# Random-Facet and Random-Bland require subexponential time even for shortest paths

Oliver Friedmann* Thomas Dueholm Hansen† Uri Zwick‡

**Abstract**

The Random-Facet algorithm of Kalai and of Matoušek, Sharir and Welzl is an elegant randomized algorithm for solving linear programs and more general LP-type problems. Its expected subexponential time of $2^{\tilde{O}(\sqrt{m})}$, where $m$ is the number of inequalities, makes it the fastest known combinatorial algorithm for solving linear programs. We previously showed that Random-Facet performs an expected number of $2^{\tilde{\Omega}(\sqrt[3]{m})}$ pivoting steps on some LPs with $m$ inequalities that correspond to $m$-action *Markov Decision Processes* (MDPs). We also showed that Random-Facet$^{1P}$, a one permutation variant of Random-Facet, performs an expected number of $2^{\tilde{O}(\sqrt{m})}$ pivoting steps on these examples. Here we show that the same results can be obtained using LPs that correspond to instances of the classical *shortest paths* problem. This shows that the stochasticity of the MDPs, which is essential for obtaining lower bounds for Random-Edge, is not needed in order to obtain lower bounds for Random-Facet. We also show that our new $2^{\tilde{\Omega}(\sqrt{m})}$ lower bound applies to Random-Bland, a randomized variant of the classical anti-cycling rule suggested by Bland.

## 1 Introduction

Linear programming (LP) is one of the most successful mathematical modeling tools. The *simplex algorithm*, introduced by Dantzig [9], is one of the most widely used methods for solving linear programs. The simplex algorithm starts at a vertex of the polytope corresponding to the linear program. (We assume, for simplicity, that the linear program is feasible, bounded, and non-degenerate, and that a vertex of the polytope is available.) If the current vertex is not optimal, then at least one of the edges incident on it leads to a neighboring vertex with a smaller objective function. A *pivoting rule* determines which one of these vertices to move to. The simplex algorithm, with any pivoting rule, is guaranteed to find an optimal solution of the linear program.

Unfortunately, with essentially all known deterministic pivoting rules, the simplex method is known to require *exponential time* on some linear programs (see, e.g., Klee and Minty [30], Amenta and Ziegler [2], and a recent subexponential bound of Friedmann [16]). While there are other polynomial time algorithms for solving LP problems, most notably the *ellipsoid algorithm* (Khachian [29]) and *interior point methods* (Karmarkar [28]), these algorithms are not *strongly* polynomial, i.e., their running time, in the unit-cost model, depends on the number of bits needed to represent the coefficients of the LP, and not just on the combinatorial size of the problem, i.e., the number of variables and the number of constraints. The question of whether there exists a strongly polynomial time algorithm for solving linear programs is of great theoretical importance.

Kalai [26, 27] and Matoušek, Sharir and Welzl [35] devised a *randomized* pivoting rule for the simplex algorithm, known as Random-Facet, and obtained a *subexponential* $2^{O(\sqrt{(m-d)\log d})}$ upper bound on the expected number of pivoting steps it performs on *any* linear program, where $d$ is the *dimension* of the linear program, and $m$ is the number of inequality constraints. Matoušek, Sharir and Welzl [35] actually obtained a dual version of the

[*] Department of Computer Science, University of Munich, Germany. E-mail: `Oliver.Friedmann@gmail.com`.

[†] Department of Management Science and Engineering, Stanford University, USA. Supported by The Danish Council for Independent Research | Natural Sciences (grant no. 12-126512). E-mail: `tdh@cs.au.dk`.

[‡] Blavatnik School of Computer Science, Tel Aviv University, Israel. Research supported by BSF grant no. 2012338 and by the The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11). E-mail: `zwick@tau.ac.il`.

Random-Facet pivoting rule and obtained an upper bound of $2^{O(\sqrt{d\log(m-d)})}$ on the number of dual pivoting steps it performs. Random-Facet is currently the fastest known pivoting rule for the simplex algorithm.

The Random-Facet pivoting rule works as follows. Let $v$ be the current vertex visited by the simplex algorithm. Randomly choose one of the *facets* of the polytope that contain $v$. Let $F$ be the facet chosen. Recursively find the optimal vertex $v'$ among all vertices of the polytope that lie on $F$. This corresponds to finding an optimal solution for a modified linear program in which the inequality corresponding to $F$ is replaced by an equality. If $v'$ is also an optimal solution of the original problem, we are done. Otherwise, it is not difficult to check that there must be a single edge leading from $v'$ to a vertex $v''$ with a smaller value. This edge is taken and the algorithm is recursively run from $v''$. A more formal description, as well as an equivalent non-recursive description, and descriptions of two variants, Random-Facet$^{1P}$ and Random-Bland, of Random-Facet are given in Section 2.

Random-Facet can be used to solve not only linear programs, but also a wider class of abstract optimization problems known as *LP-type* problems (see [35]). Matoušek [33] showed that the subexponential upper bound on the complexity of Random-Facet is essentially tight for LP-type problems. The instances used by Matoušek [33], however, are far from being linear programs.

In [17, 18], we claimed an $2^{\tilde{\Omega}(\sqrt{m})}$ lower bound on the complexity of Random-Facet for actual LPs. The lower bound presented there, however, is actually for the *one-permutation* variant Random-Facet$^{1P}$ of Random-Facet. In [17], we erroneously claimed that the expected running times of Random-Facet and Random-Facet$^{1P}$ are the same, and thus thought that our lower bound also applied to Random-Facet. The mistake is pointed out in [23, 19]. The lower bounds given in [17, 18] for Random-Facet$^{1P}$ are adapted in [23] to give an $2^{\tilde{\Omega}(\sqrt[3]{m})}$ lower bound for Random-Facet.

In this paper, we obtain an $2^{\tilde{\Omega}(\sqrt[3]{m})}$ lower bound for the original Random-Facet pivoting rule. While the new bound is smaller then the bounds obtained in [17, 18] for Random-Facet$^{1P}$, a square-root in the exponent is replaced by a cube-root, it still provides a subexponential lower bound on the running time of Random-Facet. We also obtain a new $2^{\tilde{\Omega}(\sqrt{m})}$ lower bound on the running time of Random-Bland, a randomized version of Bland's pivoting rule (Bland [7]), which is closely related to both Random-Facet and Random-Facet$^{1P}$.

The lower bound given in [17] is for *parity games*, a class of deterministic 2-player games. The lower bounds given in [18, 23] are for LPs that correspond to *Markov Decision Processes* (MDPs), a class of stochastic 1-player games. (For more on MDPs, see Puterman [38].) Here, we obtain subexponential lower bounds for the running times of Random-Facet, Random-Facet$^{1P}$ and Random-Bland on LPs that correspond to standard acyclic *shortest paths* problems, the simplest form of deterministic 1-player games.

It is interesting, and perhaps surprising, that Random-Facet, the fastest known pivoting rule for general linear programs, requires a subexponential number of pivoting steps even for acyclic shortest paths problems which can be easily solved in linear time.

In [18] we also presented MDPs on which Random-Edge, a different randomized pivoting rule, requires an expected number of $2^{\tilde{\Omega}(\sqrt[4]{m})}$ pivoting steps. In contrast with Random-Facet, it is not difficult to show that Random-Edge solves shortest paths problems in expected polynomial time.

As mentioned, Random-Facet can be used to solve not only linear programs, but also a wider class of problems known as LP-type problems (see Matoušek *et al.* [35]). Ludwig [32] was the first to show that Random-Facet can be used to solve *stochastic games*. Petersson and Vorobyov [37], and Björklund *et al.* [5, 6] used Random-Facet to solve several deterministic and stochastic games. Halman [22] showed that all these form LP-type problems.

The line of work pursued here started by Friedmann [14, 15] who proved that the *strategy iteration* algorithm for *parity games* may require exponential time. The strategy iteration algorithm is an adaptation to 2-player games of Howard's [24] *policy iteration* algorithm for solving MDPs. (For more on parity games, see also Vöge and Jurdziński [43] and Jurdziński *et al.* [25].). Fearnley [11] used a similar construction to obtain an exponential lower bound for Howard's algorithm for MDPs, a class of stochastic 1-player games. Policy iteration algorithms are similar in nature to the simplex algorithm. The main difference is that they may apply *multi-switches*, i.e., perform several pivoting steps simultaneously. In [17, 18] and here we continue this line of work and obtain lower bounds for Random-Facet applied to shortest paths problems. Although Random-Facet performs only one pivoting step at a time, its randomized nature makes it similar to more general policy iteration algorithms.

The rest of the paper is organized as follows. In Section 2 we give a more formal description of the Random-Facet pivoting rule and its variants. In Section 3 we remind the reader how shortest paths problems can be cast as linear programs and solved by the simplex algorithm. We also explain how Random-Facet works in this concrete setting. In Section 4 we describe a *randomized counter* on which our lower bounds, as well as the lower bounds in [17, 18] are based. In Section 5 we describe instances of the shortest paths problems that can be used to simulate the behavior of the randomized counter. In Section 6 we obtain the lower bound for Random-Facet. In Section 7 we obtain the lower bound of Random-Facet$^{1P}$. In Section 8 we obtain the lower bound of Random-Bland. We end in Section 9 with some concluding remarks and open problems.

## 2 Random-Facet, Random-Facet$^{1P}$ and Random-Bland

We begin with a brief introduction to linear programming problems and the simplex algorithm. For a more thorough treatment, the reader is refereed to the textbooks of Dantzig [9], Chvátal [8], Schrijver [39], Bertsimas and Tsitsiklis [4], and Matoušek and Gärtner [34]. As customary, we consider linear programming problems in standard form:

$$(P)\quad \begin{array}{lrcl} \min & \mathbf{c}^T\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} \qquad\qquad (D)\quad \begin{array}{lrcl} \max & \mathbf{b}^T\mathbf{y} & & \\ \text{s.t.} & \mathbf{A}^T\mathbf{y} & \leq & \mathbf{c} \end{array}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$. Here $n$ is the number of variables of the linear program, and $m$ is the number of linear equalities. We assume that the rows of $A$ are linearly independent. The *dimension* of the linear program is defined to be $d = n - m$. The linear program $(P)$ on the left is referred to as the *primal* linear program, and the linear program $(D)$ on the right is referred to as the *dual* linear problem. (We focus, here, mainly on the primal linear program.)

Let $B \subset [n] = \{1, 2, \ldots, n\}$, $|B| = m$. We let $\mathbf{B} = \mathbf{A}_B \in \mathbb{R}^{m\times m}$ be the $m \times m$ matrix obtained by selecting the columns of $A$ whose indices belong to $B$. If the columns of $\mathbf{B}$ are linearly independent, we refer to $B$ as a *basis* and to $\mathbf{B}$ as a *basis matrix.* There is then a unique vector $\mathbf{x} \in \mathbb{R}^n$ for which $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{x}_i = 0$, for $i \notin B$. If we let $N = [m] \setminus B$, then $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{x}_N = \mathbf{0}$. This vector $\mathbf{x}$ is the *basic solution* corresponding to $B$. If $\mathbf{x}_B \geq \mathbf{0}$, then $\mathbf{x}$ is said to be a *basic feasible solution* (bfs). A bfs is a *vertex* of the polyhedron corresponding to $(P)$. The variables in $\{\mathbf{x}_i \mid i \in B\}$, are referred to a *basic* variables, while the variables in $\{\mathbf{x}_i \mid i \in N\}$, are referred to as *non-basic* variables. A simple and standard manipulation shows that the objective function can also be expressed as $\mathbf{c}^T\mathbf{x} = \mathbf{c}_B^T\mathbf{x}_B + \bar{\mathbf{c}}^T\mathbf{x}$, where $\bar{\mathbf{c}} = \mathbf{c} - (\mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{A})^T$. The vector $\bar{\mathbf{c}} \in \mathbb{R}^n$ is referred to as the vector of *reduced costs.* Note that $\mathbf{c}_B^T\mathbf{x}_B$ here is a constant. Also note that $\bar{\mathbf{c}}_B = \mathbf{0}$, i.e., the reduced costs of the basic variables are all 0. It is also not difficult to check that if $\bar{\mathbf{c}} \geq \mathbf{0}$, then $\mathbf{x}$ is an *optimal* solution.

The simplex algorithm starts with a bfs $\mathbf{x}$ corresponding to a basis $B$. (We do not get here into the question as to how the first bfs is obtained.) If $\bar{\mathbf{c}} \geq \mathbf{0}$, then $\mathbf{x}$ is an optimal solution, and we are done. Otherwise, there must be an index $i \in N = [m] \setminus B$ for which $\bar{\mathbf{c}}_i < 0$. If the linear program is *non-degenerate*, then the set $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x}_j = 0 \text{ for } j \in N \setminus \{i\}\}$ is an *edge* of the polytope corresponding to $(P)$ that leads to a bfs $\mathbf{x}'$ corresponding to a basis $B' = B \cup \{i\} \setminus \{j\}$, for some $j \in B$. Furthermore, $\mathbf{c}^T\mathbf{x}' < \mathbf{c}^T\mathbf{x}$. The algorithm moves from $\mathbf{x}$ to $\mathbf{x}'$ and continues from there. This is referred to as a *pivoting step.* If there are several indices $i$ for which $\bar{\mathbf{c}}_i < 0$, a *pivoting rule* is used to select one of them.

Dantzig's [9] pivoting rule, chooses an index $i$ with the *smallest reduced cost*, i.e., an index $i$ which minimizes $\bar{\mathbf{c}}_i$. Bland's [7] pivoting rule chooses the *smallest* index $i$ for which $\bar{\mathbf{c}}_i < 0$. Bland's pivoting rule ensures the termination of the simplex algorithm even in the presence of degeneracies.

Random-Facet is a more complicated randomized pivoting rule introduced by Kalai [26, 27], and in a dual form by Matoušek, Sharir and Welzl [35]. Unlike Dantzig's and Bland's rules mentioned above, Random-Facet has *memory.* Its behavior depends on decisions made at previously visited bfs's, and not only on the current bfs. We begin by describing a recursive version of Random-Facet which closely follows [26, 27]. Let $\mathbf{x}$ be the initial bfs, corresponding to basis $B$. Select a random index $i \notin B$. Recursively find an optimal solution $\mathbf{x}'$ of the linear program in which the inequality constraint $\mathbf{x}_i \geq 0$ is replaced by the equality constraint $\mathbf{x}_i = 0$. (This corresponds to staying within the *facet* $\mathbf{x}_i = 0$.) This effectively *removes* the non-basic variable $\mathbf{x}_i$, the column corresponding to it in $\mathbf{A}$, and the entry corresponding to it in $\mathbf{c}$ from the linear program. If $\mathbf{x}'$ is also an optimal solution of the original linear program, we are done. Otherwise, $i$ is the only index for which $\bar{\mathbf{c}}'_i < 0$, where $\bar{\mathbf{c}}'$ is the vector of reduced costs corresponding to $\mathbf{x}'$. Perform the pivoting step in which $i$ enters the set of basic indices, and obtain a new bfs $\mathbf{x}''$. Run the algorithm recursively from $\mathbf{x}''$.

Pseudo-code of the recursive version of Random-Facet is given on the top of Figure 1. The first argument $F$ of Random-Facet is the set of indices of the variables that are *not* constrained to be 0. Initially $F = [n]$. The second argument $B$ is the current basis. Whenever Random-Facet is called, it is assumed that $B$ defines a valid bfs of the problem and that $B \subseteq F$. For simplicity, the pseudo-code given works exclusively with bases. It returns a basis $B$ that corresponds to an optimal bfs of the problem. If $F = B$, then $B$ is the only, and therefore optimal, solution of the current problem, so it is returned. Otherwise, Random-Facet chooses at random an index $i$ of a currently non-basic variable. By the recursive call $B' \leftarrow$ Random-Facet$(F \setminus \{i\}, B)$ it computes an optimal solution under the additional constraint that the $i$-th variable is required to be 0. The call Improve$(B', i)$ checks whether performing a pivot step that inserts $i$ into the basis $B'$ would yield a bfs with a smaller objective function. More specifically, Improve$(B', i)$ computes the bfs $\mathbf{x}'$ corresponding to $B'$ and its reduced cost vector $\bar{\mathbf{c}}'$, and checks whether $\bar{\mathbf{c}}'_i < 0$. If so, $B'' \leftarrow$ Pivot$(B', i)$ performs the corresponding pivot step and returns the resulting basis $B''$. (Note that $B'' = B \cup \{i\} \setminus \{j\}$, for some $j \in B$.) Finally, the second recursive call Random-Facet$(F, B'')$ computes and returns an optimal basis of the problem.

The description of Random-Facet given on the top of Figure 1 is identical to the description of Random-Facet given by Matoušek, Sharir and Welzl [35], with the only difference that Matoušek *et al.* consider it to be an algorithm for solving the dual linear program $(D)$. Note that every basis $B \subseteq [n]$, $|B| = m$ constructed by the algorithm defines not only a basic feasible primal solution, as described above, but also a basic, not necessarily feasible, dual solution $\mathbf{y} = (\mathbf{B}^T)^{-1}\mathbf{c}_B$. In fact, all dual solutions, except the last one, are infeasible. Note that $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{A}^T\mathbf{y}$. Choosing an index $i \in F \setminus B$ and adding the primal constraint $\mathbf{x}_i = 0$, corresponds to *discarding* the $i$-th dual constraint. If the solution returned by the first recursive call Random-Facet$(F \setminus \{i\}, B)$ is dual feasible, then it is an optimal solution, of both the primal and dual linear programs. Otherwise, a pivoting step is performed. The correspondence between the algorithms of Kalai [26, 27] and of Matoušek, Sharir and Welzl [35] was first noticed by Goldwasser [21].

Random-Facet makes a fresh random choice at each invocation. It is natural to wonder whether so much randomness is really needed. Two *one-permutation* variants of Random-Facet are given at the bottom of Figure 1. Both of them base their choices on a single permutation $\sigma$ of $[n]$. This permutation is randomly chosen before the first invocation. The same (random) permutation is then used in all recursive calls. The first of these variants, shown on the bottom left, referred to as Random-Facet$^{1P}$, is identical to Random-Facet in every aspect, except that instead of choosing a random $i \in F \setminus B$, it chooses the index $i$ from $F \setminus B$ for which $\sigma(i)$ is minimized. The second variant, shown on the bottom right of Figure 1, contains an additional variation. Instead of choosing the index $i$ from $F \setminus B$, it is chosen from the whole of $F$. Also, the recursion bottoms out when $F = \emptyset$, and not when $F = B$. The invariant $B \subseteq F$ is no longer maintained.

We refer to the second variant as Bland as, as we shall see, it is actually equivalent to Bland's [7] pivoting rule. When $\sigma$ is a uniformly random permutation we refer to the algorithm as Random-Bland.

It is not difficult to verify the correctness of these two variants. Random-Facet$^{1P}(F, B, \sigma)$ finds an optimal solution when all variables not in $F$ are required to be 0. If Random-Bland$(F, B, \sigma)$ returns a basis $B'$, then $B'$ is an optimal solution when all variables not in $F \cup B'$ are required to be 0. This holds for any permutation $\sigma$. The permutation $\sigma$ may determine, however, the sequence of pivoting steps performed. (In both cases, the proof follows from the fact that all pivoting steps performed reduce the value of the objective function, and that when the algorithm terminates, there are no such improving pivoting steps.)

In [17], we erroneously claimed that for every linear program, the expected number of pivoting steps performed by Random-Facet and Random-Facet$^{1P}$, with a randomly chosen permutation $\sigma$, is the same. A counterexample to this claim is given in [19]. The 'proof' given in [17] relied on the linearity of expectations, claiming that the

```
Algorithm Random-Facet($F, B$)
  if $F = B$ then
    return $B$
  else
    $i \leftarrow$ Random($F \setminus B$)
    $B' \leftarrow$ Random-Facet($F \setminus \{i\}, B$)
    if Improve($B', i$) then
      $B'' \leftarrow$ Pivot($B', i$)
      return Random-Facet($F, B''$)
    else
      return $B'$
```

```
Algorithm Random-Facet$^{1P}$($F, B, \sigma$)
  if $F = B$ then
    return $B$
  else
    $i \leftarrow \text{argmin}_{j \in F \setminus B}\, \sigma(j)$
    $B' \leftarrow$ Random-Facet$^{1P}$($F \setminus \{i\}, B, \sigma$)
    if Improve($B', i$) then
      $B'' \leftarrow$ Pivot($B', i$)
      return Random-Facet$^{1P}$($F, B'', \sigma$)
    else
      return $B'$
```

```
Algorithm Bland($F, B, \sigma$)
  if $F = \emptyset$ then
    return $B$
  else
    $i \leftarrow \text{argmin}_{j \in F}\, \sigma(j)$
    $B' \leftarrow$ Bland($F \setminus \{i\}, B, \sigma$)
    if Improve($B', i$) then
      $B'' \leftarrow$ Pivot($B', i$)
      return Bland($F, B'', \sigma$)
    else
      return $B'$
```

Figure 1: The Random-Facet, Random-Facet$^{1P}$, and Bland algorithms.

fact that the two recursive calls of Random-Facet$^{1P}$ share some random choices does not effect the expected number of pivoting steps performed. The 'proof', however, contained a subtle flaw. Thus, while Random-Facet has a subexponential upper bound on its complexity, no such subexponential upper bounds are currently known for Random-Facet$^{1P}$ and Random-Bland.

We next describe an equivalent non-recursive version of Random-Facet. Let $\mathbf{x}$ be the current bfs, corresponding to the set $B$ of basic indices, and the set $N$ of non-basic indices. The algorithm maintains a *permutation* $\langle i_1, i_2, \ldots, i_d \rangle$, where $d = n - m$, of the indices of $N$. If $\mathbf{x}$ is the initial bfs, then a random permutation of $N$ is chosen. If $\bar{\mathbf{c}} \geq \mathbf{0}$, then $\mathbf{x}$ is optimal, and we are done. Otherwise, choose the *first* index $i_j$ in the permutation for which $\bar{\mathbf{c}}_{i_j} < 0$. Perform a pivoting step from $B$ to $B' = B \cup \{i_j\} \setminus \{i'\}$, for some $i' \notin B \cup \{i_j\}$, so that $N' = N \setminus \{i_j\} \cup \{i'\}$. The permutation corresponding to $N'$ is obtained by randomly permuting $i_1, \ldots, i_{j-1}, i'$, and keeping the original order of $i_{j+1}, \ldots, i_d$. Proving, by induction, the equivalence of the recursive and non-recursive definitions of Random-Facet is an instructive exercise. The permutation of $N$ kept by the algorithm is exactly the 'memory' referred to earlier. It also corresponds to the stack kept by the recursive version of Random-Facet.

The non-recursive formulation of Random-Facet is similar, yet different, from the following randomized version of Bland's rule. Choose a random permutation $\langle i_1, i_2, \ldots, i_n \rangle$ of $[n]$. At each step, choose the *first* index $i_j$ in the permutation for which $\bar{\mathbf{c}}_{i_j} < 0$ to enter the basis. We show below that this randomized version of Bland's rule is exactly the non-recursive version of the variant Bland given above. There are two main differences between the non-recursive versions of Random-Facet and Bland. The first is that Bland uses a permutation of the indices of all variables, not only those that are currently non-basic. The second is that Bland uses the same permutation throughout the operation of the algorithm, while Random-Facet 'refreshes' the permutation at each step by randomly permuting the prefix $i_1, \ldots, i_{j-1}, i'$.

The proof that the recursive and non-recursive formulations of Bland are equivalent follows easily by induction. To get the exact equivalence, we assume that Bland chooses the *last* index $i_j$ in the permutation $\sigma$ for which $\bar{\mathbf{c}}_{i_j} < 0$ to enter the basis. Let $\sigma^{-1} = \langle i_1, i_2, \ldots, i_n \rangle$. Note that every recursive call of Bland is of the form $\textsc{Bland}(F_k, B, \sigma)$, where $F_k = \{i_k, i_{k+1}, \ldots, i_n\}$ for some $k \in [n+1]$. (We let $F_{n+1} = \emptyset$.) The non-recursive algorithm iteratively considers $i_n, i_{n-1}, \ldots$ until finding the first index $j$ for which $\textsc{Improve}(B, j)$ returns true. If no such improving switch is found, then $B$ is an optimal basis, and the algorithm terminates. We let $\textsc{Bland}'(k, B, \sigma)$ denote the non-recursive algorithm which only considers the indices $i_n, i_{n-1}, \ldots, i_k$. We claim that for every linear program, every initial basis $B \subseteq [n]$, every permutation $\sigma$, and every $k \in [n+1]$, $\textsc{Bland}(F_k, B, \sigma)$ and $\textsc{Bland}'(k, B, \sigma)$ perform exactly the same sequence of pivoting steps and hence return the same basis. Suppose that the claim is true for all larger values of $k$, and for all better bases $B$. (A basis $B'$ is better than $B$ if the value of the bfs corresponding to $B'$ is smaller than that of $B$.) If $B$ is an optimal basis, then both algorithms stop immediately. Otherwise, Bland performs the recursive call $\textsc{Bland}(F_{k+1}, B, \sigma)$. By induction, this recursive call is equivalent to $\textsc{Bland}(k+1, B, \sigma)$, so both return the same basis $B'$. Now, if $\textsc{Improve}(B', i_k)$ is false, then both $\textsc{Bland}(F_k, B, \sigma)$ and $\textsc{Bland}'(k, B, \sigma)$ are done. Otherwise, they both perform the pivoting step $B'' \leftarrow \textsc{Pivot}(B', i_k)$ and then perform $\textsc{Bland}(F_k, B'', \sigma)$ and $\textsc{Bland}'(k, B'', \sigma)$, respectively. As $B''$ is a better basis than $B$, it follows by induction that these two calls are again equivalent.

## 3 Shortest paths

In [17, 18] we showed that $\textsc{Random-Facet}^{1P}$ performs a subexponential number of pivoting steps on some linear programs that correspond to MDPs. Here we obtain similar subexponential lower bounds also for Random-Facet and Random-Bland. Furthermore, we show that such lower bounds for Random-Facet, $\textsc{Random-Facet}^{1P}$ and Random-Bland can be obtained using linear programs that correspond to the *purely combinatorial* problem of finding *shortest paths* in directed graphs. The graphs we use are even acyclic and all their edge weights are non-negative. It is slightly more convenient for us to consider shortest paths from all vertices *to* a given *target* vertex, rather than shortest paths *from* a given *source* vertex. The two problems, however, are clearly equivalent.

Let $G = (V, E, c)$ be a weighted directed graph, where $c : E \to \mathbb{R}$ is a *cost* (or *length* function) defined on its edges. Let $\textsc{t} \in V$ be a specific vertex designated as the *target* vertex. We let $n = |V \setminus \{\textsc{t}\}|$ and $m = |E|$ be the number of vertices, not counting the target, and edges in $G$, respectively. We are interested in finding a tree of shortest paths from all vertices to $\textsc{t}$. The problem, for general graphs with positive and negative edge weights, but no negative cycles, can be solved in $O(mn)$ time using a classical algorithm of Bellman and Ford [3, 13]. When the edge weights are non-negative, the problem can be solved in $O(m + n \log n)$ time using Dijkstra's algorithm [10]. When the graph is acyclic, the problem can be easily solved in $O(m + n)$ time.

The simplex algorithm, specialized to the *min cost flow* problem, is usually referred to as the *network simplex* algorithm. For a thorough treatment of the network simplex algorithm, see Chvátal [8], Ahuja *et al.* [1], and Bertsimas and Tsitsiklis [4]. As the shortest paths problem is a very special case of the min cost flow problem, it can also be solved using the network simplex algorithm.

The shortest path problem can be formulated as a min cost flow problem, and hence as a linear program, as follows. Finding a tree of shortest paths from all vertices to the target $\textsc{t}$ is equivalent to finding the min cost flow in which we have a supply of one unit at each vertex, other than $\textsc{t}$, and a demand of $n$ units at $\textsc{t}$. (Recall that $n$ is the number of non-terminal vertices.) The corresponding primal and dual linear programs are:

$$(P) \quad \begin{array}{lrcl} \min & \mathbf{c}^T\mathbf{x} & & \\ \text{s.t.} & \mathbf{A}\mathbf{x} & = & \mathbf{e} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} \qquad\qquad (D) \quad \begin{array}{lrcl} \max & \mathbf{e}^T\mathbf{y} & & \\ \text{s.t.} & \mathbf{A}^T\mathbf{y} & \leq & \mathbf{c} \end{array}$$

where $\mathbf{A} \in \mathbb{R}^{n \times m}$ is the *incidence matrix* of the graph. We assume, without loss of generality, that $V \setminus \{\textsc{t}\} = [n]$. [1] Each row of $\mathbf{A}$ corresponds to a vertex of the graph $G$. Each column of $\mathbf{A}$ corresponds to an edge of $G$. Each

[1] When considering linear programs in standard form, it is customary to use $n$ for the number of variables and $m$ the number of equality constraints. When considering graphs, it is customary to use $n$ for the number of vertices and $m$ for the number of edges.

column contain a single $+1$ entry, and possibly a $-1$ entry. If the $i$-th edge is $(j,k)$, then $\mathbf{A}_{j,i} = +1$ and $\mathbf{A}_{k,i} = -1$. All other entries of the $i$-th column are zeros. If the $i$-th edge is $(j, \text{T})$, then $\mathbf{A}_{j,i} = +1$ and all other entries of the $i$-th column are zeros. (Note that some authors define the incidence matrix to be the negation of our incidence matrix.) The vector $\mathbf{c} \in \mathbb{R}^m$ contains the edge costs. The vector $\mathbf{e} \in \mathbb{R}^n$ is the all one vector. The primal variable vector $\mathbf{x} \in \mathbb{R}^m$ is the *flow* vector, specifying the flow on each edge. The constraints $\mathbf{Ax} = \mathbf{e}$ ensure that the net flow out of each vertex is 1. Note that the target T does not appear explicitly in this formulation. Finally, $\mathbf{y} \in \mathbb{R}^n$, the dual variables vector, directly specifies stipulated distances from each vertex to T.

It is not difficult to check that every bfs $B$ of $(P)$ corresponds to a *tree* containing paths from all vertices to T. If $i \in B$, i.e., the $i$-th edge $e_i = (j,k)$ belongs to the tree, then $\mathbf{x}_i$, the flow on $e_i$, is the number of *descendants* of $j$ in this tree, including $j$ itself. The cost of $B$ is thus the sum of the lengths of the paths along the tree from all vertices to T. The cost is therefore minimized when $B$ corresponds to a shortest paths tree. The dual variables corresponding to $B$ are the *distances* along the tree. Thus, $\mathbf{y}_j$ is simply the length of the path from $j$ to T in the tree. The reduced costs $\mathbf{c}$ also have a simple combinatorial interpretation. If $e_i = (j,k)$, then $\bar{\mathbf{c}}_i = \mathbf{c}_i + \mathbf{y}_k - \mathbf{y}_j$. (If $e_i = (j, \text{T})$, then $\bar{\mathbf{c}}_i = \mathbf{c}_i - \mathbf{y}_j$.) Note that if $\bar{\mathbf{c}}_i < 0$, then $\mathbf{c}_i + \mathbf{y}_k < \mathbf{y}_j$, and thus the path from $j$ to T that starts with the edge $e_i = (j,k)$, which is currently not in the tree, is shorter than the path from $j$ to T along the tree. The tree can therefore be improved by performing a *switch* in which the edge $e_i = (j,k)$ is inserted, and the edge of the tree currently emanating from $j$ is removed. If the graph does not contain *negative* cycles, then a new and improved tree is obtained. This is exactly the pivoting step in which $i$ enters the basis.

Each tree $B$ of paths from all vertices to the terminal T is obtained by choosing one outgoing edge from each non-terminal vertex. This may also be viewed as specifying a *policy* in a 1-player game. In the case of shortest paths with no negative cycles, the set of edges chosen, i.e., the policy, is required to be *acyclic*. In more general classes of games, a policy may contain cycles.

In the sequel, we view a basis, i.e., tree, $B$, not as a set of indices but as a set of edges. We also let $y_B(j) = \mathbf{y}_j$ denote the distance in $B$ from vertex $j$ to T, for every $j \in [n]$. If $e = (i,j)$ is an edge of the graph and $c(i,j) + y_B(j) < y_B(i)$, then $e$ is said to be an *improving switch* with respect to $B$. We let $B[e]$ be the tree obtained by performing the switch, i.e., $B[e] = B \cup \{e\} \setminus \{e'\}$, where $e'$ is the edge of $B$ emanating from $i$. We let $y(j)$ denote the *distance* from $j$ to T in the graph, i.e., $y(j) = y_{B*}(j)$, where $B^*$ is a shortest paths tree. An edge $e = (i,j)$ is said to be *optimal* if it appears on some shortest path from $i$ to T. It is not difficult to see that an edge $(i,j)$ is optimal if and only if $y(i) = c(i,j) + y(j)$, and a tree $B$ is optimal, i.e., is a tree of shortest paths, if and only if all its edges are optimal.

We end the section by describing the way Random-Facet is used to find a tree of shortest paths in a weighted directed graph $G = (V, E, c)$ to a terminal T using the terminology used throughout the rest of the paper. The algorithm starts with an initial tree $B$. It randomly chooses an edge $e$ *not* in $B$. A recursive call of the algorithm is used to find a tree $B'$ of shortest paths in the graph $G \setminus \{e\}$ obtained by removing $e$. If $e$ is not an improving switch with respect to $B'$, then $B'$ is also a tree of shortest paths of the original graph $G$. Otherwise, the improving switch $B'' \leftarrow B'[e]$ is performed, and the algorithm is called recursively on the whole graph with $B''$.

# 4 A high-level description of the lower bound proofs

Our lower bounds for the Random-Facet, Random-Facet$^{1P}$, and Random-Bland algorithms are obtained by simulating the behavior of the *randomized counter* shown in Figure 2. Such a counter is composed of $n$ bits, $bit_1, \ldots, bit_n$, all initially 0. The randomized counter works in a recursive manner, focusing each time on a subset $N \subseteq [n] := \{1, \ldots, n\}$ of the bits, such that $bit_i = 0$ for all $i \in N$. Initially $N = [n]$. If $N = \emptyset$, then nothing is done. Otherwise, the counter chooses a random index $i \in N$ and recursively performs a randomized count on $N \setminus \{i\}$. When this recursive count is done, we have $bit_j = 1$, for every $j \in N \setminus \{i\}$, while $bit_i = 0$. Next, the $i$-th bit is set to 1, and all bits $j \in N \cap [i-1]$ are reset to 0. Finally, a recursive randomized count is performed on $N \cap [i-1]$.

Unfortunately, these two conventions clash in our case. We switch now to graph terminology, so $n$ is now the number of vertices, and hence the number of equality constraints, and $m$ is the number of edges, and hence the number of variables and inequality constraints.

```
Function RandCount($N$)
  if $N \neq \emptyset$ then
    $i \leftarrow$ Random($N$)
    RandCount($N \setminus \{i\}$)
    $bit_i \leftarrow 1$
    for $j \in N \cap [i-1]$ do  $bit_j \leftarrow 0$
    RandCount($N \cap [i-1]$)
```

```
Function RandCount$^{1P}$($N, \sigma$)
  if $N \neq \emptyset$ then
    $i \leftarrow \mathrm{argmin}_{j \in N}\ \sigma(j)$
    RandCount$^{1P}$($N \setminus \{i\}, \sigma$)
    $bit_i \leftarrow 1$
    for $j \in N \cap [i-1]$ do  $bit_j \leftarrow 0$
    RandCount$^{1P}$($N \cap [i-1], \sigma$)
```

Figure 2: The randomized counter. Original version on the left. One-permutation variant on the right.

Let $f(n)$ be the expected number of times the call RandCount($[n]$) sets a bit of the randomized counter to 1. It is not difficult to check that the behavior of RandCount($N$), where $|N| = k$, is equivalent to the behavior of RandCount($[k]$). It is then easy to see that $f(n)$ satisfies the following recurrence relation:

$$\begin{aligned} f(0) &= 0 \\ f(n) &= f(n-1) + 1 + \frac{1}{n}\sum_{i=0}^{n-1} f(i) \quad \text{for } n > 0 \end{aligned}$$

**Lemma 4.1** $f(n) \;=\; \displaystyle\sum_{k=1}^{n} \frac{1}{k!}\binom{n}{k}$ .

The proof of the lemma can be found in Appendix A.

According to Lemma 4.1 we can interpret $f(n)$ as the expected number of increasing subsequences in a uniformly random permutation of $[n] = \{1, \ldots, n\}$, i.e., every non-empty subset $S \subseteq [n]$ appears as an increasing subsequence with probability $1/|S|!$. The asymptotic behavior of $f(n)$ is known quite precisely:

**Lemma 4.2 ([31],[12, p. 596–597])** $f(n) \sim \dfrac{\mathrm{e}^{2\sqrt{n}}}{2\sqrt{\pi \mathrm{e}}\, n^{1/4}}$

Note, in particular, that $f(n)$ is subexponential. The challenge, of course, is to construct weighted directed graphs such that the behavior of RandCount is mimicked by the Random-Facet, Random-Facet$^{1P}$, and Random-Bland algorithms.

Just like it was natural to define one-permutation variants of the Random-Facet algorithm, the randomized counter, RandCount, can be implemented such that the random choices are based on a single, given, random permutation $\sigma : [n] \to [n]$. We refer to the one-permutation variant of RandCount as RandCount$^{1P}$. It is also shown in Figure 2. We let $f^{1P}(N, \sigma)$, where $N \subseteq [n]$, be the number of times the call RandCount$^{1P}(N, \sigma)$ sets a bit to 1. Note that RandCount$^{1P}$ is deterministic, and that:

$$\begin{aligned} f^{1P}(\emptyset, \sigma) &= 0 \\ f^{1P}(N, \sigma) &= f^{1P}(N \setminus \{i\}, \sigma) + 1 + f^{1P}(N \cap [i-1], \sigma) \qquad \text{where } \; N \neq \emptyset \; \text{ and } \; i \in \operatorname*{argmin}_{j \in N} \sigma(j) \end{aligned}$$

We let $f^{1P}(n)$ be the expected value of $f^{1P}([n], \sigma)$ when $\sigma$ is a uniformly random permutation of $[n]$. The following lemma is proved by using induction and linearity of expectation. The proof of the lemma can be found in Appendix B. [2]

**Lemma 4.3** $f(n) = f^{1P}(n)$.

[2] The fact that the expected number of steps performed by RandCount and RandCount$^{1P}$ is the same led us to (mistakenly) believe that the expected number of steps performed by Random-Facet and Random-Facet$^{1P}$ is also the same.

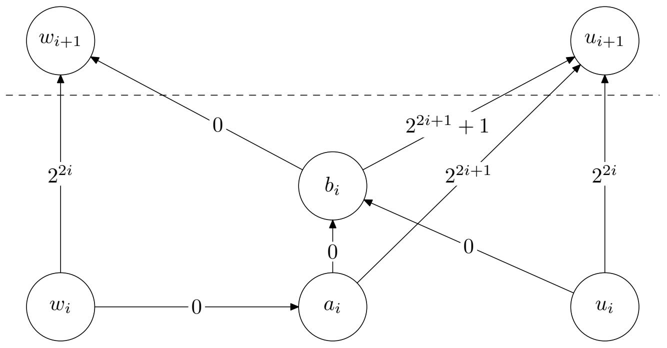

Figure 3: The $i$-th level of the graph $G_{n,r,s,t}$ when $r = s = t = 1$.

**Lower bound graphs and interpretation of trees.**

We next describe how to obtain the graphs used for our lower bounds. We use the same family of graphs for all three lower bounds. The graphs $G_{n,r,s,t}$ are parameterized by four parameters, $n, r, s, t \in \mathbb{N}$, where $n$ is the number of bits. For simplicity we initially consider the case when $r = s = t = 1$. The graphs are acyclic and are composed of $n$ levels. The $i$-th level represents the $i$-th bit of the counter, with the $n$-th bit being the most significant. Figure 3 shows the $i$-th level of the graph. The target $\textsc{t}$ is identified with $u_{n+1}$ and $w_{n+1}$. The initial tree consists entirely of edges with non-zero cost.

The critical vertex is the vertex $b_i$. A tree $B$ consists of one out-going edge from every vertex. Since the choice at $b_i$ is binary, we can interpret $B$ as a setting of the binary counter as follows. If $(b_i, w_{i+1}) \in B$ we say that the $i$-th bit is 1 and write $bit_i(B) = 1$. Otherwise we say that the $i$-th bit is 0 and write $bit_i(B) = 0$.

The critical edge is the edge $(b_i, w_{i+1})$. Recall that the Random-Facet, Random-Facet$^{1P}$, and Random-Bland algorithms work by removing edges and recursively solving sub-problems. When $(b_i, w_{i+1})$ is removed, the $i$-th bit is fixed to 0 during the following recursive call, which exactly corresponds to the behavior of RandCount. In order to argue that the recursive call sets the remaining bits to 1, we characterize optimal trees when certain key edges, such as $(b_i, w_{i+1})$, are removed.

**Optimal trees when edges are removed.**

We say that level $i$ is *stable* in a tree $B$ if either $B$ contains all the 0 cost edges of the $i$-th level, or $B$ contains all the non-zero cost edges of the $i$-th level. We also say that $B$ is stable from level $i$ if the $j$-th level is stable in $B$ for all $j \geq i$. It is not difficult to check that $y_B(u_i) = y_B(w_i)$ if $B$ is stable from level $i$. Moreover, if $B$ is stable from $i + 1$ and no edge is missing in the $i$-th level, then it is optimal to use the 0 cost edges of level $i$. On the other hand, if $(b_i, w_{i+1})$ is missing, then it is optimal to use the non-zero cost edges. In particular, the distances decrease as more and more bits are set to 1.

It follows from the above discussion that when $(b_i, w_{i+1})$ is removed the optimal tree is stable for all levels, and all bits, except $i$, are 1. After returning from the first recursive call, the Random-Facet, Random-Facet$^{1P}$, and Random-Bland algorithms all perform the improving switch $(b_i, w_{i+1})$, setting the $i$-th bit to 1 and making the level unstable. Suppose the edge $(a_i, b_i)$ is removed next, so that no path from $w_i$ to $w_{i+1}$ has cost 0. The distance from $u_i$ to $w_{i+1}$ remains 0, and for the optimal tree $B$ we get $y_B(w_i) = 2^{2i} + y_B(u_i)$. In level $i - 1$ all vertices then go to $u_i$, so that $y_B(w_{i-1}) = 2^{2(i-1)} + y_B(u_{i-1})$, and by induction all the lower bits are reset. Performing the improving switch $(a_i, b_i)$ reduces the cost of reaching $w_{i+1}$ from $w_i$ to 0, which enables the lower bits to count again. Note that the lower levels are unstable at this stage, but it is, in fact, only important that $(b_j, w_{j+1}) \notin B$ for $j < i$.

**Gadgets and full construction.**

The behavior described above requires Random-Facet, Random-Facet$^{1P}$, and Random-Bland to pick $(b_i, w_{i+1})$ before $(a_i, b_i)$, for $i \in [n]$, and to pick the remaining edges after $(a_i, b_i)$. We introduce two basic gadgets to ensure this order occurs with high probability.

The first idea is to duplicate edges whose removal should be delayed. We thus make $t$ copies of every edge other than $(b_i, w_{i+1})$ and $(a_i, b_i)$, for $i \in [n]$. The second idea is to speed up the removal of an edge by replacing it with a path; removing a single edge along the path corresponds to removing the original edge. We thus replace $(b_i, w_{i+1})$ and $(a_i, b_i)$, for $i \in [n]$, by paths of length $s$. We show that $(b_i, w_{i+1})$ and $(a_i, b_i)$, for $i \in [n]$, are removed before other edges with high probability when $s$ and $t$ are chosen appropriately. We apply the same idea to ensure that $(b_i, w_{i+1})$ is removed before $(a_i, b_i)$. We make $r$ copies of the path corresponding to $(a_i, b_i)$, and we make the $(b_i, w_{i+1})$-path $r$ times longer.

The resulting graph $G_{n,r,s,t}$ is shown in Figure 4. Every vertex on an edge-path can escape the path with an edge that corresponds to the original choice. In order for the paths to be reset correctly, the costs along the paths increase in increments of size $\epsilon = \frac{1}{rs}$, making it cheaper to escape a path as soon as possible.

**Lower bound for** $\textsc{Random-Facet}^{1P}$**.**

Recall that $\textsc{Random-Facet}^{1P}$ takes as input a set of edges $F$, a tree $B$, and a permutation $\sigma$, and that it initially removes the first edge $e \in F \setminus B$ according to $\sigma$. We interpret $F$ and $B$ as a configuration of the randomized counter as follows. For every $i \in [n]$, let $\mathbf{b}_i^1$ be the set of edges along the $(b_i, w_{i+1})$-path. If $\mathbf{b}_i^1 \not\subseteq F$ then the $i$-th bit is fixed to 0. If $\mathbf{b}_i^1 \subseteq F$ and $\mathbf{b}_i^1 \cap B = \emptyset$ then the $i$-th bit is 0, and otherwise it is 1.

Note that $B$ need not contain all of $\mathbf{b}_i^1$ for the $i$-th bit to be 1. When $i$ is fixed to 0 by removing $e \in \mathbf{b}_i^1$, the resulting optimal tree $B'$ contains the edges in $\mathbf{b}_i^1$ ahead of $e$ but not the edges behind $e$. In [17] and [18] we used stronger gadgets, that cannot be implemented for shortest paths, to ensure that $B'$ included $\mathbf{b}_i^1 \setminus \{e\}$. We observe, however, that the current tree is less important than the set of remaining edges, and this allows us to use the path gadget.

Recall that $\textsc{RandCount}^{1P}$ is a one-permutation variant of the randomized counter, and that it fixes the first available bit according to a given permutation $\hat{\sigma} : [n] \to [n]$. Recall also that, according to Lemma 4.3, $\textsc{RandCount}^{1P}$ and $\textsc{RandCount}$ perform the same expected number of increments when $\hat{\sigma}$ is uniformly random. The permutation $\sigma$ of the edges defines an induced permutation $\hat{\sigma}$ of the bits, where $\hat{\sigma}$ is obtained from $\sigma$ from the first element of $\mathbf{b}_i^1$ for every $i \in [n]$. We show that $\textsc{Random-Facet}^{1P}(F, B, \sigma)$ performs at least as many improving swithces as $\textsc{RandCount}^{1P}(N(F, B), \hat{\sigma})$, where $N(F, B)$ is the set of unfixed 0 bits. To be precise, we assume that $\sigma$ is *well-behaved* and that $B$ has the relevant structure.

A permutation is well-behaved if it ensures that, for all $i \in [n]$, an edge from $\mathbf{b}_i^1$ is removed before $w_i$ is disconnected from $b_{i,1}$, which in turn happens before any multi-edge is exhausted. Concretely, at least one edge from $\mathbf{b}_i^1$ comes before all edges from one of the $r$ copies of the $(a_i, b_i)$-path. Also, at least one edge from each $(a_i, b_i)$-path comes before the last copy of every multi-edge.

The proof is by induction, and the critical case is when an improving switch increments a bit $i$, i.e., when $\mathbf{b}_i^1 \subseteq F$, $\mathbf{b}_i^1 \cap B = \emptyset$, and the chosen edge $e$ is from $\mathbf{b}_i^1$. In fact these are the only improving switches we count in the analysis, and it is the only situation for which the count includes both recursive calls. For all other situations we specify a single recursive call that is considered for the induction step:

1. When $e \in \mathbf{b}_i^1$, $\mathbf{b}_i^1 \subseteq F$, and $\mathbf{b}_i^1 \cap B \neq \emptyset$, the count is for the second call.

2. When $e$ belongs to an $(a_i, b_i)$-path, $\mathbf{b}_i^1 \subseteq F$, and removing $e$ disconnects $w_i$ from $b_{i,1}$, the count is for the second call.

3. In all other cases the count is for first call.

Note that the structure of the graph is preserved by the second recursive call since no edge is removed. Also, due to the assumption that the permutation $\sigma$ is well-behaved, only redundant edges are removed in the third case.

Let us note that the first case is new compared to our analysis in [17] and [18]. It is needed because not all edges from $\mathbf{b}_i^1$ are immediately included in a tree where the $i$-th bit is set to 1. We observe, however, that during the first recursive call the tree is only updated by including edges along the $(b_i, w_{i+1})$-path ahead of the chosen

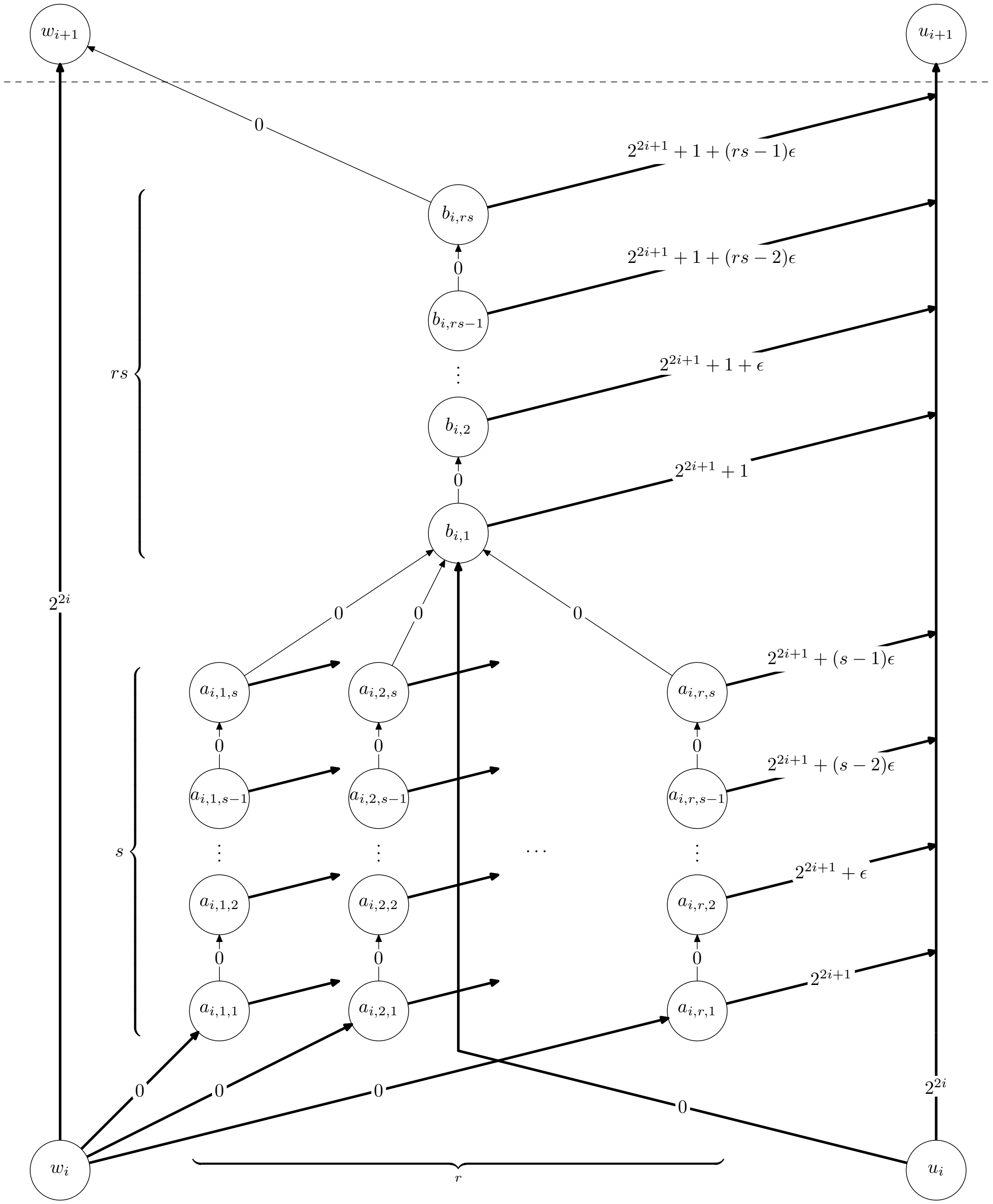

Figure 4: The $i$-th level of the graph $G_{n,r,s,t}$. The bold edges are multi-edges with multiplicity $t$.

edge $e$. The remaining edges are included when the tree is updated in a situation corresponding to the second case, which also causes the lower bits to reset. A similar issue occurs after a reset.

Finally, we show that a uniformly random permutation is well-behaved with high probability when $r = s = t = 3\lceil \log n \rceil$. Since a uniformly random well-behaved permutation gives a uniformly random induced permutation, this shows that RANDOM-FACET$^{1P}$ simulates RANDCOUNT$^{1P}$ with high probability. Thus, when the total number of edges is $M = \tilde{\Theta}(n)$, the expected number of improving switches performed is $2^{\tilde{\Omega}(\sqrt{n})}$, which proves a $2^{\tilde{\Omega}(\sqrt{M})}$ lower bound.

**Lower bound for** Random-Bland**.**

The proof for Random-Bland is similar to the proof for Random-Facet$^{1P}$. Recall that Bland$(F, B, \sigma)$ can remove edges from $F$ that are in $B$, meaning that the invariant $B \subseteq F$ is not maintained. The only guarantee is that the tree $B^\dagger$ returned by the algorithm is optimal for the subgraph defined by $F \cup B^\dagger$. The edges in $B \setminus (F \cup B^\dagger)$ are lost and can only be reintroduced higher up in the recursion.

To use the same approach as for Random-Facet$^{1P}$ we keep track of the edges in $B$ that remain in $B^\dagger$. For this purpose we introduce the notion of *fixed edges*, which are edges that are optimal for $F \cup B$. Note that, since the graph is acyclic, if an edge $e$ is removed and $B'$ is optimal for the resulting graph, then $B'$ remains optimal at every vertex ahead of $e$ when $e$ is reintroduced. In particular, when the $i$-th bit is set to 1, the edges at higher levels become fixed and maintain their configuration while the lower bits are reset and used for additional counting. This shows that the desired edges remain in $B^\dagger$.

Another challenge is to show that edges that should be removed do not remain in $B^\dagger$. Suppose, for instance, that the $i$-th bit was set to 1, and that the lower bits are being reset. If the algorithm picks an edge $e \in \mathbf{b}_j^1$ with $\mathbf{b}_j^1 \subseteq F \cap B$ and $j < i$, then the $j$-th bit should be fixed to 0, but since $e \in B$ this may not be the case. We show however that, when the permutation $\sigma$ is well-behaved, the lower bits are reset before any additional counting is done, such that the edge $e$ is truly removed.

The remainder of the proof is the same as for Random-Facet$^{1P}$, and we again set $r = s = t = 3\lceil \log n \rceil$ to get a $2^{\tilde{\Omega}(\sqrt{M})}$ lower bound, where $M = \tilde{\Theta}(n)$ is the number of edges.

**Lower bound for** Random-Facet**.**

The proof of the lower bound for Random-Facet is more involved than the proofs for Random-Facet$^{1P}$ and Random-Bland. The main complication is that nothing prevents multi-edges from being exhausted. For Random-Facet$^{1P}$ and Random-Bland, as long as one instance of a multi-edge appears sufficiently late in the given permutation, then that multi-edge is not exhausted until after the desired counting behavior is observed. For Random-Facet, on the other hand, the multi-edges are always available for removal, and the edges along $(b_i, w_{i+1})$-paths and $(a_i, b_i)$-paths are often part of the current tree and cannot be removed.

We use a technique by Gärtner [20] to bound the expected number of improving switches by bounding the probabilities that certain *computation paths* are generated by the algorithm. The call Random-Facet$(F, B)$ generates a random, binary computation tree where the nodes of the tree correspond to recursive calls. The tree is defined by the edges from the graph that are picked at the beginning of every recursive call. A computation path is a sequence of successive recursive calls that form a path from the root in a computation tree. It is described by the sequence of chosen edges, and by a sequence of directions that indicate whether the path follows the branch of the first or the second recursive call.

The edges from the graph chosen along a computation path essentially represent a permutation, which allows us to use the same type of arguments as in the proofs for Random-Facet$^{1P}$ and Random-Bland. We restrict our attention to *canonical paths* that correspond to well-behaved permutations. We also restrict our attention to computation paths where at most $\sqrt{n}$ bits are set to 1 by second recursive calls along the path. The probability of generating such shorter paths dominate the probability of generating longer paths. We use a Chernoff bound argument to bound the probability that a multi-edge is exhausted along a computation path, and the fact that we restrict our attention to shorter paths allow us to obtain a better bound.

We show that when $r = \Theta(\log n)$, $s = \Theta(\sqrt{n}\log n)$, and $t = \Theta(\log n)$, the expected number of improving switches performed by the Random-Facet algorithm is at least $2^{\Omega(\sqrt{n})}$. Thus, the number of edges is $M = \tilde{\Theta}(n^{3/2})$, which gives a lower bound of $2^{\tilde{\Omega}(\sqrt[3]{M})}$. Note that we increase the parameter $s$ instead of the parameter $t$ because adding more copies of an edge also makes those copies more likely to be removed.

# 5 The lower bound construction

In this section we formally define the family of weighted directed graphs $G_{n,r,s,t}$, where $n, r, s, t \in \mathbb{N}$, that we use to prove lower bounds for Random-Facet, Random-Facet$^{1P}$, and Random-Bland. The graph $G_{n,r,s,t}$ can be divided into $n$ levels, where the $i$-th level corresponds to the $i$-th bit of an $n$-bit binary counter. Figure 4 gives a schematic description of one level of the graph. The graph contains one additional *target* vertex $\textsc{t}$ which we identify with $u_{n+1}$ and $w_{n+1}$. We are interested in finding the shortest paths from all vertices to $\textsc{t}$.

Formally, the graph $G_{n,r,s,t} = (V, E, c)$, where $c : E \to \mathbb{R}$, has vertex set

$$V \;=\; \{\textsc{t}\} \cup \{u_i, w_i \mid i \in [n]\} \cup \{a_{i,j,k} \mid i \in [n], j \in [r], k \in [s]\} \cup \{b_{i,j} \mid i \in [n], j \in [rs]\} \;,$$

and the edges are specified in Table 1. We use $\epsilon = 1/(rs)$ to define the costs. We also assign a name to every edge. Note that the graph is acyclic, that all costs are non-negative, and that there is an edge with cost 0 that leaves every vertex. Hence, any tree composed of edges whose costs are all 0 is optimal.

| Quantification | Edge | Cost | Name |
|---|---|---|---|
| $i \in [n], j \in [r], k \in [s-1], \ell \in [t]$ | $a_{i,j,k} \to a_{i,j,k+1}$ | $0$ | $a^{1}_{i,j,k}$ |
| | $a_{i,j,k} \to u_{i+1}$ | $2^{2i+1} + (k-1)\epsilon$ | $a^{0,\ell}_{i,j,k}$ |
| $i \in [n], j \in [r], \ell \in [t]$ | $a_{i,j,s} \to b_{i,1}$ | $0$ | $a^{1}_{i,j,s}$ |
| | $a_{i,j,s} \to u_{i+1}$ | $2^{2i+1} + (s-1)\epsilon$ | $a^{0,\ell}_{i,j,s}$ |
| $i \in [n], j \in [rs-1], \ell \in [t]$ | $b_{i,j} \to b_{i,j+1}$ | $0$ | $b^{1}_{i,j}$ |
| | $b_{i,j} \to u_{i+1}$ | $2^{2i+1} + 1 + (j-1)\epsilon$ | $b^{0,\ell}_{i,j}$ |
| $i \in [n], \ell \in [t]$ | $b_{i,rs} \to w_{i+1}$ | $0$ | $b^{1}_{i,rs}$ |
| | $b_{i,rs} \to u_{i+1}$ | $2^{2i+1} + 1 + (rs-1)\epsilon$ | $b^{0,\ell}_{i,rs}$ |
| $i \in [n], \ell \in [t]$ | $u_i \to b_{i,1}$ | $0$ | $u^{1,\ell}_{i}$ |
| | $u_i \to u_{i+1}$ | $2^{2i}$ | $u^{0,\ell}_{i}$ |
| $i \in [n], j \in [r], \ell \in [t]$ | $w_i \to a_{i,j,1}$ | $0$ | $w^{j,\ell}_{i}$ |
| | $w_i \to w_{i+1}$ | $2^{2i}$ | $w^{0,\ell}_{i}$ |

Table 1: Edges, costs, and names.

It will be useful to partition the set of edges into smaller sets of edges with similar roles. We define:

$$\begin{array}{rrcl}
\forall i \in [n], \forall j \in [r] : & \mathbf{a}^{1}_{i,j} & = & \{a^{1}_{i,j,k} \mid k \in [s]\} \\
\forall i \in [n], \forall j \in [r], \forall k \in [s] : & \mathbf{a}^{0}_{i,j,k} & = & \{a^{0,\ell}_{i,j,k} \mid \ell \in [t]\} \\
\forall i \in [n] : & \mathbf{b}^{1}_{i} & = & \{b^{1}_{i,j} \mid j \in [rs]\} \\
\forall i \in [n], \forall j \in [rs] : & \mathbf{b}^{0}_{i,j} & = & \{b^{0,\ell}_{i,j} \mid \ell \in [t]\} \\
\forall i \in [n] : & \mathbf{u}^{1}_{i} & = & \{u^{1,\ell}_{i} \mid \ell \in [t]\} \\
\forall i \in [n] : & \mathbf{u}^{0}_{i} & = & \{u^{0,\ell}_{i} \mid \ell \in [t]\} \\
\forall i \in [n], \forall j \in [r] : & \mathbf{w}^{j}_{i} & = & \{w^{j,\ell}_{i} \mid \ell \in [t]\} \\
\forall i \in [n] : & \mathbf{w}^{0}_{i} & = & \{w^{0,\ell}_{i} \mid \ell \in [t]\}
\end{array}$$

Note that six of the definitions correspond to multi-edges with multiplicity $t$. It will also be helpful to define a set of multi-edges $\mathcal{M}$, i.e., $\mathcal{M}$ is a set of sets of edges:

$$\begin{aligned}
\mathcal{M} \;=\; & \{\mathbf{u}^{1}_{i}, \mathbf{u}^{0}_{i}, \mathbf{w}^{0}_{i} \mid i \in [n]\} \,\cup\, \{\mathbf{b}^{0}_{i,j} \mid i \in [n], j \in [rs]\} \,\cup \\
& \{\mathbf{a}^{0}_{i,j,k} \mid i \in [n], j \in [r], k \in [s]\} \,\cup\, \{\mathbf{w}^{j}_{i} \mid i \in [n], j \in [r]\} \;.
\end{aligned}$$

We are going to interpret certain trees $B \subseteq E$ as configurations of the binary counter. Note that the names of the edges are given superscripts, typically 0 or 1. We refer to edges with superscript 0 as zero-edges, and to the remaining edges as one-edges. Intuitively, the superscripts 0 and 1 describe whether the edges are used in trees for which the corresponding bit is 0 or 1, respectively. Note that by this convention zero-cost edges are referred to as one-edges. Let $last(\mathbf{b}_i^1, F)$ and $last(\mathbf{a}_{i,j}^1, F)$ be the largest index of an edge from $\mathbf{b}_i^1$ and $\mathbf{a}_{i,j}^1$, respectively, that is missing from $F$:

$$\begin{aligned} \forall i \in [n] : \qquad & last(\mathbf{b}_i^1, F) &=& \max(\{0\} \cup \{j \in [rs] \mid b_{i,j}^1 \notin F\}) \\ \forall i \in [n], \forall j \in [r] : \qquad & last(\mathbf{a}_{i,j}^1, F) &=& \max(\{0\} \cup \{k \in [s] \mid a_{i,j,k}^1 \notin F\}) \end{aligned}$$

**Definition 5.1** **($bit_i(F, B)$)** *For every $i \in [n]$, every tree $B$, and every $F \subseteq E$ we say that $bit_i(F, B) = 1$ if:*

- *For all $j \in [rs]$: $b_{i,j}^1 \in B$ iff $j > last(\mathbf{b}_i^1, F)$.*
- *If $\mathbf{b}_i^1 \subseteq F$ then for all $j \in [r]$ and $k \in [s]$: $a_{i,j,k}^1 \in B$ iff $k > last(\mathbf{a}_{i,j}^1, F)$.*
- *If $\mathbf{b}_i^1 \not\subseteq F$ then for all $j \in [r]$: $\mathbf{a}_{i,j}^1 \cap B = \emptyset$.*

*Similarly, we say that $bit_i(F, B) = 0$ if:*

- *$\mathbf{b}_i^1 \cap B = \emptyset$.*
- *For all $j \in [r]$: $\mathbf{a}_{i,j}^1 \cap B = \emptyset$.*

When proving our lower bound, Random-Facet, Random-Facet$^{1P}$, and Random-Bland will be given any initial tree $B_0$ composed entirely of zero-edges:

$$B_0 \;\subseteq\; \{a_{i,j,k}^{0,\ell} \mid i \in [n], j \in [r], k \in [s], \ell \in [t]\} \cup \{b_{i,j}^{0,\ell} \mid i \in [n], j \in [rs], \ell \in [t]\} \cup \{u_i^{0,\ell}, w_i^{0,,\ell} \mid i \in [n], \ell \in [t]\}$$

On the other hand, any tree composed entirely of one-edges is optimal. Hence, for the initial tree all bits are interpreted as being 0, and for the final, optimal tree all bits are interpreted as being 1.

**Definition 5.2 (Functional sets)** *A subset of edges $F \subseteq E$ is said to be* functional *if and only if it contains at least one copy of every multi-edge, i.e., $F \cap \mathbf{e} \neq \emptyset$ for all $\mathbf{e} \in \mathcal{M}$.*

Let $F \subseteq E$ be a subset of edges. It will be convenient to use the following short-hand notation:

$$\mathbf{a}_i^1 \sqsubseteq F \qquad \Longleftrightarrow \qquad \exists j \in [r] : \; \mathbf{a}_{i,j}^1 \subseteq F$$

Note that if $r = 1$ then $\mathbf{a}_i^1 \sqsubseteq F$ has the same meaning as $\mathbf{a}_{i,1}^1 \subseteq F$. When no set $\mathbf{a}_{i,j}^1$, for $j \in [r]$, is completely contained in $F$ we write $\mathbf{a}_i^1 \not\sqsubseteq F$.

**Definition 5.3 (Reset level)** *For every $F \subseteq E$ we define the* reset level *to be:*

$$reset(F) = \max\big(\{0\} \cup \{i \in [n] \mid \mathbf{b}_i^1 \subseteq F \wedge \mathbf{a}_i^1 \not\sqsubseteq F\}\big)$$

As we shall see, all bits with index lower than $reset(F)$, for some functional set $F \subseteq E$, are reset in any optimal tree for the subgraph $G_F$ defined by $F$.

Recall that an edge $e = (u, v)$ is optimal if and only if it satisfies $c(u, v) + \mathrm{y}(v) = \mathrm{y}(u)$, where $\mathrm{y}(u)$ is the optimal value of $u$, and that a tree is optimal if and only if all its edges are optimal. For every functional set $F \subseteq E$ we next define a set of edges $\mathcal{B}_F \subseteq F$ that we later show is optimal for the subgraph $G_F$, such that every tree consisting only of edges from $\mathcal{B}_F$ is optimal. The set of edges $\mathcal{B}_F$ is depicted in figures 5, 6, 7, and 8. For simplicity, the figures show the case when $r = s = t = 1$. Edges not in $\mathcal{B}_F$ are shown as dotted arrows and edges in $\mathcal{B}_F$ are shown as unbroken arrows. Edges that are not in $F$ have been removed; for the general case this should be interpreted as at least one edge missing from the corresponding path.

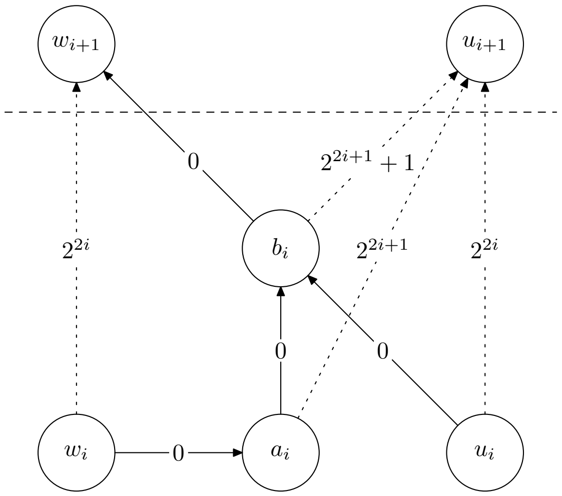

Figure 5: Case $(i)$: $i > reset(F)$ and $\mathbf{b}_i^1 \subseteq F$

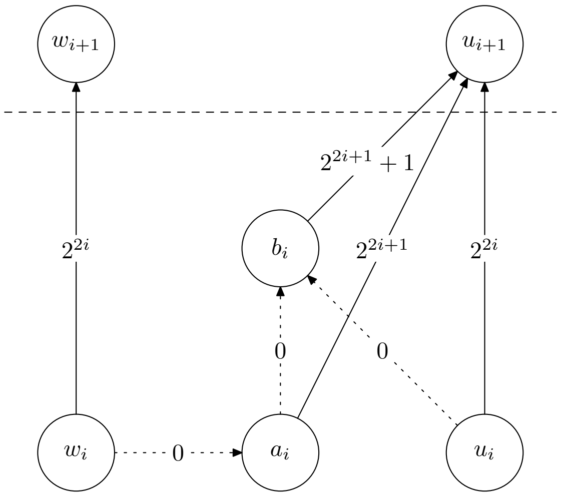

Figure 6: Case $(ii)$: $i > reset(F)$ and $\mathbf{b}_i^1 \not\subseteq F$

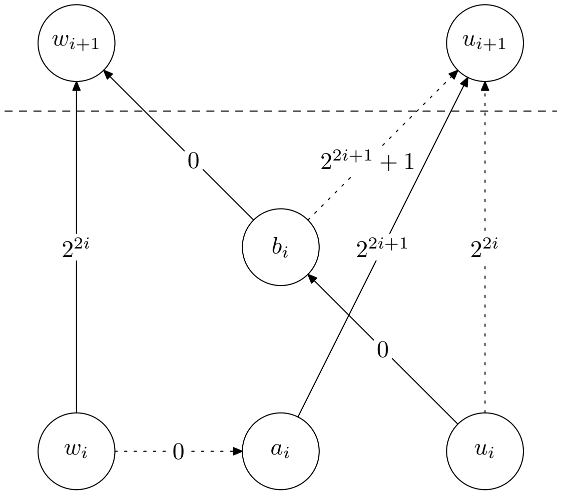

Figure 7: Case $(iii)$: $i = reset(F)$

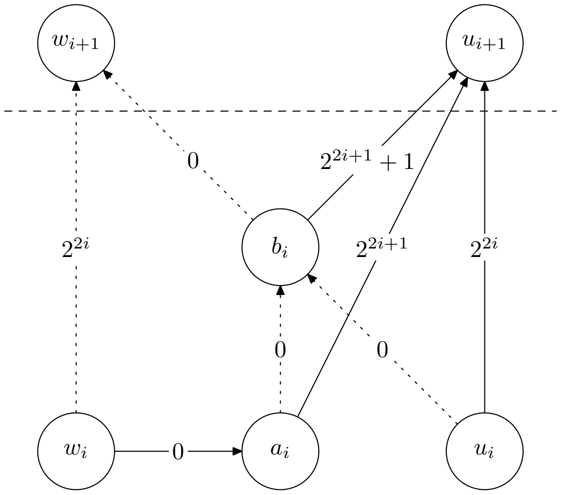

Figure 8: Case $(iv)$: $i < reset(F)$

**Definition 5.4 ($\mathcal{B}_F$)** *Let $F \subseteq E$ be functional. Define $\mathcal{B}_F \subseteq F$ to contain exactly the following edges:*

*(i) For all $i > reset(F)$ where $\mathbf{b}_i^1 \subseteq F$:*

- *For all $j \in [rs]$: $b_{i,j}^1 \in \mathcal{B}_F$.*
- *For all $j \in [r]$ and $k \in [s]$: $a_{i,j,k}^1 \in \mathcal{B}_F$ if $k > last(\mathbf{a}_{i,j}^1, F)$, and $\mathbf{a}_{i,j,k}^0 \cap F \subseteq \mathcal{B}_F$ otherwise.*
- *$\mathbf{u}_i^1 \cap F \subseteq \mathcal{B}_F$.*
- *$\mathbf{w}_i^j \cap F \subseteq \mathcal{B}_F$ for all $j$ with $\mathbf{a}_{i,j}^1 \subseteq F$.*

*(ii) For all $i > reset(F)$ where $\mathbf{b}_i^1 \not\subseteq F$:*

- *For all $j \in [rs]$: $b_{i,j}^1 \in \mathcal{B}_F$ if $j > last(\mathbf{b}_i^1, F)$, and $\mathbf{b}_{i,j}^0 \cap F \subseteq \mathcal{B}_F$ otherwise.*
- *For all $j \in [r]$ and $k \in [s]$: $\mathbf{a}_{i,j,k}^0 \cap F \subseteq \mathcal{B}_F$.*
- *$\mathbf{u}_i^0 \cap F \subseteq \mathcal{B}_F$.*
- *$\mathbf{w}_i^0 \cap F \subseteq \mathcal{B}_F$.*

*(iii) For $i = reset(F)$:*

- *For all $j \in [rs]$: $b^1_{i,j} \in \mathcal{B}_F$.*
- *For all $j \in [r]$ and $k \in [s]$: $a^1_{i,j,k} \in \mathcal{B}_F$ if $k > last(\mathbf{a}^1_{i,j}, F)$, and $\mathbf{a}^0_{i,j,k} \cap F \subseteq \mathcal{B}_F$ otherwise.*
- $\mathbf{u}^1_i \cap F \subseteq \mathcal{B}_F$.
- $\mathbf{w}^0_i \cap F \subseteq \mathcal{B}_F$.

*(iv) For all $i < reset(F)$:*

- *For all $j \in [rs]$: $\mathbf{b}^0_{i,j} \cap F \subseteq \mathcal{B}_F$.*
- *For all $j \in [r]$ and $k \in [s]$: $\mathbf{a}^0_{i,j,k} \cap F \subseteq \mathcal{B}_F$.*
- $\mathbf{u}^0_i \cap F \subseteq \mathcal{B}_F$.
- *$\mathbf{w}^j_i \cap F \subseteq \mathcal{B}_F$ for all $j \in [r]$.*

Note that when $i > reset(F)$ and $\mathbf{b}^1_i \subseteq F$ then we must also have $\mathbf{a}^1_i \sqsubseteq F$. Hence, in case $(i)$ there exists a $j \in [r]$ such that $\mathbf{a}^1_{i,j} \subseteq F$, which means that at least one edge in $\mathcal{B}_F$ is available from $w_i$. It is then not difficult to check that for every vertex $v$ there is at least one edge in $\mathcal{B}_F$ that leaves $v$. Also note that for every tree $B \subseteq \mathcal{B}_F$ we have $bit_i(F, B) = 1$ for all $i \geq reset(F)$, and $bit_i(F, B) = 0$ for all $i < reset(F)$.

**Lemma 5.5** *For every functional set $F \subseteq E$, the set of optimal edges for the subgraph $G_F$ is $\mathcal{B}_F$.*

**Proof:** In the following we let $\mathrm{y}(u)$ be the optimal value of $u$ in the graph $G_F$. Since $G_{n,r,s,t}$ is acyclic then so is $G_F$. The idea of the proof is to find the distances to the terminal vertex $\mathrm{T}$ by backward induction from $\mathrm{T}$. In the process of doing so we show that the edges allowed by $\mathcal{B}_F$ are exactly the optimal choices. Our induction hypothesis is that $\mathrm{y}(w_i) = \mathrm{y}(u_i)$ for all $i > reset(F)$, and that $\mathrm{y}(w_i) = \mathrm{y}(u_i) + 2^{2i}$ for all $i \leq reset(F)$.

We start by making the following two useful observations. We let $\mathcal{B}^*_F$ be the actual set of optimal edges, i.e., we hope to show that $\mathcal{B}_F = \mathcal{B}^*_F$.

**Claim 5.6** *Let $i \in [n]$ be given. If $y(w_{i+1}) > (2^{2i+1} + 1 + (rs-1)\epsilon) + y(u_{i+1})$, then $\mathbf{b}^0_{i,j} \cap F \subseteq \mathcal{B}^*_F$ for all $j \in [rs]$. Furthermore, $y(b_{i,j}) = (2^{2i+1} + 1 + (j-1)\epsilon) + y(u_{i+1})$ for all $j \in [rs]$. In particular, $y(b_{i,1}) = (2^{2i+1} + 1) + y(u_{i+1})$.*

**Claim 5.7** *Let $i \in [n]$ be given. If $y(w_{i+1}) < (2^{2i+1} + 1) + y(u_{i+1})$, then $b^1_{i,j} \in \mathcal{B}^*_F$ for $j > last(\mathbf{b}^1_i, F)$, and $\mathbf{b}^0_{i,j} \cap F \subseteq \mathcal{B}^*_F$ for $j \leq last(\mathbf{b}^1_i, F)$. Furthermore, $y(b_{i,j}) = y(w_{i+1})$ for all $j > last(\mathbf{b}^1_i, F)$, and $y(b_{i,j}) = (2^{2i+1} + 1 + (j-1)\epsilon) + y(u_{i+1})$ for all $j \leq last(\mathbf{b}^1_i, F)$. In particular, if $\mathbf{b}^1_i \subseteq F$ then $y(b_{i,1}) = y(w_{i+1})$, and otherwise $y(b_{i,1}) = (2^{2i+1} + 1) + y(u_{i+1})$.*

To prove the claims observe that the cost of going to $u_{i+1}$ goes up by $\epsilon$ for every step made along the chain of $b_{i,j}$ vertices. Hence, if it is optimal to leave the chain at some vertex $b_{i,j}$, then the same is true for all vertices $b_{i,j'}$ with $j' < j$. In particular, if $\mathrm{y}(w_{i+1}) > (2^{2i} + 1 + (rs-1)\epsilon) + \mathrm{y}(u_{i+1})$ then $b_{i,rs}$, and all other $b_{i,j}$ vertices, should go to $u_{i+1}$. This proves Claim 5.6. The same argument proves Claim 5.7 for the case where $j \leq last(\mathbf{b}^1_i, F)$. When $\mathrm{y}(w_{i+1}) < (2^{2i} + 1) + \mathrm{y}(u_{i+1})$ every $b_{i,j}$ vertex that can reach $w_{i+1}$ would do so with cost 0. This is cheaper than going to $u_{i+1}$, which proves Claim 5.7 for the case where $j > last(\mathbf{b}^1_i, F)$. The following two claims are proved in the same way for the $a_{i,j,k}$ vertices.

**Claim 5.8** *Let $i \in [n]$ and $j \in [r]$ be given. If $y(b_{i,1}) > (2^{2i+1} + (s-1)\epsilon) + y(u_{i+1})$, then $\mathbf{a}^0_{i,j,k} \cap F \subseteq \mathcal{B}^*_F$ for all $k \in [s]$. Furthermore, $y(a_{i,j,k}) = (2^{2i+1} + (k-1)\epsilon) + y(u_{i+1})$ for all $k \in [s]$. In particular, $y(a_{i,j,1}) = 2^{2i+1} + y(u_{i+1})$.*

**Claim 5.9** *Let $i \in [n]$ and $j \in [r]$ be given. If $y(b_{i,1}) < 2^{2i+1} + y(u_{i+1})$, then $a^1_{i,j,k} \in \mathcal{B}^*_F$ for $k > last(\mathbf{a}^1_{i,j}, F)$, and $\mathbf{a}^0_{i,j,k} \cap F \subseteq \mathcal{B}^*_F$ for $k \leq last(\mathbf{a}^1_{i,j}, F)$. Furthermore, $y(a_{i,j,k}) = y(b_{i,1})$ for all $k > last(\mathbf{a}^1_{i,j}, F)$, and $y(a_{i,j,k}) = (2^{2i+1} + (k-1)\epsilon) + y(u_{i+1})$ for all $k \leq last(\mathbf{a}^1_{i,j}, F)$. In particular, if $\mathbf{a}^1_{i,j} \subseteq F$ then $y(a_{i,j,1}) = y(b_{i,1})$, and otherwise $y(a_{i,j,1}) = 2^{2i+1} + y(u_{i+1})$.*

We are now ready to prove the lemma by backward induction in $i$. For $i = n+1$ we have $\mathrm{y}(\mathrm{T}) = \mathrm{y}(w_{n+1}) = \mathrm{y}(u_{i+1})$. We consider four cases, corresponding to the four cases in Definition 5.4. It is not difficult to verify the claims below by studying figures 5, 6, 7, and 8.

**Case *(i)*:** Assume that $i > \mathit{reset}(F)$ and $\mathbf{b}_i^1 \subseteq F$. Using claims 5.7 and 5.9, and the fact that $\mathbf{a}_i^1 \sqsubseteq F$, we see that:

$$\begin{array}{rcll}
\mathrm{y}(b_{i,1}) & = & \mathrm{y}(w_{i+1}) & \\
\mathrm{y}(u_i) & = & \mathrm{y}(b_{i,1}) \ = \ \mathrm{y}(w_{i+1}) & \\
\mathrm{y}(a_{i,j,1}) & = & \mathrm{y}(b_{i,1}) \ = \ \mathrm{y}(w_{i+1}) & \text{for all } j \in [r] \text{ with } \mathbf{a}_{i,j}^1 \subseteq F \\
\mathrm{y}(a_{i,j,1}) & = & 2^{2i+1} + \mathrm{y}(u_{i+1}) & \text{for all } j \in [r] \text{ with } \mathbf{a}_{i,j}^1 \not\subseteq F \\
\mathrm{y}(w_i) & = & \mathrm{y}(a_{i,j,1}) \ = \ \mathrm{y}(b_{i,1}) \ = \ \mathrm{y}(w_{i+1}) & \text{where } \mathbf{a}_{i,j}^1 \subseteq F
\end{array}$$

and that the optimal edges are those specified by $\mathcal{B}_F$. Note that $\mathrm{y}(w_i) = \mathrm{y}(u_i)$, which proves the induction hypothesis.

**Case *(ii)*:** Assume that $i > \mathit{reset}(F)$ and $\mathbf{b}_i^1 \not\subseteq F$. Using claims 5.7 and 5.8 we see that:

$$\begin{array}{rcll}
\mathrm{y}(b_{i,1}) & = & (2^{2i+1} + 1) + \mathrm{y}(u_{i+1}) & \\
\mathrm{y}(u_i) & = & 2^{2i} + \mathrm{y}(u_{i+1}) & \\
\mathrm{y}(a_{i,j,1}) & = & 2^{2i+1} + \mathrm{y}(u_{i+1}) & \text{for all } j \in [r] \\
\mathrm{y}(w_i) & = & 2^{2i} + \mathrm{y}(w_{i+1}) &
\end{array}$$

and that the optimal edges are those specified by $\mathcal{B}_F$. Note that $\mathrm{y}(w_i) = \mathrm{y}(u_i)$, which proves the induction hypothesis.

**Case *(iii)*:** Assume that $i = \mathit{reset}(F)$ such that $\mathbf{b}_i^1 \subseteq F$ and $\mathbf{a}_i^1 \not\sqsubseteq F$. Using claims 5.7 and 5.9 we see that:

$$\begin{array}{rcll}
\mathrm{y}(b_{i,1}) & = & \mathrm{y}(w_{i+1}) & \\
\mathrm{y}(u_i) & = & \mathrm{y}(b_{i,1}) \ = \ \mathrm{y}(w_{i+1}) & \\
\mathrm{y}(a_{i,j,1}) & = & 2^{2i+1} + \mathrm{y}(u_{i+1}) & \text{for all } j \in [r] \\
\mathrm{y}(w_i) & = & 2^{2i} + \mathrm{y}(w_{i+1}) &
\end{array}$$

and that the optimal edges are those specified by $\mathcal{B}_F$. Note that $\mathrm{y}(w_i) = 2^{2i} + \mathrm{y}(u_i)$ as desired.

**Case *(iv)*:** Assume that $i < \mathit{reset}(F)$. Note that by induction we have $\mathrm{y}(w_{i+1}) = 2^{2(i+1)} + \mathrm{y}(u_{i+1})$. Using claims 5.6 and 5.8 we see that:

$$\begin{array}{rcll}
\mathrm{y}(b_{i,1}) & = & (2^{2i+1} + 1) + \mathrm{y}(u_{i+1}) & \\
\mathrm{y}(u_i) & = & 2^{2i} + \mathrm{y}(u_{i+1}) & \\
\mathrm{y}(a_{i,j,1}) & = & 2^{2i+1} + \mathrm{y}(u_{i+1}) & \text{for all } j \in [r] \\
\mathrm{y}(w_i) & = & \mathrm{y}(a_{i,j,1}) \ = \ 2^{2i+1} + \mathrm{y}(u_{i+1}) & \text{for any } j \in [r]
\end{array}$$

and that the optimal edges are those specified by $\mathcal{B}_F$. Note that $\mathrm{y}(w_i) = 2^{2i} + \mathrm{y}(u_i)$, which proves the induction hypothesis.

This completes the proof. □

**Lemma 5.10** *Let $F \subseteq E$ be functional. Then*

*(i) Let $i \in [n]$ and $j \in [rs]$ be given. If $b_{i,j}^1 \notin F$, $b_{i,j'}^1 \in F$ for all $j' > j$, $i \geq \mathit{reset}(F)$, and $B \subseteq \mathcal{B}_F$, then $b_{i,j}^1$ is an improving switch with respect to $B$.*

*(ii) Let $i \in [n]$, $j \in [r]$, and $k \in [s]$ be given. If $a^1_{i,j,k} \notin F$, $a^1_{i,j,k'} \in F$ for all $k' > k$, $i \geq reset(F)$, $\mathbf{b}^1_i \subseteq F$, and $B \subseteq \mathcal{B}_F$, then $a^1_{i,j,k}$ is an improving switch with respect to $B$.*

**Proof:** We first prove $(i)$. Let $F' = F \cup \{b^1_{i,j}\}$. It is not difficult to see that $i \geq reset(F')$. Observe also that $last(\mathbf{b}^1_i, F') < j$. By Definition 5.4 and Lemma 5.5 every tree of $\mathcal{B}_{F'}$ must contain $b^1_{i,j}$. Thus, $B \not\subseteq \mathcal{B}_{F'}$, and there must be an edge in $F'$ which is an improving switch with respect to $B$. Since no edge in $F$ is an improving switch, $b^1_{i,j}$ must be an improving switch with respect to $B$.

The proof of $(ii)$ is similar. Let $F' = F \cup \{a^1_{i,j,k}\}$. Since $\mathbf{b}^1_i \subseteq F$ we again have $i \geq reset(F')$, and $B \not\subseteq \mathcal{B}_{F'}$, and it again follows that $a^1_{i,j,k}$ must be an improving switch with respect to $B$. □

# 6 Lower bound for Random-Facet

Before considering the behavior of the Random-Facet algorithm when applied to the lower bound construction $G_{n,r,s,t}$, we first make some general observations about the algorithm. For this purpose let $G = (V, E, c)$ be any directed weighted graph, and let $B_0$ be some initial tree.

The operation of the Random-Facet algorithm may be described by a binary *computation tree* $T$ as follows. Each node $u$ of the computation tree corresponds to a recursive call Random-Facet$(F(u), B(u))$, where $F(u) \subseteq E$ is a subset of edges assigned to $u$, and $B(u)$ is a tree assigned to $u$. For the root, $u_0$, we have $F(u_0) = E$ and $B(u_0) = B_0$, where $B_0$ is the initial tree. For every node $u$ with $F(u) \neq B(u)$, we assign an edge $e(u) \in F(u) \backslash B(u)$ to $u$. When $F(u) = B(u)$ we write $e(u) = \perp$. The edge $e(u)$ corresponds to the edge picked by the Random-Facet algorithm at the highest level of the recursion for the recursive call Random-Facet$(F(u), B(u))$.

Every node $u$ may have a left child $u_L$ and a right child $u_R$, corresponding to the first and second recursive call of the Random-Facet algorithm, respectively. The left child $u_L$ exists if and only if $F(u) \neq B(u)$, in which case $F(u_L) = F(u) \setminus \{e(u)\}$ and $B(u_L) = B(u)$. Let $u^*$ be the *rightmost* leaf in the subtree of $u$. It is obtained by following a path from $u$ that makes a right turn whenever possible until reaching a leaf. It can also be defined recursively as follows. If $u$ is a leaf then $u^* = u$. If $u_R$ exists, then $u^* = (u_R)^*$. Otherwise, $u^* = (u_L)^*$. Note that $B(u^*)$ is the tree returned by the recursive call of Random-Facet at $u$. The correctness of the algorithm thus implies that $B(u^*)$ is an optimal shortest path tree for the graph defined by $F(u)$. The right child $u_R$ exists if and only if $u_L$ exists and $e(u)$ is an improving switch with respect to $B((u_L)^*)$. If $u_R$ exists, then $F(u_R) = F(u)$ and $B(u_R) = B((u_L)^*)[e(u)]$.

The Random-Facet algorithm is of course a randomized algorithm, so the computation tree it defines is not unique. Instead, the computation tree is a random variable, and Random-Facet defines a probability distribution over computation trees. The random choices made by the algorithm manifest themselves in the edges $u(e)$ assigned to the nodes of the computation tree. Note that every right-edge $(u, u_R)$ in the computation tree corresponds to an improving switch performed by the Random-Facet algorithm. Since we are interested in the expected number of improving switches performed this motivates the following definition. For every computation tree $T$, let $switch(T)$ be the set of nodes $u$ for which there exists a right child $u_R$. Then the expected number of improving switches performed by the Random-Facet algorithm is $\mathbb{E}\left[|switch(T)|\right]$.

Let us note that every element $u$ of $switch(T)$ can be uniquely identified with the path from the root to $u$. Such a path can be described by a sequence of $L$ and $R$ labels. Since $u$ has a right child, the path obtained by appending an additional $R$ label must also appear in $T$. Furthermore, the sets of edges assigned to nodes along the path are uniquely determined by the labels and the encountered edges picked by the Random-Facet algorithm. It will therefore be helpful to include the picked edges in the description of the path. This leads us to the following definition.

**Definition 6.1 (Computation path)** *A* computation path *$P$ is a sequence of pairs $\langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k) \rangle$, where $e_0, e_1, \ldots, e_k \in E$ are distinct edges of the graph $G$, and where $d_0, d_1, \ldots, d_k \in \{L, R\}$. We let $F_0, F_1, \ldots, F_{k+1}$ be the sets of edges assigned to the nodes of the path, such that $F_0 = E$, and for all $\ell \leq k$ we have $F_{\ell+1} = F_\ell \setminus \{e_\ell\}$ if $d_\ell = L$ and $F_{\ell+1} = F_\ell$ otherwise. Furthermore, we let $paths(G)$ be the set of all computation paths for $G$ for which $d_k = R$, i.e., we require the paths of $paths(G)$ to end with a right-edge.*

If a computation path $P$ appears in a computation tree $T$ we write $P \in T$. Furthermore, we let $\mathcal{T}_{E,B_0}$ be the set of all possible computation trees generated by a call to Random-Facet$(E, B_0)$, and $p_{E,B_0}(T)$ be the probability that the computation tree $T$ is generated. Using that every right-edge of a computation tree corresponds uniquely to a path, we get the following useful lemma. This technique was first used by Gärtner [20].

**Lemma 6.2** $\mathbb{E}\left[|switch(T)|\right] = \sum_{P \in paths(G)} \mathbb{P}\left[P \in T\right]$.

**Proof:** Let $\mathbb{1}_{P \in T}$ be an indicator variable that is 1 if $P \in T$ and 0 otherwise. Then we have:

$$\begin{aligned}
\mathbb{E}\left[|switch(T)|\right] &= \sum_{T \in \mathcal{T}_{E,B_0}} p_{E,B_0}(T)\,|switch(T)| \\
&= \sum_{T \in \mathcal{T}_{E,B_0}} p_{E,B_0}(T)\,|\{P \in paths(G) \mid P \in T\}| \\
&= \sum_{P \in paths(G)} \sum_{T \in \mathcal{T}_{E,B_0}} p_{E,B_0}(T)\,\mathbb{1}_{P \in T} \\
&= \sum_{P \in paths(G)} \mathbb{P}\left[P \in T\right] .
\end{aligned}$$

□

We next focus on the behavior of the Random-Facet algorithm when applied to the graph $G = G_{n,r,s,t}$ starting from a tree $B_0$ consisting entirely of zero-edges. $n \in \mathbb{N}$ is the number of bits of the corresponding binary counter. $r, s, t \in \mathbb{N}$ are parameters that will be specified by the analysis to ensure that the expected number of improving switches performed by the Random-Facet algorithm is at least $\mathrm{e}^{\Omega(\sqrt{n})}$. We show that it suffices to use $r = \Theta(\log n)$, $s = \Theta(\sqrt{n}\log n)$, and $t = \Theta(\log n)$. Since the graph $G_{n,r,s,t}$ contains $N = \Theta(nrs) = \Theta(n^{3/2}\log^2 n)$ vertices and $M = \Theta(nrst) = \Theta(n^{3/2}\log^3 n)$ edges, we prove the following theorem.

**Theorem 6.3** *The expected number of switches performed by the* Random-Facet *algorithm, when run on the graph $G_{n,r,s,t}$, where $r = \Theta(\log n)$, $s = \Theta(\sqrt{n}\log n)$, and $t = \Theta(\log n)$, with initial tree $B_0$, is at least $\mathrm{e}^{\Omega(\sqrt{n})} = \mathrm{e}^{\Omega(M^{1/3}/\log M)}$, where $M$ is the number of edges in $G_{n,r,s,t}$.*

First, we need some definitions that allow us to identify interesting events along a computation path $P = \langle(e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k)\rangle$. Note that the edges $e_0, e_1, \ldots, e_k$ are all distinct. For every $e \in E$, we define $\sigma_P(e)$ to be the index of $e$ in $P$, if it appears in $P$, and $\infty$ otherwise:

$$\sigma_P(e) = \begin{cases} \ell & \text{if } e = e_\ell \text{ for some } \ell \in \{0, \ldots, k\} \\ \infty & \text{otherwise} \end{cases}$$

It will again be helpful to work with sets of edges rather than single edges. We therefore define $\sigma_P(\mathbf{b}_i^1)$ to be the first occurrence of an edge from $\mathbf{b}_i^1$ in $P$. We also define $\sigma_P(\mathbf{a}_i^1)$ to be the smallest index for which at least one edge from every set $\mathbf{a}_{i,j}^1$ has appeared in $P$. Note that removing an edge from $\mathbf{b}_i^1$ makes it impossible to reach $w_{i+1}$ from $b_{i,1}$, and, similarly, removing at least one edge from every set $\mathbf{a}_{i,j}^1$, for all $j \in [r]$, makes it impossible to reach $b_{i,1}$ from any vertex $a_{i,j,1}$, for $j \in [r]$. Formally, we define:

$$\begin{aligned}
\forall i \in [n]: \quad \sigma_P(\mathbf{b}_i^1) &= \min_{j \in [rs]} \sigma_P(b_{i,j}^1) \\
\forall i \in [n], \forall j \in [r]: \quad \sigma_P(\mathbf{a}_{i,j}^1) &= \min_{k \in [s]} \sigma_P(a_{i,j,k}^1) \\
\forall i \in [n]: \quad \sigma_P(\mathbf{a}_i^1) &= \max_{j \in [r]} \min_{k \in [s]} \sigma_P(a_{i,j,k}^1)
\end{aligned}$$

Next, we define computation paths for which the random selection of edges is well-behaved.

**Definition 6.4 (Canonical paths)** *Let $n \geq i_1 > i_2 > \cdots > i_p \geq 1$, let $S = \{i_1, i_2, \ldots, i_p\} \subseteq [n]$, let $P = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k) \rangle$ be a computation path, and let $F_0, F_1, \ldots, F_{k+1}$ be the sets of edges assigned to the nodes of the path. We say that $P$ is* $(i_1, i_2, \ldots, i_p)$-canonical *if and only if it satisfies:*

*(i)* $\sigma_P(\mathbf{b}^1_{i_q}) < \sigma_P(\mathbf{b}^1_{i_{q+1}})$ *for all* $q \in [p-1]$.

*(ii)* $\sigma_P(\mathbf{b}^1_i) \leq \sigma_P(\mathbf{a}^1_i)$ *for all* $i \in [n]$.

*(iii)* $F_\ell$ *is functional for all* $\ell \in \{0, \ldots, k+1\}$.

*(iv) For every* $\ell \in \{0, \ldots, k\}$, $d_\ell = R$ *if and only if* $\ell \in \mathcal{R}_{P,S}$, *where:*

$$\mathcal{R}_{P,S} \;:=\; \{\sigma_P(e) \mid e \in \mathbf{b}^1_{i_q}, q \in [p]\} \cup \{\sigma_P(e) \mid e \in \mathbf{a}^1_{i_q,j}, q \in [p], j \in [r], \mathbf{a}^1_{i_q} \not\subseteq F_{\sigma_P(e)} \setminus \{e\}\} \, .$$

*Also,* $\sigma_P(\mathbf{a}^1_{i_p}) = k$ *such that* $d_k = R$.

*We let* $canon_S(G)$ *be the set of* $(i_1, i_2, \ldots, i_p)$*-canonical computation paths.*

Note that in Definition 6.4 (*ii*) weak inequality is used to allow that $\sigma_P(\mathbf{b}^1_i) = \sigma_P(\mathbf{a}^1_i) = \infty$ for some $i$. Since every canonical path ends with a right-edge we have $canon_S(G) \subseteq paths(G)$ for every $S \subseteq [n]$. Also note that $canon_S(G) \cap canon_{S'}(G) = \emptyset$ for $S \neq S'$. Recall that $P \in T$ is the event that the computation path $P$ appears in a random computation tree $T$ generated by the Random-Facet algorithm. Let $p = \lfloor \sqrt{n} \rfloor$. It follows from Lemma 6.2 that:

$$\begin{aligned}
\mathbb{E}\left[|switch(T)|\right] \;&=\; \sum_{P \in paths(G)} \mathbb{P}\left[P \in T\right] \\
&\geq\; \sum_{S \subseteq [n]} \; \sum_{P \in canon_S(G)} \mathbb{P}\left[P \in T\right] \\
&\geq\; \sum_{S \subseteq [n] : |S| = p} \; \sum_{P \in canon_S(G)} \mathbb{P}\left[P \in T\right] \, .
\end{aligned}$$

We use this inequality to prove Theorem 6.3, i.e., we show that:

$$\sum_{S \subseteq [n] : |S| = p} \; \sum_{P \in canon_S(G)} \mathbb{P}\left[P \in T\right] \;=\; \mathrm{e}^{\Omega(\sqrt{n})} \, .$$

The main technical lemma of this section, from which the desired lower bound on the expected number of steps performed by the Random-Facet algorithm easily follows, is the following lemma.

**Lemma 6.5** *For every* $S \subseteq [n]$, *where* $|S| = p = \lfloor \sqrt{n} \rfloor$, *we have*

$$\sum_{P \in canon_S(G)} \mathbb{P}\left[P \in T\right] \;\geq\; \frac{1}{2\,p!} \, .$$

The proof of Lemma 6.5 is quite involved and is the main technical contribution of this section. Before presenting this proof, we show that Lemma 6.5 allows us to obtain the subexponential lower bound we are after.

**Proof of Theorem 6.3:** Let $p = \lfloor \sqrt{n} \rfloor$. We get from Lemma 6.2 and Lemma 6.5 that:

$$\mathbb{E}\left[|switch(T)|\right] \;\geq\; \sum_{S \subseteq [n] : |S| = p} \; \sum_{P \in canon_S(G)} \mathbb{P}\left[P \in T\right] \;\geq\; \sum_{S \subseteq [n] : |S| = p} \frac{1}{2\,|S|!} \;=\; \frac{1}{2}\frac{1}{p!}\binom{n}{p} \, .$$

It follows that $\mathbb{E}\left[|switch(T)|\right] \;\geq\; \mathrm{e}^{\Omega(\sqrt{n})}$ because $\frac{1}{2}\frac{1}{p!}\binom{n}{p} \;\geq\; \mathrm{e}^{(2-o(1))\sqrt{n}}$, as

$$\frac{1}{p!}\binom{n}{p} \;=\; \frac{n!}{(p!)^2(n-p)!} \;\geq\; \frac{(n-p)^p}{(p!)^2} \;=\; \frac{\left(n\left(1-\frac{1}{p}\right)\right)^p}{(p!)^2} \;\sim\; \frac{\frac{n^{\sqrt{n}}}{\mathrm{e}}}{2\pi\sqrt{n}\left(\frac{n}{\mathrm{e}^2}\right)^{\sqrt{n}}} \;=\; \frac{\mathrm{e}^{2\sqrt{n}}}{2\pi\mathrm{e}\sqrt{n}} \, .$$

□

The remainder of this section is devoted to the proof of Lemma 6.5. Throughout the remainder of the section we let $S = \{i_1, i_2, \ldots, i_p\} \subseteq [n]$, where $i_1 > i_2 > \cdots > i_p$, be fixed.

Let $T$ be a computation tree. Observe that in order to check whether $T$ contains an $(i_1, i_2, \ldots, i_p)$-canonical computation path it suffices to follow a single path in $T$ starting from the root. Indeed, Definition 6.4 (*iv*) specifies at every step whether to go to the left child or the right child. More precisely, let $P = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k)\rangle \in T$ be a computation path that is a proper prefix of some $(i_1, i_2, \ldots, i_p)$-canonical path. Let $u(P)$ be the last node on $P$, and let $P' = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k), (e(u(P)), L)\rangle$ be the path obtained by extending $P$ with the edge picked at $u(P)$. The choice to use $L$ here is arbitrary. Then $P$ can only be extended into an $(i_1, i_2, \ldots, i_p)$-canonical path by going to the right child of $u$ if $k + 1 \in \mathcal{R}_{S,P'}$ and to the left child of $u$ otherwise. We define:

$$d(P) \;=\; \begin{cases} R & \text{if } k+1 \in \mathcal{R}_{S,P'} \\ L & \text{otherwise} \end{cases}$$

Hence, we can check whether $T$ contains an $(i_1, i_2, \ldots, i_p)$-canonical path as follows. Starting with the empty path $P$, we repeatedly append the pair $(e(u(P)), d(P))$ to $P$. We stop when either $P$ is $(i_1, i_2, \ldots, i_p)$-canonical; Definition 6.4 (*i*), (*ii*), or (*iii*) show that $P$ can not be extended to an $(i_1, i_2, \ldots, i_p)$-canonical path; or there is no child of $u(P)$ in direction $d(P)$. In fact, Lemma 6.9 below shows that the last case never happens. This procedure is guaranteed to find an $(i_1, i_2, \ldots, i_p)$-canonical path in $T$ if it exists. Note also that $T$ can contain at most one $(i_1, i_2, \ldots, i_p)$-canonical path.

**Definition 6.6 ($P_S(T)$)** *Let $T$ be a computation tree, and let $P$ be the maximal path in $T$ that is a proper prefix of an $(i_1, i_2, \ldots, i_p)$-canonical path. We define $P_S(T)$ to be the path obtained by appending the pair $(e(u(P)), d(P))$ to $P$.*

Note that if $T$ contains an $(i_1, i_2, \ldots, i_p)$-canonical path then $P_S(T)$ is exactly this path. Also note that when $T$ is generated at random then $P_S(T)$ is a random variable. We next define events corresponding to the different ways in which we can fail to find an $(i_1, i_2, \ldots, i_p)$-canonical path in $T$. These events correspond directly to the requirements in Definition 6.4.

**Definition 6.7 (Bad events)** *Let $T$ be generated at random by the* Random-Facet *algorithm. Suppose that $P_S(T) = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k)\rangle$, and that $F_0, F_1, \ldots, F_{k+1}$ are the sets of edges assigned to the nodes of the path.*

- (*i*) *For every $q \in [p-1]$, define $Bad_1(q)$ to be the event that $\sigma_{P_S(T)}(\mathbf{b}^1_{i_q}) = \infty$ and $\sigma_{P_S(T)}(\mathbf{b}^1_{i_{q'}}) = k$ for some $q' > q$. Also define $Bad_1 := \bigcup_{q \in [p-1]} Bad_1(q)$.*
- (*ii*) *For every $i \in [n]$, define $Bad_2(i)$ to be the event that $\sigma_{P_S(T)}(\mathbf{b}^1_i) = \infty$ and $\sigma_{P_S(T)}(\mathbf{a}^1_i) = k$. Also define $Bad_2 := \bigcup_{i \in [n]} Bad_2(i)$.*
- (*iii*) *For every $\mathbf{e} \in \mathcal{M}$, define $Bad_3(\mathbf{e})$ to be the event that $F_{k+1}$ is not functional because $F_{k+1} \cap \mathbf{e} = \emptyset$. Also define $Bad_3 := \bigcup_{\mathbf{e} \in \mathcal{M}} Bad_3(\mathbf{e})$.*

*We let $Good_1$, $Good_2$, and $Good_3$ be the complements of $Bad_1$, $Bad_2$, and $Bad_3$ respectively.*

Note that the events $Bad_2$ and $Bad_3$ are disjoint since $Bad_2$ only occurs when $e_k \in \mathbf{a}^1_{i,j}$ for some $i$ and $j$, and $Bad_3$ only occurs when $e_k \in \mathbf{e}$ for some $\mathbf{e} \in \mathcal{M}$.

We now make the first step towards proving Lemma 6.5. Recall that $\sum_{P \in canon_S(G)} \mathbb{P}\left[P \in T\right]$ is the probability that the computation tree $T$ generated by Random-Facet$(E, B_0)$ contains an $(i_1, i_2, \ldots, i_p)$-canonical path.

Following the above discussion we know that this happens if and only if none of the events $Bad_1$, $Bad_2$, or $Bad_3$ occur. Since $Bad_2$ and $Bad_3$ are disjoint we get:

$$\begin{aligned}\sum_{P \in canon_S(G)} \mathbb{P}[P \in T] &= \mathbb{P}[Good_1 \wedge Good_2 \wedge Good_3] \\ &= \mathbb{P}[Good_1]\, \mathbb{P}[Good_2 \wedge Good_3 \mid Good_1] \\ &= \mathbb{P}[Good_1]\, (1 - \mathbb{P}[Bad_2 \mid Good_1] - \mathbb{P}[Bad_3 \mid Good_1])\end{aligned}$$

Before estimating $\mathbb{P}[Good_1]$, $\mathbb{P}[Bad_2 \mid Good_1]$, and $\mathbb{P}[Bad_3 \mid Good_1]$, we start by proving a couple of lemmas that describe the trees assigned to nodes along the path $P_S(T)$, and show that we always have $P_S(T) \in T$. Recall that $S = \{i_1, i_2, \ldots, i_p\}$.

**Lemma 6.8** *Let $T$ be a computation tree, let*

$$\begin{aligned} P &= P_S(T) = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k) \rangle , \\ P' &= \langle (e_0, d_0), (e_1, d_1), \ldots, (e_{k-1}, d_{k-1}) \rangle , \end{aligned}$$

*and let $F_0, F_1, \ldots, F_k$ be the sets of edges assigned to nodes on $P'$. Then:*

- *$(i)$ For all $q \in [p]$ and $\ell \in \{0, \ldots, k\}$: $\mathbf{b}^1_{i_q} \subseteq F_\ell$ and $\mathbf{a}^1_{i_q} \sqsubseteq F_\ell$.*
- *$(ii)$ For all $\ell \in \{0, \ldots, k\}$: $reset(F_\ell) = 0$.*

**Proof:** $(i)$ follows from Definition 6.4 $(iv)$, i.e., every time a right-edge is used on $P$ no edge is removed, and right-edges are used exactly in a way that ensures $(i)$.

We next prove $(ii)$. Let $\ell \in \{0, \ldots, k\}$ be given. For all $i \notin S$, Definition 6.4 $(ii)$ shows that $\sigma_P(\mathbf{b}^1_i) \leq \sigma_P(\mathbf{a}^1_i)$. In particular, if $\mathbf{a}^1_i \not\sqsubseteq F_\ell$ then we also have $\mathbf{b}^1_i \not\subseteq F_\ell$. When this is combined with $(i)$ we see that $reset(F_\ell) = 0$. □

**Lemma 6.9** *Let $T$ be a computation tree, let*

$$\begin{aligned} P &= P_S(T) = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k) \rangle; , \\ P' &= \langle (e_0, d_0), (e_1, d_1), \ldots, (e_{k-1}, d_{k-1}) \rangle , \end{aligned}$$

*and let $u_0, u_1, \ldots, u_k$ be the nodes on the path $P'$. Define $i_0 := n+1$, $\sigma_P(\mathbf{a}^1_{i_0}) := -1$, and $\sigma_P(\mathbf{b}^1_{i_{p+1}}) := \infty$. Then:*

- *$(i)$ For all $q \in [p]$: $\sigma_{P'}(\mathbf{b}^1_{i_q}) \leq \sigma_{P'}(\mathbf{a}^1_{i_q}) \leq \sigma_{P'}(\mathbf{b}^1_{i_{q+1}})$.*
- *$(ii)$ For all $q \in [p]$ and all $\sigma_{P'}(\mathbf{a}^1_{i_{q-1}}) < \ell \leq \sigma_{P'}(\mathbf{b}^1_{i_q})$ we have:*
  - *For all $i > i_{q-1}$: $bit_i(F(u_\ell), B(u_\ell)) = 1$.*
  - *$\mathbf{b}^1_{i_{q-1}} \subseteq B(u_\ell)$.*
  - *For all $i < i_{q-1}$: $bit_i(F(u_\ell), B(u_\ell)) = 0$.*

  *We refer to the interval $\sigma_P(\mathbf{a}^1_{i_{q-1}}) < \ell \leq \sigma_P(\mathbf{b}^1_{i_q})$ as the $\mathbf{b}_q$-phase.*
- *$(iii)$ For all $q \in [p]$ and all $\sigma_{P'}(\mathbf{b}^1_{i_q}) < \ell \leq \sigma_{P'}(\mathbf{a}^1_{i_q})$ we have:*
  - *For all $i \neq i_q$: $bit_i(F(u_\ell), B(u_\ell)) = 1$.*
  - *$\mathbf{a}^1_{i_q,j} \cap B(u_\ell) = \emptyset$ for all $j \in [r]$.*

  *We refer to the interval $\sigma_P(\mathbf{b}^1_{i_q}) < \ell \leq \sigma_P(\mathbf{a}^1_{i_q})$ as the $\mathbf{a}_q$-phase.*
- *$(iv)$ $P \in T$.*

**Proof:** We start by partitioning the indices $\ell \in \{0, \dots, k\}$ into phases. We later show that the phases alternate as described by the lemma. Let $\mathcal{R} = \{\sigma_P(\mathbf{a}^1_{i_q}), \sigma_P(\mathbf{b}^1_{i_q}) \mid q \in [p]\}$ be the set of critical moments when we transition from one phase to another. We say that $\ell$ is in the $\mathbf{b}_q$-phase if the largest element of $\mathcal{R}$ smaller than $\ell$ is $\sigma_P(\mathbf{a}^1_{i_{q-1}})$, i.e., we later prove that this phase leads to an edge from $\mathbf{b}^1_{i_q}$ being picked. If no element of $\mathcal{R}$ is smaller than $\ell$ we say that $\ell$ is in the $\mathbf{b}_1$-phase. Similarly, we say that $\ell$ is in the $\mathbf{a}_q$-phase if the largest element of $\mathcal{R}$ smaller than $\ell$ is $\sigma_P(\mathbf{b}^1_{i_q})$.

We prove $(i)$, $(ii)$, and $(iii)$ jointly by induction in $\ell$. For $\ell = 0$, $\ell$ is in the $\mathbf{b}_1$-phase, $F(u_0) = E$, and $B(u_0) = B_0$, a tree consisting entirely of zero-edges. Hence, $(ii)$ is satisfied for $\ell = 0$. Assume that the lemma is satisfied for some given $\ell < k$. To prove that the lemma also holds for $\ell + 1$ we consider five cases:

1. $d_\ell = L$.
2. $d_\ell = R$, $e_\ell \in \mathbf{b}^1_{i_{q'}}$, and $\ell$ is in the $\mathbf{b}_q$-phase, for some $q, q' \in [p]$.
3. $d_\ell = R$, $e_\ell \in \mathbf{a}^1_{i_{q'},j}$, and $\ell$ is in the $\mathbf{b}_q$-phase, for some $q, q' \in [p]$ and $j \in [r]$.
4. $d_\ell = R$, $e_\ell \in \mathbf{b}^1_{i_{q'}}$, and $\ell$ is in the $\mathbf{a}_q$-phase, for some $q, q' \in [p]$.
5. $d_\ell = R$, $e_\ell \in \mathbf{a}^1_{i_{q'},j}$, and $\ell$ is in the $\mathbf{a}_q$-phase, for some $q, q' \in [p]$ and $j \in [r]$.

**Case 1:** Assume that $d_\ell = L$. Recall from Definition 6.4 that $d_\ell = R$ if and only if $\ell \in \mathcal{R}_{P,S}$ where $\mathcal{R} \subseteq \mathcal{R}_{P,S}$. For the case when $d_\ell = L$ we therefore have $\ell \notin \mathcal{R}$, which means that the phase did not change from $\ell$ to $\ell+1$. We also have $B(u_{\ell+1}) = B(u_\ell)$. Recall that $F(u_{\ell+1}) = F(u_\ell) \setminus \{e_\ell\}$. Since $e_\ell \notin B(u_\ell)$ we have $\mathit{bit}_i(F(u_\ell) \setminus \{e_\ell\}, B(u_\ell)) = 1$ if $\mathit{bit}_i(F(u_\ell), B(u_\ell)) = 1$, and $\mathit{bit}_i(F(u_\ell) \setminus \{e_\ell\}, B(u_\ell)) = 0$ if $\mathit{bit}_i(F(u_\ell), B(u_\ell)) = 0$. This completes the induction step.

**Case 2:** Let $q, q' \in [p]$, and assume that $d_\ell = R$, that $e_\ell \in \mathbf{b}^1_{i_{q'}}$, and that $\ell$ is in the $\mathbf{b}_q$-phase. For all $q'' < q - 1$ we have $\mathit{bit}_{i_{q''}}(F(u_\ell), B(u_\ell)) = 1$, and since Lemma 6.8 shows that $\mathbf{b}^1_{i_{q''}} \subseteq F(u_\ell)$, it follows that $e_\ell \notin \mathbf{b}^1_{i_{q''}}$. Since $\mathbf{b}^1_{i_{q-1}} \subseteq B(u_\ell)$ it must then be the case that $q' \geq q$. Since $P'$ satisfies Definition 6.4 $(i)$ it follows that $q' = q$. Hence, $\ell = \sigma_P(\mathbf{b}^1_{i_q})$, and we transition to the $\mathbf{a}_q$-phase.

Let $B = B(((u_\ell)_L)^*)$ be the tree returned by the first recursive call of the Random-Facet algorithm at $u_\ell$. Since $P'$ satisfies Definition 6.4 $(iii)$, $F(u_\ell) \setminus \{e_\ell\}$ is functional, and Lemma 5.5 shows that $B \subseteq \mathcal{B}_{F(u_\ell)\setminus\{e_\ell\}}$. From Lemma 6.8 we know that $\mathit{reset}(F(u_\ell) \setminus \{e_\ell\}) = 0$. Lemma 5.10 then shows that $e_\ell$ is an improving switch w.r.t. $B$. Moreover, since $\mathit{reset}(F(u_\ell) \setminus \{e_\ell\}) = 0$ we get from Definition 5.4 that $B[e_\ell]$ has the form described in $(iii)$. Note in particular that $\mathbf{b}^1_{i_q} \not\subseteq F(u_\ell) \setminus \{e_\ell\}$.

**Case 3:** Let $q, q' \in [p]$ and $j \in [r]$, and assume that $d_\ell = R$, that $e_\ell \in \mathbf{a}^1_{i_{q'},j}$, and that $\ell$ is in the $\mathbf{b}_q$-phase. From the definition of $\mathcal{R}_{P,S}$ it follows that $\mathbf{a}^1_{i_{q'}} \not\sqsubseteq F(u_\ell) \setminus \{e_\ell\}$. We first prove that $q' = q - 1$, which implies that $\ell + 1$ is also in the $\mathbf{b}_q$-phase. For $q'' < q - 1$ we have $\mathit{bit}_{i_{q''}}(F(u_\ell), B(u_\ell)) = 1$, and Lemma 6.8 shows that $\mathbf{a}^1_{i_{q''}} \sqsubseteq F(u_\ell)$. Since $P'$ satisfies Definition 6.4 $(ii)$ it then follows that $q' = q - 1$. Hence, we have $\mathbf{a}^1_{i_{q-1}} \not\sqsubseteq F(u_\ell) \setminus \{e_\ell\}$ and since $\mathbf{b}^1_{i_{q-1}} \subseteq B(u_\ell)$ we get $\mathit{reset}(F(u_\ell) \setminus \{e_\ell\}) = i_{q-1}$.

The remainder of the proof is similar to the last half of the proof of case 2. Let $B = B(((u_\ell)_L)^*)$ be the tree returned by the first recursive call of the Random-Facet algorithm at $u_\ell$. Since $P'$ satisfies Definition 6.4 $(iii)$, $F(u_\ell) \setminus \{e_\ell\}$ is functional, and Lemma 5.5 shows that $B \subseteq \mathcal{B}_{F(u_\ell)\setminus\{e_\ell\}}$. Since $\mathit{reset}(F(u_\ell) \setminus \{e_\ell\}) = i_{q-1}$, Lemma 5.10 shows that $e_\ell$ is an improving switch w.r.t. $B$. Moreover, we get from Definition 5.4 that $B[e_\ell]$ has the form described in $(ii)$.

**Case 4:** Let $q, q' \in [p]$, and assume that $d_\ell = R$, that $e_\ell \in \mathbf{b}^1_{i_{q'}}$, and that $\ell$ is in the $\mathbf{a}_q$-phase. Since for all $q'' \neq q$ we have $\mathit{bit}_{i_{q''}}(F(u_\ell), B(u_\ell)) = 1$ and, by Lemma 6.8, $\mathbf{b}^1_{i_{q''}} \subseteq F(u_\ell)$ such that $\mathbf{b}^1_{i_{q''}} \subseteq B(u_\ell)$, we must have $q' = q$. In particular, $\ell + 1$ is also in the $\mathbf{a}_q$-phase. The remainder of the proof is the same as the last half of the proof of case 2.

**Case 5:** Let $q, q' \in [p]$ and $j \in [r]$, and assume that $d_\ell = R$, that $e_\ell \in \mathbf{a}^1_{i_{q'},j}$, and that $\ell$ is in the $\mathbf{a}_q$-phase. Since for all $q'' \neq q$ we have $\mathit{bit}_{i_{q''}}(F(u_\ell), B(u_\ell)) = 1$ and, by Lemma 6.8, $\mathbf{a}^1_{i_{q''}} \sqsubseteq F(u_\ell)$ such that $\mathbf{a}^1_{i_{q''}} \sqsubseteq B(u_\ell)$, we must have $q' = q$. In particular, $\ell = \sigma_P(\mathbf{a}^1_{i_q})$ such that $\ell + 1$ is in the $\mathbf{b}_{q+1}$-phase. The remainder of the proof is the same as the last half of the proof of case 3.

It remains to prove $(iv)$: $P \in T$. Per definition we have $P' \in T$, so it suffices to show that $u_k$ has a child in direction $d_k$. For $d_k = L$ we must show that there is a vertex from which more than one edge is available in $F(u_\ell)$. This follows, for instance, from the fact that $F(u_\ell)$ is functional such that any $w_i$ vertex, for $i \in [n]$, has at least two out-going edges in $F(u_\ell)$. When $d_k = R$ we proved in cases 2 to 5 that $e_\ell$ is an improving switch with respect to $B = B(((u_\ell)_L)^*)$, which shows that $u_k$ has a right child.

□

**Lemma 6.10** $\Pr[\mathit{Good}_1] \geq \frac{1}{p!}$.

**Proof:** Let $\mathit{Good}_1(q)$ be the complement of the event $\mathit{Bad}_1(q)$ for every $q \in [p-1]$. Observe first that:

$$\mathbb{P}\left[\mathit{Good}_1\right] \;=\; \prod_{q=1}^{p-1} \mathbb{P}\left[\mathit{Good}_1(q) \mid \mathit{Good}_1(1), \ldots, \mathit{Good}_1(q-1)\right] \;.$$

We will prove that $\Pr[\mathit{Good}_1(q) \mid \mathit{Good}_1(1), \ldots, \mathit{Good}_1(q-1)] \geq \frac{1}{p-q+1}$ for every $q \in [p-1]$, from which it follows that:

$$\mathbb{P}\left[\mathit{Good}_1\right] \;\geq\; \prod_{q=1}^{p-1} \frac{1}{p-q+1} \;=\; \frac{1}{p!} \;.$$

Let $q \in [p-1]$ be given, let $T$ be a computation tree generated by Random-Facet$(E, B_0)$, and let $u_0, \ldots, u_{k+1}$ be the nodes on the path $P_S(T) = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k) \rangle$. Let $\ell$ be the index of the first occurence of an edge picked from $\mathbf{b}^1_{i_{q'}}$ along $P_S(T)$ for some $q' \geq q$, i.e., $\ell = \min\{\sigma_{P_S(T)}(\mathbf{b}^1_{i_{q'}}) \mid q' \geq q\}$. If no such edge was picked along $P_S(T)$ then we get the event $\mathit{Good}_1(q)$, and we may therefore assume that $\ell$ exists. The index $\ell$ must be part of a $\mathbf{b}$-phase, since, according to Lemma 6.9, the edge $e_\ell$ would otherwise be part of the current policy $B(u_\ell)$. In fact, since we condition on $\mathit{Good}_1(1), \ldots, \mathit{Good}_1(q-1)$, $\ell$ must be part of the $\mathbf{b}_q$-phase. From Lemma 6.9 we then know that $\mathbf{b}^1_{i_{q'}} \cap B(u_\ell) = \emptyset$ for all $q' \geq q$. Since the sets $\mathbf{b}^1_{i_{q'}}$ all have the same size, the probability that the edge picked at $u_\ell$ was from $\mathbf{b}^1_q$ is $\frac{1}{p-q+1}$. In this case we again get the event $\mathit{Good}_1(q)$. □

**Lemma 6.11** *For every $i \in [n]$ we have*

$$\Pr[\mathit{Bad}_2(i) \mid \mathit{Good}_1] \;\leq\; \frac{(r!)^2}{(2r)!} \;<\; 2^{-r} \;.$$

**Proof:** Let $T$ be a randomly generated computation tree, let $P = P_S(T) = \langle (e_1, d_1), (e_2, d_2), \ldots, (e_k, d_k) \rangle$, and let $u_0, \ldots, u_{k+1}$ be the nodes on $P$ in $T$.

The event $\mathit{Bad}_2(i)$ occurs if $\sigma_P(\mathbf{a}^1_{i,j}) \leq k$, for every $j \in [r]$, and $\sigma_P(\mathbf{b}^1_i) = \infty$. Hence, when $\mathit{Bad}_2(i)$ occurs we may assume that there exists indices $\ell_j = \sigma_P(\mathbf{a}^1_{i,j}) \leq k$ for all $j \in [r]$. For all $\ell_j$ we have $\mathbf{b}^1_i, \mathbf{a}^1_{i,j} \subseteq F(u_{\ell_j})$. Hence, Lemma 6.9 shows that the index $\ell_j$ must be part of a $\mathbf{b}$-phase, since the edge $e_{\ell_j}$ would otherwise be part of the current tree $B(u_{\ell_j})$. It follows from Lemma 6.9 that $\mathbf{b}^1_i \cap B(u_{\ell_j}) = \emptyset$ for all $j \in [r]$. Since $|\mathbf{b}^1_i| = rs$ and $|\mathbf{a}^1_{i,j}| = s$, for all $j \in [r]$, it follows that the $w$-th time, for $w \in [r]$, we pick an edge from a new set $\mathbf{a}^1_{i,j}$ without picking an edge from $\mathbf{b}^1_i$, this happens with probability at most $\frac{(r-w+1)s}{rs+(r-w+1)s} = \frac{r-w+1}{2r-w+1}$. Note that conditioning on $\mathit{Good}_1$ does not affect this probability. Hence,

$$\Pr[\mathit{Bad}_2(i) \mid \mathit{Good}_1] \;\leq\; \prod_{w=1}^{r} \frac{r-w+1}{2r-w+1} \;=\; \prod_{w=1}^{r} \frac{w}{r+w} \;=\; \frac{(r!)^2}{(2r)!} \;<\; 2^{-r} \;.$$

The last inequality follows from a simple proof by induction. □

Recall that $|S| = p$.

**Lemma 6.12** *For every multi-edge* $\mathbf{e} \in \mathcal{M}$*, we have* $\Pr[\mathit{Bad}_3(\mathbf{e}) \mid \mathit{Good}_1] \le n^{-2}$*, assuming* $s = 2p(r+1)+t$ *and* $t = 15\lceil \log n \rceil = \Theta(\log n)$*.*

**Proof:** Let $T$ be a randomly generated computation tree, let $P = P_S(T) = \langle (e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k) \rangle$, and let $u_0, \ldots, u_{k+1}$ be the nodes on $P$ in $T$.

Define the set of sets of edges:

$$\mathcal{A} \;:=\; \{\mathbf{a}^1_{i_q,j} \mid q \in [p], j \in [r]\} \cup \{\mathbf{b}^1_{i_q} \mid q \in [p]\}\;.$$

Observe that if an edge from each of the sets in $\mathcal{A}$ is picked along $P$, then $P$ is $(i_1, i_2, \ldots, i_p)$-canonical and we do not get the event $\mathit{Bad}_3(\mathbf{e})$. Observe also that $|\mathcal{A}| = p(r+1)$, and that each of the sets in $\mathcal{A}$ contains at least $s$ edges. Furthermore, Lemma 6.9 shows that for every index $\ell \in \{0, \ldots, k\}$ there exists a set $A \in \mathcal{A}$ such that $\sigma_P(A) \ge \ell$, $A \subseteq F(u_\ell)$, and $A \cap B(u_\ell) = \emptyset$. Indeed, in every $\mathbf{b}_q$-phase the set $\mathbf{b}^1_{i_q}$ has this property, and in every $\mathbf{a}_q$-phase there exists some $\mathbf{a}^1_{i_q,j}$, for $j \in [r]$, with this property. Hence, there is always a set in $\mathcal{A}$ from which no edge has previously been picked and for which all edges are available to be picked.

The event $\mathit{Bad}_3(\mathbf{e})$ occurs only if we pick $t$ edges from $\mathbf{e}$ before picking edges $p(r+1)$ times from new sets from $\mathcal{A}$. Every time we either pick an edge from $\mathbf{e}$ or from a new set from $\mathcal{A}$, the edge is picked from $\mathbf{e}$ with probability at most $t/(s+t)$. Note that conditioning on $\mathit{Good}_1$ does not affect this probability. Let $X$ be the sum of $p(r+1)+t$ Bernoulli trials, each with a 'success' probability of $p = t/(s+t)$. $\mathit{Bad}_3(\mathbf{e})$ occurs only if $X \ge t$. We now apply Chernoff. $\mu = \mathbb{E}[X] = \frac{t}{s+t}(p(r+1)+t)$. As $s = 2p(r+1)+t$, we then have $\mu = \frac{t}{2}$. Then,

$$\Pr[X \ge t] \;=\; \Pr[X \ge 2\mu] \;\le\; \left(\frac{\mathrm{e}}{4}\right)^{\mu} \;=\; \left(\frac{\mathrm{e}}{4}\right)^{t/2} \;<\; n^{-2}.$$

It is possible that there are fewer than $p(r+1)+t$ trials, which only lowers the probability. □

We next prove Lemma 6.5 by showing that:

$$\mathbb{P}[\mathit{Good}_1]\,(1 - \mathbb{P}[\mathit{Bad}_2 \mid \mathit{Good}_1] - \mathbb{P}[\mathit{Bad}_3 \mid \mathit{Good}_1]) \;\ge\; \frac{1}{2\,p!}\;,$$

where $p = \lfloor \sqrt{n} \rfloor$. In order for the proof to work we pick $r = \log(4n) = \Theta(\log n)$, $t = 15\lceil \log n \rceil = \Theta(\log n)$, and, to satisfy the assumption for Lemma 6.12, $s = 2p(r+1)+t = \Theta(\sqrt{n}\log n)$.

**Proof of Lemma 6.5:** Observe first that the events $\mathit{Bad}_2(i)$ and $\mathit{Bad}_2(i')$ are disjoint for $i \ne i'$. Using Lemma 6.11, it follows that:

$$\mathbb{P}[\mathit{Bad}_2 \mid \mathit{Good}_1] \;=\; \sum_{i \in [n]} \mathbb{P}[\mathit{Bad}_2(i) \mid \mathit{Good}_1] \;\le\; n\,2^{-r} \le \frac{1}{4}\;.$$

Similarly, the events $\mathit{Bad}_3(\mathbf{e})$ and $\mathit{Bad}_3(\mathbf{e}')$ are disjoint for $\mathbf{e} \ne \mathbf{e}'$. The number of multi-edges is $|\mathcal{M}| = n(2rs + r + 3) = O(n^{3/2}\log^2 n)$. We, thus, get from Lemma 6.12 that:

$$\mathbb{P}[\mathit{Bad}_3 \mid \mathit{Good}_1] \;=\; \sum_{\mathbf{e} \in \mathcal{M}} \mathbb{P}[\mathit{Bad}_3(\mathbf{e}) \mid \mathit{Good}_1] \;\le\; |\mathcal{M}|\,n^{-2} \le \frac{1}{4}\;.$$

Using Lemma 6.10 it follows that:

$$\begin{aligned}\sum_{P \in \mathit{canon}_S(G)} \mathbb{P}[P \in T] \;&=\; \mathbb{P}[\mathit{Good}_1]\,(1 - \mathbb{P}[\mathit{Bad}_2 \mid \mathit{Good}_1] - \mathbb{P}[\mathit{Bad}_3 \mid \mathit{Good}_1]) \\ &\ge\; \frac{1}{p!}\left(1 - \frac{1}{4} - \frac{1}{4}\right) \;=\; \frac{1}{2\,p!}\;.\end{aligned}$$

□

# 7 Lower bound for Random-Facet$^{1P}$

We next prove a lower bound for the Random-Facet$^{1P}$ algorithm. Recall that Random-Facet$^{1P}$ receives, as a third argument, a *permutation function* $\sigma : E \to \mathbb{N}$ that assigns to each edge of $E$ a *distinct* natural number. See Figure 1 for a precise description of this variant of the Random-Facet algorithm. Instead of choosing a *random* edge $e$ from $F \setminus B$, Random-Facet$^{1P}(F, B, \sigma)$ chooses the edge $e \in F \setminus \sigma$ with the *smallest permutation index* $\sigma(e)$.

Let $G_{n,r,s,t} = (V, E, c)$ be defined as in Section 5. $n \in \mathbb{N}$ is the number of bits of the corresponding binary counter, and $r, s, t \in \mathbb{N}$ are parameters that will be specified by the analysis to ensure that the expected number of improving switches performed by the Random-Facet$^{1P}$ algorithm is at least $\mathrm{e}^{\Omega(\sqrt{n})}$. We show that it suffices to use $r = s = t = \Theta(\log n)$. Since the graph $G_{n,r,s,t}$ contains $N = \Theta(nrs) = \Theta(n \log^2 n)$ vertices and $M = \Theta(nrst) = \Theta(n \log^3 n)$ edges, we prove the following theorem.

**Theorem 7.1** *The expected number of switches performed by the* Random-Facet$^{1P}$ *algorithm, when run on the graph $G_{n,r,s,t}$, where $r = s = t = \Theta(\log n)$, with initial tree $B_0$, is at least $\mathrm{e}^{\Omega(\sqrt{n})} = \mathrm{e}^{\Omega(\sqrt{M}/\log^{3/2} M)}$, where $M$ is the number of edges in $G_{n,r,s,t}$.*

We let $\Sigma$ be the set of all permutation functions for $G_{n,r,s,t} = (V, E, w)$ with range $\{1, 2, \ldots, |E|\}$. As in Section 6 it is helpful to define notation that allows us to identify interesting events. Let $\sigma \in \Sigma$ be a permutation function. Recall that $\mathcal{M}$ is the set of multi-edges, with $\mathbf{e} \in \mathcal{M}$ being a set $\mathbf{e} = \{e^1, e^2, \ldots, e^t\}$ of $t$ identical edges. Define:

$$\begin{aligned}
\forall i \in [n]: \quad \sigma(\mathbf{b}_i^1) &= \min_{j \in [rs]} \sigma(b_{i,j}^1) \\
\forall i \in [n], \forall j \in [r]: \quad \sigma(\mathbf{a}_{i,j}^1) &= \min_{k \in [s]} \sigma(a_{i,j,k}^1) \\
\forall i \in [n]: \quad \sigma(\mathbf{a}_i^1) &= \max_{j \in [r]} \min_{k \in [s]} \sigma(a_{i,j,k}^1) \\
\forall \mathbf{e} \in \mathcal{M}: \quad \sigma(\mathbf{e}) &= \max_{e \in \mathbf{e}} \sigma(e)
\end{aligned}$$

**Definition 7.2 (Induced permutation function)** *For every permutation function $\sigma \in \Sigma$ we define the* induced permutation function, *$\hat{\sigma} : [n] \to [n]$, for $\sigma$ to be the (unique) permutation function that satisfies $\hat{\sigma}(i) < \hat{\sigma}(j)$ if and only if $\sigma(\mathbf{b}_i^1) < \sigma(\mathbf{b}_j^1)$, for all $i, j \in [n]$.*

**Definition 7.3 (Well-behaved permutation function)** *We say that $\sigma \in \Sigma$ is* well-behaved *if and only if:*

*(i)* $\sigma(\mathbf{b}_i^1) < \sigma(\mathbf{a}_i^1)$ *for all $i \in [n]$.*

*(ii)* $\sigma(\mathbf{a}_{i,j}^1) < \sigma(\mathbf{e})$ *for all $\mathbf{e} \in \mathcal{M}$, $i \in [n]$, and $j \in [r]$.*

The requirements $(i)$ and $(ii)$ of Definition 7.3 correspond to the requirements $(ii)$ and $(iii)$, respectively, of Definition 6.4. Recall that the one-permutation randomized counter, RandCount$^{1P}$, takes as its second argument a permutation function $\hat{\sigma} : [n] \to [n]$. Recall also that $f^{1P}([n], \hat{\sigma})$ is the number of times RandCount$^{1P}([n], \hat{\sigma})$ sets a bit to 1, and that $f^{1P}(n)$ is the expected value of $f^{1P}([n], \hat{\sigma})$ when $\hat{\sigma}$ is uniformly random. Let $g^{1P}(F, B, \sigma)$ be the number of improving switches performed by Random-Facet$^{1P}(F, B, \sigma)$, and let $g^{1P}(F, B)$ be the expected number of improving switches performed by Random-Facet$^{1P}(F, B, \sigma)$ when $\sigma \in \Sigma$ is picked uniformly at random.

The following lemma shows that Random-Facet$^{1P}$ essentially simulates RandCount$^{1P}$ when run on $G_{n,r,s,t}$. In particular, for $F = E$ and $p = n+1$, the condition described by $(i)$ is satisfied for the initial tree $B_0$, such that the lemma says that $g^{1P}(E, B_0, \sigma) \geq f^{1P}([n], \hat{\sigma})$. The proof of the lemma is very similar to the proof of Lemma 6.9. Note also that the two lemmas describe the same kind of trees.

**Lemma 7.4** *Let $\sigma \in \Sigma$ be a well-behaved permutation function, and let $F \subseteq E$ satisfy $reset(F) = 0$. Furthermore, assume that for all $\mathbf{e} \in \mathcal{M}$ we have $e \in F$ where $\sigma(e) = \sigma(\mathbf{e})$. In particular, $F$ is functional. Let $p \in [n+1]$, define $N(F,p) := \{i \in [n] \mid i < p \wedge \mathbf{b}_i^1 \subseteq F\}$, and let $B$ be a tree. Assume that one of the following two conditions are satisfied:*

*(i)*
- *For all $i > p$: $bit_i(F,B) = 1$.*
- *$\mathbf{b}_p^1 \subseteq B$, and $\mathbf{a}_{i,j}^1 \subseteq F$ where $\sigma(\mathbf{a}_{i,j}^1) = \sigma(\mathbf{a}_i^1)$.*
- *For all $i < p$: $bit_i(F,B) = 0$. Furthermore, if $\mathbf{b}_i^1 \subseteq F$ then $\mathbf{a}_{i,j}^1 \subseteq F$ where $\sigma(\mathbf{a}_{i,j}^1) = \sigma(\mathbf{a}_i^1)$.*

*(ii)*
- *For all $i \neq p$: $bit_i(F,B) = 1$.*
- *$\mathbf{b}_p^1 \subseteq F$, $\mathbf{a}_{p,j}^1 \subseteq F$ where $\sigma(\mathbf{a}_{p,j}^1) = \sigma(\mathbf{a}_p^1)$, and $\mathbf{a}_{p,j}^1 \cap B = \emptyset$ for all $j \in [r]$. Furthermore, $e \in B$ where $\sigma(e) = \sigma(\mathbf{b}_p^1)$.*

*Then $g^{1P}(F,B,\sigma) \geq f^{1P}(N(F,p),\hat{\sigma})$.*

**Proof:** The lemma is proved by induction in $|N(F,p)|$, $p$, and $|F|$, and backward induction in $|\mathbf{b}_p^1 \cap B|$ and $|\mathbf{a}_{p,j}^1 \cap B|$, where $\sigma(\mathbf{a}_{p,j}^1) = \sigma(\mathbf{a}_p^1)$. For $|N(F,p)| = 0$ or $p = 1$ we have $N(F,p) = \emptyset$, and the lemma is clearly true since $f^{1P}(\emptyset,\hat{\sigma}) = 0$.

Assume that $|N(F,p)| > 0$, and let $e = \operatorname{argmin}_{e \in F \setminus B} \sigma(e)$. We consider four cases:

1. $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ for some $i' \in [n]$ with $\mathbf{b}_{i'}^1 \subseteq F$.
2. $e \in \mathbf{b}_{i'}^1$ for some $i' \in [n]$ with $\mathbf{b}_{i'}^1 \subseteq F$, and $\sigma(e) \neq \sigma(\mathbf{b}_{i'}^1)$.
3. $e \in \mathbf{a}_{i',j}^1$ for some $i' \in [n]$ and $j \in [r]$ such that $\mathbf{b}_{i'}^1 \subseteq F$, $\mathbf{a}_{i'}^1 \sqsubseteq F$, and $\mathbf{a}_{i'}^1 \not\sqsubseteq F \setminus \{e\}$.
4. $e$ does not qualify for any of the other cases.

**Case 1:** Assume that $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ for some $i' \in [n]$ where $\mathbf{b}_{i'}^1 \subseteq F$. Let $B'$ be the tree returned by $\textsc{Random-Facet}^{1P}(F \setminus \{e\}, B, \sigma)$. Since $F \setminus \{e\}$ is functional we know from Lemma 5.5 that $B' \subseteq \mathcal{B}_F$. Furthermore, since $reset(F \setminus \{e\}) = 0$ we get from Lemma 5.10 that $e$ is an improving switch with respect to $B'$. Let $B'' = B'[e]$. It follows that:

$$g^{1P}(F,B,\sigma) \;=\; g^{1P}(F \setminus \{e\}, B, \sigma) + 1 + g^{1P}(F, B'', \sigma)\ .$$

Let $i \in \operatorname{argmin}_{i \in N(F,p)} \hat{\sigma}(i)$. Then we also have:

$$f^{1P}(N(F,p),\hat{\sigma}) \;=\; f^{1P}(N(F,p) \setminus \{i\}, \hat{\sigma}) + 1 + f^{1P}(N(F,p) \cap [i-1], \hat{\sigma})\ .$$

Using the induction hypothesis, we show that $g^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p) \setminus \{i\}, \hat{\sigma})$ and $g^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p) \cap [i-1], \hat{\sigma})$ which proves the induction step.

We next prove that $F$, $B$, and $\sigma$ satisfy $(i)$. Indeed, for $(ii)$ we have $bit_i(F,B) = 1$ for all $i \neq p$, and if $\sigma(e) = \sigma(\mathbf{b}_p^1)$ then $e \in B$. It follows that every edge $e$ with $\sigma(e) = \sigma(\mathbf{b}_i^1)$ is part of the current tree in $(ii)$, such that $e \notin F \setminus B$. A similar argument for $(i)$ shows that $i' < p$. Since $\mathbf{b}_{i'}^1 \subseteq F$ it follows that $i' \in N(F,p)$. Moreover, $i' \in \operatorname{argmin}_{i \in N(F,p)} \sigma(\mathbf{b}_i^1) = \operatorname{argmin}_{i \in N(F,p)} \hat{\sigma}(i)$. Hence, $i'$ is also the index picked by $\textsc{RandCount}^{1P}(N(F,p),\hat{\sigma})$. Observe that $N(F \setminus \{e\}, p) = N(F,p) \setminus \{i'\}$, that $F \setminus \{e\}$ is functional, that $reset(F \setminus \{e\}) = 0$, and that $F \setminus \{e\}$, $B$, and $\sigma$ satisfy $(i)$. Hence, for the first recursive call, $\textsc{Random-Facet}^{1P}(F \setminus \{e\}, B, \sigma)$, it follows by induction that $g^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p) \setminus \{i'\}, \hat{\sigma})$.

Lemma 5.5 shows that $F$, $B''$, and $\sigma$ satisfy $(ii)$ where $i'$ plays the role of $p$. Since $i' < p$ it follows by induction that $g^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p) \cap [i'-1], \hat{\sigma})$.

**Case 2:** Assume that $e \in \mathbf{b}_{i'}^1$ for some $i' \in [n]$ with $\mathbf{b}_{i'}^1 \subseteq F$, and $\sigma(e) \neq \sigma(\mathbf{b}_{i'}^1)$. Let $B'$ be the tree returned by $\textsc{Random-Facet}^{1P}(F \setminus \{e\}, B, \sigma)$, and let $B'' = B'[e]$. As in case 1 we have:

$$g^{1P}(F,B,\sigma) \;=\; g^{1P}(F \setminus \{e\}, B, \sigma) + 1 + g^{1P}(F, B'', \sigma)\ .$$

Using the induction hypothesis, we show that $g^{1P}(F, B'', \sigma) \geq f^{1P}(N(F, p), \hat{\sigma})$ which proves the induction step, i.e., in this case we only count the improving switches performed during the second recursive call.

We first show that $F$, $B$, and $\sigma$ satisfy $(ii)$. Indeed, for $(i)$ we have $\mathbf{b}_i^1 \subseteq B$ for all $i \geq p$ with $\mathbf{b}_i^1 \subseteq F$, which implies that $i' < p$. On the other hand, $\mathbf{b}_i^1 \cap B = \emptyset$ for all $i < p$. Since $\mathbf{b}_{i'}^1 \subseteq F$ it follows that $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ which is a contradiction. Moreover, since $F$, $B$, and $\sigma$ satisfy $(ii)$ such that $bit_i(F, B) = 1$ for all $i \neq p$, we must have $i' = p$. Lemma 5.5 then shows that $F$, $B''$, and $\sigma$ satisfy $(ii)$. Furthermore, we have $|\mathbf{b}_p^1 \cap B''| > |\mathbf{b}_p^1 \cap B|$, and we get by induction that $g^{1P}(F, B'', \sigma) \geq f^{1P}(N(F, p), \hat{\sigma})$. Note that if $|\mathbf{b}_p^1 \cap B| = rs$ such that $\mathbf{b}_p^1 \subseteq B$, then it was not possible to pick $e \in \mathbf{b}_p^1$, and case 2 could not happen. The base-case $|\mathbf{b}_p^1 \cap B| = rs$ then follows from the proof of the other three cases.

**Case 3:** Assume that $e \in \mathbf{a}_{i',j}^1$ for some $i' \in [n]$ and $j \in [r]$ such that $\mathbf{b}_{i'}^1$, $\mathbf{a}_{i'}^1 \sqsubseteq F$, and $\mathbf{a}_{i'}^1 \not\sqsubseteq F \setminus \{e\}$. The most important observation for this case is that $reset(F \setminus \{e\}) = i'$. Let $B'$ be the tree returned by Random-Facet$^{1P}(F \setminus \{e\}, B, \sigma)$. Since $F \setminus \{e\}$ is functional we know from Lemma 5.5 that $B' \subseteq \mathcal{B}_F$. Furthermore, since $reset(F \setminus \{e\}) = i'$ we get from Lemma 5.10 that $e$ is an improving switch with respect to $B'$. Let $B'' = B'[e]$. As in cases 1 and 2 we therefore have:

$$g^{1P}(F, B, \sigma) \;=\; g^{1P}(F \setminus \{e\}, B, \sigma) + 1 + g^{1P}(F, B'', \sigma)\ .$$

Using the induction hypothesis, we show that $g^{1P}(F, B'', \sigma) \geq f^{1P}(N(F, p), \hat{\sigma})$ which proves the induction step, i.e., as in case 2 we again only count the improving switches performed during the second recursive call.

We next show that for both $(i)$ and $(ii)$ we must have $i' = p$. Consider first the case when $F$, $B$, and $\sigma$ satisfy $(i)$. Since $bit_i(F, B) = 1$ for all $i > p$ and $reset(F) = 0$, we must have $\mathbf{a}_i^1 \sqsubseteq B$ if $\mathbf{b}_i^1 \subseteq B$. It follows that we can not have $\mathbf{a}_{i'}^1 \not\sqsubseteq F \setminus \{e\}$ if $i' > p$. Suppose that $i' < p$. Since $\mathbf{b}_{i'}^1 \subseteq F$ we have $\mathbf{a}_{i',j}^1 \subseteq F$ where $\sigma(\mathbf{a}_{i',j}^1) = \sigma(\mathbf{a}_{i'}^1)$. Hence, we must have $e \in \mathbf{a}_{i',j}^1$, which means that $\sigma(e) \geq \sigma(\mathbf{a}_{i'}^1)$. Since $\sigma$ is well-behaved it satisfies $\sigma(\mathbf{b}_{i'}^1) < \sigma(\mathbf{a}_{i'}^1) \leq \sigma(e)$. On the other hand, for all $i < p$ we have $\mathbf{b}_i^1 \cap B = \emptyset$. Hence, the edge $e'$ where $\sigma(e') = \sigma(\mathbf{b}_{i'}^1)$ is available and would have been picked instead of $e$; a contradiction. For $(ii)$ we get that $i' = p$ from the fact that $reset(F) = 0$ and $bit_i(F, B) = 1$ for all $i \neq p$.

Lemma 5.5 then shows that $F$, $B''$, and $\sigma$ satisfy $(i)$. Note that for both $(i)$ and $(ii)$ we must have $e \in \mathbf{a}_{p,j}^1$ where $\sigma(\mathbf{a}_{p,j}^1) = \sigma(\mathbf{a}_p^1)$. Hence, $|\mathbf{a}_{p,j}^1 \cap B''| > |\mathbf{a}_{p,j}^1 \cap B|$, and we get by induction that $g^{1P}(F, B'', \sigma) \geq f^{1P}(N(F, p), \hat{\sigma})$. Note that if $|\mathbf{a}_{p,j}^1 \cap B| = s$ such that $\mathbf{a}_{p,j}^1 \subseteq B$, then it was not possible to pick $e \in \mathbf{a}_{p,j}^1$, and case 3 could not happen. The base-case $|\mathbf{a}_{p,j}^1 \cap B| = s$ then follows from the proof of the other three cases.

**Case 4:** Assume that $e \notin \mathbf{b}_i^1$ for some $i$ where $\mathbf{b}_i^1 \subseteq F$, and that $e \notin \mathbf{a}_{i,j}^1$ for some $i \in [n]$ and $j \in [r]$ such that $\mathbf{b}_i^1 \subseteq F$, $\mathbf{a}_i^1 \sqsubseteq F$, and $\mathbf{a}_i^1 \not\sqsubseteq F \setminus \{e\}$. We show that $g^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F, p), \hat{\sigma})$, i.e., in this case we only count the improving switches performed during the first recursive call.

Observe that $N(F \setminus \{e\}, p) = N(F, p)$ since $e \notin \mathbf{b}_i^1$ for some $i$ where $\mathbf{b}_i^1 \subseteq F$. Also observe that $reset(F \setminus \{e\}) = 0$ since $e \notin \mathbf{a}_{i,j}^1$ for some $i \in [n]$ and $j \in [r]$ such that $\mathbf{b}_i^1 \subseteq F$, $\mathbf{a}_i^1 \sqsubseteq F$, and $\mathbf{a}_i^1 \not\sqsubseteq F \setminus \{e\}$. For the same reason we see that for all $i \in [n]$, if $\mathbf{b}_i^1 \subseteq F \setminus \{e\}$ then $\mathbf{a}_{p,j}^1 \subseteq F \setminus \{e\}$ where $\sigma(\mathbf{a}_{p,j}^1) = \sigma(\mathbf{a}_p^1)$. It follows that if $F$, $B$, and $\sigma$ satisfy $(i)$ or $(ii)$, respectively, then $F \setminus \{e\}$, $B$, and $\sigma$ satisfy $(i)$ or $(ii)$, correspondingly. It remains to show that for all $\mathbf{e} \in \mathcal{M}$ we have $e' \in F \setminus \{e\}$ where $\sigma(e') = \sigma(\mathbf{e})$. Once this has been shown it follows by induction that $g^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F, p), \hat{\sigma})$, since $|F \setminus \{e\}| < |F|$. Note that it is not possible to reach a situation where $F \cap \mathbf{e} = \emptyset$ for some $\mathbf{e} \in \mathcal{M}$. Hence, the induction always goes back to one of the other base-cases.

Suppose that $e \in \mathbf{e}$ for some $\mathbf{e} \in \mathcal{M}$, and assume for the sake of contradiction that $\mathbf{e} \cap (F \setminus \{e\}) = \emptyset$. This is only possible if $\sigma(e) = \sigma(\mathbf{e})$. Since $p > 1$ and $N(F, p) \neq \emptyset$ there exists some set $\mathbf{a}_{i,j}^1$ such that $\mathbf{a}_{i,j}^1 \subseteq F$ and $\mathbf{a}_{i,j}^1 \cap B = \emptyset$. For $(i)$ this is true for some $i < p$, and for $(ii)$ it is true for $i = p$. Since $\sigma$ is well-behaved it must be the case that $\sigma(\mathbf{a}_{i,j}^1) < \sigma(\mathbf{e})$. Hence, an edge from $\mathbf{a}_{i,j}^1$ would be picked before an edge from $\mathbf{e}$, and we get a contradiction. □

We next prove that a uniformly random permutation function $\sigma \in \Sigma$ is well-behaved with high probability. The proof of the following lemma resembles that of Lemma 6.11.

**Lemma 7.5** *Let $\sigma \in \Sigma$ be chosen uniformly at random. If $r = s = t = 3\lceil \log n \rceil$, then $\sigma$ is well-behaved with probability at least $1/2$.*

**Proof:** We start by upper bounding the probability that $\sigma$ fails to satisfy Definition 7.3 $(i)$. Let $i \in [n]$ be given, and define:

$$\begin{aligned}\forall j \in [r]: \qquad \mathbf{b}^1_{i,j} &= \{b^1_{i,1+(j-1)s}, b^1_{i,2+(j-1)s}, \dots, b^1_{i,s+(j-1)s}\}\\ \forall j \in [r]: \quad \sigma(\mathbf{b}^1_{i,j}) &= \min_{e \in \mathbf{b}^1_{i,j}} \sigma(e)\end{aligned}$$

Note that the sets $\mathbf{b}^1_{i,j}$, for $j \in [r]$, partition $\mathbf{b}^1_i$ into $r$ sets of size $s$. In particular, all the sets $\mathbf{b}^1_{i,j}$ and $\mathbf{a}^1_{i,j}$, for $j \in [r]$, have the same size. Hence, we may view $\sigma$ as defining a uniformly random permutation $\sigma'$ of the set $\mathcal{S} = \{\mathbf{b}^1_{i,1}, \dots, \mathbf{b}^1_{i,r}, \mathbf{a}^1_{i,1}, \dots, \mathbf{a}^1_{i,r}\}$, such that for all $s_1, s_2 \in \mathcal{S}$, $\sigma'(s_1) < \sigma'(s_2)$ if and only if $\sigma(s_1) < \sigma(s_2)$. In order to get the event $\sigma(\mathbf{b}^1_i) > \sigma(\mathbf{a}^1_i)$ it must be the case that all the $\mathbf{a}^1_{i,j}$ elements of $\mathcal{S}$ are placed before all the $\mathbf{b}^1_{i,j}$ elements of $\mathcal{S}$. This happens with probability:

$$\mathbb{P}\left[\sigma(\mathbf{b}^1_i) > \sigma(\mathbf{a}^1_i)\right] \;=\; \frac{(r!)^2}{(2r)!}$$

Hence, the probability that $\sigma$ fails to satisfy Definition 7.3 $(i)$ is at most $n\frac{(r!)^2}{(2r)!} \leq n2^{-r} \leq 1/4$, where the first inequality follows from a simple proof by induction.

We next upper bound the probability that $\sigma$ fails to satisfy Definition 7.3 $(ii)$. Let $\mathbf{e} \in \mathcal{M}$, $i \in [n]$, and $j \in [r]$ be given. Recall that $\mathbf{e}$ contains $t$ copies of a multi-edge. Similarly, $\mathbf{a}^1_{i,j}$ contains $s$ edges. The event $\sigma(\mathbf{a}^1_{i,j}) > \sigma(\mathbf{e})$ occurs if and only if all the edges of $\mathbf{a}^1_{i,j}$ are assigned higher indices than all the edges of $\mathbf{e}$. Since $\sigma$ is uniformly random this happens with probability:

$$\mathbb{P}\left[\sigma(\mathbf{a}^1_{i,j}) > \sigma(\mathbf{e})\right] \;=\; \frac{s!\,t!}{(s+t)!}$$

Since $|\mathcal{M}| = n(2rs+r+3)$, it follows that $\sigma$ fails to satisfy Definition 7.3 $(ii)$ with probability at most $n^2r(2rs+r+3)\frac{s!\,t!}{(s+t)!} \leq 6n^2\log^3 n\, 2^{-3\log n} \leq 1/4$.

Let $p_{n,r,s,t}$ be the probability that $\sigma$ is well-behaved. We have shown that:

$$p_{n,r,s,t} \;\geq\; 1 - n\frac{(r!)^2}{(2r)!} - n^2r(2rs+r+3)\frac{s!\,t!}{(s+t)!} \;\geq\; \frac{1}{2}\,.$$

□

**Proof of Theorem 7.1:** If $\sigma \in \Sigma$ is picked uniformly at random then we know from Lemma 7.5 that $\sigma$ is well-behaved with probability at least 1/2. Moreover, since every set $\mathbf{b}^1_i$, for $i \in [n]$, has the same size, the induced permutation function $\hat{\sigma}$ is also uniformly random. It then follows from Lemma 4.3 and Lemma 4.2 that:

$$g^{1P}(F,B) \;\geq\; \frac{1}{2}f^{1P}(n) \;=\; \frac{1}{2}f(n) \;\sim\; \frac{\mathrm{e}^{2\sqrt{n}}}{4\sqrt{\pi\mathrm{e}}\,n^{1/4}} \;=\; \mathrm{e}^{\Omega(\sqrt{n})}\,.$$

□

# 8 Lower bound for Random-Bland

We next prove a lower bound for the Random-Bland algorithm. The proof of this lower bound is very similar to the corresponding proof for Random-Facet$^{1P}$. In fact, the only difference is that Lemma 7.4 needs to be proved slightly differently. Recall that Random-Bland is the algorithm obtained by running Bland with a uniformly random permutation. Since, we remove edges and consider subproblems, it is again convenient to operate with a *permutation function* $\sigma : E \to \mathbb{N}$ that assigns to each edge of $E$ a *distinct* natural number. Recall that $\Sigma$ is the set of permutation functions with range $[|E|]$. The algorithm works by repeatedly performing the improving switch with *largest* permutation index $\sigma(e)$. We study the recursive formulation of Bland shown in Figure 1. Bland is different from Random-Facet$^{1P}$ in the following ways. The edge chosen to be removed from $F$ is always the edge in $F$ with *smallest* permutation index $\sigma(e)$, regardless of whether $e \in B$ or not. This means that it is possible to have $B \not\subseteq F$, which means that the termination criterion is changed to $F = \emptyset$.

Let $G_{n,r,s,t} = (V, E, c)$ be defined as in Section 5. The parameters $n, r, s, t \in \mathbb{N}$ play the same role as in Section 7. In particular, we use $r = s = t = \Theta(\log n)$, such that $G_{n,r,s,t}$ contains $N = \Theta(nrs) = \Theta(n \log^2 n)$ vertices and $M = \Theta(nrst) = \Theta(n \log^3 n)$ edges. We also use the same notation as in Section 7. The following theorem is again proved by showing that $\textsc{Bland}(E, B_0, \sigma)$ simulates $\textsc{RandCount}^{1P}([n], \hat{\sigma})$, where $\hat{\sigma}$ is the induced permutation function, when $\sigma$ is well-behaved. Recall that Lemma 7.5 shows that $\sigma$ is well-behaved with probability at least 1/2.

**Theorem 8.1** *The expected number of switches performed by the* Random-Bland *algorithm, when run on the graph $G_{n,r,s,t}$, where $r = s = t = \Theta(\log n)$, with initial tree $B_0$, is at least $\mathrm{e}^{\Omega(\sqrt{n})} = \mathrm{e}^{\Omega(\sqrt{M}/\log^{3/2} M)}$, where $M$ is the number of edges in $G_{n,r,s,t}$.*

Let $B'$ be the tree returned by $\textsc{Bland}(F, B, \sigma)$. An important difference between Bland and $\textsc{Random-Facet}^{1P}$ is that $B' \subseteq F \cup B$ is optimal for $G_{F \cup B'}$ for Bland, whereas it is optimal for $G_F = G_{F \cup B}$ for $\textsc{Random-Facet}^{1P}$. This means that it is harder to keep track of which edges are present in $F \cup B'$. There are, however, some edges in $B$ that are guaranteed to also be in $B'$, namely the edges $(u, v) \in B$ used at all vertices $u$ whose distance to the target $\mathrm{T}$ in $B$ is optimal for $G_{F \cup B}$. Since $B'$ is obtained from $B$ by performing improving switches such edges must remain in $B'$. This leads us to the following definition.

**Definition 8.2 (Fixed edges)** *Let $B$ be a tree and $F \subseteq E$. We say that an edge $(u, v) \in B$ is* fixed *for $B$ with respect to $F$ if $y_B(u) = y(u)$ for the graph $G_{F \cup B}$.*

Recall that $G_{n,r,s,t}$ is acyclic such that it is not possible to get from higher bits to lower bits. For any edge $(u, v) \in F \subseteq E$, let $B'$ be returned by $\textsc{Bland}(F \setminus \{(u, v)\}, B, \sigma)$. Then all edges used in $B'$ at vertices that can not reach $u$ in $G_{n,r,s,t}$ are fixed with respect to $F$. We let $V(b_{i',j'}) \subseteq V$ and $V(a_{i',j',k'}) \subseteq V$ be vertices that can not reach $b_{i',j'}$ and $a_{i',j',k'}$, respectively, including $b_{i',j'}$ and $a_{i',j',k'}$ themselves:

$$\begin{aligned} V(b_{i',j'}) &= \{u_i, w_i \mid i > i'\} \cup \{a_{i,j,k} \mid i > i', j \in [r], k \in [s]\} \cup \{b_{i,j} \mid i > i', j \in [rs]\} \cup \{b_{i',j} \mid j \geq j'\} \\ V(a_{i',j',k'}) &= \{u_i, w_i \mid i > i'\} \cup \{a_{i,j,k} \mid i > i', j \in [r], k \in [s]\} \cup \{b_{i,j} \mid i \geq i', j \in [rs]\} \cup \\ &\quad\ \{a_{i',j,k} \mid j \neq j', k \in [s]\} \cup \{a_{i',j',k} \mid k \geq k'\} \end{aligned}$$

We say that $V(b_{i',j'})$ is fixed for $B$ with respect to $F$ if every edge $(u, v) \in B$, for $u \in V(b_{i',j'})$, is fixed with respect to $F$. Similarly, we say that $V(a_{i',j',k'})$ is fixed for $B$ with respect to $F$ if every edge $(u, v) \in B$, for $u \in V(a_{i',j',k'})$, is fixed with respect to $F$. Note that if $V(a_{i',j',k'})$ is fixed for $B$ with respect to $F$, for some $i', j', k'$, then $V(b_{i',1})$ is fixed for $B$ with respect to $F$. Since the choices at $b_{i,j}$ and $a_{i,j,k}$, for some $i, j, k$, are essentially binary, i.e., the edges in $\mathbf{b}^0_{i,j}$ and $\mathbf{a}^0_{i,j,k}$, respectively, are identical, the above discussion proves the following lemma.

**Lemma 8.3** *Let $B$ be a tree, let $F \subseteq E$, and let $\sigma \in \Sigma$. Assume that $e = b^1_{i,j} \in F$, for some $i \in [n]$ and $j \in [r]$, let $B'$ be the tree returned by $\textsc{Bland}(F \setminus \{e\}, B, \sigma)$, assume that $e$ is an improving switch w.r.t. $B'$, and let $B'' = B'[e]$. Then $V(b_{i,j})$ is fixed for $B''$ w.r.t. $F$. Similarly, if $e = a^1_{i,j,k}$ then $V(a_{i,j,k})$ is fixed for $B''$ w.r.t. $F$.*

Let $B'$ be the tree returned by $\textsc{Bland}(F, B, \sigma)$. It is also useful to know which edges are, in a sense, lost when going from $B$ to $B'$, i.e., edges $e$ such that $e \in F \cup B$ but $e \notin F \cup B'$. The following lemma will be useful for this purpose. Let $\sigma^{-1}(\ell)$ be the edge $e$ with index $\ell$, i.e., $\sigma(e) = \ell$. Define $F(\sigma, \ell) := \{e \in E \mid \sigma(e) \geq \ell\}$. Note that the set $F$ for $\textsc{Bland}(F, B, \sigma)$ is always equal to $F(\sigma, \ell)$ for some $\ell \geq 1$. In particular, $F(\sigma, 1) = E$.

**Lemma 8.4** *Let $\sigma \in \Sigma$ be well-behaved, let $\ell \geq 1$, let $F = F(\sigma, \ell)$, let $B$ be a tree, and let $p \in [n]$. Assume that $reset(F \cup B) < p$, that $bit_i(F \cup B, B) = 1$ for all $i > p$, and that $\ell \leq \sigma(\mathbf{a}^1_p)$. Assume also that there is some $j' \in [rs]$ such that for all $j \geq j'$ we have $b^1_{p,j} \in B$, for all $j < j'$ we have $b^1_{p,j} \in F$, and $V(b_{p,j'})$ is fixed for $B$ w.r.t. $F$. In particular, $\mathbf{b}^1_p \subseteq F \cup B$. Finally, let $B'$ be the tree returned by* $\textsc{Bland}(F, B, \sigma)$*. Then for all $i < p$, $\mathbf{b}^1_i \cap B' \subseteq F$.*

**Proof:** Consider the computation path obtained by repeatedly entering the first branch of the recursion until reaching a recursive call of the form $\textsc{Bland}(F(\sigma,\ell'),B,\sigma)$, where either $\sigma^{-1}(\ell') \in \mathbf{b}_p^1 \setminus B$ or $\ell' = \sigma(\mathbf{a}_p^1)$. Note that since $\sigma$ is well-behaved and $\ell' \leq \sigma(\mathbf{a}_p^1)$ the set $F(\sigma,\ell'+1)$ is functional. Let $B_1'$ be the tree returned by $\textsc{Bland}(F(\sigma,\ell'+1),B,\sigma)$. Then $B_1'$ is optimal for $G_{F(\sigma,\ell'+1)\cup B_1'}$, and by Lemma 5.5 we have $B_1' \subseteq \mathcal{B}_{F(\sigma,\ell'+1)\cup B_1'}$.
Consider the case when $\sigma^{-1}(\ell') \in \mathbf{b}_p^1 \setminus B$, that is, $\sigma^{-1}(\ell') = b_{p,j}^1$ for some $j$. Then, since $\mathit{reset}(F\cup B) < p$ and $\mathit{bit}_i(F,B) = 1$ for all $i > p$, we have $\mathit{reset}(F(\sigma,\ell'+1)\cup B) < p$, and Lemma 5.10 shows that $b_{p,j}^1$ is an improving switch w.r.t. $B_1'$. Moreover, by Lemma 8.3, $V_{(p,j)}$ is fixed for $B_1'' = B_1'[b_{p,j}^1]$ w.r.t. $F(\sigma,\ell')$. Furthermore, Lemma 5.5 shows that $F(\sigma,\ell')$ and $B_1''$ have the properties that were assumed about $F$ and $B$ in the lemma. Hence, we may assume, by induction, that $\ell' = \sigma(\mathbf{a}_p^1)$.

Consider next the case when $\ell' = \sigma(\mathbf{a}_p^1)$. Since $\mathbf{b}_p^1 \subseteq F(\sigma,\ell'+1)\cup B$ and $V(b_{p,j'+1})$ is fixed for $B$ w.r.t. $F$, where $j' = \min\{j \in [rs] \mid b_{p,j}^1 \notin B\}$, it follows that $\mathbf{b}_p^1 \subseteq F(\sigma,\ell'+1)\cup B_1'$. On the other hand, we have $\mathbf{a}_p^1 \not\subseteq F(\sigma,\ell'+1)\cup B$, which means that $\mathit{reset}(F(\sigma,\ell'+1)\cup B_1') = p$. It then follows from Lemma 5.5 that $\mathbf{b}_i^1 \cap B_1' = \emptyset$ for all $i < p$. Since the tree $B'$ returned by $\textsc{Bland}(F,B,\sigma)$ is obtained from $B_1'$ by performing improving switches from $F$ it follows that $\mathbf{b}_i^1 \cap B' \subseteq F$ for $i < p$. □

Let $\sigma \in \Sigma$ be a permutation function. Recall that $f^{1P}([n],\hat{\sigma})$ is the number of times $\textsc{RandCount}^{1P}([n],\hat{\sigma})$ sets a bit to 1, and that $f^{1P}(n)$ is the expected value of $f^{1P}([n],\hat{\sigma})$ when $\hat{\sigma}$ is uniformly random. Let $h^{1P}(F,B,\sigma)$ be the number of improving switches performed by $\textsc{Bland}(F,B,\sigma)$, and let $h^{1P}(F,B)$ be the expected number of improving switches performed by $\textsc{Bland}(F,B,\sigma)$ when $\sigma \in \Sigma$ is picked uniformly at random, i.e., $h^{1P}(F,B)$ is the expected number of improving switches performed by $\textsc{Random-Bland}$.

The following lemma shows that $\textsc{Bland}$ essentially simulates $\textsc{RandCount}^{1P}$ when run on $G_{n,r,s,t}$. In particular, for $F = E$ and $p = n+1$, the condition described by $(i)$ is satisfied for the initial tree $B_0$, such that the lemma says that $h^{1P}(E,B_0,\sigma) \geq f^{1P}([n],\hat{\sigma})$. The lemma corresponds to Lemma 7.4, and the proof is very similar. It is used to prove Theorem 8.1 in the same way as Lemma 7.4 was used to prove Theorem 7.1.

**Lemma 8.5** *Let $\sigma \in \Sigma$ be a well-behaved permutation function, let $B$ be a tree, let $\ell \geq 1$, let $F = F(\sigma,\ell) \subseteq E$, assume that $\mathit{reset}(F\cup B) = 0$, and assume that $F$ is functional. Let $p \in [n+1]$, and define $N(F,p) := \{i \in [n] \mid i < p \wedge \mathbf{b}_i^1 \subseteq F\}$. Assume that one of the following two conditions are satisfied:*

*(i)*
- *For all $i > p$: $\mathit{bit}_i(F\cup B,B) = 1$.*
- *$\mathbf{b}_p^1 \subseteq B$.*
- *Let $j' \in [r]$ satisfy $\sigma(\mathbf{a}_{p,j'}^1) = \sigma(\mathbf{a}_p^1)$. There is some $k' \in [s]$ such that for all $k \geq k'$ we have $a_{p,j',k}^1 \in B$, for all $k < k'$ we have $a_{p,j',k}^1 \in F$, and $V(a_{p,j',k'})$ is fixed for $B$ w.r.t. $F$. In particular, $\mathbf{a}_{p,j'}^1 \subseteq F\cup B$.*
- *For all $i < p$: $\mathit{bit}_i(F\cup B,B) = 0$.*
- *$\ell \leq \sigma(\mathbf{b}_p^1)$.*

*(ii)*
- *For all $i \neq p$: $\mathit{bit}_i(F\cup B,B) = 1$.*
- *There is some $j' \in [rs]$ such that for all $j \geq j'$ we have $b_{p,j}^1 \in B$, for all $j < j'$ we have $b_{p,j}^1 \in F$, and $V(b_{p,j'})$ is fixed for $B$ w.r.t. $F$. In particular, $\mathbf{b}_p^1 \subseteq F\cup B$.*
- *$\ell \leq \sigma(\mathbf{a}_p^1)$.*

*Then $h^{1P}(F,B,\sigma) \geq f^{1P}(N(F,p),\hat{\sigma})$.*

**Proof:** The lemma is proved by induction in $|N(F,p)|$, $p$, and $|F|$, and backward induction in $|\mathbf{b}_p^1 \cap B|$ and $|\mathbf{a}_{p,j}^1 \cap B|$, where $\sigma(\mathbf{a}_{p,j}^1) = \sigma(\mathbf{a}_p^1)$. For $|N(F,p)| = 0$ or $p = 1$ we have $N(F,p) = \emptyset$, and the lemma is clearly true since $f^{1P}(\emptyset,\hat{\sigma}) = 0$.

Assume that $|N(F,p)| > 0$, and let $e = \operatorname{argmin}_{e\in F} \sigma(e) = \sigma^{-1}(\ell)$. We consider five cases:

1. $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ for some $i' \in N(F,p)$, and $(i)$ is satisfied.
2. $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ for some $i' \in N(F,p)$, and $(ii)$ is satisfied.

3. $e \in \mathbf{b}_p^1 \setminus B$ and $\sigma(e) \neq \sigma(\mathbf{b}_p^1)$.
4. $e \in \mathbf{a}_{p,j}^1 \setminus B$ for some $j \in [r]$ such that $\mathbf{a}_p^1 \not\sqsubseteq (F \setminus \{e\}) \cup B$.
5. $e$ does not qualify for any of the other cases.

For each case we consider the cases where $(i)$ and $(ii)$ are satisfied separately.

In the following case analysis we let $B'$ be the tree returned by $\textsc{Bland}(F \setminus \{e\}, B, \sigma)$, and we let $B'' = B'[e]$.

**Case 1:** Assume that $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ for some $i' \in N(F,p)$, and assume that $(i)$ is satisfied. Since $F \setminus \{e\}$ is functional and $B'$ is optimal for $G_{(F\setminus\{e\})\cup B'}$, we know from Lemma 5.5 that $B' \subseteq \mathcal{B}_{(F\setminus\{e\})\cup B'}$. Since $reset(F \cup B) = 0$ and $V(b_{p,1})$ is fixed for $B$ w.r.t. $F$ we get that $\mathbf{b}_i^1 \subseteq B'$ if $\mathbf{b}_i^1 \subseteq F \cup B$, for all $i \geq p$. Similarly, $\mathbf{a}_i^1 \sqsubseteq B'$ if $\mathbf{b}_i^1 \subseteq F \cup B$, for all $i > p$. It follows that $reset((F \setminus \{e\}) \cup B') \geq p$, and we get from Lemma 5.10 that $e$ is an improving switch with respect to $B'$. Let $B'' = B'[e]$. Hence, we have:

$$h^{1P}(F,B,\sigma) \;=\; h^{1P}(F \setminus \{e\}, B, \sigma) + 1 + h^{1P}(F, B'', \sigma) \;.$$

Let $i \in \operatorname{argmin}_{i \in N(F,p)} \hat{\sigma}(i)$. Then we also have:

$$f^{1P}(N(F,p),\hat{\sigma}) \;=\; f^{1P}(N(F,p) \setminus \{i\}, \hat{\sigma}) + 1 + f^{1P}(N(F,p) \cap [i-1], \hat{\sigma}) \;.$$

Using the induction hypothesis, we show that $h^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p) \setminus \{i\}, \hat{\sigma})$ and $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p) \cap [i-1], \hat{\sigma})$ which proves the induction step.

Observe that $i' \in \operatorname{argmin}_{i \in N(F,p)} \sigma(\mathbf{b}_i^1) = \operatorname{argmin}_{i \in N(F,p)} \hat{\sigma}(i)$ such that $\textsc{RandCount}^{1P}(N(F,p), \hat{\sigma})$ also picks the index $i'$. Furthermore, observe that $N(F \setminus \{e\}, p) = N(F,p) \setminus \{i'\}$, that $F \setminus \{e\}$ is functional, that $reset((F \setminus \{e\}) \cup B) = 0$, and that $F \setminus \{e\}$ and $B$ satisfy $(i)$. Hence, for the first recursive call, $\textsc{Bland}(F \setminus \{e\}, B, \sigma)$, it follows by induction that $h^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p) \setminus \{i'\}, \hat{\sigma})$.

Let $j'$ be the index such that $e = b_{i',j'}^1$. Note that Lemma 8.3 shows that $V(b_{i',j'})$ is fixed for $B''$ w.r.t. $F$. Lemma 5.5 shows that $F$, $B''$, and $\sigma$ satisfy $(ii)$ where $i'$ plays the role of $p$. Since $i' < p$ it follows by induction that $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p) \cap [i'-1], \hat{\sigma})$.

**Case 2:** Assume that $\sigma(e) = \sigma(\mathbf{b}_{i'}^1)$ for some $i' \in N(F,p)$, and assume that $(ii)$ is satisfied. By an argument analogous to the one given in case 1 $(i)$ we see that:

$$\begin{aligned} h^{1P}(F,B,\sigma) \;&=\; h^{1P}(F \setminus \{e\}, B, \sigma) + 1 + h^{1P}(F, B'', \sigma) \\ f^{1P}(N(F,p),\hat{\sigma}) \;&=\; f^{1P}(N(F,p) \setminus \{i\}, \hat{\sigma}) + 1 + f^{1P}(N(F,p) \cap [i-1], \hat{\sigma}) \;, \end{aligned}$$

where $i \in \operatorname{argmin}_{i \in N(F,p)} \hat{\sigma}(i)$. Using the induction hypothesis, we again show that $h^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p) \setminus \{i\}, \hat{\sigma})$ and $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p) \cap [i-1], \hat{\sigma})$ which proves the induction step.

We again observe that $i' \in \operatorname{argmin}_{i \in N(F,p)} \hat{\sigma}(i)$ is also the index picked by $\textsc{RandCount}^{1P}(N(F,p), \hat{\sigma})$, that $N(F \setminus \{e\}, p) = N(F,p) \setminus \{i'\}$, that $F \setminus \{e\}$ is functional, and that $reset((F \setminus \{e\}) \cup B) = 0$. It follows that $F \setminus \{e\}$, $B$, and $\sigma$ satisfy $(ii)$. Hence, for the first recursive call, $\textsc{Bland}(F \setminus \{e\}, B, \sigma)$, it follows by induction that $h^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p) \setminus \{i'\}, \hat{\sigma})$.

Note that $\mathbf{b}_{i'}^1 \not\subseteq F \setminus \{e\}$. Since $i' < p$, Lemma 8.4 shows that $\mathbf{b}_{i'}^1 \cap B' \subseteq F \setminus \{e\}$, and we therefore have $\mathbf{b}_{i'}^1 \not\subseteq (F \setminus \{e\}) \cup B'$. Let $j'$ be the index such that $e = b_{i',j'}^1$. Note also that Lemma 8.3 shows that $V(b_{i',j'})$ is fixed for $B''$ w.r.t. $F$. It follows that Lemma 5.5 shows that $F$, $B''$, and $\sigma$ satisfy $(ii)$ where $i'$ plays the role of $p$. Let $j'$ be the index such that $e = b_{i',j'}^1$. Since $i' < p$ it follows by induction that $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p) \cap [i'-1], \hat{\sigma})$.

**Case 3:** Assume that $e \in \mathbf{b}_p^1$ and that $\sigma(e) \neq \sigma(\mathbf{b}_p^1)$. In particular, we must have $\sigma(e) > \sigma(\mathbf{b}_p^1)$ which means that the last requirement for $(i)$ is violated such that condition $(ii)$ must be satisfied instead. By an argument analogous to the one given in case 1 $(i)$ we see that:

$$h^{1P}(F,B,\sigma) \;=\; h^{1P}(F \setminus \{e\}, B, \sigma) + 1 + h^{1P}(F, B'', \sigma) \;.$$

Using the induction hypothesis, we show that $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p), \hat{\sigma})$, i.e., we only count improving switches performed during the second recursive call.

The remainder of the argument is the same as the last part of the argument for case 1 $(i)$ where $i'$ is replaced by $p$. Note, however, that in order to apply the induction hypothesis we need the observation that $|\mathbf{b}_p^1 \cap B''| > |\mathbf{b}_p^1 \cap B|$. Note also that if $|\mathbf{b}_p^1 \cap B| = rs$ such that $\mathbf{b}_p^1 \subseteq B$, then it was not possible to pick $e \in \mathbf{b}_p^1$, and case 2 could not happen. The base-case $|\mathbf{b}_p^1 \cap B| = rs$ then follows from the proof of the other four cases.

**Case 4:** Assume that $e \in \mathbf{a}_{p,j}^1$ for some $j \in [r]$ such that $\mathbf{a}_p^1 \not\subseteq (F \setminus \{e\}) \cup B$. By an argument analogous to the one given in case 1 $(i)$ we see that:

$$h^{1P}(F, B, \sigma) \ = \ h^{1P}(F \setminus \{e\}, B, \sigma) + 1 + h^{1P}(F, B'', \sigma) \ .$$

Using the induction hypothesis, we show that $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p), \hat{\sigma})$, i.e., we only count improving switches performed during the second recursive call.

The most important observation for this case is that $\mathit{reset}((F \setminus \{e\}) \cup B') = p$. To prove this, observe that $\mathbf{b}_p^1 \subseteq F \cup B'$. For $(i)$ this follows from the fact that $V(b_{p,1})$ is fixed for $B$ w.r.t. $F$. For $(ii)$ some edges in $\mathbf{b}_p^1$ are fixed and the remaining edges are in $F$. Lemma 5.5 then shows that $F$, $B''$, and $\sigma$ satisfy $(i)$. Note that for both $(i)$ and $(ii)$ we must have $e \in \mathbf{a}_{p,j'}^1$ where $\sigma(\mathbf{a}_{p,j'}^1) = \sigma(\mathbf{a}_p^1)$. Hence, $|\mathbf{a}_{p,j'}^1 \cap B''| > |\mathbf{a}_{p,j'}^1 \cap B|$, and we get by induction that $h^{1P}(F, B'', \sigma) \geq f^{1P}(N(F,p), \hat{\sigma})$. Note that if $|\mathbf{a}_{p,j}^1 \cap B| = s$ such that $\mathbf{a}_{p,j}^1 \subseteq B$, then it was not possible to pick $e \in \mathbf{a}_{p,j}^1 \setminus B$, and case 3 could not happen. The base-case $|\mathbf{a}_{p,j}^1 \cap B| = s$ then follows from the proof of the other four cases.

**Case 5:** Assume that $e$ does not qualify for any of the first four cases. We show that $h^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p), \hat{\sigma})$, i.e., in this case we only count the improving switches performed during the first recursive call.

Observe that $N(F \setminus \{e\}, p) = N(F, p)$ since, due to cases 1 and 2, $e \notin \mathbf{b}_{i'}^1$ for some $i' \in N(F,p)$. We also claim that $\mathit{reset}((F \setminus \{e\}) \cup B) = 0$. Assume for the sake of contradiction that there exists an index $i \in [n]$ such that $\mathbf{b}_i^1 \subseteq (F \setminus \{e\}) \cup B$ and $\mathbf{a}_i^1 \not\subseteq (F \setminus \{e\}) \cup B$. Then it must be the case that $\mathit{bit}_i(F \cup B, B) \neq 1$, since otherwise the relevant edges are part of $B$. On the other hand, we can also not have $\mathit{bit}_i(F \cup B, B) = 0$ since $\sigma$ is well-behaved. The only remaining possibility is $i = p$, but this case has been ruled out due to case 4. The claim follows. Observe also that due to cases 1, 2, and 4 we can not have $\ell = \sigma(\mathbf{b}_p^1)$ or $\ell = \sigma(\mathbf{a}_p^1)$. The remaining requirements for conditions $(i)$ and $(ii)$ are not affected by removing $e$ from $F$, and it follows that if $F$, $B$, and $\sigma$ satisfy $(i)$ or $(ii)$, respectively, then $F \setminus \{e\}$, $B$, and $\sigma$ satisfy $(i)$ or $(ii)$, correspondingly. It remains to show that $F \setminus \{e\}$ is functional. This follows from $\sigma$ being well-behaved and $\ell \leq \sigma(\mathbf{a}_p^1)$.

Since $|F \setminus \{e\}| < |F|$ it follows by induction that $h^{1P}(F \setminus \{e\}, B, \sigma) \geq f^{1P}(N(F,p), \hat{\sigma})$. Note that it is not possible to reach a situation where $F$ is not functional. Hence, the induction always goes back to one of the other base-cases. The base-case where $F$ is not functional therefore follows from the proof of the other four cases. □

# 9 Concluding remarks and open problems

We obtained an $2^{\tilde{\Omega}(\sqrt[3]{m})}$ lower bound for Random-Facet. The question whether this can be improved to $2^{\tilde{\Omega}(\sqrt{m})}$, which would then be tight, is an interesting open problem. We also obtained $2^{\tilde{\Omega}(\sqrt{m})}$ lower bounds for Random-Facet$^{1P}$ and Random-Bland. No subexponential upper bounds are currently known for these two algorithms. Obtaining such upper bounds, or further improving our lower bounds, are also interesting open problems.

Our lower bounds were all obtained using shortest paths problems. Another interesting open problem is whether similar lower bounds can be obtained using graphs in which the out-degree of each vertex is 2. Such *binary* instances would supply explicit *Acyclic Unique Sink Orientations* (AUSOs) on which Random-Facet takes subexponential time. (For more on AUSOs, see Gärtner [20], Matoušek [33], Matoušek and Szabó [36], Schurr and Szabó [40, 41], and Szabó and Welzl [42].)

# Acknowledgement

We would like to thank Bernd Gärtner, Günter Rote and Tibor Szabó for various discussions on the Random-Facet algorithm and its variants that helped us realize that the expected number of pivoting steps performed by Random-Facet and Random-Facet$^{1P}$ are *not* the same.

# References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network flows – Theory, algorithms and applications.* Prentice Hall, 1993.

[2] N. Amenta and G. Ziegler. Deformed products and maximal shadows of polytopes. In *Advances in Discrete and Computational Geometry*, pages 57–90, Providence, 1996. Amer. Math. Soc. Contemporary Mathematics 223.

[3] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[4] D. Bertsimas and J. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific, 1997.

[5] H. Björklund and S. Vorobyov. Combinatorial structure and randomized subexponential algorithms for infinite games. *Theoretical Computer Science*, 349(3):347–360, 2005.

[6] H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.

[7] R. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977.

[8] V. Chvátal. *Linear programming.* A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York, 1983.

[9] G. Dantzig. *Linear programming and extensions.* Princeton University Press, 1963.

[10] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[11] J. Fearnley. Exponential lower bounds for policy iteration. In *Proc. of 37th ICALP*, pages 551–562, 2010.

[12] P. Flajolet and R. Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009.

[13] L. R. Ford. Network flow theory. Technical Report P-923, RAND Corporation, 1956.

[14] O. Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *Proc. of 24th LICS*, pages 145–156, 2009.

[15] O. Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *Logical Methods in Computer Science*, 7(3), 2011.

[16] O. Friedmann. A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In *Proc. of 15th IPCO*, pages 192–206, 2011.

[17] O. Friedmann, T. D. Hansen, and U. Zwick. A subexponential lower bound for the random facet algorithm for parity games. In *Proc. of 22nd SODA*, pages 202–216, 2011.

[18] O. Friedmann, T. D. Hansen, and U. Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proc. of 43th STOC*, pages 283–292, 2011.

[19] O. Friedmann, T. D. Hansen, and U. Zwick. Errata for: A subexponential lower bound for the random facet algorithm for parity games. 2014. Available at: `www.cs.au.dk/~tdh/papers/errata.pdf`.

[20] B. Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures and Algorithms*, 20(3):353–381, 2002.

[21] M. Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACT News*, 26(2):96–104, 1995.

[22] N. Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007.

[23] T. D. Hansen. *Worst-case Analysis of Strategy Iteration and the Simplex Method.* PhD thesis, Aarhus University, 2012. Available at: `www.cs.au.dk/~tdh/papers/dissertation.pdf`.

[24] R. Howard. *Dynamic programming and Markov processes.* MIT Press, 1960.

[25] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008.

[26] G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proc. of 24th STOC*, pages 475–482, 1992.

[27] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Mathematical Programming*, 79:217–233, 1997.

[28] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[29] L. Khachiyan. A polynomial time algorithm in linear programming. *Soviet Math. Dokl.*, 20:191–194, 1979.

[30] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, New York, 1972.

[31] V. Lifschitz and B. Pittel. The number of increasing subsequences of the random permutation. *Journal of Combinatorial Theory, Series A*, 31(1):1–20, 1981.

[32] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.

[33] J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures and Algorithms*, 5(4):591–608, 1994.

[34] J. Matoušek and B. Gärtner. *Understanding and using linear programming.* Springer, 2007.

[35] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

[36] J. Matoušek and T. Szabó. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 204(1):262–277, 2006.

[37] V. Petersson and S. Vorobyov. A randomized subexponential algorithm for parity games. *Nord. J. Comput.*, 8(3):324–345, 2001.

[38] M. Puterman. *Markov decision processes.* Wiley, 1994.

[39] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

[40] I. Schurr and T. Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004.

[41] I. Schurr and T. Szabó. Jumping doesn't help in abstract cubes. In *Proc. of 9th IPCO*, pages 225–235, 2005.

[42] T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc. of 42th FOCS*, pages 547–555, 2001.

[43] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games (Extended abstract). In *International Conference on Computer-Aided Verification, CAV 2000*, volume 1855 of *LNCS*, pages 202–215, 2000.

# A Proof of Lemma 4.1

**Proof:** We must show that:

$$f(n) \;=\; \sum_{k=1}^{n} \frac{1}{k!}\binom{n}{k} \,.$$

Observe that:

$$\frac{1}{n}\sum_{i=0}^{n-1}\sum_{k=1}^{i}\frac{1}{k!}\binom{i}{k} \;=\; \frac{1}{n}\sum_{k=1}^{n-1}\frac{1}{k!}\sum_{i=k}^{n-1}\binom{i}{k} \;=\; \frac{1}{n}\sum_{k=1}^{n-1}\frac{1}{k!}\binom{n}{k+1} \;=\; \sum_{k=1}^{n-1}\frac{1}{(k+1)!}\binom{n-1}{k} \;=\; \sum_{k=2}^{n}\frac{1}{k!}\binom{n-1}{k-1} \,.$$

Then by induction:

$$\begin{aligned} f(n) &= f(n-1)+1+\frac{1}{n}\sum_{i=0}^{n-1} f(i) \\ &= \sum_{k=1}^{n-1}\frac{1}{k!}\binom{n-1}{k} + 1 + \frac{1}{n}\sum_{i=0}^{n-1}\sum_{k=1}^{i}\frac{1}{k!}\binom{i}{k} \\ &= \sum_{k=1}^{n-1}\frac{1}{k!}\binom{n-1}{k} + 1 + \sum_{k=2}^{n}\frac{1}{k!}\binom{n-1}{k-1} \\ &= \sum_{k=1}^{n}\frac{1}{k!}\binom{n-1}{k} + \sum_{k=1}^{n}\frac{1}{k!}\binom{n-1}{k-1} \\ &= \sum_{k=1}^{n}\frac{1}{k!}\binom{n}{k} \,. \end{aligned}$$

□

# B Proof of Lemma 4.3

**Proof:** We must show that $f(n) = f^{1P}(n)$.

The lemma is proved by using induction and linearity of expectation. For $n = 0$ we have $f(0) = f^{1P}(0) = 0$. Let $\Sigma(N)$, for $N \subseteq [n]$, be the set of permutations of $N$, i.e., every $\sigma \in \Sigma(N)$ is a map $\sigma : N \to [|N|]$. Note that $|\Sigma(N)| = |N|!$. Note also that $f^{1P}(N,\sigma) = f^{1P}(N,\sigma')$ where $N \subseteq [n]$ and $\sigma \in \Sigma([n])$, and where $\sigma' \in \Sigma(N)$ is obtained by compressing $\sigma$. We see that:

$$\begin{aligned} f^{1P}(n) &= \frac{1}{n!}\sum_{\sigma\in\Sigma([n])} f^{1P}([n],\sigma) \\ &= \frac{1}{n}\sum_{i\in[n]}\frac{1}{(n-1)!}\sum_{\sigma\in\Sigma([n]\setminus\{i\})} f^{1P}([n]\setminus\{i\},\sigma)+1+f^{1P}([i-1],\sigma) \\ &= \frac{1}{n}\sum_{i\in[n]} f^{1P}(n-1)+1+f^{1P}(i-1) \\ &= f(n-1)+1+\frac{1}{n}\sum_{i\in[n]} f(i-1) \\ &= f(n) \,. \end{aligned}$$

□