

A Tool for Optimal Weak Sense of Direction

Paolo Boldi

Sebastiano Vigna *

Abstract

Weak sense of direction is a property of the labelling of (possibly anonymous) networks that allows one to assign coherently local identifiers to other processors on the basis of the route followed by incoming messages. Though there exists a linear algorithm that allows one to assign a weak sense of direction to any given network, the number of colours used in such construction may be as large as the number of processors. It is an open, difficult, yet intriguing problem that of establishing an *optimal* weak sense of direction for a given graph, that is, a weak sense of direction using as few colours as possible. To attack this problem, we have developed an implicit enumeration algorithm that searches for optimal weak sense of direction; the algorithm is implemented in a publicly available tool, `optwsod`. Although `optwsod` can only deal with graphs that are reasonably small, we think that the results obtained (some examples are presented at the end of the paper) are sometimes surprising and may be used as an inspiration to determine optimal weak sense of direction for classes of graphs. This paper reports some details of the design and implementation of `optwsod`, and some experimental data about its behaviour.

1 Introduction and motivations

The topological structure of distributed systems can be described by graphs, with nodes representing agents and arcs representing links. Each node has a local (partial) view of the system, and it associates a different label (colour) to each of its incident links. The solution to many problems in a distributed system can be greatly simplified by using colourings with special properties. In particular, in this paper we study *weak sense of direction* (WSOD for short), a property of global consistency [FMS98]. In a previous paper [BV00], the authors characterized weak sense of direction in a combinatorial manner, and showed that it can be decided efficiently in parallel; as a consequence, there is a sequential polynomial-time algorithm for deciding weak sense of direction.

A challenging problem is that of finding a colouring which gives WSOD to a given graph using as few colours as possible: such a colouring is called *optimal WSOD*¹, and the number of colours used will be called *WSOD-chromatic number* of the graph. Very little is known about finding optimal WSODs, and no attempt was ever made to estimate the WSOD-chromatic number of a given graph, not even for very small graphs.

*Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milano MI, Italia. email: {boldi,vigna}@dsi.unimi.it.

¹The term “optimal” used here is not in line with the common usage in the literature about combinatorial optimization, where it would be replaced with “minimum”; unfortunately, the latter might be easily confused with *minimal* WSOD. We should perhaps find a better term than “minimal”, and then use “minimum” instead of “optimal”.

A closely related problem is that of finding a *minimal WSOD*, that is, a WSOD using as many colours as the maximum outdegree of the graph; since every WSOD must be a deterministic colouring, it is impossible to find a WSOD with fewer colours, thus minimal WSODs are also optimal. In [BV97] the authors proved that an outregular graph admits a minimal WSOD if and only if it is a Cayley graph² (and a minimal WSOD can be obtained by using the natural Cayley colouring of the graph); thus, the optimal WSOD (and *a fortiori* the minimal WSOD) problem is at least as hard as Cayley graph recognition.

The search for the holy grail of optimal sense of direction is a relatively recent story, but—like almost all problems of graph colouring—it immediately appears as a surprisingly difficult and frustrating task. With respect to typical colouring problems, the peculiarity one has to face is the difficulty of determining at first glance (even for very small graphs) whether a given colouring is a WSOD or not. Extremely difficult problems (like MINIMUM GRAPH COLOURING or MAXIMUM CLIQUE) have solutions that are very easy to check, even manually. This easiness provides fundamental intuitions guiding the design of algorithms that are efficient (at least, in practice), and the construction of inapproximability proofs. On the contrary, it is almost impossible to check manually whether a graph with, say, more than four nodes has WSOD, unless the colouring is constructed *ad hoc* so to enjoy particular properties (e.g., the Cayley colouring of a Cayley graph). The algorithm of [BV00] is in this case of no help, since even for a graph with three nodes (see Figure 1 and 2) one has to close transitively 9×9 matrices³.

These are the reasons that finally convinced us of the necessity to develop a program that searches an optimal WSOD for a given graph: the aim of this program is to suggest and inspire techniques for proving optimality results about weak sense of direction. Although the naive approach to the algorithm is fatally superexponential in nature, there are many issues that can be carefully taken into account, giving rise to a program that works within reasonable time on sufficiently small graphs. For instance, the complete analysis of the search tree for the Petersen graph (see Section 8) required about 28s on an SGI workstation (2471 nodes out of a tree with 846749014511809332450147 leaves were explored), while the optimal colouring of the 4-inputs butterfly (Figure 5) required 183s (7524 nodes out of a tree with 128064670049908713818925644 leaves were explored).

2 Definitions

A (*directed*) graph G is given by a set N of n nodes and a set $A \subseteq N \times N$ of arcs. We write $P[x, y] \subseteq A^*$ for the set of paths from the node x to the node y .

An (*arc*) colouring of a graph G is a function $\lambda : A \rightarrow \mathcal{L}$, where \mathcal{L} is a finite set of colours. We say that λ is *deterministic* iff

$$\lambda(\langle x, y \rangle) = \lambda(\langle x, z \rangle) \implies y = z,$$

that is, if the automaton described by the transition graph G with colouring λ is deterministic. We say that λ is *codeterministic* iff

$$\lambda(\langle y, x \rangle) = \lambda(\langle z, x \rangle) \implies y = z.$$

²A special case of this result, for symmetric connected graphs with a symmetric colouring, has been obtained independently in [FRS96].

³It should be noted that the situation is even worse for *sense of direction*, a stronger consistency property: in this case, the only known algorithm [BV00] works in $O(n^{18})$, and it is utterly unfeasible even for verifying mechanically small graphs.

Our (coloured) graphs will be always represented by (coloured) adjacency matrices, that is, by $n \times n$ matrices such that the entry indexed by $x, y \in V$ contains 0 if no arc connects x to y ; otherwise, it contains a positive integer representing the colour of the arc (or 1, if the graph has no colouring).

Given a graph G deterministically coloured by λ , let

$$L(x, y) = \{\lambda^*(\pi) \mid \pi \in P[x, y]\},$$

where $\lambda^* : A^* \rightarrow \mathcal{L}^*$ is defined by

$$\lambda^*(\langle x_1, x_2 \rangle \langle x_2, x_3 \rangle \cdots \langle x_{k-1}, x_k \rangle) = \lambda(\langle x_1, x_2 \rangle) \lambda(\langle x_2, x_3 \rangle) \cdots \lambda(\langle x_{k-1}, x_k \rangle).$$

In other words, $L(x, y)$ is the language recognized by G when x is the initial state and y is the final state. For all $I \subseteq V^2$ let

$$L_I = \bigcup_{(x,y) \in I} L(x, y)$$

(of course, $L(x, y) = L_{\{(x,y)\}}$). Notice that $\varepsilon \in L(x, x) \neq \emptyset$.

A *local naming* for G is a family of injective functions $\beta = \{\beta_x : V \rightarrow \mathcal{S}\}_{x \in V}$, with \mathcal{S} a finite set, called the *name space*. Intuitively, each node x of G gives to each other node y a name $\beta_x(y)$ taken from the name space. Since we require injectivity, we have that necessarily $|\mathcal{S}| \geq n$. We shall also write $\beta : V \times V \rightarrow \mathcal{S}$ for the “unindexed” local naming, that is, $\beta(x, y) = \beta_x(y)$.

Given a coloured graph endowed with a local naming, a function $f : L_{V^2} \rightarrow \mathcal{S}$ is a (consistent) *coding function* iff

$$\forall x, y \in V \quad \forall \pi \in P[x, y] \quad f(\lambda^*(\pi)) = \beta_x(y).$$

A coding function translates the colouring of the path along which two nodes x, y are connected into the name that x gives to y . Note that although the resulting name is *local* (i.e., x and z might choose different elements of the name space for the same node y), the coding function is *global*.

A colouring λ is a *weak sense of direction (WSOD)* for a graph G iff for some local naming there is a coding function⁴. We shall also say that a coloured graph *has* weak sense of direction, or that λ *gives* weak sense of direction to G .

A colouring λ is an *optimal WSOD* for a graph G if it uses as few colours as possible; in this case, the number of colours used is called the *WSOD-chromatic number* of G , and it is denoted by $\zeta(G)$. A graph G admits a *minimal WSOD* iff $\zeta(G)$ is equal to the outdegree of G .

We shall use the notions of equivalence relation and partition interchangeably, and equivalence relations will be represented using boolean matrices. Thus, if R is an equivalence relation, we shall indifferently write $x R y$, $R(x, y) = 1$ or $x, y \in I \in R$ (i.e., x and y belong to the same class I of the partition induced by R). If R and S are equivalence relations, we shall write $R \leq S$ whenever R is finer than S as an equivalence relation, that is, if $R \subseteq S$ (we shall also say that S is coarser than R). We shall use the same notation, with the same meaning, for the associated partitions and boolean matrices.

⁴Elsewhere weak sense of direction has been defined in a slightly different way, by considering only nonempty paths (the so-called *nonhomonymous WSOD*). It is easy to check that the algorithms described here can be immediately adapted to that definition, just by assuming $\varepsilon \notin L(x, x)$ and, consequently, performing a transitive (nonreflexive) closure of the matrix M_λ in Section 3.

3 Verifying WSOD

We recall briefly the results of [BV00]. Given a coloured graph, we firstly define a $n^2 \times n^2$ matrix M_λ (whose rows and columns are indexed by pairs of nodes) so that

$$M_\lambda(\langle x, x' \rangle, \langle y, y' \rangle) = 1 \iff \langle x, y \rangle, \langle x', y' \rangle \in A \wedge \lambda(\langle x, y \rangle) = \lambda(\langle x', y' \rangle).$$

Then we set the matrix Q_λ as follows:

$$Q_\lambda(\langle x, y \rangle, \langle x', y' \rangle) = M_\lambda^*(\langle x, x' \rangle, \langle y, y' \rangle),$$

where the star denotes reflexive-transitive closure. Finally, we set

$$T_\lambda = Q_\lambda^*.$$

Theorem 1 (BV00) Let G be a graph with colouring λ . Then,

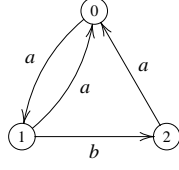
1. $Q_\lambda(\langle x, y \rangle, \langle x', y' \rangle) = 1$ iff $L(x, y) \cap L(x', y') \neq \emptyset$;
2. λ is a WSOD iff for all nodes x, y, z such that $y \neq z$ we have $T_\lambda(\langle x, y \rangle, \langle x, z \rangle) = 0$, that is, if the n matrices $n \times n$ along the diagonal of T_λ are identities (this condition is called *monodromy* of the matrix T_λ).

The previous theorem gives a polynomial-time algorithm for testing whether a given colouring is a WSOD or not⁵. In Figure 1 we show a graph with a colouring that is not a WSOD⁶, and the associated matrices. The same graph, with a slight change in the colouring, is shown in Figure 2, and the new colouring is a WSOD (and it is minimal). Note that even for such small graphs, the quartic increase in the number of matrix entries defies any manual computation. Indeed, all claims of WSOD for graphs in this paper have been verified mechanically. Weak sense of direction can be elusive, and it is easy to get confused by a colouring that is apparently regular enough, but that is not really so. On the other hand, the sequential algorithm for verifying WSOD is very slow. The two crucial time-consuming steps are the reflexive-transitive closures of M_λ and Q_λ . The speed of these steps depends essentially on the fastest boolean matrix multiplication algorithm, which is currently $O(m^{2.376})$ [CW90] for matrices of size $m \times m$. Since our matrices have size $n^2 \times n^2$, where n is the number of nodes in the graph, we obtain that $O(n^{4.752} \log n)$ steps are sufficient (reflexive-transitive closure can be computed using the well-known identity $A^* = (I + A)^n = (I + A)^{2^{\lceil \log n \rceil}}$, and the latter requires $\lceil \log n \rceil$ squarings; transitive closure can be computed as AA^*).

There is however an important *caveat*: both M_λ and Q_λ have special properties that can be used to make the computation of their transitive closure much faster. First of all, M_λ is very *sparse*: since the number of ones in the row $\langle x, y \rangle$ cannot exceed the minimum of the degrees of x and y (λ is deterministic), the number of ones in M_λ cannot be more than $n^2 d$, where d is the maximum outdegree of the graph; thus, the density of M_λ is bounded by $d/n^2 \leq 1/n - 1/n^2$ (the latter bound is very rough indeed, and it is reached only on complete graphs with a minimal colouring). Thus, we can exploit O’Neil and O’Neil’s sparse boolean multiplication algorithm [OO73], which performs as $O(m^2)$ on the average on $m \times m$ matrices. The very simple idea behind the algorithm is that one

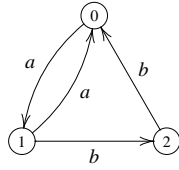
⁵In fact, it can be implemented as an AC¹ algorithm that verifies in logarithmic time WSOD, using n^6 processors.

⁶However, maybe surprisingly, it is a nonhomonymous WSOD.



$$M_\lambda = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad Q_\lambda = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad T_\lambda = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Figure 1: A colouring that is *not* a WSOD.



$$M_\lambda = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad Q_\lambda = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad T_\lambda = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 2: A colouring that is a WSOD.

builds for each row of the first matrix a list of the indices k_1, \dots, k_l of the columns that contain a one on that row. Then multiplication by a column can be performed just checking the entries of row indices k_1, \dots, k_l in the column (of course, the search stops as soon as a one is found).

The second important observation is that Q_λ is symmetric, and in this case there is a very nice (although definitely opaque) folklore algorithm that is very fast in practice⁷:

```

for  $i = 1$  to  $m$  do
  begin
     $j \leftarrow \min\{s > i \mid B[i, s] = 1\}$ ;
    if  $j \neq \infty$  then  $B[j] \leftarrow B[j] \vee B[i]$ ;
  else
    begin
       $B[i, i] \leftarrow 1$ ;
    
```

⁷The first appearance of this algorithm in the literature is due to Fischer and Paterson [FP80]; there, the authors claim to have been unable to trace a published source for the algorithm, although it was part of the folklore at least since 1973.

```

    for  $k = 1$  to  $i - 1$  do
      if  $B[i, k] = 1$  then  $B[k] \leftarrow B[i]$ 
    end
  end
end

```

Here m is the size of the matrix, and j is ∞ exactly when the part of the i th row after the diagonal is entirely zero; $A[i]$ is the i th row of A . The inner loop on k is executed rarely if the matrix is sufficiently dense, so the algorithm runs as $O(m^2)$ on the average.

Using the knowledge above it is easy to write down a verifier that can easily handle graphs with up to few hundreds of nodes. The high-polynomial nature of the algorithm then becomes a barrier. Indeed, polynomiality is usually associated with a low degree, but at present time there are no known verifications algorithms for sense of direction that are better than $O(n^4)$, even on the average.

4 Searching for optimality

Clearly, the property of being a WSOD only depends on the *fibres* of the colouring function (i.e., on the equivalence relation on A defined by letting two arcs be equivalent iff they have the same colour). Hence, a naive attempt to search for an optimal WSOD consists in enumerating all partitions of the set A of arcs; letting $|A| = a = O(n^2)$, the number of configurations to be explored would be equal to the a -th Bell number [GKP94], that is,

$$\frac{1}{e} \sum_{k \geq 0} \frac{k^a}{k!} \sim f(a)^a \frac{e^{f(a)-a-\frac{1}{2}}}{\sqrt{\ln a}},$$

where $f(x) \ln f(x) = x - 1/2$ (the sum is superexponential). The last simple consideration explains why the brute-force solution of enumerating exhaustively all partitions and testing for WSOD is unrealistic (with 20 arcs, we should explore 51724158235372 configurations).

From a more general point of view, the problem of determining an optimal WSOD for a graph is a special instance of the following problem:

MINIMUM SET PARTITION

Instance: A set A , together with a predicate $\mathcal{P} \subseteq \text{Part}(A)$ (where $\text{Part}(A)$ denotes the set of partitions of A).

Output: A partition $P \in \mathcal{P}$ such that $|P|$ is minimum among the partitions in \mathcal{P} .

In general, this problem is NP-hard, since, for instance, PARTITION INTO CLIQUES [GJ79] may be seen as a special case of MINIMUM SET PARTITION (just let A be the set of edges of the graph, and \mathcal{P} be the set of partitions into cliques). Nevertheless, we shall introduce some heuristics that will reduce the burden of enumeration by exploiting implicit enumeration techniques, so that optimal WSODs may be found within reasonable time at least for small graphs.

5 An enumeration scheme for partition problems

When trying to consider MINIMUM SET PARTITION in its full generality, one has to deal with “partially specified” equivalence relations. Indeed, one can try to build finer and finer partial specifications

of the partition, ruling out the ones that are nonpromising or nonconsistent. For this reason, we introduce the following notion: an *equivalence-inequivalence constraint* on a set A is a pair $\langle E, I \rangle$ of relations on A such that E is an equivalence relation, I is a symmetric relation, and $E \cap I = \emptyset$. Intuitively, E specifies which elements must be equal, whereas I determines which elements must be inequivalent (clearly, the two requirements are mutually orthogonal). Note that there are usually pairs of elements whose (in)equivalence status is left unspecified.

If W is an equivalence relation satisfying $E \leq W$ and $I \cap W = \emptyset$, we say that W is a *witness* of $\langle E, I \rangle$. Witnesses are simply the equivalence relations that are coarser than E , but do not violate the inequality constraints of I . We define

$$\langle E, I \rangle^? = A^2 \setminus \left(E \cup \bigcap_W W^c \right),$$

where W ranges over the witnesses of $\langle E, I \rangle$. This set contains *all* pairs that are left free by the constraint. Clearly, witnesses are “sandwiched” between E and $E \cup \langle E, I \rangle^?$, so the following proposition holds:

Proposition 1 An equivalence-inequivalence constraint $\langle E, I \rangle$ on A has exactly one witness (namely, E) iff $\langle E, I \rangle^?$ is empty.

A constraint $\langle E, I \rangle$ satisfying any of the two equivalent conditions above is called *complete*. For incomplete constraints we have the following:

Proposition 2 Let $\langle E, I \rangle$ be an equivalence-inequivalence constraint on A , and $\langle a, b \rangle \in \langle E, I \rangle^?$. Then, the following pairs

$$\begin{aligned} \langle E, I \rangle + \langle a, b \rangle &= \langle (E \cup \{\langle a, b \rangle, \langle b, a \rangle\})^*, I \rangle \\ \langle E, I \rangle - \langle a, b \rangle &= \langle E, I \cup \{\langle a, b \rangle, \langle b, a \rangle\} \rangle \end{aligned}$$

are equivalence-inequivalence constraints; in particular, the set of witnesses of $\langle E, I \rangle + \langle a, b \rangle$ and the one of $\langle E, I \rangle - \langle a, b \rangle$ are distinct proper subsets of the witnesses of $\langle E, I \rangle$.

Proof. There must exist witnesses W and W' of $\langle E, I \rangle$ such that $a W b$ and not $a W' b$. Hence W is also a witness of $\langle E, I \rangle + \langle a, b \rangle$ (but not of $\langle E, I \rangle - \langle a, b \rangle$) and W' is a witness of $\langle E, I \rangle - \langle a, b \rangle$ (but not of $\langle E, I \rangle + \langle a, b \rangle$). ■

Intuitively, the proposition above says that we can enrich an incomplete equivalence-inequivalence constraint by adding a free pair (i.e., a pair belonging to $\langle E, I \rangle^?$) either to the positive part or to the negative part of the constraint. (It is immediate to check that adding pairs not belonging to $\langle E, I \rangle^?$ either is of no use, or violates the definition of constraint.)

Finally, a *heuristic* for a set A is a function that, given an incomplete equivalence-inequivalence constraint $\langle E, I \rangle$, returns an element of $\langle E, I \rangle^?$. Given a set A and a heuristic h for A , we define a binary tree $T(A, h)$ (whose nodes are equivalence-inequivalence constraints on A) as follows:

- the root of T is the pair $\langle \text{identity}, \emptyset \rangle$;
- a node $\langle E, I \rangle$ is a leaf iff it is complete;
- if $\langle E, I \rangle$ is not a leaf, the children of $\langle E, I \rangle$ are $\langle E, I \rangle + h(E, I)$ and $\langle E, I \rangle - h(E, I)$.

This definition is correct by virtue of Proposition 2. Since our interest is the enumeration of all partitions of A , we must show that the tree $T(A, h)$ suits our needs:

Theorem 2 Every equivalence relation on A appears exactly once as the first member of a leaf of $T(A, h)$.

Proof. Let E be an equivalence relation on A , and consider the maximal branch $\langle E_0, I_0 \rangle, \langle E_1, I_1 \rangle, \dots, \langle E_k, I_k \rangle$ of $T(A, h)$ defined as follows: $\langle E_0, I_0 \rangle = \langle \text{identity}, \emptyset \rangle$ and $\langle E_{i+1}, I_{i+1} \rangle$ is $\langle E_i, I_i \rangle + h(E_i, I_i)$ or $\langle E_i, I_i \rangle - h(E_i, I_i)$ according to whether $h(E_i, I_i) \in E$ or not. Clearly E is the (unique) witness of $\langle E_k, I_k \rangle$. Conversely, let $\langle E'_0, I'_0 \rangle, \langle E'_1, I'_1 \rangle, \dots, \langle E'_l, I'_l \rangle$ be another maximal branch of the tree, and let $\langle E'_{i+1}, I'_{i+1} \rangle$ be the first difference with respect to the former branch; assume, without loss of generality, that $\langle E_{i+1}, I_{i+1} \rangle = \langle E_i, I_i \rangle + h(E_i, I_i)$ and $\langle E'_{i+1}, I'_{i+1} \rangle = \langle E_i, I_i \rangle - h(E_i, I_i)$ (where $h(E_i, I_i) \in E$). Every witness of $\langle E'_{i+1}, I'_{i+1} \rangle$ (hence of $\langle E'_l, I'_l \rangle$) does not contain $h(E_i, I_i)$, so E cannot be the partition induced by $\langle E'_l, I'_l \rangle$. ■

6 Implicit enumeration techniques for partition problems

Of course, the tree $T(A, h)$ of the previous section is preposterously huge, even for a very small set A (the leaves are as many as the partitions of A). However, we can exploit *implicit enumeration* [Geo67] to *fathom* parts of the tree, and give them for granted even if they have not been explicitly enumerated. In particular, in this section we study fathoming due to the detection of unpromising solutions, and to the detection of unfeasible solutions when the predicate \mathcal{P} is downward closed (with respect to the “finer than” relation on partitions).

6.1 Fathoming based on chromatic numbers

A subtree of $T(A, h)$ can be *fathomed* when we can somehow detect that we can skip safely all partitions in the subtree (i.e., the partitions are enumerated implicitly, but not actually). A possible fathoming strategy is based on optimality: suppose that, during our visit of the search tree we have already found a solution (partition) with c classes. If we can guarantee that all the solutions on a certain subtree contain c classes at least, we can safely prune the whole subtree and backtrack. Thus, we need an answer to the following question: how can we lower bound the number of classes of the equivalence relations on the leaves of a given subtree?

Consider an equivalence-inequivalence constraint $\langle E, I \rangle$ on A , and let $H(E, I)$ be the undirected graph having as vertices the equivalence classes of E , with C adjacent to C' iff there are elements $a \in C, a' \in C'$ such that $a I a'$.

Theorem 3 Let $\langle E, I \rangle$ be a node of the tree $T(A, h)$. Then, the witnesses of the nodes of the subtree rooted at $\langle E, I \rangle$ all contain at least $\chi(H(E, I))$ classes (where χ gives the chromatic number of an undirected graph).

Proof. Let W be a witness of some node in the subtree; W is *a fortiori* a witness of $\langle E, I \rangle$ as well. Suppose, by contradiction, that W has $k < \chi(H(E, I))$ classes C_1, C_2, \dots, C_k . Since E is the minimum witness of $\langle E, I \rangle$, W is certainly coarser than E ; hence, each equivalence class C of E is included in a (unique) class C_i of W . Now, colour the vertices of $H(E, I)$ on the set $\{1, 2, \dots, k\}$ by

using for the vertex (equivalence class) C the colour i iff $C \subseteq C_i$. By hypothesis, there must be two classes C', C'' that are adjacent (i.e., $a' I a''$ for some $a' \in C'$ and $a'' \in C''$) such that $C', C'' \subseteq C_i$. But then $a' W a''$ and thus W is not a witness of $\langle E, I \rangle$ (for $a' I a''$ means that no witness of $\langle E, I \rangle$ relates a' and a''), a contradiction. ■

As a consequence, if the current optimal solution has c classes and the chromatic number of the graph associated with a node is at least c , the whole subtree rooted at that node can be fathomed, because no solution in the subtree can be better than the current optimum.

6.2 Lower bounds for χ

The previous section introduced a way of fathoming subtrees by computing the chromatic number of the reduced graph of the current node. Although this problem is NP-hard (even inapproximable [FK98, BGS98]), we can try to exploit some techniques to compute lower bounds (clearly, this is sufficient for our purposes). Of course, there is a trade-off between the accuracy (i.e., strictness) of the lower bound and the time needed to compute it; it is thus crucial to choose lower bounds that can be computed quickly enough (because the computation of such lower bounds has to be carried out at *all* nodes of the search tree) but that are still good enough (because stricter lower bounds can successfully fathom more subtrees, thus reducing the explicit search space).

Spectral lower bounds. There are some well-known techniques for computing lower bounds for $\chi(H)$ based on the spectrum of the graph H (i.e., the spectrum of the adjacency matrix of H ; a good survey of these results is contained in [CDS78]). Some of these bounds require the computation of the whole spectrum, whereas others just need the knowledge of some eigenvalues. For example, Hoffman [Hof70] proves that

$$\chi(H) \geq 1 - \frac{\lambda_1}{\lambda_n},$$

where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ are the eigenvalues of the graph H (with n nodes). The bound above can be easily refined (essentially with the same proof [CDS78]) to the one we use:

$$\chi(H) \geq 1 + \min \left\{ K \mid \lambda_1 + \sum_{i=1}^K \lambda_{n-i+1} \leq 0 \right\}.$$

Greedy lower bounds. Edwards and Elphick [EE83] give a lower bound for the clique number (hence, also for the chromatic number) of a graph, using a greedy technique. More precisely, they build a maximal clique in the following manner: they first choose one of the vertices having maximum degree; then, if the clique is nonmaximal, they extend the clique by adding a vertex of degree as large as possible. The size of the resulting clique can of course be used as a lower bound for the chromatic number of the graph.

Lovász's ϑ function. A very sophisticated technique for computing a lower bound for $\chi(H)$ is based on Lovász's ϑ function [GLS88]. The ϑ function satisfies the so-called Sandwich Theorem, that is,

$$\omega(H) \leq \vartheta(\overline{H}) \leq \chi(H),$$

where $\omega(H)$ denotes the clique number of H , and \bar{H} is the complement of H . Knuth [Knu94] contains a very deep account about the possible ways to define and compute⁸ ϑ , and about the various properties of the function. Of course, $\vartheta(\bar{H})$ gives always a better lower bound than Edwards and Elphick’s one.

A comparison. From a computational point of view, the greedy lower bound presented above is the easiest to compute, whereas the spectral bounds require a number of operations that use double-precision arithmetic; an even larger overhead is required for the computation of the Lovász’s ϑ function. In the latter two cases, moreover, attention should be paid to avoid numeric instability (although the computation of the eigenvalues of a graph is an easy matter, since the adjacency matrix is real symmetric). Surprisingly enough, during tests on thousands on graphs actually produced by the algorithm the greedy lower bound was very good, and its difference with ϑ was never more than 2 (and, usually, no more than 1). Nonetheless, ϑ was sometimes able to cut a major part of the search tree, and this fact convinced us to make `optwsod` consider the lower bound of ϑ only on demand, as an option passed to the program.

Upper bounds vs. lower bounds. Since the computation of lower bounds (in particular of ϑ) is usually time-consuming, it would be wise to compute firstly some very simple upper bounds for the chromatic number, so to avoid the lower bound computation at least for those graphs with an upper bound that is smaller than the current optimum.

A trivial upper bound for the chromatic number of a graph is obtained by the maximum degree plus one⁹. Another upper bound, which is usually better than the previous one (although not as simple to obtain) is given by Liu [Liu89], who proves that

$$\chi(\bar{G}) \leq n - \frac{m^2}{\sum_{i=1}^n d_i^2 - m},$$

where d_i is the degree of vertex i (in fact, the bound can be made stronger if the independence number of G is known, but this is of course not practical in our case).

As in the lower bound case, however, the best upper bound can often be obtained by using a greedy algorithm, Brélez’s `DSATUR` [Bré79], which, starting from a vertex of maximum degree, colours the vertex adjacent to the largest number of vertices with different colours, using the minimum colour available. `DSATUR` is very fast, but gives a much better upper bound than the previous ones. This is particularly useful if the computation of Lovász’s ϑ function is enabled, as a large number of slow floating-point computations can be skipped.

6.3 Fathoming based on feasibility

When the predicate \mathcal{P} is downward closed (i.e., when $P \in \mathcal{P}$ implies $Q \in \mathcal{P}$ for all $Q \leq P$) we can further exploit the structure of $T(A, h)$: indeed, all witnesses W of the descendants of a node $\langle E, I \rangle$ satisfy necessarily $W \geq E$. Thus, if E is not feasible (i.e., $E \notin \mathcal{P}$) the whole subtree under $\langle E, I \rangle$ can be fathomed, because it contains no feasible solutions. Fortunately, this is our case, as shown by the following

⁸The most efficient way to compute ϑ is by means of semidefinite programming—see again [Knu94].

⁹More precisely, by Brooks’ Theorem, if d is the maximum degree of a connected graph H , then $d \leq \chi(H)$, unless H is a complete graph or an odd cycle.

Proposition 3 Let G be a graph, and P, Q be partitions of the arcs of G . If $P \leq Q$ and P does not give weak sense of direction, then the same is true of Q .

Proof. Simply note that all the matrix operations used in Theorem 1 are *monotonic* with respect to pointwise ordering (i.e., $B \leq C$ iff $B(i, j) \leq C(i, j)$ for all i and j), which specializes, in the case of matrices representing equivalence relations, to the “finer than” ordering. ■

7 An implementation

In this section we briefly discuss the algorithmic core of the WSOD verification/search tool `optwsod`. A number of subtleties are involved in the algorithm, and some tricks (beside a careful exploitation of the properties described in the previous sections) can be used to reduce (sometimes in a dramatic way) the overall time needed for the search; they are described in the source code. Here we give instead a very high-level (but actually very faithful¹⁰) description of our implementation, and a number of sparse remarks:

```

const  n : N; /* Number of nodes */
        m : N; /* Number of arcs */
        d : N; /* Maximum outdegree */
        s, t : array 1 .. m of 1 .. n; /* Source/target of each arc */

procedure WSOD( $E, I$  : relation of 1 .. m,  $o$  : N) /*  $o$  is the current optimum solution */
var  F, J : relation of 1 .. m;
begin
  if  $\langle E, I \rangle$  is complete then return;
  if Edwards and Elphick’s lower bound for  $\chi(H(E, I)) \geq o$  return;
  if Hoffman’s lower bound for  $\chi(H(E, I)) \geq o$  return;
  if Lovász’s lower bound for  $\chi(H(E, I)) \geq o$  return;
   $\langle F, J \rangle \leftarrow \langle E, I \rangle + h(E, I)$ ;
  if  $F$  gives WSOD then
    begin
      if number of classes of  $F < o$  then
        begin
          output( $F$ );
           $o \leftarrow$  number of classes of  $F$  ;
          if  $o = d$  then stop
        end
      end
      WSOD( $F, J, o$ );
    end
   $\langle F, J \rangle \leftarrow \langle E, I \rangle - h(E, I)$ ;
  WSOD( $F, J, o$ )
end

```

¹⁰Essentially, the only part of the algorithm not represented here is the computation of the upper bounds on $\chi(H(E, I))$ and the subsequent skip of the lower-bound computation.

```

begin /* Main */
  WSOD(identity, {⟨a, b⟩ | s(a) = s(b)}, m)
end

```

- The base constraint set up by our algorithm is given by the trivial inequivalence relation imposed by determinism of λ (see Section 2). However, if the graph under examination is strongly connected λ must also be codeterministic¹¹. The base constraint and the value of d can be modified accordingly.
- We used the standard LAPACK (Linear Algebra Package) library to compute the eigenvalues that are necessary for Hoffman’s bound. LAPACK returns suitable error codes if the computation fails, but this is never the case due to the good linear properties of adjacency matrices of undirected graphs.
- We used Brian Borchers’s CSDP (C Semidefinite Programming) package [Bor99] to compute Lovász’s ϑ function (CSDP is built on LAPACK). The computation, however, must be turned on with an option.
- The search for an optimum solution stops if a solution is found that cannot be improved. However, very little is known about lower bounds for the number of colours that can give WSOD to a graph. Indeed, besides the obvious bounds imposed by determinism (and possibly codeterminism), we just know that a graph that is outregular, but not Cayley, cannot be given minimal sense of direction [BV97], and this condition is clearly of almost no practical use. The precomputation of any better lower bound for WSOD would reduce significantly the search space.
- Currently, the function h that we use chooses a pair of arcs so to maximize the product of the degrees of the corresponding vertices of $H(E, I)$; this heuristic helps Edwards and Elphick’s lower bound to grow quickly.
- There may be differences in the actual enumeration on different machines: `optwsod` uses `libc’s qsort()` function in several occasions, and the behaviour of `qsort()` is undefined on equal keys.

In Figure 3 we show the tree explored by an execution of the algorithm on the graph of Figure 2. At each node we display a synthetic representation of the corresponding equivalence-inequivalence constraint, and on the right the associated graph (the reader can easily match the nodes of the graph with the equivalence classes). The arc $\langle x, y \rangle$ is simply written as xy . A real execution would have stopped as soon as the solution with two colours had been found, but for demonstration purposes we showed also the remaining part of the tree.

8 Experimenting with Weak Sense of Direction

In this section we give two examples of optimality results obtained by `optwsod` (of course, we cannot include all our experimental data). As a first example, consider Petersen’s graph. Since it is not Cayley, we need at least 4 colours to give it sense of direction, and indeed `optwsod` has found the

¹¹This is not true, however, if nonhomonymous WSOD is considered.

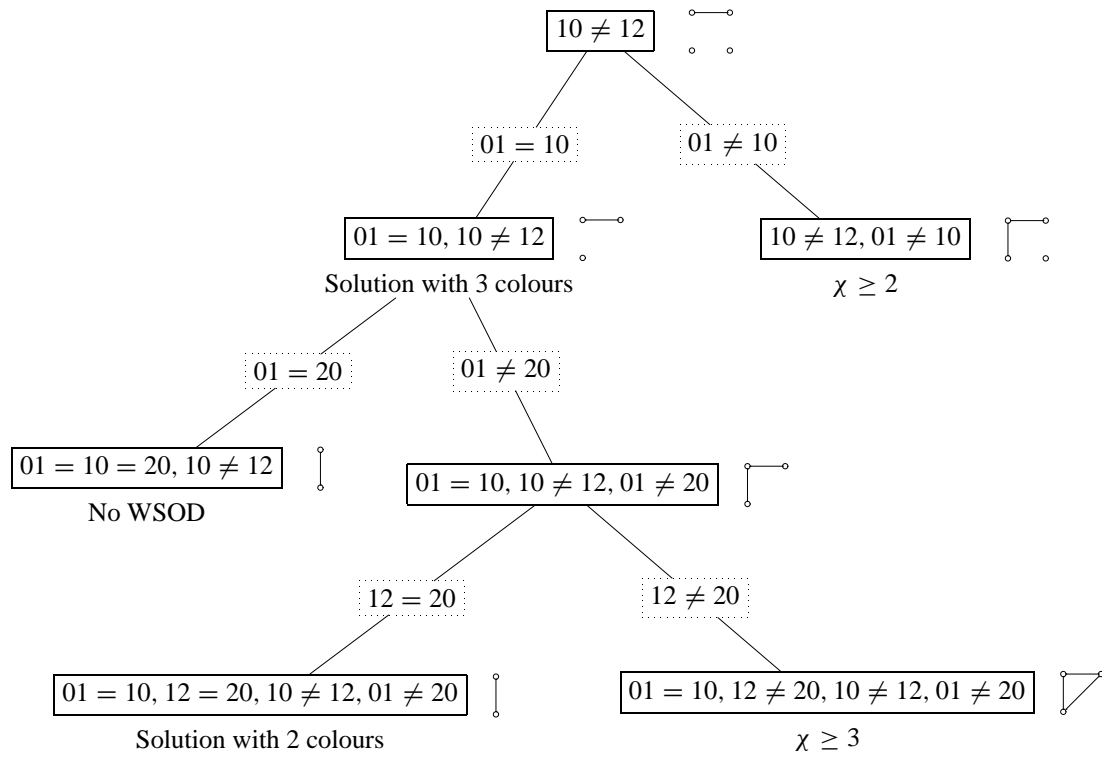


Figure 3: An execution of the algorithm on the graph of Figure 2.

colouring shown¹² in Figure 4. Another interesting example is given by a butterfly with four inputs.

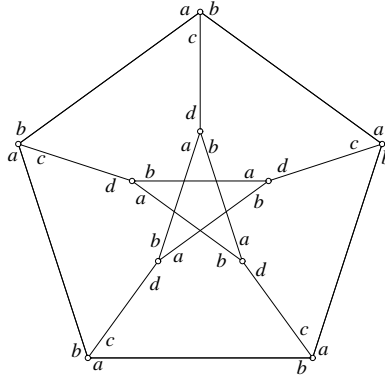


Figure 4: An optimal (but not minimal) WSOD for Petersen’s graph.

Using the techniques described in [BV], it is easy to show that such a butterfly can be given WSOD using six colours, and the corresponding (very regular) colouring is shown on the left of Figure 5. However, `optwsod` gives the (minimal, optimal and symmetric, but not so regular) solution on the right, which is by all means nontrivial and defies manual verification. Finally, in Figure 6 we show a

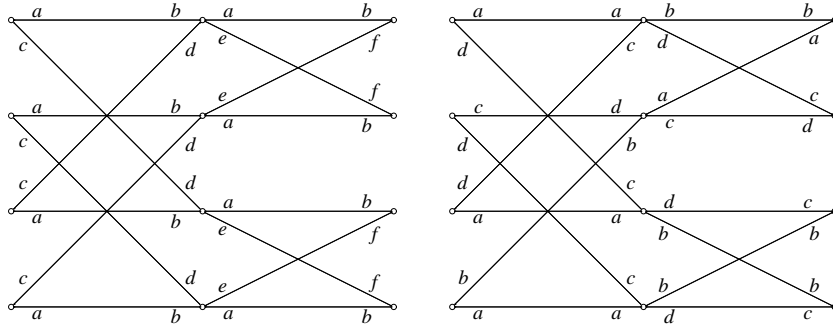


Figure 5: Two WSODs for a butterfly (the second one is minimal).

simple shuffle-exchange network with eight nodes: in this case, the optimal WSOD is not minimal, as four colours are necessary (if the usual loops at the extremal nodes were added, the gap would be larger by one). Also this result may come as a surprise; indeed, a measure of our ignorance about WSOD optimality is given by the fact that this is the *first* nontrivial lower bound ever obtained for a nonregular graph, and that the first example of a loopless graph whose optimal WSOD exceeds by more than one its maximum indegree or outdegree was obtained only recently using `optwsod`.

¹²Following a widely accepted convention, we represent pairs of parallel symmetric arcs as undirected edges with two colours; the colour $\lambda((x, y))$ is the one that appears close to y , whereas the colour $\lambda((y, x))$ appears close to x .

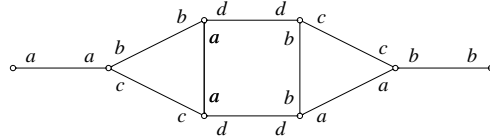


Figure 6: An optimal (but not minimal) WSOD for a shuffle-exchange network.

9 Conclusions and acknowledgements

Our (very preliminary) experimental results suggest that `optwsod` might be fruitfully used to support the intuition about weak sense of direction, and to help the research concerning optimality. Even though the algorithm already works quickly on sufficiently small graphs, there is still room for improvements: in particular, we think that a suitable choice of the heuristic used can make the computation faster.

We would like to acknowledge Alain Hertz (Ecole Polytechnique Fédérale de Lausanne), Brian Borchers (New Mexico Tech) and Giuliano Grossi (Università di Milano) for their cooperation.

References

- [BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.
- [Bor99] Brian Borchers. CSDP, a C library for semidefinite programming. *Optim. Methods Softw.*, 11(1):613–623, 1999.
- [Bré79] Daniel Brélez. New methods to color the vertices of a graph. *Comm. ACM*, 22(4):251–256, 1979.
- [BV] Paolo Boldi and Sebastiano Vigna. Coverings that preserve sense of direction. To appear in *Inform. Process. Lett.*
- [BV97] Paolo Boldi and Sebastiano Vigna. Minimal sense of direction and decision problems for Cayley graphs. *Inform. Process. Lett.*, 64(6):299–303, 1997.
- [BV00] Paolo Boldi and Sebastiano Vigna. Complexity of deciding sense of direction. *SIAM J. Comput.*, 29(3):779–789, 2000.
- [CDS78] Dragoš M. Cvetković, Michael Doob, and Horst Sachs. *Spectra of Graphs*. Academic Press, 1978.
- [CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9(3):251–280, 1990.
- [EE83] C.S. Edwards and C.H. Elphick. Lower bounds for the clique and the chromatic number of a graph. *Discrete Appl. Math.*, 5(1):51–64, 1983.
- [FK98] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *J. Comput. System Sci.*, 57(2):187–199, 1998.
- [FMS98] Paola Flocchini, Bernard Mans, and Nicola Santoro. Sense of direction: Definitions, properties, and classes. *Networks*, 32(3):165–180, 1998.
- [FP80] Michael J. Fischer and Michael S. Paterson. The fast skew-closure algorithm. *Enseign. Math. (2)*, 26(3–4):345–360, 1980.

- [FRS96] Paola Flocchini, Alessandro Roncato, and Nicola Santoro. Complete symmetries and minimal sense of direction in labeled graphs. In *Proc. 27th SE Conference on Combinatorics, Graph Theory and Computing*, volume 121 of *Congressus Numerantium*, pages 3–18, 1996.
- [Geo67] Arthur M. Geoffrion. Integer programming by implicit enumeration and Balas’ method. *SIAM Rev.*, 7:178–190, 1967.
- [GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
- [GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison–Wesley, second edition, 1994.
- [GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer–Verlag, 1988.
- [Hof70] Alan J. Hoffman. On eigenvalues and colorings of graphs. In B. Harris, editor, *Graph Theory and its Applications (Proc. Advanced Sem., Math. Research Center, Univ. of Wisconsin, Madison, Wis., 1969)*, pages 79–91. Academic Press, 1970.
- [Knu94] Donald E. Knuth. The sandwich theorem. *Electron. J. Combin.*, 1, 1994.
- [Liu89] Ru Ying Liu. An upper bound on the chromatic number of a graph. *J. Xinjiang Univ. Natur. Sci.*, 6(2):24–27, 1989.
- [OO73] Patrick O’Neil and Elizabeth J. O’Neil. A fast expected time algorithm for Boolean matrix multiplication and transitive closure. *Information and Control*, 22(2):132–138, 1973.