 2366-8539. [DOI:10.17815/CD.2021.131].

[21] Maurantonio Caprolu, Matteo Pontecorvi, Matteo Signorini, Carlos Segarra, and Roberto Di Pietro. Analysis and patterns of unknown transactions in bitcoin. In *IEEE Intl. Conf. on Blockchain*, pages 170–179, 2021. [DOI:10.1109/Blockchain53845.2021.00031].

[22] Yassine Issaoui, Azeddine Khiat, Ayoub Bahnasse, and Hassan Ouajji. Smart logistics: Blockchain trends and applications. *Journal of Ubiquitous Systems & Pervasive Networks*, 12(2):09–15, March 2020. [DOI:10.5383/juspn.12.02.002].

[23] Gary L Miller and John H Reif. Parallel tree contraction part 2: Further applications. *SIAM Journal on Computing*, 20(6):1128–1147, 1991. [DOI:10.1137/0220070].

[24] Cynthia A. Philips. Parallel graph contraction. In *Proc. 1st ACM Symp. on Parallel Algorithms and Architectures*, SPAA '89, pages 148–157, 1989. ISBN 089791323X. [DOI:10.1145/72935.72952].

[25] Henning Meyerhenke, Peter Sanders, and Christian Schulz. Parallel graph partitioning for complex networks. *IEEE Transactions on Parallel and Distributed Systems*, 28(9): 2625–2638, 2017. [DOI:10.1109/TPDS.2017.2671868].

[26] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960. [DOI:10.1515/9781400841356.38].

[27] Colin McDiarmid and Fiona Skerman. Modularity of Erdös-Rényi Random Graphs. In *29th Intl. Conf. on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA 2018)*, pages 31:1–31:13, 2018. ISBN 978-3-95977-078-1. [DOI:10.4230/LIPIcs.AofA.2018.31].

[28] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. [snap.stanford.edu/data], June 2014.

[29] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding, 2019. [dataset].

[30] Wolfram Research, Inc. Mathematica, Version 13.1, 2022. [wolfram.com/mathematica].