Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 306 (2003) 1–18

Theoretical
Computer Science

www.elsevier.com/locate/tcs

The Go polynomials of a graph

G.E. Farr[1]

School of Computer Science and Software Engineering, Monash University (Clayton Campus), Clayton, Victoria 3168, Australia

Received 23 May 2002; received in revised form 14 October 2002; accepted 21 October 2002
Communicated by A.S. Fraenkel

Abstract

This paper introduces graph polynomials based on a concept from the game of Go. Suppose that, for each vertex of a graph, we either leave it uncoloured or choose a colour uniformly at random from a set of available colours, with the choices for the vertices being independent and identically distributed. We ask for the probability that the resulting partial assignment of colours has the following property: for every colour class, each component of the subgraph it induces has a vertex that is adjacent to an uncoloured vertex. In Go terms, we are requiring that every group is uncaptured. This definition leads to *Go polynomials* for a graph. Although these polynomials are based on properties that are less "local" in nature than those used to define more traditional graph polynomials such as the chromatic polynomial, we show that they satisfy recursive relations based on local modifications similar in spirit to the deletion–contraction relation for the chromatic polynomial. We then show that they are #P-hard to compute in general, using a result on linear forms in logarithms from transcendental number theory. We also briefly record some correlation inequalities.

1. Introduction

This paper concerns graph polynomials based on a concept from the game of Go.

In the game (known as *Go* or *Igo* in Japan, *Weiqi* in China and *Baduk* in Korea), two players (Black and White) take turns to place stones of their respective colours on the vertices of a 19×19 square grid graph (with 19^2 vertices altogether). Fig. 1 shows a portion of this grid, together with a few stones of each colour. Each player

[1] Part of the work of this paper was done while the author was visiting the Department of Computer Science, Royal Holloway, University of London.

E-mail address: gfarr@csse.monash.edu.au (G.E. Farr).

0304-3975/03/$ - see front matter © 2002 Elsevier B.V. All rights reserved.
doi:10.1016/S0304-3975(02)00831-9

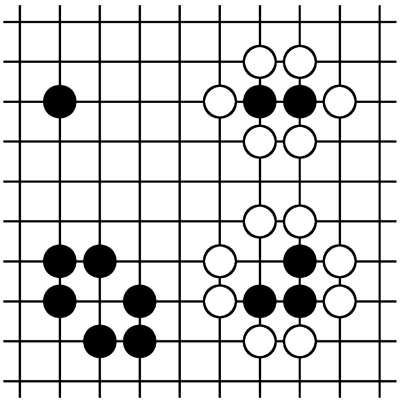

Fig. 1. Five black groups on part of a Go board. All are free except the top right one.

aims to enclose (in a particular sense) as much territory on the board as possible, and to capture the opponent's stones by surrounding them. We will not be concerned with most details of this game, and refer the interested reader to books on the subject such as [15]. The ideas that are important to us are the notion of a *group* of stones and recognising when a group is *captured.*

A player's stones on the board induce a subgraph of the square grid graph in the natural way, and a *group* is a connected component of this subgraph. Thus, a group consists of a maximal set of stones (or their corresponding vertices) that are all of one colour and form a connected subgraph of the grid. A group is *captured* if every vertex not in it but adjacent to it is occupied by a stone of a different colour. For example, see the black group in the top right area of Fig. 1. A captured group is immediately removed from the board. There are rules about how capture is done and in what circumstances, but these do not concern us here. Suffice to say that, in any legal Go position (and apart from the brief periods of time during which captures are physically carried out and captured stones are removed from the board), every group has a stone that is adjacent to an unoccupied vertex.

This last concept is our key definition. We say that such a group is *free* or *uncaptured*. The definition extends easily to general graphs and more than two players, and is a perfectly natural graph-theoretic concept.

Suppose each vertex of a graph is given a stone of a randomly chosen colour (from some fixed palette), or possibly left vacant, and that this is done independently and with the same probability distribution for all the vertices of the graph. We can ask for the probability that the resulting configuration is *free* in the sense that it has no captured groups. We take particular interest in (i) the case of two players, with each colour equally likely, and the probability of receiving a colour is varied, and (ii) λ players, and hence colours, with all colours (and being vacant or "uncoloured") equally likely, in which case we are essentially just counting the number of free configurations.

These probabilities can be written as polynomials in the variables of interest, and are the subject of this paper. It is natural to ask whether these *Go polynomials* have

any interesting theory, and whether they behave much like other known families of graph polynomials.

There is a considerable literature on graph polynomials. Many of the most interesting ones (e.g., chromatic and flow polynomials, percolation probability, reliability polynomials, Ising and Potts model partition functions, Jones polynomial) arise as partial evaluations of the Tutte polynomial (or Whitney rank generating function); see [8,30,31]. Analogues of the Tutte polynomial or its specialisations have been introduced in many contexts (see, e.g., [11,12,17,23–28,32,33]). Some of these polynomials have links with knot polynomials [16,23]. Other graph polynomials of interest include polynomials for counting cliques or stable sets (see [14] for the earliest paper I have found on this kind of polynomial, or [10, Section 1] for other references), matchings [19, Section 8.5], and dominating sets [1].

One common theme in the study of these diverse polynomials is the development of simple relations (usually linear in some sense) that express the polynomial in question, for some given graph G, in terms of a small number (typically two or three) polynomials for graphs derived from G by simple local modifications. The classic example is the deletion-contraction rule for the chromatic polynomial: writing $P(G;k)$ for the number of k-colourings of G, we have

$$P(G;k) = P(G-e;k) - P(G/e;k),$$

where $G-e$ is obtained by deleting e from G, and G/e is obtained by identifying the endpoints of e and deleting e. Rules of this sort allow the polynomial to be computed, albeit in exponential time. Although not very efficient, it should be borne in mind that most of these polynomials are #P-hard to compute exactly [16,29], and so probably do not have polynomial time algorithms. Such rules are also very important in the study of the polynomial's properties, and in fact can sometimes be used to characterise classes of polynomials (see, e.g., [31]).

Rules of this kind are often very elementary. They generally depend on the property in question (e.g., validity of a putative k-colouring) being testable by a conjunction of independent local tests on the graph (e.g., for colouring, testing whether each edge has different colours on its endpoints).

The situation is not so straightforward for Go polynomials. Whether or not a configuration of stones is free depends on knowing about groups of stones, and such groups may be quite large. The property does not seem so "local" in character, and it does not seem obvious whether a rule of the kind discussed above, using a constant number of locally modified graphs, exists.

Our first aim in this paper is to show (in Section 3) that such rules can, in fact, be derived for Go polynomials. We find it necessary to move beyond the class of graphs by adding a few kinds of labels to vertices and edges. Also, we need more than one rule if we want to calculate the Go polynomial recursively in terms of trivial base cases. We divide the calculation of the Go polynomial into several stages. Each stage has its own rule, and repeated application of that rule allows the elimination of a certain kind of edge or vertex from the graph, often at the cost of introducing vertices or edges of a new kind. When application of the rule for that stage is finished, a new rule is applied in the next stage, and this is used to eliminate some of the new kinds of vertices or

edges introduced in earlier stages. We eventually find ourselves asking for chromatic polynomials of graphs, so the rule for that stage is just the usual deletion-contraction one, and of course that stage then ends with trivial base cases (null graphs). Taken successively, these stages show how to calculate Go polynomials by simple recursive rules, based on purely local modifications of graphs, with trivial base cases. Of course, this procedure for calculating the Go polynomial takes exponential time, though in general it is faster than the naïve method based on exhaustively checking all partial λ-assignments.

In Section 4 we consider further the complexity of computing the Go polynomial, and show that it is #P-hard if $\lambda \geqslant 2$. The proof uses a result on linear forms in logarithms from transcendental number theory.

There is considerable literature on mathematical and computational aspects of Go, though this present work is not directly related to it. That literature can be divided into about three main streams. Firstly, Go is studied using the theory of games developed by Berlekamp, Conway and others (see, e.g., [4–6]). Secondly, it has been studied as a problem in computational complexity: determining which player can win from a given position was shown to be PSPACE-hard by Lichtenstein and Sipser [18], and for a recent refinement see [9]. Thirdly, it is studied in the field of artificial intelligence, and programs are written to play it: see [7,21,22].

Although inspired by Go, we do not suggest that the work of this paper is applicable to playing the game, but hope to show that it is of some interest in its own right.

2. Partial λ-assignments and Go polynomials

This section gives formal definitions of our main concepts, and briefly records some natural correlation inequalities for the Go polynomials.

Let $G=(V,E)$ be a graph, and $\lambda \in \mathbb{N} \cup \{0\}$. A *partial assignment* is a function $f : W \to \Lambda$, where $W \subseteq V$ and Λ is a finite set (we often use $\{1,\ldots,\lambda\}$). If $|\Lambda| = \lambda$, we may call f a *partial λ-assignment*. If $W = V$ then f is simply an *assignment* or a *λ-assignment*. We refer to the members of Λ as *colours*, and for each $i \in \Lambda$ the set of vertices receiving colour i is denoted by C_i or $C_i(f)$ and called a *colour class*. Note that the colour classes here do not have to be independent sets. (An *independent* or *stable* set of vertices is one in which no two vertices are adjacent.) The set of uncoloured vertices is denoted by $U = U(f)$. An edge of G is *good* (respectively, *bad*) under f if both its endpoints are coloured and these colours are different (the same).

A colour class C under a particular λ-assignment is said to be *free in G* if, for each component H of the subgraph $\langle C \rangle$ induced by C, some vertex of H is adjacent to an uncoloured vertex. Clearly, C is free if and only if for each $v \in C$ there is a path $P(v)$ in G, beginning at v, ending at some uncoloured vertex w, and with all vertices of $P(v) - w$ belonging to C. We say that a partial λ-assignment is *free in G* if all its colour classes are free in G.

Referring back to the game, any configuration of stones on the graph constitutes a partial 2-assignment, with each player's stones giving a colour class in the obvious

way. A *group* is (in our language) a component of a colour class. Note that it is quite possible for a group to be free in this sense, even though it might be "doomed" in the sense that the opponent can certainly capture the group on a subsequent move. In this latter case, a Go player might well deem the group to be dead, even though we (with a far simpler purpose) do not. We are only concerned with the *prima facie* legality of a position, ignoring prior history (including whether or not our position could arise in an actual game that begins with an empty board) and any questions of viability of uncaptured groups.

Fig. 1 illustrates these remarks. It contains five black groups (note that the bottom left region contains two black groups of three stones each). All are free except the one at top right. The top right group is captured by White and is immediately removed from the board. The bottom right group is doomed, but not yet captured so it is still free for our purposes. The partial 2-assignment represented by this configuration of stones is not free, because it has a group that is not free. However, after the captured group is removed, the partial 2-assignment corresponding to the remaining stones is free.

We consider hypothetical Go positions generated not from actual play but by random partial assignments of colours to vertices, and define polynomials that give the probability that such a random partial assignment is free.

We use the following model of random partial assignment. Let $(r, p_1, \ldots, p_\lambda)$ be a probability distribution. For each $v \in V(G)$, assign it colour i with probability p_i (and no colour with probability $r = 1 - \sum_{i=1}^{\lambda} p_i$), with the choices for the vertices being independent and identically distributed. The colour classes C_i and the uncoloured set U are now set-valued random variables. Let f be the partial λ-assignment generated randomly in this way. In this paper we will be interested in

$$\Pr(f \text{ is free in } G).$$

If $\lambda = 2$ and $p_1 = p_2 = p \leqslant \frac{1}{2}$, then this probability is a polynomial in p. We refer to it as the *Go polynomial* of G, and denote it by $\mathrm{Go}(G; p)$. We will also take an interest in the number of free partial λ-assignments in G (for general λ). This is a polynomial in λ, and will be written $\mathrm{Go}^{\#}(G; \lambda)$. It is just $(\lambda+1)^n \Pr(f \text{ is free in } G)$ for the case $r = p_1 = \cdots = p_\lambda = 1/(\lambda+1)$, where $n = |V(G)|$.

Here are some elementary examples.

$$\begin{aligned}
\mathrm{Go}(K_n; p) &= 1 - (2p)^n, \\
\mathrm{Go}(\overline{K_n}; p) &= (1-2p)^n, \\
\mathrm{Go}(K_{1,k}; p) &= 1 - 2p + 2p((1-p)^k - p^k), \\
\mathrm{Go}(C_4; p) &= r^4 + 8r^3 p + 24r^2 p^2 + 4r(1-2p^3), \quad (\text{where } r = 1-2p), \\
\mathrm{Go}^{\#}(K_n; \lambda) &= (\lambda+1)^n - \lambda^n, \\
\mathrm{Go}^{\#}(\overline{K_n}; \lambda) &= 1, \\
\mathrm{Go}^{\#}(K_{1,k}; \lambda) &= \lambda^k + \lambda(2^k - 1), \\
\mathrm{Go}^{\#}(C_4; \lambda) &= 1 + 4\lambda + 6\lambda^2 + 4(\lambda^3 - \lambda(\lambda-1) - \lambda(\lambda-1)(\lambda-2)), \\
&= 1 + 14\lambda^2.
\end{aligned}$$

We also note the obvious fact that if G and H are disjoint graphs then

$$\mathrm{Go}(G \cup H; p) = \mathrm{Go}(G; p)\mathrm{Go}(H; p),$$
$$\mathrm{Go}^{\#}(G \cup H; \lambda) = \mathrm{Go}^{\#}(G; \lambda)\mathrm{Go}^{\#}(H; \lambda).$$

The Go polynomials give the probability that *all* colour classes, in a random partial assignment, are free. It is natural to consider the probability that certain designated colour classes are free (with other colour classes allowed to be free or not), and ask how these probabilities compare.

Write $\mathrm{Go}_1(G; p)$ for the probability that colour class C_1 is free, under the model used for $\mathrm{Go}(G; p)$. If $L \subseteq \Lambda$, write $\mathrm{Go}^{\#}_L(G; \lambda)$ for the number of partial λ-assignments in which colour classes C_i, $i \in L$, are free.

The following are straightforward corollaries of a lemma of McDiarmid [20], and are reminiscent of the main inequality of [10].

Theorem 1.
(a) $\mathrm{Go}(G; p) \leqslant \mathrm{Go}_1(G; p)^2$.
(b) *If* $L_1, L_2 \subseteq \Lambda$ *and* $L_1 \cap L_2 = \emptyset$, *then* $\mathrm{Go}^{\#}_{L_1 \cup L_2}(G; \lambda) \leqslant \lambda^{-n}\mathrm{Go}^{\#}_{L_1}(G; \lambda)\mathrm{Go}^{\#}_{L_1}(G; \lambda)$.
(c) $\mathrm{Go}^{\#}(G; \lambda) \leqslant \lambda^{-n(n-1)}(\mathrm{Go}^{\#}_1(G; \lambda))^{n-1}$.

3. Relations based on local changes

We now find some relations involving Go polynomials that are somewhat analogous to deletion–contraction expressions for Tutte polynomials (see, e.g., [30]).

3.1. Intermediate graphs

All our relations require us to move beyond the class of ordinary graphs, though in recursively applying them to compute Go polynomials we eventually end up dealing just with ordinary graphs again, as we shall see.

An *intermediate graph* is a graph with exactly one label from $\{\mathtt{J}, \mathtt{L}, \mathtt{A}, \mathtt{a}\}$ on each edge, and zero, one or two labels from $\{\mathtt{C}, \mathtt{S}\}$ on each vertex. We briefly discuss the interpretation of the labels to assist reading the subsequent definitions.

Observe that, when determining whether a group in an ordinary graph is free, the edges serve two distinct purposes: *joining* vertices of the same colour together in order to help form a group, and being a *lifeline* by making a coloured vertex adjacent to an uncoloured one in order to preserve the former's group. This provides the motivation for our first two edge labels, `J` and `L`, as they allow us to separate the two roles of a normal (unlabelled) edge.

If an edge is labelled `A`, then we forbid its endpoints from receiving the same colour, but allow them to receive two different colours. In addition, one or both endpoints may remain uncoloured. Label `a` is similar, except that at most one of the edge's endpoints can be uncoloured. Vertex label `C` means that a vertex so labelled cannot be left uncoloured, and a vertex labelled `S` guarantees the freedom of any group of which it

Table 1

Label	Short for	Explanation
Edge labels		
J	Joining	Joins identically coloured vertices into groups
L	Lifeline	Allows a coloured vertex (and hence its group) to be free, by being linked to an uncoloured vertex
A	Antichromatic (first type)	Endpoints cannot get same colour
a	Antichromatic (second type)	Ditto, and cannot both be uncoloured
Vertex labels		
C	Coloured	Must receive a colour
S	Safe	Can survive without a lifeline

is a part, regardless of whether the group has a lifeline to an uncoloured vertex. The need for the various labels will become apparent in subsequent subsections. Table 1 may help keep track of them.

In practice, we only deal with three or four types of intermediate graph, each of which only uses a particular subset of these allowed labels. However, it is convenient to state all the definitions for intermediate graphs in general, rather than for the several specific types we actually use, to avoid repetition.

Let G be an intermediate graph. If $\mathtt{Z} \in \{\mathtt{J}, \mathtt{L}, \mathtt{A}, \mathtt{a}\}$ then we write $E_{\mathtt{Z}} = E_{\mathtt{Z}}(G)$ for the set of edges of G with label Z. These edge sets partition $E(G)$. Similarly, if $\mathtt{Z} \in \{\mathtt{C}, \mathtt{S}\}$, then $V_{\mathtt{Z}} = V_{\mathtt{Z}}(G)$ denotes the set of vertices with label Z, although these sets can now be arbitrary subsets of $V(G)$ (in particular, they do not have to partition $V(G)$).

The definitions of partial assignments, colour classes and so on extend immediately to intermediate graphs.

A colour class C of a partial λ-assignment of an intermediate graph G is *free in G* if

(a) for each component H of the subraph $\langle C \rangle$ of $(V, E_{\mathtt{J}})$ induced by C, *either* some vertex of H is adjacent via an edge of $E_{\mathtt{L}}$ to an uncoloured vertex, *or* some vertex of H is labelled S;

(b) it is an independent set in the subgraph $(V(G), E_{\mathtt{A}} \cup E_{\mathtt{a}})$.

Thus, C is free in G if and only if (a) for all $v \in C$ there exists a path $P(v)$ in G from v to some vertex w such that all vertices of $P(v) - w$ belong to C, all edges but the last are joining, and *either* w is uncoloured and the last edge of $P(v)$ is a lifeline, *or* w is labelled S and the last edge of $P(v)$ is also joining (which includes the case $w = v$ when $P(v)$ is trivial); and (b) the endpoints of an antichromatic edge do not both belong to C.

A partial assignment is *free* in G if (a) all its colour classes are, (b) no edge labelled a has both endpoints uncoloured, and (c) every vertex labelled C is coloured: $V_{\mathtt{C}} \subseteq \operatorname{dom} f$.

If f is free then it must be a proper colouring of the subgraph of $(V(G), E_{\mathtt{A}} \cup E_{\mathtt{a}})$ induced by $\operatorname{dom} f$.

We let f be a random partial λ-assignment, consider $\Pr(f \text{ is free in } G)$, and define the Go polynomials from this as in Section 2.

Let $\mathtt{Z}$ be a label. If $e \in E(G)$ then $G[e \leftarrow \mathtt{Z}]$ denotes the intermediate graph obtained by replacing the label of e by $\mathtt{Z}$. If $v \in V(G)$ then $N_G^{(\mathtt{Z})}(v)$ denotes the set of those neighbours w of v such that vw has label $\mathtt{Z}$. If $v \in V(G)$ then $G[v \leftarrow \mathtt{Z}]$ is obtained from G by giving v the label $\mathtt{Z}$ if it does not have it already, in addition to any other labels it may already have. If $W \subseteq V$ then $G[W \leftarrow \mathtt{Z}]$ is obtained from G analogously, by giving the additional label $\mathtt{Z}$ to all vertices in W that are not already so labelled.

G/e is the intermediate graph formed from G by contracting e and giving all labels held by its endpoints to the new vertex so formed.

If $\mathscr{L} \subseteq \{\mathtt{J}, \mathtt{L}, \mathtt{A}, \mathtt{a}, \mathtt{C}, \mathtt{S}\}$ then an *intermediate $\mathscr{L}$-graph* is an intermediate graph in which all labels belong to $\mathscr{L}$, and the class of such graphs is denoted by $\mathscr{G}(\mathscr{L})$. We frequently drop braces when the meaning is clear: for example, we may write $\mathscr{G}(\mathtt{J}, \mathtt{L})$ for $\mathscr{G}(\{\mathtt{J}, \mathtt{L}\})$.

Ordinary graphs can be regarded as intermediate $\{\mathtt{J}, \mathtt{L}\}$-graphs of a particular kind. To see this, let G be an ordinary graph. Replace each edge of G by a pair of parallel edges, one labelled $\mathtt{J}$ and the other labelled $\mathtt{L}$. Such a parallel pair makes explicit the "double act" performed by an ordinary edge: it can join like-coloured vertices to help form a group, and it can serve as a lifeline to an uncoloured vertex. The resulting graph is called the *intermediate version* of G. It is clear that a partial assignment f is free in G if and only if it is free in the intermediate version of G. If we wish to compute $\Pr(f \text{ is free in} G)$, we may as well start with the intermediate version of G.

Fig. 2 gives an overview of how the main results of this section allow recursive computation of the Go polynomials, using intermediate graphs. Each box represents a class of intermediate $\mathscr{L}$-graphs, with the label set $\mathscr{L}$ shown at the top of the box, and any additional structure for that class specified within the box. An arrow from one box to another means that repeated application of the indicated result allows the Go polynomial of an intermediate graph of the first type (i.e., from the class indicated in the first box) to be expressed in terms of Go polynomials of graphs of the second type. A solid line means that both polynomials Go( ) and Go$^{\#}$( ) are covered by the result; a dashed line means that the result only applies to Go$^{\#}$( ).

3.2. A four-term relation

The proof of our first relation (Corollary 4) rests on the following Lemma.

Lemma 2. *If $G \in \mathscr{G}(\mathtt{J}, \mathtt{L})$, f is a partial λ-assignment of G, and $e_{\mathtt{J}} \in E_{\mathtt{J}}(G)$ and $e_{\mathtt{L}} \in E_{\mathtt{L}}(G)$ are parallel edges, then:*

(a) *f is free in $G - e_{\mathtt{J}}$ or $G - e_{\mathtt{L}}$ if and only if f is free in G.*

(b) *f is free in both $G - e_{\mathtt{J}}$ and $G - e_{\mathtt{L}}$ if and only if f is free in $G - e_{\mathtt{J}} - e_{\mathtt{L}}$.*

Proof. The forward implication for (a) and the reverse implication for (b) are clear.

Suppose both $e_{\mathtt{J}}$ and $e_{\mathtt{L}}$ have endpoints $v_1, v_2 \in V(G)$.

(a) Suppose f is free in G. Exactly one of the following is true:

1. v_1, v_2 are both coloured, and $f(v_1) = f(v_2)$.

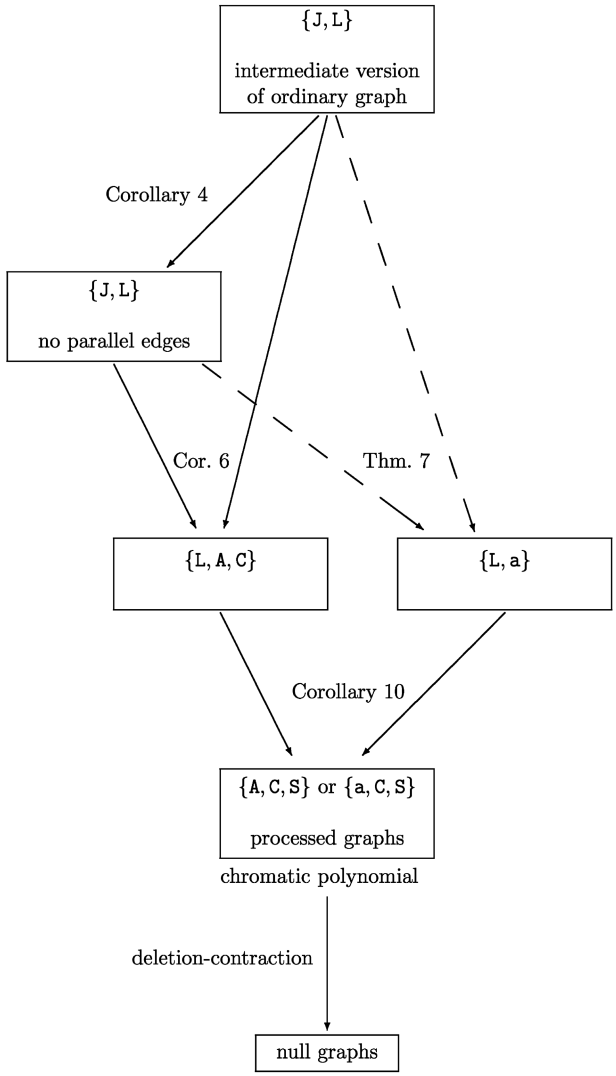

Fig. 2. Overview of computation of Go polynomials using intermediate graphs.

2. Exactly one of v_1, v_2 is uncoloured.
3. v_1, v_2 are both coloured and $f(v_1) \neq f(v_2)$, *or* v_1 and v_2 are both uncoloured.

If the first of these holds, then f is free in $G - e_L$. If the second holds, then f is free in $G - e_J$. If the third holds, then f is free in $G - e_J - e_L$ and hence in both $G - e_J$ and $G - e_L$. In any of these cases, f is free in at least one of $G - e_J$ and $G - e_L$.

(b) Suppose f is free in both $G - e_J$ and $G - e_L$. Again, exactly one of conditions 1–3 above holds. If the first holds, the fact that f is free in $G - e_J$ implies that f is

also free in $G - e_J - e_L$. If the second holds, the fact that f is free in $G - e_L$ implies that f is also free in $G - e_J - e_L$. If the third holds, then the fact that f is free in both $G - e_J$ and $G - e_L$ implies that f is free in $G - e_J - e_L$. □

Theorem 3. *If $G \in \mathscr{G}(J,L)$, f is a random partial λ-assignment of G, and $e_J \in E_J(G)$ and $e_L \in E_L(G)$ are parallel edges, then*

$$\begin{aligned}&\Pr(f \text{ is free in } G) + \Pr(f \text{ is free in } G - e_J - e_L)\\ &\quad = \Pr(f \text{ is free in } G - e_L) + \Pr(f \text{ is free in } G - e_J).\end{aligned}$$

Proof.

$$\Pr(f \text{ is free in } G) = \Pr((f \text{ is free in } G - e_L) \cup (f \text{ is free in } G - e_J)) \tag{1}$$

$$\begin{aligned}&= \Pr(f \text{ is free in } G - e_L) + \Pr(f \text{ is free in } G - e_J)\\ &\quad - \Pr((f \text{ is free in } G - e_L) \cap (f \text{ is free in } G - e_J))\\ &= \Pr(f \text{ is free in } G - e_L) + \Pr(f \text{ is free in } G - e_J)\\ &\quad - \Pr(f \text{ is free in } G - e_J - e_L),\end{aligned} \tag{2}$$

where (1) and (2) follow by Lemma 2(a) and (b), respectively. □

Corollary 4. *If G is a stage 1 graph and $e_J \in E_J(G)$ and $e_L \in E_L(G)$ are parallel edges then*

$$\begin{aligned}\mathrm{Go}(G; p) + \mathrm{Go}(G - e_J - e_L; p) &= \mathrm{Go}(G - e_L; p) + \mathrm{Go}(G - e_J; p).\\ \mathrm{Go}^{\#}(G; \lambda) + \mathrm{Go}^{\#}(G - e_J - e_L, \lambda) &= \mathrm{Go}^{\#}(G - e_L; \lambda) + \mathrm{Go}^{\#}(G - e_J; \lambda).\end{aligned}$$

Repeated application of Corollary 4 allows us to express the Go polynomial of a graph G (specifically, its intermediate version) in terms of the Go polynomials of a number of intermediate $\{J,L\}$-graphs, with no parallel edges, derived from it. The number of these derived graphs will in general be exponential in the number of edges of the initial (ordinary) graph G, in fact $3^{|E(G)|}$ in the worst case.

3.3. Relations that eliminate joining edges

We now look at relations that use contraction and relabelling of joining edges. These introduce antichromatic edges and (except for the last relation of the subsection) one type of vertex label.

Theorem 5. *Let $G \in \mathscr{G}(J,L,C,A)$ and $e = uv \in E_J(G)$. Let f be a random partial λ-assignment whose distribution satisfies $p_1 = \cdots = p_\lambda = p \leqslant 1/\lambda$. Then*

$$\Pr(f \text{ free in } G) = p \Pr(f \text{ free in } (G/e)[u \leftarrow C]) + \Pr(f \text{ free in } G[e \leftarrow A]).$$

Proof. In the following, f is randomly generated, while g just stands for some non-random partial assignment.

$$\begin{aligned}
\Pr(f \text{ free in } G) &= \sum_{g \text{ free in } G} p^{\Sigma_i |C_i(g)|}(1-\lambda p)^{n-\Sigma_i |C_i(g)|} \\
&= \sum_{\substack{g \text{ free in } G: \\ g(u)=g(v)}} p^{\Sigma_i |C_i(g)|}(1-\lambda p)^{n-\Sigma_i |C_i(g)|} \\
&\quad + \sum_{\substack{g \text{ free in } G: \\ \{u,v\}\cap U(g)\neq\emptyset, \\ \text{or } g(u)\neq g(v)}} p^{\Sigma_i |C_i(g)|}(1-\lambda p)^{n-\Sigma_i |C_i(g)|} \\
&= p \sum_{\substack{g' \text{ free in } G/uv: \\ u\notin U(g')}} p^{\Sigma_i |C_i(g')|}(1-\lambda p)^{(n-1)-\Sigma_i |C_i(g')|} \\
&\quad + \Pr(f \text{ free in } G[e \leftarrow \mathtt{A}]) \\
&\qquad\qquad \text{(noting that, in the first summand,} \\
&\qquad\qquad \textstyle\bigcup_i C_i(g') = \left(\bigcup_i C_i(g)\right) \setminus \{v\}\text{)} \\
&= p \Pr(f \text{ free in } (G/e)[u \leftarrow \mathtt{C}]) \\
&\quad + \Pr(f \text{ free in } G[e \leftarrow \mathtt{A}]). \qquad \square
\end{aligned}$$

Corollary 6. *If* $G \in \mathscr{G}(\mathtt{J},\mathtt{L},\mathtt{C},\mathtt{A})$ *and* $e = uv \in E_{\mathtt{J}}(G)$ *then*

$$\begin{aligned}
\mathrm{Go}(G;p) &= p\mathrm{Go}((G/e)[u \leftarrow \mathtt{C}];p) + \mathrm{Go}(G[e \leftarrow \mathtt{A}];p), \\
\mathrm{Go}^{\#}(G;\lambda) &= \mathrm{Go}^{\#}((G/e)[u \leftarrow \mathtt{C}];\lambda) + \mathrm{Go}^{\#}(G[e \leftarrow \mathtt{A}];\lambda).
\end{aligned}$$

This can be applied repeatedly to express a Go polynomial of an intermediate $\{\mathtt{J},\mathtt{L}\}$-graph G in terms of Go polynomials of intermediate $\{\mathtt{L},\mathtt{A},\mathtt{C}\}$-graphs. Of course, the number of these latter graphs is again exponential, this time at most $2^{|E_{\mathtt{J}}(G)|}$. Observe that, for any intermediate $\{\mathtt{L},\mathtt{A},\mathtt{C}\}$-graph H so produced,

$$|V_{\mathtt{C}}(H)| + |E_{\mathtt{A}}(H)| \leqslant |E_{\mathtt{J}}(G)|, \tag{3}$$

since every application of Corollary 6 produces graphs with one fewer joining edge and either at most one new vertex label or exactly one new antichromatic edge (though some existing antichromatic edges may become redundant as a side-effect of some contractions).

Corollary 6 may be used on intermediate versions of ordinary graphs, since it still works when the edge e is parallel to a lifeline, and can then be used repeatedly until no joining vertices are left. At that stage, between every pair of adjacent vertices there will be two parallel edges, with labels $\mathtt{L}$ and $\mathtt{A}$.

Alternatively (but generally less efficiently), beginning again with an intermediate version of an ordinary graph, Corollary 4 may be applied repeatedly until all graphs under consideration have no parallel edges, and then Corollary 6 may be applied repeatedly to those graphs.

For the polynomials $\mathrm{Go}^{\#}$, another relation may be used to remove joining edges. To do so, we must use antichromatic edges of a different kind to those used above. It is straightforward to prove

Theorem 7. *Let* $G \in \mathscr{G}(\mathtt{J},\mathtt{L},\mathtt{a})$ *and* $e \in e_{\mathtt{J}}(G)$. *Then*

$$\mathrm{Go}^{\#}(G;\lambda) = \mathrm{Go}^{\#}(G/e;\lambda) + \mathrm{Go}^{\#}(G[e \leftarrow \mathtt{a}];\lambda).$$

This may be used to express $\mathrm{Go}^{\#}$ for an intermediate $\{\mathtt{J},\mathtt{L}\}$-graph G in terms of Go polynomials of intermediate $\{\mathtt{L},\mathtt{a}\}$-graphs. We can begin with an intermediate version of an ordinary graph if we wish.

3.4. Relations based on removal of vertices and lifelines

For graphs with no joining edges, there is no notion of vertices of like colour joining together, so each coloured vertex is just a singleton group. This enables us to prove another type of relation.

Theorem 8. *If* $G \in \mathscr{G}(\mathtt{L},\mathtt{A},\mathtt{a},\mathtt{C},\mathtt{S})$ *and* $v \in V \setminus V_{\mathtt{C}}$ *then*

$$\begin{aligned}\Pr(f \text{ free in } G) = r\Pr(f \text{ free in } (G-v)[N_G^{(\mathtt{L})}(v) \leftarrow \mathtt{S}]) \\ + \Pr(f \text{ free in } G[v \leftarrow \mathtt{C}]).\end{aligned}$$

Proof.

$$\begin{aligned}\Pr(f \text{ free in } G) &= \Pr(v \in U)\Pr(f \text{ free in } G \mid v \in U) \\ &\quad + \Pr((f \text{ free in } G) \cap (v \notin U)) \\ &= r\Pr(f \text{ free in } (G-v)[N_G^{(\mathtt{L})}(v) \leftarrow \mathtt{S}]) \\ &\quad + \Pr(f \text{ free in } G[v \leftarrow \mathtt{C}]),\end{aligned}$$

since $v \in U$ implies that its neighbours must be free (if coloured), and $v \notin U$ means it must be coloured. □

Proposition 9. *If* $G \in \mathscr{G}(\mathtt{L},\mathtt{A},\mathtt{a},\mathtt{C},\mathtt{S})$ *and* $v \in V_{\mathtt{C}} \cap V_{\mathtt{S}}$ *then*

$$\Pr(f \text{ free in } G) = \Pr(f \text{ free in } G - E_{\mathtt{L}}(\{v\}, N_G^{(\mathtt{L})}(v))).$$

(Note that $G - E_{\mathtt{L}}(\{v\}, N_G^{(\mathtt{L})}(v))$ is the intermediate graph obtained from G by deleting all lifelines incident with v.)

Proof. A lifeline plays no role when one endpoint is a safe coloured vertex. □

Corollary 10. *Let* $G \in \mathcal{G}(\mathtt{L}, \mathtt{A}, \mathtt{a}, \mathtt{C}, \mathtt{S})$.
(a) *If* $v \in V \setminus V_{\mathtt{C}}$ *then*

$$\mathrm{Go}(G; p) = (1-2p)\,\mathrm{Go}((G-v)[N_G^{(\mathtt{L})}(v) \leftarrow \mathtt{S}]; p) + \mathrm{Go}(G[v \leftarrow \mathtt{C}]; p),$$
$$\mathrm{Go}^{\#}(G; \lambda) = \mathrm{Go}^{\#}((G-v)[N_G^{(\mathtt{L})}(v) \leftarrow \mathtt{S}]; \lambda) + \mathrm{Go}^{\#}(G[v \leftarrow \mathtt{C}]; \lambda).$$

(b) *If* $v \in V_{\mathtt{C}} \cap V_{\mathtt{S}}$ *then*

$$\mathrm{Go}(G; p) = \mathrm{Go}(G - E_{\mathtt{L}}(\{v\}, N_G^{(\mathtt{L})}(v)); p),$$
$$\mathrm{Go}^{\#}(G; \lambda) = \mathrm{Go}^{\#}(G - E_{\mathtt{L}}(\{v\}, N_G^{(\mathtt{L})}(v)); \lambda).$$

Remarks. 1. If G has a lifeline with both endpoints labelled $\mathtt{C}$ (regardless of whether either endpoint has label $\mathtt{S}$ as well), then this lifeline plays no role and can be deleted. 2. If G has a vertex labelled $\mathtt{C}$ but not $\mathtt{S}$ and with no incident lifelines, then no partial assignment can be free in G, so $\Pr(f \text{ is free in } G)$ and the corresponding Go polynomials are all zero.

Repeated application of Corollary 10 and the subsequent remarks allows one to express a Go polynomial of an intermediate $\{\mathtt{L}, \mathtt{A}, \mathtt{C}\}$-graph in terms of the Go polynomials of a number of intermediate $\{\mathtt{A}, \mathtt{C}, \mathtt{S}\}$-graphs in which all vertices are labelled both $\mathtt{C}$ and $\mathtt{S}$. This number is of course exponential in $|V(G) \setminus (V_{\mathtt{C}} \cap V_{\mathtt{S}})|$.

Similar remarks apply to the Go polynomial $\mathrm{Go}^{\#}$ of an intermediate $\{\mathtt{L}, \mathtt{a}\}$-graph. Note that, once all vertices are labelled both $\mathtt{C}$ and $\mathtt{S}$, the edge labels $\mathtt{A}$ and $\mathtt{a}$ have the same effect.

3.5. Counting colourings again

Suppose we have an intermediate $\{\mathtt{A}, \mathtt{C}, \mathtt{S}\}$-graph with all vertices labelled both $\mathtt{C}$ and $\mathtt{S}$. Call such a graph a *processed graph*. As usual, we look to express its Go polynomial in terms of some simpler local modifications of the graph. But this is familiar territory, since a partial assignment f is free in a processed graph G if and only if it is a proper colouring, in the usual sense, of G (now unlabelled). So, if G is a processed graph, then

$$\mathrm{Go}(G; p) = p^n P(G; 2),$$
$$\mathrm{Go}^{\#}(G; \lambda) = P(G; \lambda).$$

These Go polynomials can therefore use the deletion–contraction expression for the chromatic polynomial (though $P(G; 2)$ can of course be calculated much more easily).

3.6. Recursive computation of Go polynomials

The naïve approach to computing Go polynomials is to work directly from the definition and enumerate all partial λ-assignments, picking out those that are free in G, adding up the appropriate probability for each. This will need to be done for sufficiently

many λ to determine the polynomial. The complexity will be dominated by a factor $(\lambda+1)^n$, or something close to it (where $n=|V(G)|$).

The recursive rules presented in this section allow faster computation of the Go polynomials, although it still takes exponential time. Let G be a graph with n vertices and m edges. Starting with its intermediate version, we apply Corollary 6 (respectively, Theorem 7) repeatedly, until we have a set of $\leqslant 2^m$ intermediate $\{\mathtt{L},\mathtt{A},\mathtt{C}\}$-graphs (a set of intermediate $\{\mathtt{L},\mathtt{a}\}$-graphs). For each such graph, apply Corollary 10 and the subsequent remarks repeatedly until we are left just with processed graphs, $\leqslant 2^n$ in number. This phase will give up to 2^{m+n} processed graphs altogether. Finally, for each processed graph, we find its chromatic polynomial using the usual deletion–contraction rule, which will generate $\leqslant 2^m$ base cases per processed graph. The complexity will thus be dominated by a factor of 2^{2m+n}.

4. Complexity

In this section, we show that calculating $\mathrm{Go}^{\#}(G;\lambda)$ is #P-complete for $\lambda\geqslant 2$. We do not undertake a comprehensive study of the complexity of Go polynomials. We just introduce the topic and prove a first result on it.

Observe that Section 3.5 gives us the #P-completeness of $\mathrm{Go}^{\#}(G;\lambda)$ when G is a certain kind of processed graph—roughly, those for which partial λ-assignments are just proper colourings, so little is really said about complexity. It is of more interest to consider $\mathrm{Go}^{\#}(G;\lambda)$ for ordinary graphs G, when partial λ-colourings are quite different things to proper colourings, so #P-completeness is not immediate.

We will need the following special case of a result of Fel'dman on linear forms in logarithms.

Lemma 11 (Fel'dman [13]; see also Baker [2, Theorem 3.1], Baker [3]). *If integers s,t,u are not all zero, with absolute values $\leqslant m$ and $s+t+u=0$, then*

$$|s\ln 2 + t\ln 3 + u\ln(\lambda+1)| > m^{-c},$$

where c depends only on λ.

Theorem 12. *Computing $\mathrm{Go}^{\#}(G;\lambda)$ is #P-complete for $\lambda\geqslant 2$.*

Proof. Membership of #P is clear. To begin with, suppose $\lambda\geqslant 3$. It is then known that computing $P(G;\lambda)$ is #P-complete. We show that there is a polynomial time Turing reduction from $P(G;\lambda)$ to $\mathrm{Go}^{\#}(G;\lambda)$.

Let $G=(V,E)$ be any graph, with n vertices and m edges, whose λ-colourings we want to count. Let $k_1=\lceil m^c(n\ln(\lambda+1)+\ln((m+1)(m+2)/2))\rceil$, where c is the constant from Lemma 11. Let $k_2=\lceil (n+mk+1)\ln(1+\lambda)/\ln(1+\lambda^{-1})\rceil$. (The exact values of k_1 and k_2 are unimportant. They need to be sufficiently large for purposes explained below, yet polynomially bounded in m and n.)

We will construct a graph $G'(k_1)$ from G, and then construct a graph $G''(k_1,k_2)$ from $G'(k_1)$. We will use the Go polynomial $\mathrm{Go}^{\#}(G''(k_1,k_2);\lambda)$ to compute $P(G;\lambda)$.

The graph $G'(k_1)$ is constructed as follows. For each edge $e=v_1v_2$ of G, add k_1 *new vertices* $w_{e,1},\ldots,w_{e,k_1}$ joined only to v_1 and v_2. So all new vertices have degree 2, and $G'(k_1)$ has $n+mk_1$ vertices and $m+2mk_1$ edges.

Now form $G''(k_1,k_2)$ by adding a copy of K_{k_2}, disjoint from $G'(k_1)$, and then adding all possible edges between vertices in that copy and vertices of G. (So vertices in $V(G'(k_1))\backslash V(G)$ still have degree 2.)

The free partial λ-assignments of $G''(k_1,k_2)$ can be divided into two sets: those that leave some vertex in the K_{k_2} uncoloured, and those that give colours to all those vertices. Let the number of these be g_1 and g_2, respectively. Now it is easy to see that $g_2\leqslant\lambda^{k_2}(\lambda+1)^{n+mk_1}$. Furthermore, $g_1=((\lambda+1)^{k_2}-\lambda^{k_2})\mathrm{Go}^{\#}(G'(k_1)[V(G)\leftarrow \mathtt{S}],\lambda)$.

Our oracle gives us $\mathrm{Go}^{\#}(G''(k_1,k_2),\lambda)=g_1+g_2$. Our choice of k_2 ensures that $g_2<(\lambda+1)^{k_2}-\lambda^{k_2}$, so it is easy to work out $\mathrm{Go}^{\#}(G'(k_1)[V(G)\leftarrow \mathtt{S}],\lambda)$. It remains to compute $P(G;\lambda)$ from this.

Let $b_{st}^{(k_1)}$ be the number of partial λ-assignments that are free in $G'(k_1)[V(G)\leftarrow \mathtt{S}]$ and that induce a partial λ-assignment of G with s bad edges, t good edges, and $u=m-s-t$ edges with at least one endpoint uncoloured. Let a_{st} be the number of partial λ-assignments of G that have s bad edges and t good edges.

Due to the safety of the vertices of G, a partial λ-assignment is free in $G'(k_1)[V(G)\leftarrow \mathtt{S}]$ if and only if it gives each vertex in $V(G'(k_1))\backslash V$ the same colour as one of its two neighbours or no colour at all. Hence

$$b_{st}^{(k_1)}=a_{st}(2^s3^t(\lambda+1)^{m-s-t})^{k_1}.$$

Now let the quantities $2^s3^t(\lambda+1)^{m-s-t}$ be arranged in decreasing order, and rename them h_j, $1\leqslant j\leqslant(m+1)(m+2)/2$, where $h_1\geqslant\cdots\geqslant h_{(m+1)(m+2)/2}$. It is easy to show that $i\neq j$ implies $h_i\neq h_j$, so they are actually in strictly decreasing order. Consider the ratio between two successive h_j. Suppose $h_j=2^{s_j}3^{t_j}(\lambda+1)^{u_j}$ and $h_{j+1}=2^{s_{j+1}}3^{t_{j+1}}(\lambda+1)^{u_{j+1}}$ $(1\leqslant j<(m+1)(m+2)/2)$. Then this ratio

$$\begin{aligned}\rho_j=h_j/h_{j+1}&=2^{s_j-s_{j+1}}3^{t_j-t_{j+1}}(\lambda+1)^{u_j-u_{j+1}}\\&=\exp((s_j-s_{j+1})\ln 2+(t_j-t_{j+1})\ln 3+(u_j-u_{j+1})\ln(\lambda+1))\\&>\mathrm{e}^{m^{-c}},\end{aligned}\tag{4}$$

for some $c\geqslant 1$ depending only on λ, by Lemma 11.

We know (from earlier) the value

$$\begin{aligned}\mathrm{Go}^{\#}(G'(k_1)[V(G)\leftarrow \mathtt{S}],\lambda)&=\sum_{j=1}^{(m+1)(m+2)/2}b_{s_jt_j}^{(k_1)}\\&=\sum_{j=1}^{(m+1)(m+2)/2}a_{s_jt_j}h_j^{k_1}.\end{aligned}$$

Observe that

$$\begin{aligned} h_j^{k_1} &= \rho_j^{k_1} h_{j+1}^{k_1} \\ &> \mathrm{e}^{k_1 m^{-c}} h_{j+1}^{k_1} && \text{(by (4))} \\ &> (\lambda+1)^n((m+1)(m+2)/2) h_{j+1}^{k_1} && \text{(by choice of } k_1) \\ &> (\lambda+1)^n \sum_{l \geqslant j+1} h_l^{k_1} \\ &\geqslant \sum_{l \geqslant j+1} a_{s_l t_l} h_l^{k_1}. \end{aligned}$$

It is therefore straightforward to work out the a_{st}, much as one works out representations of numbers to some base. Once this is done, we know $a_{0m} = P(G;\lambda)$, which is what we wanted.

The procedure takes polynomial time. Thus, we have a polynomial time Turing reduction from the chromatic polynomial to the polynomial $\mathrm{Go}^{\#}$, so the latter is #P-complete.

We sketch the proof for $\lambda = 2$, which uses reduction from counting 3-colourings. Given a graph G, form $G'(k_1)$ as above, except that each pair of vertices adjacent in G is now linked by k_1 paths of length 3 (instead of length 2), disjoint except at their endpoints, in addition to the edges of G. Then form $G''(k_1,k_2)$ as above. Make k_1 and k_2 sufficiently large yet polynomially bounded. The oracle gives the number of free partial 2-assignments of $G''(k_1,k_2)$, and from this one can easily obtain the number of free partial 2-assignments in $G'(k_1)[V(G) \leftarrow \mathrm{S}]$. Let a_{st} be the number of partial 2-assignments of G with s bad edges and t edges that are either good (both endpoints coloured) or have exactly one endpoint uncoloured. These have $u = m - s - t$ edges with both endpoints uncoloured. Let $b_{st}^{(k)}$ be the corresponding number of free partial 2-assignments in $G'(k_1)[V(G) \leftarrow \mathrm{S}]$. Show that $b_{st}^{(k)} = a_{st} 6^s 8^t 9^{m-s-t}$. Appeal to a lemma similar to Lemma 11 (with $\ln 2$, $\ln 3$ and $\ln(\lambda+1)$ replaced by $\ln 6$, $\ln 8$ and $\ln 9$, respectively). Reasoning as in the previous proof, we can find the a_{st}. This yields $a_{0m} = P(G;3)$. □

The polynomial transformation used in this proof for $\lambda = 2$ could have been used to deal with all $\lambda \geqslant 2$ together, at the cost of a longer proof. Not only is the construction (and associated counting arguments) for $\lambda = 2$ a bit more complex, but when using that technique for $\lambda \geqslant 3$ it is necessary to consider four kinds of edges instead of three, and more care is needed in ensuring that the resulting linear forms in logarithms have the required properties.

Acknowledgements

I am grateful to Rod Worley and the referees for their helpful comments.

References

[1] J.L. Arocha, B. Llano, Mean value for the matching and dominating polynomial, Discuss. Math. Graph Theory 20 (2000) 57–69.

[2] A. Baker, Transcendental Number Theory, Cambridge University Press, Cambridge, 1975.

[3] A. Baker, The theory of linear forms in logarithms, in: A. Baker, D.W. Masser (Eds.), Transcendence Theory: Advances and Applications, Academic Press, London, 1977, pp. 1–27.

[4] E.R. Berlekamp, Introductory overview of mathematical Go endgames, in: R.K. Guy (Ed.), Combinatorial Games, Proc. Symp. Appl. Math., Vol. 43, Columbus, Ohio, 6–7 August 1990, American Mathematical Society, Providence, RI, 1991, pp. 73–100.

[5] E.R. Berlekamp, Y. Kim, Where is the "Thousand-Dollar Ko"? in: R.J. Nowakowski (Ed.), Games of No Chance, Math. Sci. Res. Inst. Publ., Vol. 29, Berkeley, Ca, 11–21 July 1994, Cambridge University Press, Cambridge, 1996, pp. 203–226.

[6] E.R. Berlekamp, D. Wolfe, Mathematical Go: Chilling Gets the Last Point, A. K. Peters, Wellesley, MA, 1994.

[7] B. Bouzy, T. Cazenave, Computer Go: an AI oriented survey, Artif. Intell. 132 (2001) 39–103.

[8] T.H. Brylawski, J.G. Oxley, The Tutte polynomial and its applications, in: N. White (Ed.), Matroid Applications, Encyclopedia Mathematical Applications, Vol. 40, Cambridge University Press, Cambridge, 1992, pp. 123–225.

[9] M. Crâşmaru, J. Tromp, Ladders are PSPACE-complete, in: T.A. Marsland, I. Frank (Eds.), Computers and Games: Second International Conference, CG 2000, Hamamatsu, Japan, 26–28 October 2000, Lecture Notes in Computer Science, Vol. 2063, Springer, Berlin, 2001, pp. 241–249.

[10] G.E. Farr, A correlation inequality involving stable set and chromatic polynomials, J. Combin. Theory (Ser. B) 58 (1993) 14–21.

[11] G.E. Farr, A generalization of the Whitney rank generating function, Math. Proc. Cambridge Philos. Soc. 113 (1993) 267–280.

[12] G.E. Farr, Some results on generalised Whitney functions, Adv. Appl. Math., to appear.

[13] N.I. Fel'dman, An improvement of the estimate of a linear form in the logarithms of algebraic numbers (Russian), Mat. Sbornik 77 (1968) 423–436 (English translation: Math. USSR Sbornik 6 (1968) 393–406).

[14] T. Helgason, Aspects of the theory of hypermatroids, in: Hypergraph Seminar, Lecture Notes in Mathematics, Vol. 411, Springer, Berlin, 1974, pp. 191–213.

[15] K. Iwamoto, Go for Beginners, Ishi Press, 1972, and Penguin Books, Harmondsworth, 1976.

[16] F. Jaeger, D.L. Vertigan, D.J.A. Welsh, On the computational complexity of the Jones and Tutte polynomials, Math. Proc. Cambridge Philos. Soc. 108 (1990) 35–53.

[17] J.P.S. Kung, The Rédei function of a relation, J. Combin. Theory (Ser. A) 29 (1980) 287–296.

[18] D. Lichtenstein, M. Sipser, GO is polynomial-space hard, J. Assoc. Comput. Mach. 27 (1980) 393–401.

[19] L. Lovász, M.D. Plummer, Matching Theory, Annals of Discrete Mathematics, Vol. 29, North-Holland Mathematical Studies, Vol. 121, North-Holland, Amsterdam, 1986.

[20] C. McDiarmid, On a correlation inequality of Farr, Combin. Probab. Comput. 1 (1992) 157–160.

[21] M. Müller, Review: computer Go 1984–2000, in: T.A. Marsland, I. Frank (Eds.), Computers and Games: Second International Conference, CG 2000, Hamamatsu, Japan, 26–28 October 2000, Lecture Notes in Computer Science, Vol. 2063, Springer, Berlin, 2001, pp. 405–413.

[22] M. Müller, Computer Go, Artif. Intell. 134 (2002) 145–179.

[23] S.D. Noble, D.J.A. Welsh, A weighted graph polynomial from chromatic invariants of knots, in: Symposium à la Mémoire de François Jaeger (Grenoble, 31 August–4 September 1998), Ann. Inst. Fourier (Grenoble) 49 (1999) 1057–1087.

[24] J.G. Oxley, D.J.A. Welsh, The Tutte polynomial and percolation, in: J.A. Bondy, U.S.R. Murty (Eds.), Graph Theory and Related Topics, Academic Press, New York, 1979, pp. 329–339.

[25] J.G. Oxley, G.P. Whittle, Tutte invariants for 2-polymatroids, in: N. Robertson, P.D. Seymour (Eds.), Graph Structure Theory, Contemporary Mathematics, Vol. 147, Seattle, 1991, American Mathematical Societ, Providence, RI, 1993, pp. 9–19.

[26] J.G. Oxley, G.P. Whittle, A characterization of Tutte invariants of 2-polymatroids, J. Combin. Theory (Ser. B) 59 (1993) 210–244.
[27] R.P. Stanley, A symmetric function generalization of the chromatic polynomial of a graph, Adv. Math. 111 (1995) 166–194.
[28] R.P. Stanley, Graph colorings and related symmetric functions: ideas and applications: a description of results, interesting applications, & notable open problems, Discrete Math. 193 (1998) 267–286.
[29] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (1979) 410–421.
[30] D.J.A. Welsh, Matroid Theory, London Mathematical Society Monograph No. 8, Academic Press, New York, 1976.
[31] D.J.A. Welsh, Complexity: Knots, Colourings and Counting, in: London Mathematical Society Lecture Note Series, Vol. 186, Cambridge University Press, Cambridge, 1993.
[32] G.P. Whittle, Characteristic polynomials of weighted lattices, Adv. Math. 99 (1993) 125–151.
[33] G.P. Whittle, The critical problem for polymatroids, Quart. J. Math. Oxford Ser. (2) 45 (1994) 117–125.