Lower Bounds for On-line Graph Coloring

Magnús M. Halldórsson* Mario Szegedy†

April 27, 1994

Abstract

An algorithm for vertex-coloring graphs is said to be *on-line* if each vertex is irrevocably assigned a color before later vertices are considered. We show that for every such algorithm, there exists a $\log n$-colorable graph for which the algorithm uses at least $2n/\log n$ colors. This also holds for randomized algorithms, to within a constant factor, against an oblivious adversary.

We then show that various means of relaxing the constraints of the on-line model do not reduce these lower bounds. The features include presenting the input in blocks of up to $\log^2 n$ vertices, recoloring any fraction of the vertices, presorting vertices by degree, and disclosing the adversary's previous coloring.

1 Introduction.

1.1 On-line Computation.

An *on-line algorithm* answers a sequence of requests under the following, informally specified constraints:

- A request must be answered before the next request is arrives.
- No information about future requests is available, including the number and the ordering of the requests.
- Each answer is irrevocable.

Once the questioning is over the answers are compared to the answers given by an off-line algorithm, and the algorithm is ranked by the "quality" of his answers relative to an optimal off-line answer sequence. The requests can be assumed to originate from an all-knowing, devious adversary.

There are several important reasons for studying on-line computation. The most commonly cited reason is that it corresponds naturally to the real world: time is uni-directional, past events cannot be reversed, the future is uncertain, and Murphy's Law rules.

Another reason is that on-line algorithms nicely complement many well-studied algorithmic frameworks, such as real-time computation, incremental/dynamic algorithms, prefix solutions of discrete structures, highly recursive computation, and single pass, greedy, and first-fit algorithms.

A third reason is that on-line computation forms an elegant framework for analyzing algorithms with incomplete information or incomplete access to the input.

1.2 Graph coloring

The problem of *coloring* a graph is that of assigning the vertices to fewest bins possible so that no two vertices assigned to the same bin are adjacent. In the on-line version of the problem, a vertex is given along with its edges to the previous vertices; the algorithm must irrevocably assign the vertex to a bin before proceeding to the next vertex, but we do not impose any restrictions on the power of the algorithm. The *performance ratio* of a coloring algorithm is the maximum ratio of the number of bins used to the chromatic number (the minimum number of colors required), ranging over all input graphs.

This problem has been much studied, particularly for specific classes of graphs [10, 4, 7]. Lovász, Saks and Trotter [11] gave an algorithm for general graphs that obtains a performance ratio of $\mathcal{O}(n/\log^* n)$, slightly improving the trivial bound of n. Vishwanathan [15] gave a randomized algorithm which attains a performance ratio of $\mathcal{O}(n/\sqrt{\log n})$ against an oblivious adversary. His algorithm was modified in [5] to improve the performance ratio to $\mathcal{O}(n/\log n)$.

In this paper, we prove a $2n/\log^2 n$ lower bound for deterministic on-line graph coloring, which holds to within a constant factor for randomized algorithms. The previous best lower bounds known were $\Omega(\log n)$ for trees [1, 4, 10], and $\Omega(\log^k n)$ for k-colorable graphs, where k is fixed [15].

1.3 Variations of the on-line models.

On-line computing places strong constraints on the algorithm; we would like to extend our lower bounds to a more general class of algorithms obtained by weakening some of the restrictions. In the context of the motivating applications, it is in fact natural to relax various conditions.

Take for instance the "answer before next request" condition. The most important property of a real-time algorithm is the ability to respond within a prescribed delay; such a delay may be sufficient to allow for some *lookahead*: the viewing of a few of the subsequent requests. Similarly, in a system with a two-level store, the input can be expected to appear in blocks rather than in individual units. And, a single-pass algorithm may have enough memory to keep a limited number of requests in a queue for later processing, which has, for example, been shown to be useful in a server problem with excursions [2].

The "irrevocability" condition can also often be made more flexible. Decisions may be reversible as long as the changes are infrequent and localized. In the case of the bin packing problem, fewer bins are needed if constant amount of repacking per item is allowed [3]. It is also well known that inputs sorted by non-increasing item size – a relaxation of the "unknown ordering" principle – require significantly fewer bins [8].

We consider the above variations in the context of the graph coloring problem, and show them all to yield limited or no improvement. Our lower bound holds to within a constant factor even if the algorithm is given the advantage of $\log^2 n$-size lookahead (or blocked input or input buffer), allowed to reassign colors to a constant fraction of the vertices, or allowed to reorder the input based on vertex degrees. This is optimal in the sense that a greater lookahead/recoloring would trivialize the problem.

All of our results hold in a model that significantly restricts the power of the adversary. The adversary must construct her own coloring on-line and reveal it after the algorithm answers. The results are optimal within that framework. Finally, the constructions also reveal in advance the input length and the number of colors required, both of which the standard framework specifies to be *a priori* unknown to the algorithm.

The rest of the article is organized as follows. The following section contains definitions of our "transparent" on-line model and related terminology. Section 3 starts with essential properties of the definitions, leading to the main results: lower bounds for deterministic and randomized

on-line coloring. Section 4 continues with lower bounds for the various relaxations of the on-line model, and an explanation of the optimality of the basic construction. Some ideas for further work are suggested at the end.

2 Definitions

We consider graphs with an imposed ordering on the vertices $\langle v_1, v_2, \ldots, v_n \rangle$. A *pre-neighbor* of a vertex v_t is an adjacent vertex that precedes v_t in the given ordering. The *pre-adjacencies* of v_t, denoted $Adj^-(v_t)$, is a list of its pre-neighbors. Note that a sequence of all the pre-adjacencies fully specifies the graph. A function f of the vertices is a *proper coloring* if $v_j \in Adj^-(v_t)$ implies $f(v_j) \neq f(v_t)$, for each v_j, v_t.

Transparent on-line coloring is a combinatorial game between two players: the *adversary* A, and the *algorithm* B. The game consists of an undetermined, but finite, number of request-answer-reply rounds $1, 2, \ldots, n$, where round t is as in fig. 1.

A poses a pair $(v_t, Adj^-(v_t))$, where $Adj^-(v_t) \subseteq \{v_1, \ldots, v_{t-1}\}$
B answers with an integer $Bin(v_t)$, a proper coloring of v_t.
A responds with an integer $Col(v_t)$, a proper coloring of v_t.

Figure 1: *Transparent On-Line Coloring Game*

Without the last step in each round, the game is equivalent to the usual on-line coloring problem. We say that an adversary is *transparent* if, as above, it reveals its decisions following the algorithm. In this paper, we fix in advance the number of rounds n, with the performance evaluation occurring only at the end. We generally describe only the t-th round, intending the description to be applicable to each round.

We say that the algorithm colors with *bins* and the adversary with *colors* in order to easily distinguish the two. Both of these are objects from some finite sets, which we associate with the positive integers. The term *bin* will refer both to the cardinal assigned to vertices, as well as the set of vertices that have been assigned that bin number. For a vertex v_t, the set $Avail(v_t)$ of *admissible colors* consists of the colors *not* used by its pre-neighbors.

Define the *hue* of a bin to be the set of colors of the vertices in the bin, $Hue(b) = \{Col(v_i) : Bin(v_i) = b\}$. A *hue collection* H is a set of all the non-empty hues. Both of these are dependent on an implicit point in time.

The term k will denote the number of colors sufficient to properly color the graph constructed. We shall assume in this paper that k is divisible by 16 in order to simplify the presentation. Let $[k]$ denote the set $\{1, 2, \ldots, k\}$, and $[k]^{(k/2)}$ the collection of all subsets of $[k]$ of size $k/2$. Finally, let *Random* be a function that gets as input a finite set, and selects an item with uniform probability.

3 Main course.

3.1 Preliminaries.

Our task as the adversary is to decide on the adjacencies and the color of each vertex. The adjacencies are determined by first selecting a set of admissible colors, and then applying the following rule: A vertex v_t shall be adjacent to a previous vertex v_i iff the color of v_i does not belong to the set of admissible colors. The idea is that if we make the vertex adjacent to any

previous node of a given color it doesn't hurt to make it adjacent to all nodes of that color. From this definition, we get the following property.

Observation 1 $Hue(Bin(v_t)) \subseteq Avail(v_t)$.

Our task is to select a set of admissible colors, and to select one of those as the vertex color, so that the algorithm is bound to introduce many bins. Our measure of progress is the number of distinct pairs of bin and color values assigned to the vertices, or $|\{(Bin(v_i), Col(v_i)) : 1 \leq i \leq t\}|$.

To see that this number is actually indicative of the progress made, notice that the number of bins is at least the number of bin/color pairs divided by the maximum number of pairs with the same bin number. But the latter is bounded by the number of admissible colors for the last vertex assigned to that bin, by Obs. 1, giving us the following bound.

Observation 2 *If m distinct bin/color pairs have been assigned after round t, then at least $m/(\max_{i \leq t} |Avail(v_i)|)$ bins have been used.*

It is useful to note that if the bin hue is a strict subset of the set of admissible colors, then we are in a situation to make progress, since then the vertex can be assigned a color different from those of other vertices in the bin.

Observation 3 *If $Col(v_t) \in Avail(v_t) - Hue(Bin(v_t))$ then progress is made.*

3.2 Adversary against deterministic algorithms.

Let k be a positive even number, and $n = k/2 \cdot \binom{k}{k/2}$ be the number of rounds. Let *any* denote a non-deterministic selection operator that gets as input a set and returns some member. Fig. 2 gives the adversary's adaptive strategy; set H is the current set of hues.

$$\begin{aligned} Avail(v_t) &= any([k]^{(k/2)} - H) \\ Adj^{-}(v_t) &= \{v_i : Col(v_i) \notin Avail(v_t) \text{ and } i < t\} \\ Col(v_t) &= any(Avail(v_t) - Hue(Bin(v_t))) \end{aligned}$$

Figure 2: ADVERSARY STRATEGY: *Adaptive Construction*

Theorem 1 *The performance ratio of any deterministic on-line coloring algorithm is at least $2n/\log^2 n$.*

Proof. Each round of the game contributes at most one element to the combined membership of the hues in H. As a result, as long as the game is played, i.e. as long as $t \leq k/2\binom{k}{k/2}$, at least one of the $\binom{k}{k/2}$ sets of size $k/2$ is not a member of H, and can therefore be assigned to $Avail(v_t)$. Since $Avail(v_t)$ cannot equal any bin hue (by definition) but must include the hue of the selected bin (by Obs. 1), the inclusion must be strict, and $Col(v_t)$ is thus well-defined. Each round therefore makes progress (by Obs. 3), and the number of bins must be at least $n/(k/2)$ (by Obs. 2). Since the number of colors is at most k, the performance ratio is at least $2n/k^2$, where $k = \log n(1 - o(1))$. ∎

To appreciate the simplicity of the construction, the reader is encouraged to work through an example.

3.3 Adversary against randomized algorithms.

Randomized on-line algorithms can be evaluated in at least three different ways, depending on the power of the adversary in question. The weakest of these, the *oblivious* adversary, must construct the whole input in advance before feeding it to the algorithm. It is against this adversary that the algorithms of [15, 5] are successful.

We show in this section that, for any randomized on-line coloring algorithm there exists a k-colorable graph on which the algorithm will use at least expected n/k bins, where $k = \mathcal{O}(\log n)$. By Yao's lemma [16], it suffices to show that there exists a distribution of k-colorable graphs for which the average number of colors used by any *deterministic* algorithm is at least n/k.

We construct a distribution of k-colorable graphs, in $n = 2^{k/4}$ rounds as in fig. 3. Observe that the construction is oblivious, since decisions made by the algorithm, such as bin assignments, are never consulted.

$$\begin{aligned} Avail(v_t) &= Random([k]^{(k/2)}) \\ Adj^{-}(v_t) &= \{v_i : Col(v_i) \notin Avail(v_t) \text{ and } i < t\} \\ Col(v_t) &= Random(Avail(v_t)) \end{aligned}$$

Figure 3: ADVERSARY STRATEGY: *Oblivious Construction*

Consider a given choice for $Avail(v_t)$. The probability that the random color assignment yields a successful round equals

$$\frac{|Avail(v_t) - Hue(Bin(v_t))|}{|Avail(v_t)|}.$$

Since the graphs are constructed in advance, we cannot assume anything about the actual values of the hues. Instead, we shall show that for a randomly chosen $Avail(v_t)$ and any fixed collection H of at most n hues, the difference $|Avail(v_t) - h|$ minimized over all hues h in H, can be expected to be large. For a $k/2$-set p, and a collection H of subsets of $[k]$, define

$$dist(p, H) = \min_{h \in H} |p - h|.$$

We have that for any hue collection H,

$$|Avail(v_t) - Hue(Bin(v_t))| \;\geq\; dist(Avail(v_t), H).$$

If we now let $Avail(v_t)$ vary, the probability of a successful round is at least

$$\frac{E[dist(Avail(v_t), H)]}{|Avail(v_t)|}.$$

The performance ratio lower bound follows easily from the following bound on this distance function.

Lemma 1 *Let H be any collection of subsets of $[k]$, and let p be a randomly chosen subset of $[k]$ of size $k/2$. If $|H| \leq 2^{k/4}$ and k is large enough, then*

$$E[dist(p, H)] \geq k/4. \tag{1}$$

Proof. We claim that

$$\Pr[dist(p,H) \le s] \le \frac{\binom{k/2+s}{s}|H|}{\binom{k}{k/2}}. \tag{2}$$

This implies, using Stirling's approximation, that

$$\Pr[dist(p,H) \le 0.28k] \le 1.01^{-k}$$

for $|H| \le 2^{k/4}$. Now,

$$E[dist(p,H)] \;\ge\; 0.28k \cdot Pr[dist(p,H) \ge 0.28k] \;\ge\; 0.25k$$

for k large enough.

To see that Inequality 2 holds, let H' contain the sets in H whose size is at least $k/2-s$. Each set in H' is a subset of at most $\binom{k/2+s}{s}$ sets of size $k/2$. Therefore, we get $\Pr[dist(p,H') \le s] \le \binom{k/2+s}{s}|H'|/\binom{k}{k/2}$ by summing up the probabilities of the sets in H'. But none of the elements of $H - H'$ affect these probabilities, since they are too far away from p. Hence the claim and the lemma. ∎

Theorem 2 *The performance ratio of any randomized on-line coloring algorithm is at least* $n/(16\log^2 n)$.

Proof. Consider the execution of a fixed deterministic algorithm on the graphs drawn from the distribution constructed above. The probability that a node makes progress is at least

$$\frac{E[dist(Avail(v_t),H)]}{|Avail(v_t)|} \ge \frac{k/4}{k/2} = 1/2.$$

Hence, the expected number of bin/color pairs is at least $n/2$, by linearity of expectation, and the expected number of bins at least n/k by Obs. 2. Thus the performance guarantee is at least n/k^2, where $k = 4\log n$. ∎

4 Bells and Whistles.

4.1 Blocked input.

One of the caveats of computing on-line is that an answer is often proved wrong or ineffective almost immediately after it's uttered. One hope would be that if just a little bit was known about what's coming up on the horizon, far better decisions could be made. No such luck. We can show that even with moderate visibility, the performance would not improve.

We construct the input in an oblivious manner, in blocks of size $k^2/128$ organized as a sequence of $k/16$ disjoint cliques of size $k/8$. As before, the vertices are adjacent to vertices in previous blocks according to a randomly chosen set of admissible colors.

The formal definition is given in fig. 4. If vertex v_t is in the b^{th} block and inside this block is in the c^{th} clique, then $t = bk^2/128 + ck/8 + t'$, where $0 \le t' < k/8$ (blocks and cliques are counted from 0). The block-number and clique-number of v_t are $blk_t = b$ and $cliq_t = c$. Let $cc(v_t) = \{Col(v_i) : i < t \text{ and } blk_i = blk_t \text{ and } cliq_i = cliq_t\}$ represent the colors of the previous vertices in the same clique. Let $n = 2^{k/4}$ be the number of rounds.

Lemma 2 *Against any randomized algorithm with blocked input of size up to* $k^2/128$*, an expected constant fraction of the rounds in fig. 4 will make progress.*

$$
\begin{aligned}
Avail(v_t) &= Random([k]^{(k/2)}) \\
Adj^{-}(v_t) &= \{v_i : (blk_i < blk_t \text{ and } Col(v_i) \notin Avail(v_t)) \text{ or } (blk_t = blk_i \text{ and } cliq_t = cliq_t)\} \\
Col(v_t) &= Random(Avail(v_t) - cc(v_t))
\end{aligned}
$$

Figure 4: ADVERSARY STRATEGY: *Blocked Input*

Proof. As before, the probability of success is a function of the difference between the set of admissible colors and the largest valid bin hue. The expected size of the bin hue at the beginning of the lookahead block is at most $k/4$, by Lemma 1, and the number of vertices added since then cannot exceed the independence number of the subgraph in the block, or $k/16$. The number of admissible colors is $|Avail(v_t) - cc(v_t)|$ and ranges from $k/2 - (k/8 - 1)$ to $k/2$. The probability of success of a random color assignment is thus at least

$$\frac{|Avail(v_t) - cc(v_t)| - k/4 - k/16}{|Avail(v_t) - cc(v_t)|} \geq \frac{k/2 - k/4 - k/8 - k/16}{k/2 - k/8} = 1/6.$$

∎

Given the bound on the success probability of the preceding lemma, the following theorem follows easily. Notice that in our construction k is of order $\log n$, hence the bound holds even if we measure the block size in terms of n.

Theorem 3 *The performance ratio of any randomized algorithm, when the input is presented in blocks of size $\mathcal{O}(\log^2 n)$, is $\Omega(n/\log^2 n)$.*

This lower bound is the best possible, since there is a simple deterministic method to take advantage of blocks of larger size. Off-line coloring each block of size ℓ optimally (possibly using exponential time) requires at most $\chi(G) \cdot \lceil n/\ell \rceil$ colors, where $\chi(G)$ is the minimum number of colors required to color G, which implies an n/ℓ performance ratio. Our lower bound matches this for any $\ell = \Omega(\log^2 n)$, to within a constant factor, by padding each input block (adding $\ell - k^2/128$ dummy vertices). Therefore, in this problem, the effect of blocked input on the performance guarantee is a threshold function.

Similar threshold-like behavior has been shown for the on-line problems of bipartite matching [9], and coloring inductive graphs [7].

4.2 Lookahead and buffering.

We can treat two interesting variations of blocked input similarly. An algorithm is *on-line with lookahead ℓ* if it bases its answer to request t only on the first $t + \ell$ requests. And it is *on-line with a buffer of size ℓ* if at step $t + \ell$ at least t requests have been answered.

It is easy to see that lookahead is more powerful (as an algorithmic feature) than blocked input, and that an input buffer is more powerful than lookahead. Also, we can always simulate lookahead with blocked input of double the block size: follow every block of actual input, with an equivalent amount of dummy input.

We are also able to deal with buffered coloring effectively. Construct blocks twice the size of the buffer; at least one bufferful – a half a blockful – must be answered before the end of the block. By Inequality 2, the probability that *every* vertex in the block succeeds is high, or at least

$$1 - \frac{\binom{k/2+k/16}{k/16}|H|}{\binom{k}{k/2}} \leq 1 - 1.38^{-k}$$

which is asymptotically 1. Thus, no matter which half of the block the algorithm answers, the expected amount of success is not decreased, and the bound for lookahead holds also for buffering to within a factor of two.

This result differs from results on other problems, such as the problem of balancing vectors in a metric space, where lookahead is of no help while buffering yields dramatic improvements [14, p.69].

4.3 Recolorings.

One suggested alternative to lookahead is to allow the algorithm to "recolor" a portion of the nodes, i.e. to independently reassign them bins at some point in the coloring process. This would be useful if only a few nodes were the cause of a poor coloring while the assignments were overall reasonable.

We find that significant amounts of recolorings are of limited use. The adaptive adversary in Thm. 1 makes progress in every round, and thus forces the algorithm to use at least $r/\log^2 r$ colors, when counting only the r vertices never recolored. Thus, unless almost all, $n - o(n)$, of the vertices are recolored, the $\Omega(n/\log^2 n)$ lower bound on the performance ratio still holds. We note, however, that this claim does not apply to our lower bound argument for randomized algorithms as stated.

4.4 Preprocessing: sorting vertices by degree.

One hope for a more effective on-line algorithm would be to pre-massage the input into a pliable ordering. The most natural ordering criterias for graphs are those based on the degrees of the vertices. Such strategies have been extensively evaluated both experimentally and analytically in association with the ubiquitous, inherently on-line First-Fit coloring algorithm (e.g. [13]).

We can easily circumvent any such attempt by padding the input so that all vertices will be of the same degree. With $n/(k-1)$ extra vertices for each of the k colors, each original vertex can then be made adjacent to up to n new vertices without destroying the k-colorability property. The randomized constructions can do with even less padding, due to the highly convergent nature of random selection.

4.5 Transparency and optimality.

In the construction of section 3.2, the adversary makes her coloring assignments on-line, and can without harm reveal those decisions following the algorithm's answers. When given the advantage of this extra information, there is a simple, effective algorithmic strategy: Allocate bins for every non-empty subset of $[k]$. Assign the current vertex to the bin whose hue is a maximal subset of the admissible colors for the vertex. This ensures that no more than $2^k - 1$ bins are used, and in fact $k2^{k-1}$ vertices are required for all of them to be used.

This can be matched precisely, obtaining a minimax value for the game, if, in the deterministic construction, the adversary selects any *minimal* hue (no longer restricting herself to $k/2$-sets) unoccupied by a bin. The details can be found in [6].

The strong lower bound obtained here for the transparent model is in stark contrast with its ineffectiveness for other problems. For the 3-coloring problem, exactly 7 bins are needed, a far cry from the (nevertheless weak) $\Omega(\log^2 n)$ lower bound for the standard model [15]. Similarly, for the k-server problem [12], we can give a simple algorithm with a performance ratio of 3, for any $k \geq 3$, compared to the lower bound and conjectured upper bound of k for the standard model.

5 Conclusions.

We have presented strong lower bounds for on-line graph coloring. Chances are that further lower bounds can be found to bridge the gap that remains in the deterministic case, but not with the methods of this paper.

We have also given matching bounds for several variants of the standard on-line model. Fundamental questions about properties and the applicability of these and other variants are yet to be studied in a general framework. For instance, is the value of lookahead always a threshold function?

Acknowledgements.

The first-mentioned author would like to thank Jaikumar Radhakrishnan and Gábor Tardos for many helpful discussions and insights.

An earlier version of this paper, with different results, appeared in [6].

References

[1] D. Bean. Effective coloration. *J. Symbolic Logic*, 41:469–480, 1976.

[2] F. R. K. Chung, R. L. Graham, and M. E. Saks. A dynamic location problem for graphs. *Combinatorica*, 9(2):111–131, 1989.

[3] G. Gambosi, A. Postiglione, and M. Talamo. New algorithms for on-line bin packing. In *First Italian Conference on Algorithms*, pages 44–59, 1990.

[4] A. Gyárfás and J. Lehel. On-line and first fit colorings of graphs. *J. Graph Theory*, 12(2):217–227, 1988.

[5] M. M. Halldórsson. Parallel and on-line graph coloring. In *Proc. of 3rd Intl. Symp. on Algorithms And Computation*, pages 61–70, Nagoya, Japan, Dec. 1992. Springer-Verlag LNCS.

[6] M. M. Halldórsson and M. Szegedy. Lower bounds for on-line graph coloring. In L. A. McGeoch and D. D. Sleator, editors, *Proc. of DIMACS Workshop on On-Line Algorithms.* AMS-ACM, Feb. 1991. Series in Discrete Mathematics and Theoretical Computer Science, Volume 7.

[7] S. Irani. On-line coloring inductive graphs. In *Proc. 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 470–479, Oct. 1990.

[8] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, Dec. 1974.

[9] M.-Y. Kao and S. R. Tate. Online matching with blocked input. *Inform. Process. Lett.*, 38:113–116, May 1991.

[10] H. A. Kierstead and W. T. Trotter. An extremal problem in recursive combinatorics. In *Proc. 12th Southeastern Conf. on Combinatorics, Graph Theory, and Computing. Congressus Numerantium XXXIII*, pages 143–153, 1981.

[11] L. Lovász, M. Saks, and W. T. Trotter. An online graph coloring algorithm with sublinear performance ratio. *Discrete Math.*, 75:319–325, 1989.

[12] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, Competitive algorithms for on-line problems, *J. Alg.* **11** (1990) 208–230.

[13] D. W. Matula and L. L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30:417–427, 1983.

[14] J. Spencer. *Ten Lectures on the Probabilistic Method*, volume 52. SIAM, 1987.

[15] S. Vishwanathan. Randomized online graph coloring. *J. Algorithms*, 13:657–669, Dec. 1992.

[16] A. C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 18th Ann. IEEE Symp. on Found. of Comp. Sci.*, pages 535–542, 1977.