Experimental analysis of simple, distributed vertex coloring algorithms
(Extended Abstract)

Irene Finocchi*§ Alessandro Panconesi* Riccardo Silvestri*

Abstract

We perform an extensive experimental evaluation of very simple, distributed, randomized algorithms for $(\Delta + 1)$- and so-called Brooks-Vizing vertex colorings, i.e., colorings using considerably fewer than Δ colors. We consider variants of algorithms known from the literature, boosting them with a distributed independent set computation. Our study clearly determines the relative performance of the algorithms w.r.t. the number of communication rounds and the number of colors. The results are confirmed by all the experiments and instance families. The empirical evidence shows that some algorithms are extremely fast and very effective, thus being amenable to be used in practice.

1 Introduction

In this paper we perform an extensive experimental analysis of very simple, distributed vertex coloring algorithms. Their appealing features are simplicity of implementation, paired with a rather good performance in terms of colors used, and great speed. In particular, our algorithms can quickly compute vertex colorings using many fewer than Δ colors, where Δ denotes the maximum degree of the input graph. Because of these characteristics, we expect them to be quite useful in application areas, such as parallel sparse matrices computation or protocols for wireless networks.

1.1 The algorithms. In some sense this is a study of just one algorithm. It is by changing its few parameters and by switching on and off a heuristic step that several algorithms are obtained, with quite different characteristics. We now describe the basic algorithm. The input is an undirected graph G in which every vertex u has its own unique ID and is initially given a list (or palette) of available colors, denoted as L_u^0. The colors can be assumed to be consecutive natural numbers. The algorithm simply repeats the following *basic iteration*, until all vertices are colored. The current iteration is denoted by r.

The basic iteration:

1. *Wake up!* At the beginning of the iteration every vertex is *asleep*. In parallel, each uncolored vertex u *wakes up* with probability p_u^r.
2. *Try!* Each awaken vertex u, in parallel, selects a *tentative* color t_u from its list L_u^r uniformly at random.
3. *Conflict resolution.* If no neighbor of u has selected the same tentative color, t_u becomes the *final* color of u. Otherwise the attempt fails and u will try again at the next iteration.
4. *Deliverance?* Vertices that obtain a final color exit the algorithm. The others, in parallel, update their lists by removing all colors assigned as final colors to the neighbors.
5. *Feed the hungry.* If the palette of a vertex u runs out of colors, fresh new colors are introduced in the following way. Let c be the greatest color used as final color in u's neighborhood. The new list of u is set to $[\min\{c+1, \Delta+1\}]$.
6. *Back to square one.* All vertices go back to sleep.

It is apparent that the algorithm is distributed. Every iteration (also called "round") only requires $O(\log n)$ message size and can be implemented in a constant number of rounds in the synchronous, message-passing model of computation, which is the one adopted here.

This is a rough but useful theoretical model that approximates the behavior of those real distributed architectures for which the cost of routing messages is typically orders of magnitude greater than that of performing local computations. For instance, it has been used by several authors to analyze algorithm performance in wireless networks [2, 24, 25] .

Whereas this simulation is, in terms of resources and programming effort, relatively inexpensive, an implementation on a real distributed architecture would

*Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", via Salaria 113, 00198 Rome, Italy. E-mail: {finocchi,ale,silvestri}@dsi.uniroma1.it.

§Author supported in part by the Italian Ministry of University and Scientific Research (Project "Algorithms for Large Data Sets: Science and Engineering").

be much more demanding. Moreover, a truly distributed implementation would introduce factors that would make the experimental results more informative for the specific applications considered, but probably unsuitable for drawing general conclusions about the behavior of the algorithm.

In the course of our experiments we also made use of the following heuristic, dubbed here the *hungarian-folk step* (HS step).

> Given a color c, let G_c be the graph induced by all vertices that selected c as their tentative color at the current iteration.
> In parallel, for all colors c, compute an independent set of G_c as follows. Let π be a random permutation of the vertices of G_c: a vertex u enters the independent set if and only if it comes first than its neighbors in the total ordering induced by π.

When this is used, it replaces the conflict-resolution step of the basic algorithm. A crucial feature of the HS step is that it can be implemented very cheaply in our distributed setting. It does not require extra communication rounds and the message size is $O(\log n)$. It is well-known that, given a graph with n vertices, the HS step computes an independent set whose expected size is at least $n/(\hat{d}+1)$, $\hat{d}$ being the average degree of the graph. The very cute analysis appears to be part of hungarian folklore [19]. Luby showed that by iterating the HS step a maximal independent set is computed in $O(\log n)$ expected many rounds [22].

The random permutation of the HS step is needed to generate a random total ordering of the vertices. Strictly speaking, this requires global communication. If n, the number of vertices of the input network, is known, then it suffices to choose a random number in a large enough interval for a high probability analysis to go through. But in practice this is not a problem, because for any realistic network a few dozen random bits are enough to generate unique ID's. In fact, our experiments strongly suggest that using the original ID's works as well as truly (pseudo) random generated ones.

1.2 Our results. In this paper we perform two kinds of study. The first concerns algorithms for $(\Delta+1)$-vertex coloring, while the second concerns so-called *Brooks-Vizing colorings*, i.e., colorings using "many fewer" than Δ colors [10].

Algorithms for $(\Delta+1)$-vertex coloring. In this study the only dependent variable is the *running time* of the basic algorithm, measured as the number of communication rounds (basic iterations). The independent variables are three: the *wake-up probability*, the *initial palette setting*, and the HS step. The first is set to be a constant in the interval $(0, 1]$, while for the second the possibilities are two. If the initial lists are set to $[\Delta+1]$, for all vertices, then we have the *global setting*; otherwise, if $L_u^0 := [\deg(u)+1]$, for all vertices u, we have the so-called *local setting*. Likewise we shall speak of the *global* or the *local* version of the algorithm. Since both versions never run out of colors the Feed-the-hungry step can be omitted. Notice that in either case the algorithm has as many as $\Delta+1$ colors at its disposal and since in practice, as we shall see, they will always all be used, it is not meaningful to consider the number of used colors as a dependent variable. Finally, the HS step can be either set on or off, i.e., it can be used as a conflict resolution mechanism instead of Step 3.

There exist theoretical analyses of the *trivial algorithm*, corresponding to setting the wake-up probability to 1 for all vertices and iterations, and of *Luby's algorithm* corresponding to setting the wake-up probability to 1/2, for all vertices and iterations [15, 21]. Both algorithms compute a $(\Delta+1)$-coloring in $O(\log n)$ many expected iterations, and in fact do so with high probability [5, 17].

The main findings of our experimental study of these $(\Delta+1)$-coloring algorithms are the following.

- The closer the wake-up probability to 1, the faster the algorithm, regardless of other parameters. The basic trend is exemplified by Figure 1a, showing the running time as a function of the wake-up probability for a random graph with 1000 nodes and edge probability 0.1. We remark that the same behavior was exhibited in all our tests, regardless of the graph. Thus the trivial algorithm is the algorithm of choice for $(\Delta+1)$-coloring. In particular, when compared with Luby's algorithm it consistently turned out to be 2–3 times faster.

  The available asymptotic analyses do not explain this behavior [21, 15] and we leave it as an interesting open question.

- Globalization does not help, meaning that the fully distributed local version is not slower and often results in non-negligible color savings.

- As remarked, the algorithms with the hungarian-folk heuristic work just as well if vertex ID's are used instead of generating a random permutation.

- The algorithms are extremely fast. It is often the case that graphs with a thousand vertices or more are colored within 5 communication rounds.

Algorithms for Brooks-Vizing colorings. The second part of our study concerns so-called *Brooks-Vizing colorings*, i.e., colorings using "many fewer" than Δ colors [10]. Here we have two dependent variables, the *running time*, measured as the number of communication rounds (basic iterations), and the *number of colors* actually used by the algorithm. The latter could exceed the initial allotment because the lists can run out of colors. The independent variables are three: the *wake-up probability*, the *shrinking factor*, and the HS step. For this study we considered the local setting only. Each vertex u is initially given a list of $\deg(u)/s$ colors, where $s > 1$ is the shrinking factor. Essentially our study focuses on three algorithms.

- `Algorithm GP`. This is the algorithm from [10]. Here the wake-up probability of vertex u at iteration r is set to be $p_u^r := |L_u^r|/\deg_r(u)$, where $\deg_r(u)$ is the number of uncolored neighbors of u at iteration r. This choice (with high probability) mantains the invariant that the number of neighbors vying for coloring at any given round equals the current size of the color lists.

- `Algorithm Hungarian-GP (HGP)`. This is the same as above, with the conflict-resolution step replaced by the hungarian-folk heuristic.

- `Algorithm Constantly-Hungarian (CH)`. Here the wake-up probability is a constant in the interval $(0, 1]$ and the hungarian-folk step replaces the conflict-resolution step.

The available theoretical evidence concerning Brooks-Vizing colorings can be summarized as follows. If G is square- or triangle-free then it is always possible to color G with $O(\Delta/\log\Delta)$ colors [18, 14]. Moreover, if G is not only triangle-free but also Δ-regular and of high enough degree (i.e., $\Delta \gg \log n$) then, with high probability algorithm `GP` colors the input graph with as few as $O(\Delta/\log\Delta)$ colors within $O(\log n)$ rounds [10]. The performance regarding the number of colors is the best possible in view of a result of Bollobás showing that there exist graphs of arbitrarily large girth whose chromatic number is $\Omega(\Delta/\log\Delta)$ [3].

The main conclusions of our study can be summarized as follows.

- The best algorithm for Brooks-Vizing colorings is the simplest of them all, namely algorithm `CH` with the wake-up probability set to 1. We called it algorithm `Trivially-Hungarian (TH)`. In all our tests it outperformed the competitors in terms of speed, while computing colorings of the same quality (same number of colors). While we are able to provide a (hopefully) convincing heuristic explanation of why the HS step is so effective, a rigorous analysis is lacking. For instance, it would be very interesting if the results of [10] could be extended to algorithm `TH`.

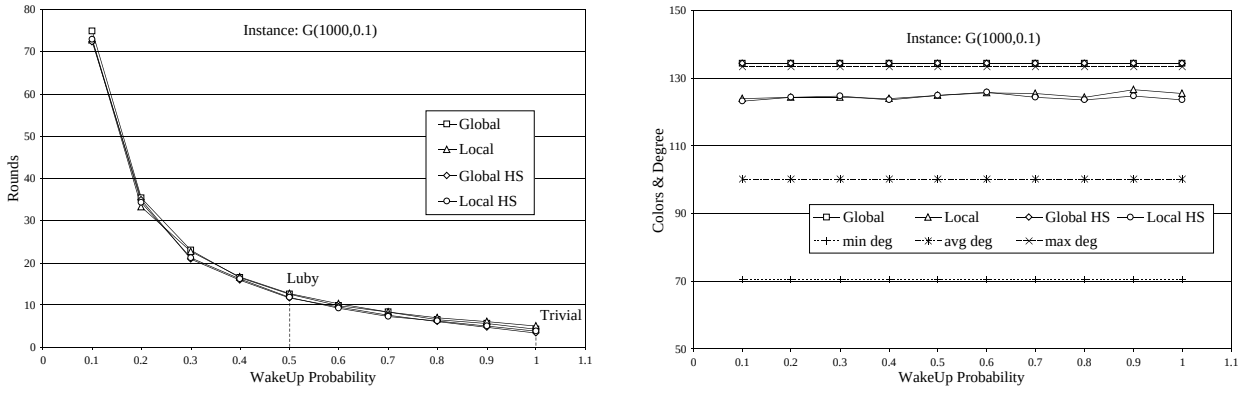
- A realistic value for the shrinking factor is a value inbetween 4 and 6, even though in many cases greater savings can be obtained. While the running time of algorithms `GP` and `HGP` grows rather quickly as s increases, algorithm `TH`'s running time grows quite slowly. In practice, algorithm `TH` can be made to run with values of s up to 20 or more. Even if this might not result in colorings better than those obtained with smaller values of the shrinking factor, it might be worth a try since the running time stays quite small (between 20 and 30 rounds for graphs with several thousands of edges).

- The algorithms are very fast. Even using the largest (feasible) shrinking factor, graphs with hundreds or even thousands of vertices are typically colored within 20-30 rounds.

- While the theoretical analyses only deal with triangle-free graphs, the empirical evidence shows that Brooks-Vizing colorings can be quickly computed distributively also in the presence of many triangles. It would be nice if the theoretical analysis could be extended to give characterizations of classes of graphs admitting such colorings. At present the only case known to us is that of line graphs [11].

1.3 Comparison with previous work. In spite of a quite extensive literature on vertex-coloring (see for instance Culberson's bibliography [7]), there are just a few experimental studies of parallel or distributed algorithms. Some are not quite related to the present work inasmuch as their experiments regard parallel algorithms designed for yielding very good colorings and thus they are neither simple nor fast (see, e.g., [20]).

Then, as far as we know, there are very few remaining papers experimenting with simple and fast parallel or distributed heuristics. Some of the studied heuristics derive from Luby's parallel algorithm for finding maximal indipendent sets [22]. The aim of the experimental study carried out by Jones and Plassmann [16] is showing that a specific parallel implementation of such an heuristic could be effective for bounded degree graphs. The performance of the heuristic is compared with that of simple sequential greedy heuristics (e.g., First Fit [12], Saturation Degree Ordering [4], Incidence Degree Ordering [6]). But no attempt is done to



(a) (b)

Figure 1: Algorithms for $(\Delta + 1)$colorings: Effects of varying the wake up probability with local and global palettes: for each algorithm the best wake up probability is 1.

understand the behavior of the parallel heuristic as a function of the characteristics of the input graphs. Of a similar flavor is the experimental study by Allwright *et al.* [1], in which heuristics based on Luby's algorithm and on parallel variations of a greedy strategy are compared. The algorithms were only tested on random triangulated meshes. More recently, Gebremedhin and Manne [9] proposed a simple parallel heuristic and they studied a specific parallel implementation of it on just a few input graphs arising from numerical problems.

We note that the emphasis of all these studies is on testing specific parallel implementations of simple heuristics with respect to some restricted class of input graphs. Differently from this paper, the general behavior of the proposed heuristics is not analysed. For instance, which characteristics of the input graphs (e.g. maximum degree, average degree, k-partiteness, etc.) affect the number of used colors or the number of communication rounds. Under this aspect the experimental study reported here is much more similar in spirit to that of [23].

2 More on the experimental design

The algorithms considered in the experiments were described in sufficient detail in the introduction. In the following we give more details concerning the implementation and the instances that we have been using.

Implementation details. The algorithms were implemented in ANSI C. The performance indicators used in the experiments are not affected by machine and compiler details, since we simulate a message-passing distributed network for which the running time is given by the number of communication rounds. We used the pseudo-random generator provided by the ANSI C standard function `rand`, using odd seeds and randomly generating the sequence of seeds for each test starting from a base value. To increase the statistical confidence of our results, in all the experiments we performed 5 to 10 runs of the algorithms on the same instance. Moreover, in the experiments on randomly generated graphs we used several trials per data point, i.e., we ran the algorithms on several instances by the same parameters.

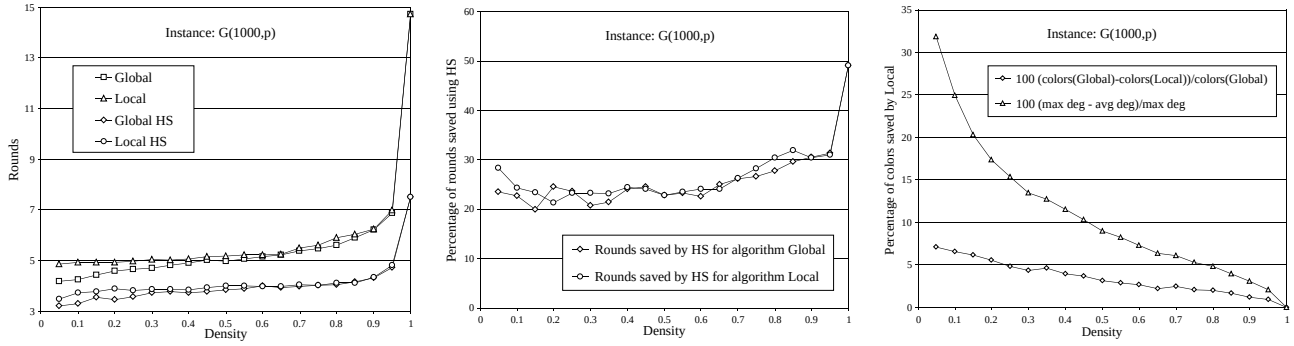
Instances. We tested the algorithms on random and synthetic graph families as well as on real test sets. In addition to random graphs from $G(n, p)$, i.e., random graphs with n vertices and edge probability p, we considered uniform random k-partite graphs from $G(n, p, k)$, where vertices are randomly assigned to one of k partition elements as nearly equal in size as possible (the smallest sets being one less than the larger) and a vertex pair uv is assigned an edge with probability p, provided u and v are from distinct partition elements. We have been also using Culberson's generators for coloring instances [7] and other test sets available from the FTP site related to the second DIMACS implementation challenge [8].

Lack of space prevents us from showing all our data in this extended abstract. We remark that, unless stated otherwise, the data we show exemplify the general pattern.

In Section 3 and Section 4 we discuss the behavior of the algorithms on random graphs. DIMACS benchmarks and real test sets are briefly considered in Section 5. The interested reader can also find additional charts and tables with the experimental package, that is available over the Web at the URL `http://www.dsi.uniroma1.it/~finocchi/coloring/`.

3 Experimental results: $\Delta + 1$ colorings

In this section we experimentally analyze distributed algorithms for computing $\Delta + 1$ colorings. The words "list" and "palette" are used as synonyms.



(a) (b) (c)

Figure 2: Behavior of the trivial algorithm with varying edge density. Very dense instances are difficult; the independent set heuristic saves approximately 25% rounds; the benefit of local palettes is greater on sparser instances.

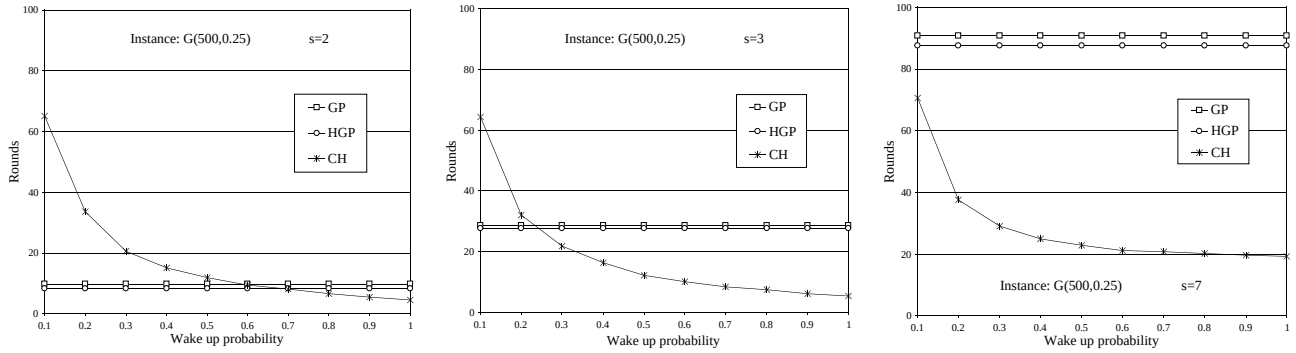
Effects of varying the wake up probability. We first show that the wake up probability, denoted here as w, has a significant impact on the number of rounds required to compute a $\Delta + 1$ coloring. The charts in Figure 1 have been obtained by running the distributed algorithms with global and local palettes on random instances from $G(1000, 0.1)$. As shown in Figure 1a, the number of rounds of each algorithm is *inversely proportional* to the wake up probability, thus obtaining the fastest convergence for $w = 1$. *The very same trend was observed in all our experiments.* The choices $w = \frac{1}{2}$ and $w = 1$ correspond to Luby's algorithm and to the trivial algorithm, respectively, analyzed in [21, 15]. In both cases the theoretical analysis shows that the probability that a vertex colors at a certain round is $\frac{1}{4}$. This contrasts with the typical trend observed in the experiments and exemplified by Figure 1a, that provides strong empirical evidence that the trivial algorithm is always faster, thus suggesting that the analysis may not be tight. The following explanation is accurate for the first round only, becoming less and less so as the algorithm progresses. Neverthless it offers a plausible explanation. Since $w = 1/2$ in Luby's algorithm, the expected degree of a vertex u in the graph induced by the vertices that are awake is $d_u/2$, while u's list has $d_u + 1 \sim d_u$ colors. Thus, a vertex colors with probability $\sim 1/2e^{\frac{1}{2}}$. On the other hand, in the trivial algorithm every vertex is awake and therefore a vertex obtains a final color with probabilty $\sim 1/e > 1/2e^{\frac{1}{2}}$. Notice that neither the HS step nor the global list initialization yield faster running times.

Intuitively, if the number of vertices is much larger than Δ, in the global case all available colors will be used (see Figure 1b). In the local case on the other hand, the smaller the number of high degree vertices, the higher the probability that the final color will use less than $\Delta + 1$ colors. This intuition is confirmed by the chart in Figure 1b: using local palettes leads to slightly better colorings, though the number of colors used is always greater than the average degree of the graph. (Figure 2c will report on the color savings that can be obtained using the local version as a function of the edge probability.)

Experiments with different graph densities. In order to analyze how the number of rounds and colors depends on the density of the graph, we ran the trivial algorithm with global and local palettes on several graph instances of varying edge density. We consider the trivial algorithm only, because it consistently proved itself to be the best. Figure 2 reports the results for random instances from $G(1000, p)$, with p increasing from 0.05 to 1. Although global palettes guarantee faster convergence (Figure 2a), the effect is once again rather negligible. Notice that the algorithms are extremely fast.

The number of rounds does not depend very much on the density of the graph except for densities beyond 0.9. After this point the number of rounds grows considerably. In particular cliques require significantly more rounds to be colored. This behavior has been confirmed also on random k-partite graphs (charts are not reported here due to the lack of space). On these instances the number of rounds stays almost constant. This is because the edge density cannot come too close to 1. Figure 2b shows the percentage of rounds saved when the HS step is used. The saving is roughly 25%, with both local and global palette, and therefore significant.

We observed that the number of colors linearly increases with the density, following the trend of the maximum degree curve, and that the benefit of using local palettes is bigger for sparser graphs, as shown by Fig-



(a) (b) (c)

Figure 3: Brooks-Vizing colorings: effects of varying the wake up probability.

ure 2c. The percentage of colors saved is approximately 7% for $p = 0.1$ and decreases linearly with the density down to 0 for $p = 1$. Colorings obtained on cliques by local and global palettes are obviously the same.

4 Experimental results: Brooks-Vizing colorings

In this section we experimentally analyze distributed algorithms for computing colorings with significantly fewer than Δ colors. Recall that we tested the local version only, in which each vertex u is initially given a list of $\deg(u)/s$ colors, s being the shrinking factor. The algorithms tested were: (a) algorithm `GP`, in which the wake up probability at round r is set to be $|L_u^r|/\deg_r(u)$; (b) algorithm `HGP`, which is the same as above with the HS step; and (c) algorithm `CH`, in which the wake-up probability w is constant and the HS step is always used. Algorithm `TH` is algorithm `CH` with $w = 1$.

Effects of varying the wake up probability. The charts in Figure 3 have been obtained by running algorithms `GP`, `HGP`, and `CH` on random instances from $G(500, 0.25)$ and plot the number of rounds required to compute $\frac{\Delta}{s}$ colorings for different values of the shrinking factor s. The basic message is that, once again, the best choice for the wake-up parameter is 1 (this applies to algorithm `CH` only).

Though the trend of each curve is maintained, the actual performance changes as s and w vary. The running times of algorithms `GP` and `HGP`, not depending on w, appear in Figure 3 as straight lines. `GP` appears to benefit only to a limited extent from the independent set computation and the benefit is greater for bigger values of s. As expected, the independent set turns out to be more useful when there are numerous conflicts, and this happens for smaller palettes. Algorithm `CH` can be worse than `GP` and `HGP` only for $s = 2$, and becomes much better for larger values of s. The running time of algorithm `CH` exhibits a good behavior. It is *inversely proportional* to the wake up probability, obtaining the fastest convergence for $w = 1$. In subsequent experiments we will therefore report only on the behavior of `TH`. We remark that if $w = 1$ algorithm `CH` is always faster than `GP` and `HGP`.

Effects of varying the shrinking factor. Figure 3 and Figure 4 show how the running time (number of communication rounds) grows as s increases. Figure 3 shows a marked deterioration of the running time of `GP` and `HGP` as s takes the values $2, 3, 7$. The running time of algorithm `CH` also grows, but much more slowly, especially that of algorithm `TH`. The same conclusion is reinforced by Figure 4a, in which the running time of both versions (local and global) of the trivial $(\Delta + 1)$-coloring algorithm is also reported. While the trivial algorithm is considerably faster than the reduced palette algorithms, it also uses many more colors (Figure 4b). Notice that algorithms `GP` and `TH` use approximately the same number of colors, but `GP` is much slower. The seemingly asymptotic trend of the number of colors used as a function of the shrinking factor is commented upon in a paragraph below.

Figure 4c shows that algorithm `TH` saves around $70\%-80\%$ of the number of rounds of `GP` (the percentage is smaller only for $s = 2$). Since the same relative performance of `GP` and `TH` has been observed in all our experiments, we can definitely conclude that algorithm `TH` is the algorithm of choice for computing Brooks-Vizing colorings in a distributed setting.

A possible explanation to account for this marked difference in speed is the following. In the initial rounds of algorithm `GP` a vertex wakes up with probability $\sim 1/s$. A vertex that wakes up has on average a number of neighbors vying for coloring equal to the size of its list of available colors. Therefore a vertex colors itself with probability $\sim 1/(se)$. In contrast, algorithm `TH` first partitions the vertex set into a certain number of classes, say k. The average degree of a conflict graph G_c– the

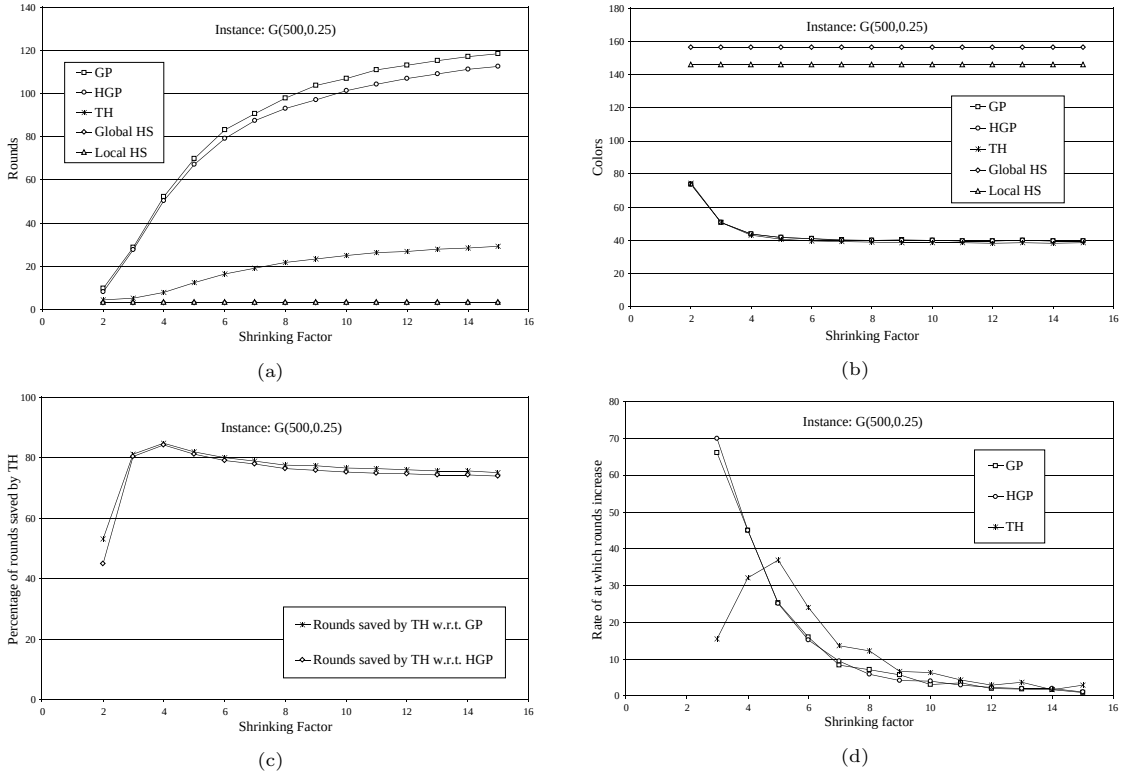


Figure 4: Brooks-Vizing colorings: effects of varying the shrinking factor.

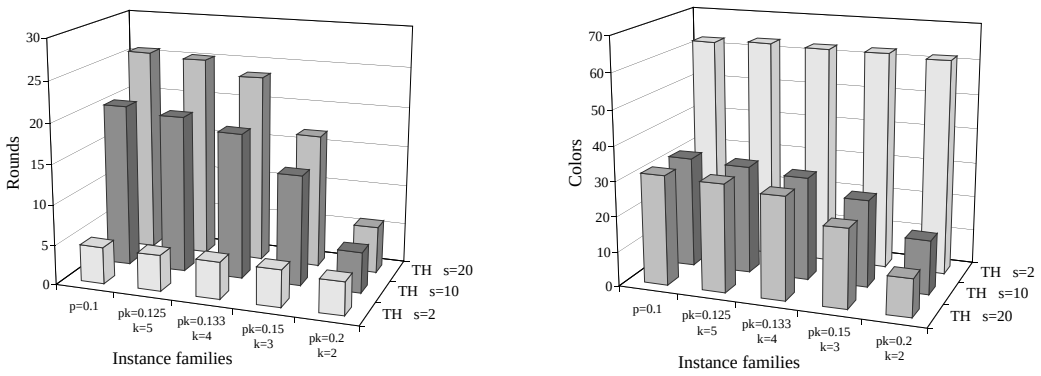
graph induced by all vertices that picked tentative color c– is s. Recall that an application of the HS step in a graph with x vertices and average degree d yields an independent set of size $\sim x/d$. Therefore on average n/s vertices will get a final color after one step of algorithm `TH`, as opposed to $\sim n/(se)$ with algorithm `GP`. Although this rough analysis is plausible only for the initial rounds, to some extent it might explain the very different running times.

Finally, Figure 4d plots $r(s)$, the rate at which the number of rounds increases. The chart should be interpreted as follows: $r(s) = k$ means that when shrinking factor s is used instead of $s-1$ the number of rounds increases by $k\%$. Both `GP` and `TH` exhibit a good behavior w.r.t. this measure, as curves are decreasing for $s \geq 5$.

Is the best shrinking factor small? Figure 4b seems to suggest that the shrinking factor tends to an asymptotic value, i.e., larger values of s do not yield better colorings. This is not always the case however. Often larger values of s do result in better colorings. In particular, we investigated this point on random k-partite instances and observed a surprising phenomenon. We fixed a distribution $G(n,p)$ (i.e., we fixed n and p) and generated k-partite random graphs from $G(n,p_k,k)$ for different values of k. These graphs all had the same expected density, and the same minimum, average and maximum degree of graphs from $G(n,p)$ (this is done by setting $p_k = k/(k-1)p$). Therefore the local properties were the same but the chromatic number– a global property– differed. We then ran algorithm `TH` on these instances with different shrinking factors. The outcome is reported in Figure 5.

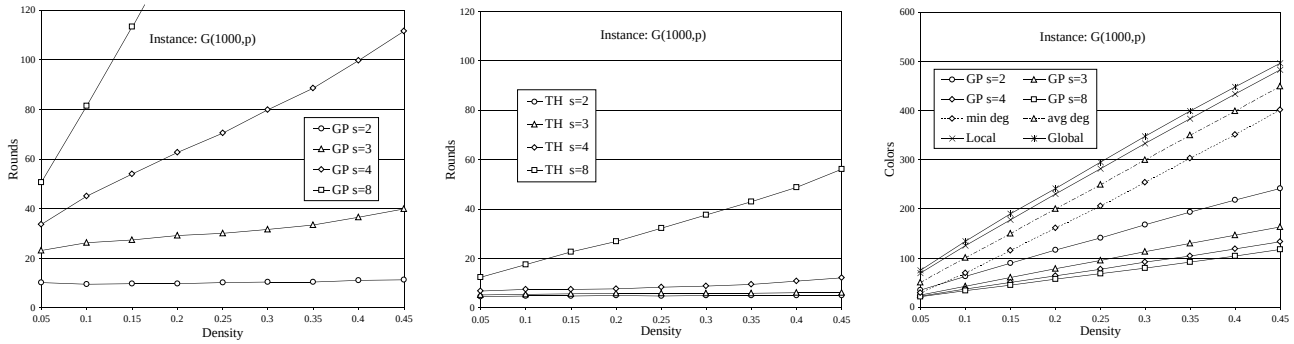
The figure shows that it pays off to consider bigger values of s and that in spite of the fact that locally the graphs look the same, the number of colors used by the algorithm differs, being lower for graphs with smaller chromatic number. Presently we are not able to offer any plausible explanation of this behavior.

Experiments for different graph densities. Figure 6 is concerned with reduced palette algorithms on random instances $G(1000,p)$ for $0.05 \leq p \leq 0.45$ and is the companion to Figure 2. Figure 6a and Figure 6b report on the number of rounds of `GP` and `TH`, respectively, for different shrinking factors. In both cases the number of rounds increase linearly with the density and curves related to bigger shrinking factors are steeper. The range of the values on the y-axes in the two charts is the same, making it easier to grasp the advantage of `TH` over `GP`: though not reported in the chart, we point out that `GP s=8` requires 293 rounds for $p = 0.45$, instead of 56 rounds used by `TH s=8`. As `GP` and `TH` use the same



(a) (b)

Figure 5: Algorithm TH for different shrinking factors on k-partite instances with the same density and degrees: the smaller k, the easier the instance.



(a) (b) (c)

Figure 6: Brooks-Vizing colorings: experimenting different graph densities.

number of colors, Figure 6c reports only on the coloring of GP. Differently from $(\Delta + 1)$-colorings, where the number of colors is sharply concentrated near the maximum degree, the curves related to reduced palettes are less steep and, as expected, their slope is smaller for bigger values of the shrinking factor.

5 Results with DIMACS instances

In this section we briefly report on the performance of algorithms HGP and TH on a subset of the instances used in the second DIMACS challenge [13], on other test sets available from the DIMACS FTP site, and on graphs with decreasing degree variance generated using Culberson's clique-driven generator. We consider both colors and rounds and, with respect to colors, in the case of DIMACS benchmarks we compare HGP and TH against two well known algorithms: Iterated Greedy and Hybrid [13]. These algorithms are known to produce very good colorings, though can require a considerable amount of time (minutes or even hours). Our distributed algorithms compare rather well.

We ran algorithms HGP and TH at least 10 times per instance and we averaged the values of colors and rounds over the runs. The value of the shrinking factor used on each instance is reported between parentheses (results for different shrinking factors are available with the experimental package).

The table in Figure 7 confirms that TH is considerably faster than HGP and shows that TH usually obtains slightly better colorings. It also reports the standard deviation of the number of rounds, that we were precluded from discussing throughout the paper due to space limitations: note that the standard deviation of TH is always much smaller than the standard deviation of HGP.

References

[1] J.R. Allwright, R. Bordawekar, P.D. Coddington, K. Dincer, and C.L. Martin. A Comparison of Parallel Graph Coloring Algorithms. Technical Report SCCS-666, Northeast Parallel Architecture Center, Syracuse University, 1995.

[2] S. Basagni, Finding a maximal weighted independent set in wireless networks, *Telecommunication Systems*, 18:1,2, 155-168, 2001.

Instance	HGP Rounds (Std. Dev.)	TH Rounds (Std. Dev.)	HGP Colors (s)	TH Colors (s)	IG Colors	HY Colors
C2000.5.col	661.2 (33.55)	122.2 (1.47)	229.6 (10)	226.0 (10)	190	DNR
flat300_20_0.col	119.2 (6.03)	32.6 (1.02)	49.1 (10)	47.7 (10)	20.2	20
flat300_26_0.col	123.6 (10.2)	32.8 (1.4)	50 (10)	48.3 (10)	37.1	32.4
flat300_28_0.col	125.6 (8.55)	33.1 (0.94)	49.8 (10)	48.2 (10)	37	33
flat1000_50_0.col	355.3 (13.91)	76.7 (1)	131 (10)	128.8 (10)	65.6	97
flat1000_60_0.col	365.4 (12.06)	76.9 (1.22)	130.5 (10)	128.9 (10)	102.5	97.8
flat1000_76_0.col	362.4 (3.77)	76.6 (1.02)	130.6 (10)	128.6 (10)	103.6	99
latin_square_10.col	514.7 (25.52)	91.9 (2.11)	159.6 (10)	159.9 (10)	106.7	109.25
le450_15a.col	43.8 (4.19)	13.9 (0.7)	23.5 (10)	21.8 (10)	17.9	15
le450_15b.col	45.1 (2.98)	14.8 (0.98)	23.4 (10)	22.3 (10)	17.9	15
le450_15c.col	69.7 (3.52)	18.8 (0.98)	33 (10)	30.9 (10)	25.6	16.6
le450_15d.col	70.8 (5.81)	19.0 (0.77)	33.1 (10)	30.9 (10)	25.8	16.8
mulsol.i.1.col	72.5 (5.64)	38.7 (1.55)	49.3 (10)	49.1 (10)	49	49
school1.col	104.1 (6.02)	26.8 (1.07)	46 (15)	43.4 (15)	14	14
school1_nsh.col	85.1 (5.41)	20.9 (2.3)	40.4 (10)	39.8 (10)	14.1	14
inithx.i.1.col	50.2 (6.55)	17.4 (3.92)	62.7 (10)	56.2 (10)		
inithx.i.2.col	28 (2.0)	4.9 (0.94)	50.9 (10)	47.2 (10)		
inithx.i.3.col	30.8 (5.74)	4.9 (0.7)	52 (10)	46.6 (10)		
fpsol2.i.1.col	89.5 (7.48)	42.8 (0.98)	65.5 (10)	65.1 (10)		
fpsol2.i.2.col	34.6 (3.26)	6.2 (0.87)	41.9 (10)	35.2 (10)		
fpsol2.i.3.col	33.6 (5.42)	6.2 (0.74)	40.3 (10)	34.2 (10)		
degree_variance.1	70.4 (7.49)	26.6 (4.22)	46.8 (20)	43.4 (20)		
degree_variance.2	66.2 (7.44)	21.6 (3.13)	43.6 (20)	41.0(20)		
degree_variance.3	59.6 (6.46)	20.6 (3.2)	41.8 (20)	40.0 (20)		
degree_variance.4	61.8 (9.51)	19.8 (2.31)	42.2 (20)	40.6 (20)		
degree_variance.5	58.8 (6.4)	25.4 (3.26)	40.6 (20)	40.0 (20)		
degree_variance.6	56.6 (8.4)	27.0 (2.6)	40.6 (20)	40.0 (20)		
degree_variance.7	55.6 (7.73)	22.6 (5.38)	40.0 (15)	40.0 (15)		
degree_variance.8	44.2 (9.8)	17.2 (5.52)	39.0 (15)	39.0 (15)		

Figure 7: Results on DIMACS instances.

[3] B. Bollobás. Chromatic number, girth, and maximal degree. *Discrete Mathematics* 24:311–314, 1978.

[4] D. Brélaz. New Methods to Color Vertices of a Graph. *Communications of the ACM*, 22:251–256, 1979.

[5] S. Chaudhuri and D. Dubhashi. Probabilistic recurrence relations revisited. *Theoretical Computer Science*, to appear.

[6] T.F. Coleman and J.J. Moré. Estimation of Sparse Jacobian Matrices and Graph Coloring Problems. *SIAM Journal on Numerical Analysis*, 20:187–209, 1983.

[7] J.C. Culberson. `http://web.cs.ualberta.ca/~joe/Coloring/`.

[8] FTP site of DIMACS implementation challenges. `ftp://dimacs.rutgers.edu/pub/challenge/`.

[9] A.H. Gebremedhin and F. Manne. Scalable Parallel Graph Coloring Algorithms. *Concurrency: Practice and Experience*, 12:1131–1146, 2000.

[10] D.A. Grable and A. Panconesi. Fast distributed algorithms for Brooks-Vizing colourings. *Journal of Algorithms*, 37:85–120, 2000. Special issue for SODA 98, the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms.

[11] D.A. Grable and A. Panconesi. Nearly optimal distributed edge colouring in $O(\log \log n)$ rounds. *Random Structures and Algorithms*, 10(3):385–405, 1997. Preliminary version in Proceedings of the Eight Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 97).

[12] G.R. Grimmet and C.J.H. McDiarmid. On Colouring Random Graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 77:313–324, 1975.

[13] D.S. Johnson and M.A. Trick. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[14] A.R. Johansson. Asymptotic choice number for triangle-free graphs. Preprint, DIMACS, September 30, 1996.

[15] Ö. Johansson. Simple distributed $\Delta + 1$-coloring of graphs. *Information Processing Letters* 70:229–232, 1999.

[16] M.T. Jones and P.E. Plassmann. A Parallel Graph Coloring Heuristics. *SIAM Journal on Scientific Computing*, 14(3):654–669, 1993.

[17] R.M. Karp. Probabilistic recurrence relations. In proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 91), 190–197, 1991.

[18] J.H.Kim, On Brooks' Theorem for sparse graphs, *Combinatorics Probability and Computing*, 4 (1995) 97-132.

[19] J. Körner. Private communication.

[20] G. Lewandowski and A. Condon. Experiments with Parallel Graph Coloring Heuristics and Applications of Graph Coloring. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[21] M. Luby. Removing randomness in parallel without processor penality. *Journal of Computer and System Sciences*, 47(2):250–286, 1993.

[22] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[23] M.V. Marathe, A. Panconesi, and L.D. Risinger jr. An Experimental Study of a Simple, Distributed Edge Coloring Algorithm. In proceedings of the 12th ACM Symposium on Parallel Algorithms and Architectures (SPAA'00), 166–175, 2000.

[24] P. Sinha, R. Sivakumar, and V. Bharghavan, Enhancing ad hoc routing with dynamical virtual infrastructures, Proceedings of IEEE Infocom 2001, 1-10.

[25] P. Sinha, R. Sivakumar, and V. Bharghavan, CEDAR: a core-extraction distributed ad-hoc routing algorithms, *IEEE Journal on Selected Areas in Communication*, 17, 8, 1454-1458, August 1999.