Efficient algorithms for three-dimensional
axial and planar random assignment problems

Alan Frieze*   Gregory B. Sorkin†

October 9, 2013

Abstract

Beautiful formulas are known for the expected cost of random two-dimensional assignment problems, but in higher dimensions even the scaling is not known. In three dimensions and above, the problem has natural "Axial" and "Planar" versions, both of which are NP-hard. For 3-dimensional Axial random assignment instances of size n, the cost scales as $\Omega(1/n)$, and a main result of the present paper is a linear-time algorithm that, with high probability, finds a solution of cost $O(n^{-1+o(1)})$. For 3-dimensional Planar assignment, the lower bound is $\Omega(n)$, and we give a new efficient matching-based algorithm that with high probability returns a solution with cost $O(n \log n)$.

1 Introduction

An instance of the (two-dimensional) assignment problem may be thought of as an $n \times n$ cost array $M_{i,j}$, a candidate solution is a permutation $\pi \colon [n] \mapsto [n]$, its cost is $\sum_{i=1}^{n} M_{i,\pi(i)}$, and an optimal solution is one minimizing the cost. If the cost matrix represents, for example, the costs of assigning various jobs i to machines j, where each machine can accommodate only one job, then the problem's solution represents the cheapest way of assigning the jobs to machines. It may equivalently be formulated as an integer linear program, minimizing the sum of selected elements consistent with the selection of exactly one element from each row and from each column, i.e., minimizing $\sum_{i,j} M_{i,j} X_{i,j}$ where $X_{i,j} \in \{0,1\}$, $(\forall i)$ $\sum_{j} X_{i,j} = 1$ and $(\forall j)$ $\sum_{i} X_{i,j} = 1$. This is a network flow problem, thus its linear relaxation with $X_{i,j} \in [0,1]$ has integer extreme points, and the problem may be solved in polynomial time.

The *random assignment problem*, in its most popular form, is the case when the entries of the cost matrix C are i.i.d. Exp(1) random variables (independent, identically distributed exponential random variables with parameter 1). Since the problem can be solved in polynomial time, the focus for the random case is on the cost's expectation as a function of n,

$$f(n) = \mathbf{E}\left[\min_{\pi} \sum_{i=1}^{n} M_{i,\pi(i)} \right] = \mathbf{E}\left[\min_{X_{i,j}} \sum_{i,j} M_{i,j} X_{i,j} \right]$$

with $X_{i,j} \in \{0,1\}$ subject to the row and column constraints above. This problem has received a great deal of study over several decades. It was considered from an operations research perspective in the 1960s [Don69], an asymptotic conjecture $f(n) \to \pi^2/6 = \zeta(2)$ was formulated by statistical

physicists Mézard and Parisi in the 1980s based on the mathematically sophisticated but non-rigorous "replica method" [MP85, MP87], an exact conjecture $f(n) = \sum_{i=1}^{n} 1/i^2$ was hazarded by Parisi in the late 1990s [Par98] a generalization to partial matchings and non-square matrices was made by Coppersmith and Sorkin [CS99], the Mézard–Parisi conjecture was proved by Aldous in a pair of papers in 1992 and 2001 [Ald92, Ald01], and the Coppersmith–Sorkin conjecture was proved simultaneously in 2004 by two papers using two very different methods, by Nair, Prabhakar and Sharma [NPS05], and Linusson and Wästlund [LW04]. A further generalisation of these conjectures was made by Buck, Chan and Robbins in 2002 [BCR02] and proved by Wästlund in 2005 [Wäs05]. The study of other aspects of the random assignment problem and related problems is ongoing, for example by Wästlund in [Wäs09].

In higher dimensions there are two natural generalizations of the assignment problem. For example in three dimensions, the Axial assignment problem is, given an $n \times n \times n$ matrix (or "tensor" or "array") C, to find a solution $X_{i,j,k}$ minimizing $\sum_{i,j,k} M_{i,j,k} X_{i,j,k}$ where $X_{i,j,k} \in \{0,1\}$ and there is one selected value per "plane" of the array, of which there are three types, 1-, 2-, and 3-planes according to which coordinate is fixed:

$$(\forall i) \sum_{j,k} X_{i,j,k} = 1, \quad (\forall j) \sum_{i,k} X_{i,j,k} = 1, \quad (\forall k) \sum_{i,j} X_{i,j,k} = 1. \tag{1}$$

Equivalently, the Axial problem is to determine $\min_{\pi,\sigma} \sum_{i=1}^{n} M_{i,\pi(i),\sigma(i)}$, the minimum taken over a pair of permutations. The Planar three-dimensional assignment problem is similar but with one selected value per "line" of the array, with three types of lines:

$$(\forall i,j) \sum_{k} X_{i,j,k} = 1, \quad (\forall j,k) \sum_{i} X_{i,j,k} = 1, \quad (\forall i,k) \sum_{j} X_{i,j,k} = 1. \tag{2}$$

The generalizations to higher dimensions are clear. In three dimensions and higher, the Axial and Planar assignment problems are both NP-hard. The Axial case in three dimensions was one of the original problems listed by Karp [Kar72]. The complexity of the Planar problem was established by Frieze [Fri83].

The *multi-dimensional random assignment problem* we consider here is the case when the entries of the cost matrix are i.i.d. Exp(1) random variables. In this random setting, there are two natural questions. First, are there polynomial-time algorithms that find optimal or near-optimal solutions w.h.p.?[1] Second, what is the expected cost of a minimum assignment? A random two-dimensional assignment instance has limiting expected cost $\zeta(2)$, and Frieze showed that the expected cost of a minimum spanning tree in the complete graph with random Exp(1) edge weights tends to $\zeta(3)$ [Fri85], so it is tantalizing to wonder if there might be similarly beautiful expressions for the expected cost in multi-dimensional versions of the random assignment problem. However, we do not even know how the cost scales with n.

Some of the characteristics and applications of these problems are discussed in a recent book by Burkard, Dell'Amico and Martello [BDM09]. Little is known about the probabilistic behavior of these problems for dimension at least three, and even less about polynomial-time algorithms for constructing good solutions. First consider the Axial problem with constraints (1). Grundel, Krokhmal and Pardalos [KGP07] replace the Exp(1) assumption with more general distributions. Where $F(x)$ is the probability that $M_{i,j,k} \leq x$, their result most relevant to our discussion is that if $F^{-1}(x) = O(x^{\beta})$ for some $\beta > 0$ as $x \to 0^+$ then the minimum value $Z_{d,n}^{A} \to 0$ w.h.p. It is not difficult to prove $Z_{d,n}^{A} \to 0$ w.h.p. for $M_{i,j,k} = \text{Exp}(1)$ (see Remark 2), but we are interested in

---

[1]A sequence of events $\mathcal{E}_n, n \geq 0$ is said to occur *with high probability* (w.h.p.) if $\lim_{n \to \infty} \mathbf{Pr}(\mathcal{E}_n) = 1$.

the precise convergence rate. Kravtsov [Kra05] describes a class of greedy algorithms that work well if Exp(1) is replaced by uniform over $\{1, 2, \ldots, n^\alpha\}$ where $\alpha < 1$ depends on the particular algorithm. The lower bound of 1 means that the minimum is at least n and this is not a difficult target, asymptotically.

The Planar problem with constraints (2) was considered by Dichkovskaya and Kravtsov [DK07]. Here they discuss a "greedy" algorithm similar to that proposed by us. Their analysis is quite different and their distribution is (as with [Kra05]) uniform over $\{1, 2, \ldots, n^\alpha\}$ where $\alpha < 1$. This makes the minimum at least n^2 and again this is not a difficult target, asymptotically.

Statistical physicists Martin, Mézard and Rivoire conjecture in [MMR05] that the Axial problem has an asymptotic expected cost of c/n, based on "cavity" calculations; see also [MMR04]. However, there is no nice constant like $\zeta(2)$ predicted, and no certainty that the conjectured scaling is correct.

2 Summary of results, methods, and limitations, and outline

2.1 Axial assignment

For the Axial D-dimensional assignment problem, there is an easy lower bound of $\Omega(1/n^{D-2})$ on the expected cost (see Theorem 1). Our main result (Theorem 2) is for the case $D = 3$. Here we give an algorithm that runs in time linear in n and yields w.h.p. a solution of cost $O(1/n^{1-o(1)})$, an $n^{o(1)}$ approximation to the best possible. Not only is this the first nearly tight upper bound obtained algorithmically, it is the only good bound except for one (see Theorem 1) following from a recent non-constructive result on hypergraph factors by Johansson, Kahn and Vu [JKV08].

Our algorithm may be compared with one in [CS99] for 2-dimensional assignment. There, a bipartite matching was augmented by an alternating path of bounded length, with care taken in regard to "conditioning" of the cost matrix. Here, partial assignments are augmented with a "bounded-depth alternating-path tree", a tree in which a newly added element displaces two previously selected elements, those two elements are replaced in a way displacing four selected elements, and so on, until all the displaced elements are replaced by elements in a non-conflicting, "unassigned" set (see Figure 1).

For dimensions $D \geq 4$ the algorithm can still be applied, but it produces a solution whose expected cost is of order $\Omega(n^{-1})$, just as in the 3-dimensional case; for $D \geq 4$ this is far from the upper bound of $O(n^{2-D} \log n)$ given in Theorem 1.

2.2 Planar assignment

Our second main result (Theorem 3), establishes high-probability lower and upper bounds for the cost of a Planar 3-dimensional assignment. A lower bound of $\Omega(n)$ comes from attending to only the first constraint in (2). We then show that a 3-dimensional Planar assignment consists of n 2-dimensional assignments, with constraints between them. An upper bound is derived from a greedy algorithm that solves the 2-dimensional assignments sequentially, respecting the constraints between them. These 2-dimensional assignment instances have a somewhat complex structure, but a general result of Dyer, Frieze, and McDiarmid [DFM86] is insensitive to the details and yields an upper bound of $O(n \log n)$.

As for the Axial case, though, our approach to the Planar problem falters for dimensions $D \geq 4$. The natural generalization is to a greedy algorithm that sequentially solves n $(D-1)$-dimensional instances, but even for $D = 4$ such an algorithm can fail, reaching an instance that has no solution, regardless of cost.
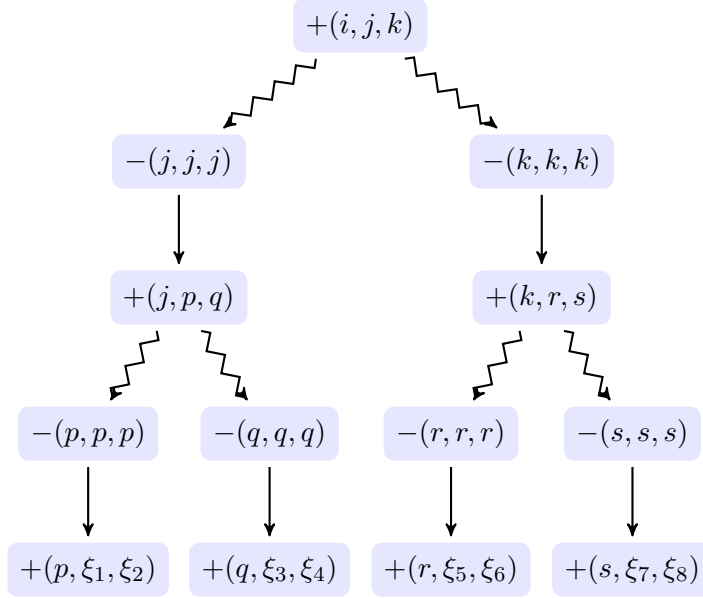
Figure 1: Diagram showing how a depth-2 "augmentation" (or alternating-path tree) increases an assignment's cardinality by 1. This is the basis of algorithm BDAPTA(2); see Section 4.2. Without loss of generality we assume that all current assignment elements are of the form (u, u, u) Adding new first coordinate i to the partial assignment, using element (i, j, k), implies deletion of previous assignment elements (j, j, j) and (k, k, k). Their first coordinates j and k are then reassigned respectively to elements (j, p, q) and (k, r, s), displacing four more existing assignment elements (p, p, p) etc. Their first coordinates are finally reassigned to unused second and third coordinates by using elements (p, ξ_1, ξ_2) etc.

2.3 Structure of the paper

Our results for Axial assignment are formally stated in Section 3. There, Theorem 1 gives a simple lower bound, and a non-constructive upper bound that is also proved easily but only by relying on some heavy machinery. Theorem 2 gives our main Axial result, a constructive upper bound for 3-dimensional assignment ($D = 3$) coming from a bounded-depth alternating-path tree algorithm BDAPTA=BDAPTA(d), where $2d$ is the depth of the search tree. The algorithm and its analysis are undeniably complicated. To provide intuition, BDAPTA(2) is described and analyzed in Section 4. The general algorithm BDAPTA(d) is analyzed in Section 5, completing the proof of Theorem 2. Difficulties for dimensions $D > 3$ are sketched in Section 6.

The Planar problem is considered in Section 7. In Section 7.1, Theorem 3 states a simple lower bound and a constructive upper bound. These are proved in Sections 7.2 and 7.3 respectively. Difficulties for dimensions $D > 3$ are sketched in Section 8.

3 Multi-dimensional Axial assignment results

3.1 Simple bounds

Theorem 1. *The minimum, $Z_{D,n}^A$, of the D-dimensional Axial random assignment problem satisfies*

$$\Omega\left(\frac{1}{n^{D-2}}\right) \le Z_{D,n}^A \le O\left(\frac{\log n}{n^{D-2}}\right) \quad w.h.p.$$

4

Proof. Clearly

$$Z_{D,n}^A \geq \sum_{i_1=1}^n \min_{i_2,\dots,i_D} M_{i_1,\dots,i_D}.$$

Each summand is distributed as $\text{Exp}(n^{D-1})$ and so has expectation $1/n^{D-1}$ and variance $1/n^{2D-2}$. The summands are independent, and the Chebyshev inequality can be used to show that the sum is concentrated around the mean.

For the upper bound we use a recent result of Johansson, Kahn and Vu [JKV08] on perfect matchings in random D-uniform hypergraphs. Their result implies that w.h.p. there is a solution that only uses D-tuples of weight at most $\frac{K \log n}{n^{D-1}}$, thereby giving an upper bound for "Shamir's problem". While [JKV08] does not deal with D-partite hypergraphs, we have verified that their result can be extended to this case; this was also done independently by Gerke and McDowell for different purposes [GM13]. The upper bound for D-dimensional Axial assignment follows immediately. $\square$

The proof in [JKV08] is emphatically non-constructive.

3.2 Main result

We remind the reader that a description of the algorithm $\texttt{BDAPTA}(d)$ for 3-dimensional Axial assignment ($D = 3$) will follow in Section 4 (for $d = 2$) and Section 5 (for general d). Throughout the paper, we associate the element $M_{i,j,k}$ with the "triple" (i, j, k).

Theorem 2. *Suppose that $1 \leq d \leq \varepsilon \log_2 \log n$ where where $0 < \varepsilon < 1/2$ is a constant. For a random 3-dimensional Axial assignment instance, w.h.p.:*

(a) *Algorithm $\texttt{BDAPTA}(d)$ runs in time $O(n^3)$.*

(b) *The cost $M(T)$ of the set of triples T output by $\texttt{BDAPTA}(d)$ satisfies $M(T) = O(2^{4d} n^{-1+\eta_d} \log n)$, where $\eta_d = \frac{1}{2^{d+1}-1}$.*

By taking $d = \varepsilon \log_2 \log n$ we see that have a linear-time algorithm that w.h.p. gives a solution of value $O(n^{-1+o(1)})$. (The exponent's $o(1)$ is $\frac{1}{2} \log^{-\varepsilon} n$ plus lower-order terms.) As already noted, we are not aware of any other polynomial-time algorithm that will w.h.p. find a solution of value $O(n^{-1+o(1)})$, or indeed any non-trivial bound including the $O(n^{-6/7})$ that $\texttt{BDAPTA}(2)$ gives.

4 3-Dimensional Axial assignment 2-level algorithm, $\texttt{BDAPTA}(2)$

In this section we consider a two-level version of the algorithm $\texttt{BDAPTA}$ for 3-dimensional Axial assignment. In this way we hope to make it easier to understand the general version described in Section 5. With reference to Theorem 2, the two-level version means taking $d = 2$, $\eta = \eta_2 = 1/7$. The algorithm has three phases, treated respectively in the next three subsections.

4.1 Greedy Phase

This phase uses a simple greedy procedure, shown below as Algorithm 1, to construct a low-cost partial assignment of cardinality $(1 - o(1))n$. Let

$$n_1 = n - n^{1-\eta} \qquad \text{and} \qquad w_0 = 2n^{-2(1-\eta)} \log n. \tag{3}$$

In the following we will pretend that n_1 is integral, as we will do throughout when it makes no difference. There will be cases where integrality is significant, and there we will be precise.

Algorithm 1 GREEDYPHASE: construct a cardinality-n_1 partial assignment

---

1: Let $B := C := [n]$, and $T := \emptyset$ (respectively the set of unassigned 2- and 3-coordinates, and the set of triples assigned so far)
2: **for** $i = 1, \ldots, n_1$ **do**
3: Locate $M_{i,j,k} = \min \left\{ M_{i,j',k'} : j' \in B, k' \in C \right\}$
4: **if** $M_{i,j,k} > w_0$ **then** FAIL
5: Add (i, j, k) to T and remove j from B and k from C
6: Return the set of triples in T as a partial assignment

Lemma 1. *With high probability the procedure does not fail, and yields*

$$Z_1 := \sum_{(i,j,k) \in T} M_{i,j,k} \leq \frac{2}{n^{1-\eta}}.$$

Proof. We observe that if $(i, j, k) \in T$ then $M_{i,j,k}$ is the minimum of $(n - i + 1)^2$ independent copies of Exp(1) and is therefore distributed as Exp$((n - i + 1)^2)$. Furthermore, the random variables $M_{i,j,k}, (i, j, k) \in T$ are independent (each drawn from a different 1-plane as given by i).

The procedure fails only if some $M_{i,j,k} > w_0$. Even in the last round $i = n_1$, the chance that all $(n^{1-\eta})^2$ elements considered are larger than w_0 is $\exp(-n^{2(1-\eta)} w_0) = \exp(-2 \log n)$. By the union bound, the probability of a failure in any round is $\leq n \exp(-2 \log n) = n^{-1}$. We now ignore this aspect and imagine running the algorithm without the failure condition.

Using the fact that an Exp(λ) random variable has mean $1/\lambda$ and variance $1/\lambda^2$,

$$\mathbf{E}(Z_1) = \sum_{i=1}^{n_1} \frac{1}{(n - i + 1)^2} \leq \int_{x=1}^{n_1+1} \frac{dx}{(n - x + 1)^2} \leq \frac{1}{n^{1-\eta}}.$$

Now

$$\mathbf{Var}(Z_1) = \sum_{i=1}^{n_1} \frac{1}{(n - i + 1)^4} \leq \frac{3}{n^{3(1-\eta)}} = o(\mathbf{E}(Z_1)^2)$$

and the lemma follows from the Chebyshev inequality. $\qquad \square$

Remark 2. *If we replace n_1 by $n - \omega$ where $\omega = \omega(n) \to \infty$ and then do exhaustive search for the optimum solution in the remaining ω-size problem then we see that (i) the expected optimum value is bounded by $O\left(\frac{1}{\omega} + \frac{\log \omega}{\omega}\right)$ (greedy plus exhaustive search costs), and (ii) the running time is bounded by $O\left(n^3 + \omega^{2\omega}\right)$. So, if $\omega = O(\log n / \log \log n)$, the algorithm is polynomial and produces a solution with a cost that is $o(1)$, in expectation and w.h.p..*

This justifies our introductory remark that it is not difficult to prove $Z_{d,n}^A \to 0$ w.h.p. for $M_{i,j,k} = \text{Exp}(1)$. (We do not even need the breakthrough result of [JKV08] for this. The early reult of Schmidt-Pruzan and Shamir [SPS85] would give $O\left(\frac{1}{\omega} + \frac{\log \omega}{\omega^{1/2}}\right)$ for the cost).

4.2 Main Phase

This phase will increase the size of the partial assignment defined by T to $n - 3$. The phase will proceed in *rounds*. At the start of each round we assume that the assignment elements T_t are $(1, 1, 1), \ldots, (n_t, n_t, n_t)$, which we can arrange simply by permuting the array M. Let $A = A(T)$ be the set of 1-coordinates assigned in T and define B and C analogously for the 2- and 3-coordinates.

Of course initially $A = B = C = \{1, \ldots, n_t\}$ but the notation will be convenient. Also, let $\bar{A}(T) = [n] - A(T)$, the set of unmatched 1-indices, and likewise define $\bar{B}$ and $\bar{C}$.

Round t will reduce the number of unmatched elements $|\bar{A}|$ by a factor β, from x_t to x_{t+1}, while increasing the total cost of the matching by an acceptably small amount, terminating in the round τ where $|\bar{A}|$ is reduced to 3. Because the sizes x we are dealing with here become small, we cannot decently ignore integrality even though its effects are minor. Specifically, then, we take

$$\alpha = 1/65, \qquad \beta = 1 - \alpha,$$

$$x_1 = n - n_1 = \lfloor n^{1-\eta} \rfloor = \lfloor n^{6/7} \rfloor, \text{ and } x_t = \lfloor \beta^{t-1} x_1 \rfloor \text{ for } t \geq 2. \qquad (4)$$

Note that (4) means that x_t decreases by 1 in each of the last few rounds, so the last round begins with $x_\tau = 4$. It is clear that

$$x_t \leq \beta^{t-1} x_1, \quad \text{thus} \quad \tau \leq \log_{1/\beta}(x_1/4) = O(\log n). \qquad (5)$$

Recall from the Greedy Phase that

$$w_0 = n^{-12/7} \log n,$$

and, for $t \geq 1$, let

$$\left. \begin{array}{l} \rho_t = n^{-6/7} x_t^{-8/7} \log n, \\ w_t = -\log(1 - \rho_t) = (1 + o(1))\rho_t, \\ W_t = w_0 + w_1 + \cdots + w_t. \end{array} \right\} \qquad (6)$$

At the start of round t we will have revealed all elements (and only elements) with values $\leq W_{t-1}$. This is true in round $t = 1$ by the Greedy Phase's definition. In later rounds t we use the following "good" elements, comprising a set G_t:

1. Elements in the partial assignment T_t at the start of the round.

2. Elements with $M_{i,j,k} \in (W_{t-1}, W_t]$. Conditioned upon $M_{i,j,k} > W_{t-1}$ and all history, a non-assignment element falls into this category with probability exactly $\rho_t = 1 - \exp(-w_t)$.

3. Previously revealed non-assignment elements, each taken with probability ρ_t, the selection made independently over all such elements.

(In contrast to this "sprinkling" approach, [CS99] used a slightly more complicated "refreshing" approach to obtain small improvements to the constants.) This selection assures the following property.

Property 3. G_t *comprises only elements of weight* $\leq W_t$*, contains all elements in the partial assignment, and (conditioned upon all history) contains each non-assignment element, independently, with probability* ρ_t*.*

At the start of round t let $A_t = A(T_t)$, $\bar{A}_t = \bar{A}(T_t)$, and likewise for B and C. In round t we will add $[n - x_t + 1, n - x_{t+1}]$ to A_t. To add i to A_t we replace 6 of the triples in T_t by 7 new triples (see Figure 1):

$$+ (i, j, k) - (j, j, j) - (k, k, k) + (j, p, q) + (k, r, s) - (p, p, p) - (q, q, q) - (r, r, r) - (s, s, s) +$$
$$(p, \xi_1, \xi_2) + (q, \xi_3, \xi_4) + (r, \xi_5, \xi_6) + (s, \xi_7, \xi_8). \qquad (7)$$

We call a collection of triples as in (7) an *augmentation*. Here + indicates addition of a new triple to the assignment, − indicates deletion of an existing assignment element, $j, k, p, q, r, s \in A_t$ are all distinct, $\xi_1, \ldots, \xi_7 \in \bar{B}_t$ are all distinct, $\xi_2, \ldots, \xi_8 \in \bar{C}_t$ are all distinct, and each of the triples added in (7) belongs to G_t (thus has cost at most W_t). To explain, we assign a new 1-coordinate i to a *previously used* 2-coordinate j and 3-coordinate k (unused ones are too rare and therefore too costly); these uses of j and k conflict with the previous assignment elements (j, j, j) and (k, k, k) so we remove those elements from the assignment; 1-coordinates j and k are re-added as (j, p, q) and (k, r, s) thus colliding with the previous assignment elements (p, p, p), (q, q, q), (r, r, r), and (s, s, s); and finally 1-coordinates p, q, r, s are re-added as (p, ξ_1, ξ_2) etc., where the ξ_i are 2- and 3-coordinates *not* previously assigned. One may think of (7) as a binary tree version of an alternating-path construction; we will control the cost despite the tree's expansion.

We realise that the algorithm is not yet completely specified but we postpone questions on index selection for a while longer. We first confirm that if we can always find $j, k, \ldots, \xi_8$ as in (7) then the cost of the assignment produced by the Main Phase is acceptable. Each application of (7) increases the cost by $\leq 7W_t$. Success in a round means doing this $x_t - x_{t+1}$ times, in which case the additional cost of the Main Phase will be at most 7 times

$$\sum_{t=1}^{\tau}(x_t - x_{t+1})W_t = x_1 w_0 - x_{\tau+1}W_\tau + \sum_{t=1}^{\tau} x_t w_t,$$

which (recalling (5)) is at most $7 + o(1)$ times

$$n^{-6/7}\log n + \log n \sum_{t=1}^{\tau} x_t(n^{-6/7}x_t^{-8/7})$$

$$\leq n^{-6/7}\log n + n^{-6/7}\log n \sum_{t=0}^{\tau-1}(\beta^t x_1)^{-1/7}.$$

The last is an increasing geometric series, summing to $\leq ((1/\beta)^\tau / x_1)^{1/7}/(1 - \beta^{1/7})$. By (5), $(1/\beta)^\tau \leq x_1/4$, so the expression above is

$$\leq n^{-6/7}\log n + n^{-6/7}\log n \frac{(1/4)^{1/7}}{1 - \beta^{1/7}}$$

$$= O(n^{-6/7}\log n). \tag{8}$$

With reference to Figure 2, we now complete the description of the algorithm by explaining how we search for $j, k, \ldots, \xi_8$ that satisfy (7). We show later that w.h.p. all of our searches succeed. Since we are considering just round t, we drop the superscripts and write, for example, B rather than B_t.

We will first show how to construct sets $J = S_1^1, \ldots, \Xi_8 = S_8^3$, then show how to draw from these sets αx collections of elements $j \in J$, $\ldots$, $\xi_8 \in \Xi_8$, each such collection augmenting the assignment as per (7). Within each augmenting set, each chosen element will force the choice of the two elements below it. For convenience, especially in the general case but even for the depth $k = 2$ case now, it is convenient to think of these sets as $S_1^1, \ldots, S_8^3$ as shown in the figure.

We construct the sets S_m^ℓ from bottom up. At the bottom level, $\ell = 3$, "equipartition" $\bar{B}$ into sets $\Xi_1, \Xi_3, \Xi_5, \Xi_7$. To be precise, partition $\bar{B}$ into sets $\Xi_1, \ldots, \Xi_7$ each of size $\lfloor |\bar{B}|/4 \rfloor$, and a fifth "discard" set. (Normally we would ignore integrality, but for instance in the second-last
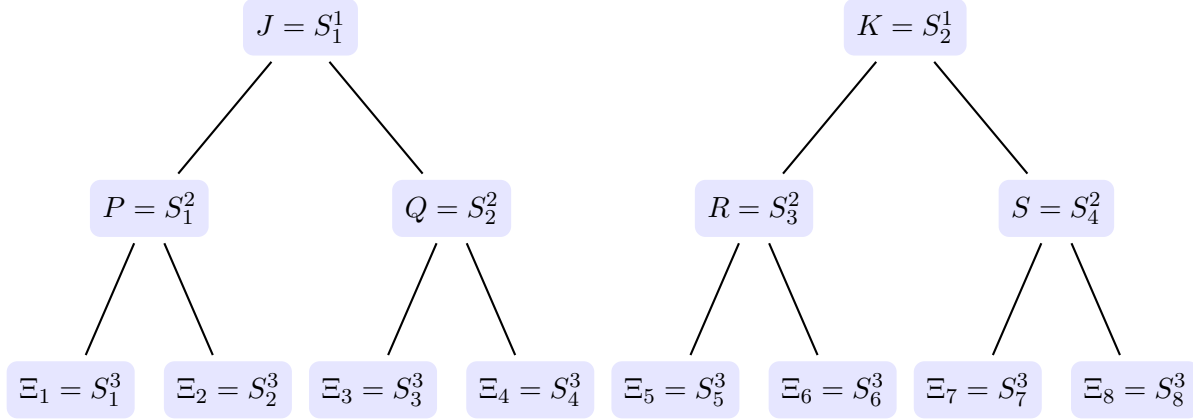
Figure 2:

round $\bar{B}$ will have size 5.) Similarly, equipartition $\bar{C}$ into $\Xi_2, \Xi_4, \Xi_6, \Xi_8$. Note that for $x \geq 4$, $x - 4 \lfloor x/4 \rfloor < 4 \leq 4 \lfloor x/4 \rfloor$, thus $x < 8 \lfloor x/4 \rfloor$, and thus each set Ξ has cardinality at least $x/8$.

For $\ell < 3$, each set $U = S_m^\ell$ in Figure 2 is constructed from its two child sets $V = S_{2m-1}^{\ell+1}$ and $W = S_{2m}^{\ell+1}$ and from a set X of "available" 1-coordinates (soon to be defined precisely) by invoking a procedure TRIPLES(X, V, W). Recall that $G = G_t$ is the set of "good" elements for this round. Intuitively, we want TRIPLES(X, V, W) to be something like the set of all elements $x \in X$ for which there are a $v \in V$ and $w \in W$ such that (x, v, w) is of low cost, i.e., something like $\text{proj}_1 ((X \times V \times W) \cap G)$, where proj_i denotes the projection of a set onto its ith coordinate. The more complicated definition of TRIPLES below enables Property 4 and simplifies the analysis later.

Algorithm 2 TRIPLES: construct set U from sets X, V, W of available 1-, 2-, and 3-indices

1: Input V, W, X.
2: **if** $|X| < n - n^{19/20}$ **then** FAIL
3: Partition $X = X' \cup X''$ where $|X'| = n' := n - 2n^{19/20}$ and $|X''| \geq n^{19/20}$ (think of X'' as a "reserve" set from which elements are shifted into X' to preserve $|X'|$)
4: Let $U^\star := \emptyset$
5: **for** each pair of elements $v \in V, w \in W$ in turn **do**
6:    **for** each element $x \in X$, in turn **do**
7:       **if** $(x, v, w) \in G$ **then**
8:          $U^\star := U^\star \cup \{(x, v, w)\}$
9:          **if** $X'' = \emptyset$ **then** FAIL
10:          Where x'' is the first element of X'', let $X' := (X' \setminus x) \cup \{x''\}$, and let $X'' := X'' \setminus x''$
11: Let $U := \text{proj}_1(U^\star)$
12: Return $(U^\star, U, X)$

Claim 11 will show that failures within TRIPLES are highly unlikely.

Property 4. TRIPLES *has the properties that (1) each element $u \in U$ appears in exactly one triple $(u, v, w) \in U^\star$ and thus is linked to a unique $v \in V$ and $w \in W$, and (2) each pair v, w occurs in* $\text{Bin}(n', \rho)$ *triples in $U^\star$ and these binomials are independent for all v and w.*

Proof. The first property is immediate. The second comes from observing that in the algorithm's Lines 6–10 the size of X' is kept constant at n', while Property 3 means that each of the n' elements considered is, independently, present in G with probability ρ. $\qquad\square$

Given TRIPLES, it is straightforward to define all the sets in Figure 2. This is done by SET-CONSTRUCT, shown as Algorithm 3 below.

Algorithm 3 SETCONSTRUCT: for Main Phase round t, construct all sets S_m^ℓ

1: Input: sets A, B, and C of assigned 1-, 2-, and 3-indices. (Elsewhere, notationally, we have assumed $A = B = C = [n - x_t]$.)
2: Let $X := A$ and $x := n - |X|$
3: "Equipartition" $\bar{B}$ into $\Xi_1 \cup \cdots \cup \Xi_7 = S_1^3 \cup \cdots \cup S_7^3$ each of size $\lfloor x/4 \rfloor$, discarding the $x - 4\lfloor x/r \rfloor$ remaining elements of $\bar{B}$
4: Likewise, equipartition $\bar{C}$ into $\Xi_2 \cup \cdots \cup \Xi_8 = S_2^3 \cup \cdots \cup S_8^3$
5: Remembering that each application of TRIPLES makes X smaller for the next one, let

$$
\begin{aligned}
(P^\star, P, X) &= ((S_1^2)^\star, S_1^2, X) &:= & \text{ TRIPLES}(X, S_1^3, S_2^3) &= & \text{ TRIPLES}(X, \Xi_1, \Xi_2) \\
(Q^\star, Q, X) &= ((S_2^2)^\star, S_2^2, X) &:= & \text{ TRIPLES}(X, S_3^3, S_4^3) &= & \text{ TRIPLES}(X, \Xi_3, \Xi_4) \\
(R^\star, R, X) &= ((S_3^2)^\star, S_3^2, X) &:= & \text{ TRIPLES}(X, S_5^3, S_6^3) &= & \text{ TRIPLES}(X, \Xi_5, \Xi_6) \\
(S^\star, S, X) &= ((S_4^2)^\star, S_4^2, X) &:= & \text{ TRIPLES}(X, S_7^3, S_8^3) &= & \text{ TRIPLES}(X, \Xi_7, \Xi_8)
\end{aligned}
$$

$$
\begin{aligned}
(J^\star, J, X) &= ((S_1^1)^\star, S_1^1, X) &:= & \text{ TRIPLES}(X, S_1^2, S_2^2) &= & \text{ TRIPLES}(X, P, Q) \\
(K^\star, K, X) &= ((S_2^1)^\star, S_2^1, X) &:= & \text{ TRIPLES}(X, S_3^2, S_4^2) &= & \text{ TRIPLES}(X, R, S)
\end{aligned}
$$

6: Return all sets S_m^ℓ and $(S^\star)_m^\ell$ defined above

---

Given a partial assignment T with $i - 1 < n$ elements, augmenting T to include 1-index i as in (7) and Figure 1 is easy as if we can find an element $(i, j, k) \in G$ with $j \in J$, $k \in K$. For, if this is so, by construction of $J^\star$ there is a unique element $(j, p, q) \in J^\star$ with $p \in P$ and $q \in Q$. Then, by construction of $P^\star$ there is a unique element $(p, \xi_1, \xi_2) \in P^\star$ with $\xi_1 \in \Xi_1$ and $\xi_2 \in \Xi_2$. Likewise, q leads uniquely to ξ_3, ξ_4, and k leads uniquely to r, s and these to $\xi_5, \ldots, \xi_8$. This collection, corresponding to the triples added in (7), immediately determines those subtracted: $-(j, j, j)$, $-(k, k, k)$, etc., and thus the full augmentation. This procedure for producing an augmentation is spelled out as MAKETREE (i, j, k), Algorithm 4. Identifying $\xi_1, \ldots, \xi_8$ with $s_1^1, \ldots, s_8^1$, this is written in a way that can be quoted for the general case.

Algorithm 4 MAKETREE (i, j, k): produce an augmentation AugOut

---

1: Input: a collection of sets as in Figure 2 as output by Algorithm 3 (MAKETREE), and a triple (i, j, k) with $i \in \bar{A}$, $j = s_1^1 \in S_1^1 = J$, $k = s_2^1 \in S_2^1 = K$.
2: Let AugOut $:= +(i, s_1^1, s_2^1)$
3: **for** $\ell = 1$ to 2 **do**
4:    **for** $m = 1$ to 2^ℓ **do**
5:       (Note that $s_m^\ell \in S_m^\ell$ is already defined)
6:       Let $(s_m^\ell, s_{2m-1}^{\ell+1}, s_{2m}^{\ell+1})$ be the unique triple in $(S^\star)_m^\ell$ with 1-coordinate s_m^ℓ (thus defining the values for Line 5 when ℓ is incremented in Line 3)
7:       Let AugOut $:=$ AugOut, $-(s_m^\ell, s_m^\ell, s_m^\ell), +(s_m^\ell, s_{2m-1}^{\ell+1}, s_{2m}^{\ell+1})$, i.e., add these two triples to the output, one negated, one positive
8: Return AugOut

Property 5. *In any augmentation constructed by* MAKETREE, *the first indices of all positive triples are distinct, as are all the second indices, and all the third indices.*

Proof. All first indices in the whole collection of sets S_m^ℓ (and therefore within one augmentation) are distinct because they are drawn from a set X ($X = A$ initially), and each first index produced is immediately removed from X.

Any second index v comes from a triple in some set $(S^\star)_m^\ell \subset X \times S_{2m-1}^{\ell+1} \times S_{2m}^{\ell+1}$, so that $v \in S_{2m-1}^{\ell+1}$. We treat this with two cases. If $\ell < d$, then $v \in S_{2m-1}^{\ell+1}$ is a first index in $(S^\star)_{2m-1}^{\ell+1}$. Such values are all distinct from one another by distinctness of first indices. If on the other hand $\ell = d$, the previous case does not apply (there are no sets $S^\star$ at depth $d+1$), but $v \in S_{2m-1}^{d+1} = \Xi_{2m-1}$ and thus belongs to one of the parts of the equipartition of $\bar{B}$. Since in an augmentation each triple is drawn from a different set Ξ, and the sets come from different parts of the equipartition of $\bar{B}$, all these values are distinct from one another. Finally, no two values from different cases can be equal: those in the first case belong to A and those in the second case belong to $\bar{B}$, and these sets are disjoint since $A = B$ by our notational assumption that the partial assignment T consists of triples of the form (i, i, i). Third indices are treated by the same argument as second indices. $\qquad\square$

We can then augment the assignment via (7) the output of MAKETREE. We now give a formal proof that MAKETREE allows a feasible augmentation. Again, we write this in a way that can be quoted for the general case.

Property 6. *The output of* MAKETREE *defines a valid assignment augmentation.*

Proof. Working backwards through the elements of AugOut we show inductively that each makes a valid change. (MAKETREE works top-down; this proof works bottom-up.) Consider adding a bottom-most triple $+(s_m^d, s_{2m-1}^{d+1}, s_{2m}^{d+1})$. Its second and third coordinates can be added, as they belong to the sets S_{2m-1}^{d+1} and S_{2m}^{d+1} which contain only unassigned indices, and (over all bottom-most triples) there are no collisions amongst these by Property 5. The first coordinate would clash with the existing assignment element, but this is removed by the AugOut element $-(s_m^d, s_m^d, s_m^d)$, also freeing up s_m^d as a 2- and 3-index. So, adding the level $\ell = d$ elements, pairing the positives and negatives (so the assignment size does not change), ensures that all their first coordinates at level $\ell = d$ are available as 2- and 3-indices at level $\ell - 1$. We refer to this as property $\mathcal{Q}$ at level $\ell = d$.

We now show that property $\mathcal{Q}$ at level $\ell + 1$ implies property $\mathcal{Q}$ at level ℓ. Consider a level-ℓ element $+(s_m^\ell, s_{2m-1}^{\ell+1}, s_{2m}^{\ell+1})$. Referring to MAKETREE's Line 6, this element's 2-index $s_{2m-1}^{\ell+1}$ (in iteration ℓ) is another element's 1-index (in iteration $\ell + 1$), and thus, by Property $\mathcal{Q}$ at level $\ell + 1$, is available for assignment. This 2-index does not conflict with any other, by Property 5. The same holds for the 3-indices. The 1-index s_m^ℓ would conflict with an existing assignment element, but this is taken care of by $-(s_m^\ell, s_m^\ell, s_m^\ell)$. That also frees up s_m^ℓ as an available 2- and 3-index, completing property $\mathcal{Q}$ at level ℓ.

When $\ell = 1$, (i, s_1^1, s_2^1) can be added: the 2- and 3-indices are available by Property $\mathcal{Q}$, while 1-index i is available (indeed the whole point of the procedure is to assign it). $\qquad\square$

However, to make more than one such augmentation using the same data structure (the same output of SETCONSTRUCT) we must ensure, for example, that no two augmentations use the same value ξ_1. To this end, if a first augmentation uses "leaf" indices $\xi_1, \ldots, \xi_8$, we regard the values $\xi_1, \ldots, \xi_8$ as "poisoned"; they in turn poison all ancestors depending on them, including this augmentation's values $j, \ldots, s$, but also other values. Contrapositively, any "healthy" (non-poisoned) elements $j' \in J$ and $k' \in K$' depend only on healthy elements ξ.

To be precise, a poisoned ξ_1 in turn poisons any ancestor $p' \in P$ for which there exists a triple $(p', \xi_1, \xi_2') \in P^\star$. Likewise the poisoned ξ_2 poisons any ancestor $p' \in P$ for which there exists a triple $(p', \xi_1', \xi_2) \in P^\star$. Note that the poisoned P values include the original p but may include

more values. (Each p is associated with a unique ξ_1 and ξ_2, but the opposite is by no means true.) We continue upward. Any poisoned $p \in P$ poisons any $j' \in J$ for which there exists a triple $(j', p, q') \in J^\star$, and poisoned q values poison additional elements of J. The poisoned J values certainly include j. Likewise, the poisoned values $\xi_5, \ldots, \xi_8$ poison subsets of R and S, which in turn poison some of K.

In general, quite simply, any poisoned element poisons all its ancestors, where ancestry is the transitive closure of the parent relation in which, for any $(u, v, w) \in (S^\star)_m^\ell$, u is a parent of v and w. (Again, Property 4 means that u has only these children, but v and w will typically each have additional parents.) The step-by-step poisoning description given in the previous paragraph will be useful for probabilistic analysis, and is shown as Algorithm 5 in a form usable for $\texttt{BDAPTA}(d)$ generally.

Algorithm 5 POISONPROPAGATE: poison all ancestors of elements used in an augmentation

---

1: Input: the output of SETCONSTRUCT with some elements poisoned, and an augmentation.
2: Poison the augmentation's leaf values $s_1^3, \ldots, s_8^3$
3: **for** $\ell = 2$ to 1 **do**
4:     **for** $m = 1$ to 2^ℓ **do**
5:         **for** each $(u, v, w) \in (S^\star)_m^\ell$ **do**
6:             **if** $v \in S_{2m-1}^{\ell+1}$ or $w \in S_{2m}^{\ell+1}$ is poisoned **then**
7:                 Poison $u \in S_m^\ell$

With this, we can present the complete Main Phase algorithm.

Algorithm 6 MAINPHASE: augment GREEDYPHASE assignment to one of cardinality $n - (2^d - 1)$

1: **for** rounds $t = 1, \ldots, \tau$ **do**
2:     (Recall that the partial assignment has cardinality $n - x_t$)
3:     Permute the cost matrix so that the partial assignment is
    $T_t = \{(1, 1, 1), \ldots, (n - x_t, n - x_t, n - x_t)\}$
4:     Let $A = B = C = [n - x_t]$ be the sets of assigned 1-, 2-, and 3-coordinates
5:     Call SETCONSTRUCT (A, B, C) and consider all elements in all its sets to be healthy
6:     (We will now assign the first $\lceil \alpha x_t \rceil$ unassigned 1-indices i, in turn)
7:     **for** $i = n - x_t + 1, \ldots, n - x_{t+1}$ **do**
8:         Let J' and K' denote the healthy subsets of J and K
9:         **if** $I' := G_t \cap (\{i\} \times J' \times K') = \emptyset$ **then** FAIL
10:         Let (i, j, k) be the lexicographically first element of I'
11:         Call MAKETREE (i, j, k) to produce an augmentation Φ
12:         Augment the assignment with Φ
13:         Call POISONPROPAGATE (Φ)

Claim 10 will show that failures within MAINPHASE are highly unlikely. First we show that if it succeeds, it returns a valid partial assignment.

Property 7. *Each augmentation in* MAINPHASE *Line 12 is valid.*

Proof. For the first augmentation in a round, this has already been justified by Property 6. To justify a later augmentation, we will show that none of its indices has appeared, in the same coordinate, in any previous augmentation within the round. Consider first a value u that appears

in a positive triple in an augmentation Φ and a previous augmentation Φ'. Bear in mind that the sets S_m^ℓ ($\ell = 1, \ldots, d+1$), the output of SETCONSTRUCT, are fixed for the round.

Consider first u drawn from S_m^ℓ for augmentation Φ, with $\ell \leq d$. By an argument like that used in the proof of Property 5, within Φ', u must have been drawn from the same set S_m^ℓ: these sets are disjoint from one another, and disjoint from each set S_m^{d+1}. But this is a contradiction, since use of Φ' would have poisoned $u \in S_m^\ell$, and Φ uses only healthy elements.

Next consider u drawn from S_{2m-1}^{d+1} for Φ, making u a 2-index in Φ. Again, by disjointness of the various sets, as a 2-index in Φ', u can only have been drawn from the same set S_{2m-1}^{d+1}. (The only other place u can appear is in a set $S_{2m'}^{d+1}$, but that would make it a 3-index.) This is a contradiction, since use of Φ' would have poisoned $u \in S_m^{d+1}$, and Φ uses only healthy elements.

The same argument goes for u drawn from S_{2m}^{d+1} for Φ.

Finally, consider the 1-index i itself whose assignment is the purpose of the augmentation Φ. It cannot appear as a 1-index in any earlier or later augmentation Φ': it does not have the same unique "root" role for any other augmentation Φ', it cannot have been picked from any set S_m^ℓ with $\ell \leq d$ since these are subsets of A while $i \notin A$, and if picked from any set S_m^{d+1} it is a 2- or 3-index not a 1-index.

We now consider a conflict involving a negative triple (u, u, u) in Φ, implying that u appears as a 1-coordinate (and not i) in a positive triple in Φ. Such a conflict with another augmentation Φ' cannot involve a negative triple (since then u would appear also as a 1-coordinate in Φ', which we have already excluded), so in Φ', u must appear in a positive triple. It cannot appear as a 1-index (already excluded). The the only 2- and 3-indices in Φ' that are not also 1-indices are those drawn from sets S_m^{d+1}, implying $u \in \bar{B} = \bar{C}$ and thus $u \notin A$, contradicting the appearance of u as a 1-index in Φ. $\qquad\square$

This concludes a proof of correctness of MAINPHASE. It remains to show that the various failure conditions are unlikely, notably that the sets J and K, initially and when restricted to their healthy elements later, are sufficiently large that failure in MAINPHASE Line 9 is unlikely. To that end we first estimate the sizes of all the sets at the beginning of a round and then analyze their losses due to poisoning.

We start by showing that the following "nominal sizes" are good estimates of the sizes of the sets S at corresponding depths:

$$
\left.
\begin{aligned}
\sigma_3 &= \lfloor x/4 \rfloor = \Theta(x) \\
\sigma_2 &= \rho n'(\sigma_3)^2 = \Theta(\rho n x^2) = \Theta(n^{1/7} x^{6/7} \log n), \text{ and} \\
\sigma_1 &= \rho n'(\sigma_2)^2 = \Theta(n^{3/7} x^{4/7} \log^2 n).
\end{aligned}
\right\}
\tag{9}
$$

Let

$$
\delta = n^{-1/100}.
\tag{10}
$$

We know that the sets $S_1^3, \ldots, S_8^3$ have size precisely σ_3, and we will show that q.s.[2] the sets $S_1^2, \ldots, S_4^2$ all have cardinality $(1 \pm \delta)\sigma_2$, and S_1^1 and S_2^1 have cardinality $(1 \pm \delta)^2 \sigma_1$, where the notation $A = (1 \pm \delta)^r B$ is shorthand for $(1 - \delta)^r B \leq A \leq (1 + \delta)^r B$. This is part of the following more detailed claim, for which we need one definition.

Definition 8. *For sets X, S', S'', and $(S^\star, S, X) = \text{TRIPLES}(X, S', S'')$, let $S^\star(\cdot, s', \cdot) \subseteq S^\star$ be those elements generated by $s' \in S'$ (that is, having s' as their second coordinate). Symmetrically, let $S^\star(\cdot, \cdot, s'') \subseteq S^\star$ be those elements generated by $s'' \in S''$.*

Claim 9. *For each depth $\ell \in \{1, \ldots, d+1\}$ and each $m \in \{1, \ldots, 2^l\}$:*

1. *The nominal sizes in (9) satisfy $\sigma_1 > \sigma_2 > \sigma_3 = \Theta(x)$.*

2. *With δ as in (10), q.s., $|S_m^\ell| = (1 \pm \delta)^{2^{d+1-\ell}-1} \sigma_\ell$.*

3. *For every pair of sets $S' = S_{2m-1}^\ell$, $S'' = S_{2m}^\ell$ (e.g., P, Q), with $S^\star = (S^\star)_m^{\ell-1}$ (e.g., $J^\star$), q.s. every $s' \in S'$ gives $|S^\star(\cdot, s', \cdot)| = (1 \pm \delta)\rho n' |S''|$ and symmetrically every $s'' \in S''$ gives $|S^\star(\cdot, \cdot, s'')| = (1 \pm \delta)\rho n' |S'|$.*

Proof. The first assertion is immediate from $n \gg x$ and thus $\rho n' x \gg 1$.[3] We prove the rest of the claim by induction on ℓ, starting with $\ell = 3$ and working backwards to $\ell = 1$. For $\ell = 3$, the second statement is immediate.

For any ℓ, the first two statements imply the third. To see this, Property 4 implies that given S'', distributionally,

$$\left|S^\star(\cdot, s', \cdot)\right| = \sum_{S''} \text{Bin}(n', \rho) = \text{Bin}(|S''| \, n', \rho). \tag{11}$$

To this we apply the Chernoff inequality

$$\mathbf{Pr}(|\text{Bin}(n, p) - np| \geq \delta np) \leq 2e^{-\delta^2 np/3} \qquad \text{for } 0 \leq \delta \leq 1 \tag{12}$$

(an easy consequence of Alon and Spencer [AS99, Theorem A.1.15]), substituting $|S''| \, n'$ for n, ρ for p, and δ for δ. Recalling from (6) that $\rho = n^{-6/7} x^{-8/7} \log n$, and from TRIPLES (Algorithm 2) that $n' = n - 2n^{19/20}$, the expectation of the binomial in (11) is

$$\left|S''\right| n' \rho = \Omega(\sigma_\ell n \rho) = \Omega(x n \rho) = \Omega(n^{1/7} x^{-1/7} \log n) = \Omega(n^{1/49} \log n).$$

From this and (10) it follows that $\delta^2 |S''| \, n' \rho = \omega(\log n)$, and thus the exponent in the Chernoff inequality (12) is of order $-\omega(\log n)$. This assures that for a given s', q.s.,

$$\left|S^\star(\cdot, s', \cdot)\right| = (1 \pm \delta) \left|S''\right| \rho n'. \tag{13}$$

Likewise, for a given s'', $|S^\star(\cdot, \cdot, s')| = (1 \pm \delta) |S'| \rho n'$. By definition of q.s. (failure probability smaller than any polynomial), the union bound shows that q.s. (13) holds for *every* s', yielding the third part of the claim.

Finally, truth of the second and third parts of the claim for ℓ implies that of the second part for $\ell - 1$. From (13) it follows that

$$\begin{aligned}
\left|S_m^{\ell-1}\right| = |S^\star| &= \sum_{s' \in S'} \left|S^\star(\cdot, s', \cdot)\right| \\
&= (1 \pm \delta)\rho n' \left|S'\right| \left|S''\right| \\
&= (1 \pm \delta)\rho n' \left|S_{2m-1}^\ell\right| \left|S_{2m}^\ell\right| \\
&= (1 \pm \delta)\rho n' ((1 \pm \rho)^{2^{d-\ell}-1} \sigma_\ell)^2 \qquad \text{by the inductive hypothesis} \\
&= (1 \pm \delta)^{2^{d-(\ell-1)}-1} \sigma_{\ell-1}.
\end{aligned} \tag{14}$$

$\square$

---

[3]The notation $f(n) \gg g(n)$ is equivalent to $f(n) = \omega(g(n))$, i.e., $f(n)/g(n) \to \infty$.

Claim 10. *There is q.s. no failure in* MAINPHASE *(Algorithm 6 Line 9).*

Proof. Consider how the sizes of the sets S_m^ℓ are affected by poisoning during a round. Claim 9 part (3) shows that each element $s' \in S'$ has almost exactly equal representation in the parent set $S^\star$. Specifically (see also (13) and (14)), q.s., every ratio $|S^\star(\cdot, s', \cdot)| \,/\, |S^\star|$ satisfies

$$\frac{|S^\star(\cdot, s', \cdot)|}{|S^\star|} = \frac{(1 \pm \delta)\,|S''|\,\rho n'}{(1 \pm \delta)\rho n'\,|S'|\,|S''|} = \frac{(1 \pm \delta)^2}{|S'|}.$$

Thus each poisoned $s' \in S'$ poisons $|S^\star(\cdot, s', \cdot)| = (1 \pm \delta)^2 |S^\star|/\,|S'|$ triples associated with the parent $S^\star$. If $\lambda\,|S'|$ elements are poisoned (a λ fraction of S'), they poison $\leq \lambda(1 + \delta)^2\,|S^\star|$ elements of $S^\star$ (a λ fraction of $S^\star$, up to the small error factor); the inequality allows for possible repeated poisonings of the same elements.

After $\lambda\sigma_3$ iterations within a round, a λ fraction of $S_1^3 = \Xi_1$ is poisoned. It therefore poisons a $\leq \lambda(1 + \delta)^2$ fraction of S_1^2. These poisoned elements in turn poison a $\leq \lambda(1 + \delta)^4$ fraction of S_1^1. The same holds for S_3^3, S_5^3, S_7^3, so, summing, we see that a $\leq 4\lambda(1 + \delta)^4$ fraction of S_1^1 is poisoned. Similarly, a $\leq 4\lambda(1 + \delta)^4$ fraction of S_2^1 is poisoned.

As long as these fractions are both at most $1/2$, at least half the elements of J are healthy, likewise for K. In this case the number of good triples (i, j, k), with i given by the round and iteration, and $j \in J$ and $k \in K$ both healthy, dominates $\mathrm{Bin}(\frac{1}{4}\sigma_1^2, \rho)$, whose expectation is

$$\frac{1}{4}\sigma_1^2\,\rho = \Theta\left((n^{3/7}x^{4/7}\log^2 n)^2\,(n^{-6/7}x^{-8/7}\log n)\right) = \Theta(\log^5 n).$$

Thus q.s. there is some good triple (i, j, k).

The poisoned fractions of J and K, each $\leq 4\lambda(1 + \delta)^4$, are $\leq 1/2$ if $\lambda \leq (1 - \delta)^4/8$. Earlier we parametrized the total number of iterations as αx, so $\lambda\sigma_3 \leq \alpha x$, for $\lambda \leq \alpha x/\sigma_3 < 8\alpha$ (the equipartitioning gives $\sigma_3 > x/8$). Taking $\alpha = (1/64)(1 - \delta)^4$, we see that the round can q.s. continue even in the iteration after αx. We thus perform $\lceil \alpha x \rceil$ iterations within the round, all succeeding q.s. $\square$

Claim 11. *There is q.s. no failure in* TRIPLES *(Algorithm 2), in Lines 2 or 9.*

Proof. Recall that TRIPLES is called by SETCONSTRUCT (Algorithm 3), in turn called by MAIN-PHASE Line 5. From Claim 9 and (9), the total size of all sets $S_1^1, \ldots, S_8^3$ is dominated by the sizes of $J = S_1^1$ and $K = S_2^1$, and q.s. is $O(\sigma_1) = O(n^{3/7}x^{4/7}\log^2 n)$. Recalling that $x = x_t \leq x_1 = n^{6/7}$, this is $O(n^{46/49})$.

Within SETCONSTRUCT, at the start of round t we have $|X| = |A| = n - x_t$; this is smallest for round 1, where $|A| = n - x_1 \geq n - n^{6/7}$. X is depleted precisely of the elements that TRIPLES adds to the sets S_m^ℓ, so by the paragraph above, even at the end of SETCONSTRUCT, q.s. $|X| \geq n - n^{6/7} - O(n^{46/49}) > n - n^{19/20}$. Thus Line 2 of TRIPLES q.s. never fails. Also, Line 9 fails iff the size of U reaches $n^{19/20}$, but again U is one of the sets S_m^ℓ, and the first paragraph shows that its size is q.s. smaller than this.

A union bound ensures that q.s. there is no failure within the polynomial-time execution of MAINPHASE. $\square$

4.3   Final Phase

We now have to add only 3 indices to I. At this point, when the number of unused indices is $|\bar{B}| = |\bar{C}| = x < 4$, the previous approach does not work: to begin with, there are not even enough

unused indices to define nonempty sets $\Xi_1, \ldots, \Xi_8$. In the extreme case $x = 1$, only n is available to fill the roles of $\xi_1, \ldots, \xi_7$ and $\xi_2, \ldots, \xi_8$, but (within each of the two groups) we relied on all these values being distinct to obtain a valid assignment (see Figure 1). We now show how to modify our approach in order to deal with this problem.

Because there are only $O(1)$ indices to add, we can now afford to have a round consist of the addition of a single triple. It is easy to check that $W_\tau = O(n^{-6/7} \log n)$, so for the remaining rounds t (namely $t = \tau + 1$, $\tau + 2$, and $\tau + 3$), in lieu of (6) we may as well let

$$\rho_t = \rho = n^{-6/7} \log n, \tag{15}$$

defining w_t and W_t as in (6). Then, if successful, the cost of this phase is $O(W_\tau + w_\tau) = O(n^{-6/7} \log n)$.

As usual, by permuting the cost matrix we can assume that $T = \{(i, i, i) \colon i \in A\}$, so $A = B = C$. For notational convenience we further assume that $A = [n - 1]$. We can do so because, when adding one more element to T, we can confine ourselves (in the algorithm and the analysis) to using indices at most $|A| + 1$; when $x = 3$, for example, this is like "pretending" that n is $n - 2$ and x is 1.

Consider the following relabeling of Figure 1. This indicates our strategy for completing the assignment. It is easy to check that if $n, j, \ldots, s$ are distinct then this is a valid augmentation, so the task is to find such an augmentation of low cost.
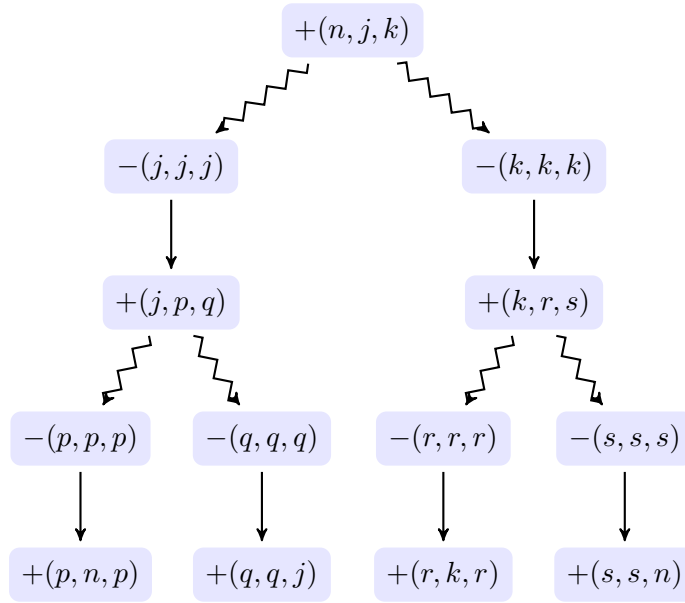

Figure 3:

It will be convenient for this and the corresponding general case to equipartition $[n - 1]$ into sets $N_{\ell,m}$ of size $n_0 = n/(2^{d+1} - 2)$ for $1 \le \ell \le d$ and $1 \le m \le 2^\ell$ (6 sets in this $d = 2$ case). Drawing our indices (in this case $j, \ldots, s$) from corresponding sets ensures distinctness. Here all sizes are large and we will ignore integrality.

Algorithm 7 FINALPHASE: augment the assignment by one
___
1: Let $P = \{p \in N_{2,1} : (p, n, p) \in G_t\}$
2: **for** $j \in N_{1,1}$ **do**
3:    Let $Q(j) = \{q \in N_{2,2} : (q, q, j) \in G_t\}$
4: Let $J = \{j \in N_{1,1} : (\exists p \in P, q \in Q(j))$ with $(j, p, q) \in G_t\}$

5: Let $S = \{s \in N_{2,4} : (s, s, n) \in G_t\}$
6: **for** $k \in N_{1,2}$ **do**
7:    Let $R(k) = \{r \in N_{2,3} : (r, k, r) \in G_t\}$
8: Let $K = \{k \in N_{1,2} : (\exists r \in R(k), s \in S)$ with $(k, r, s) \in G_t\}$

9: **if** $G_t \cap (\{n\} \times J \times K)) = \emptyset$ **then** FAIL
10: Let (n, j, k) be an element in this set. Let p, q be values satisfying Line 4 for j, and let r, s be values satisfying Line 8 for k.
11: Augment the assignment (as in Figure 3), adding $+(n, j, k)$, $+(j, p, q)$, $+(k, r, s)$, $+(p, n, p)$, $+(q, q, j)$, $+(r, k, r)$, $+(s, s, n)$ and removing $-(j, j, j)$, ..., $-(s, s, s)$.
___

Claim 12. *There is q.s. no failure in* FINALPHASE *(Algorithm 7) in Line 9.*

Proof. Observe first that $|P|$ is distributed as $\text{Bin}(n_0, \rho)$. This has expectation $\Omega(n^{1/7} \log n)$ and so the Chernoff bounds imply that q.s. $|P| = (1 \pm \delta) n_0 \rho$, with δ as in (10). Given P and $j \in N_{1,1}$, the size of $Q(j)$ is distributed as $\text{Bin}(n_0, \rho)$ and so q.s. $|Q(j)| = (1 \pm \delta) n_0 \rho$. The trials for choosing a $Q(j)$ are independent of those for choosing P as they use distinct first indices (drawn respectively from $N_{2,2}$ and $N_{2,1}$). The trials for the various $Q(j)$ are all independent as they use distinct last indices. Condition on the sets P and $Q(j)$ for $j \in N_{1,1}$ and that they are as large as q.s. claimed.

The trials to determine if $j \in N_{1,1}$ belongs to J are the only trials to look at matrix elements with first index j, so they are independent of the trials for P, for every $Q(j)$, and for every other j'. Fix $j \in N_{1,1}$ and let α_j be the (conditional) probability that $j \in J$. Then

$$\alpha_j = 1 - (1 - \rho)^{|P||Q(j)|} \geq \frac{2}{3}|P||Q(j)|\rho \geq \frac{1}{2}n_0^2\rho^3.$$

Because the trials for each $j \in N_{1,1}$ are independent, $|J|$ dominates $\text{Bin}(n_0, n_0^2\rho^3/2)$ and q.s. this is at least $n_0^3\rho^3/3$. Conditioning on all the foregoing, the same analysis shows that q.s. $|K| \geq n_0^3\rho^3/3$. Given this, and recalling ρ from (15), the probability that no (n, j, k) is good is at most

$$(1 - \rho)^{n_0^6\rho^6/9} \leq \exp(-\rho n_0^6\rho^6/9) = \exp(-\Theta(n^6\rho^7)) = \exp(-\Theta(\log^7 n)).$$

$\square$

This completes the analysis of BDAPTA when there are two levels, BDAPTA(2).

5   3-Dimensional Axial assignment general algorithm, BDAPTA(d)

We follow the same three-phase strategy as for BDAPTA(2) but with depth d a positive integer satisfying

$$2 \leq d \leq \varepsilon \log_2 \log n \quad \text{where} \quad 0 < \varepsilon < 1/2. \tag{16}$$

For the Greedy and Main Phases the ideas are the same as before; only the calculations are more difficult. For the Final Phase, new ideas are needed to generalize the construction illustrated in Figure 3.

17

5.1 Greedy Phase

This is much as before. Proceed as in Section 4.1 but take

$$\eta = \eta_d = \frac{1}{2^{d+1} - 1}, \qquad n_1 = n - n^{1-\eta}, \qquad w_0 = n^{-2(1-\eta)} \log n.$$

Lemma 1 continues to hold.

5.2 Main Phase: parameters and cost bound

In parallel with Section 4.2, let

$$\left. \begin{aligned} \alpha &= \tfrac{1}{10} 2^{-2d}, \qquad \beta = 1 - \alpha, \\ x_1 &= \lfloor n^{1-\eta} \rfloor, \qquad x_t = \lfloor \beta x_{t-1} \rfloor \ \text{ for } t \geq 2, \\ \rho_t &= x_t^{-1-\eta} n^{\eta - 1} \log n, \ \text{ for } t \geq 1, \\ \tau &= \operatorname{argmax}_t \left\{ x_t \geq 2^d \right\}, \qquad w_t = -\log(1 - \rho_t), \quad W_t = w_0 + w_1 + \cdots + w_t. \end{aligned} \right\} \tag{17}$$

Note that

$$x_t \leq \beta^{t-1} x_1, \qquad \tau \leq \log_{1/\beta}(x_1/2^d) = O(\log^2 n), \quad \text{and} \quad w_t = (1 + o(1))\rho_t.$$

As before let T_t be the partial matching (set of triples) at the start of round t of the Main Phase, let $A_t = \operatorname{proj}_1(T_t)$ be the set of 1-indices assigned and $\bar{A}_t$ those not assigned, similarly for B and C, and again for notational convenience assume that $T_t = (1,1,1), \ldots, (n - x_t, n - x_t, n - x_t)$. Round t consists of $\lceil \alpha x_t \rceil$ iterations, each increasing the cardinality of the partial assignment by 1, so that the round reduces the number of unassigned elements from x_t to $\lfloor \beta x_t \rfloor = x_{t+1}$. As before, the small size of α means that the last several rounds each each run for a single iteration and thus the last phase will have $x_\tau = 2^d$ exactly. In analogy with (7), each augmentation will add $2^{d+1} - 1$ triples each of cost at most W_t, and remove $2^{d+1} - 2$ triples.

Altogether the Main Phase increases the partial assignment's cost by at most

$$(2^{d+1} - 1) \sum_{t=1}^{\tau} (x_t - x_{t+1}) W_t \leq 2^d \left(x_1 w_0 + \sum_{t=1}^{\tau} x_t w_t \right)$$

$$= O(2^d) \left(n^{\eta - 1} \log n + n^{\eta - 1} \log n \sum_{t=1}^{\tau} x_t^{-\eta} \right).$$

$$= O(2^{4d} n^{\eta - 1} \log n).$$

The last line above follows in precise analogy to (8), using

$$\sum_{t=1}^{\tau} (x_t)^{-\eta} \leq \sum_{t=0}^{\tau-1} (\beta^t x_1)^{-\eta} \leq \frac{((1/\beta)^\tau)/x_1)^\eta}{1 - \beta^\eta} \leq \frac{(2^{-d})^\eta}{1 - \beta^\eta} \leq O(2^{3d}), \tag{18}$$

whose last inequality comes from $d\eta = \Theta(1)$ and $\beta^\eta = (1 - \alpha)^\eta = 1 - \Omega(\alpha\eta) = 1 - \Omega(2^{-3d})$.

5.3 Main Phase: algorithms and properties

The algorithm follows lines that we hope are clear from the $d = 2$ case, with only small generalizations needed.

Property 3 holds just as before.

Algorithm 2, TRIPLES, constructing a set U of triples from two child sets V, W of triples and a set X of available 1-indices, is as before with the sole exception that we replace $n^{19/20}$ with $n^{1-\eta/10}$

Property 4 holds for the same reason as before.

Algorithm 3, SETCONSTRUCT, is a trivial extension of its $d = 2$ special case. It is given as Algorithm 3 below, and constructs the obvious generalization of Figure 2.

Algorithm 8 SETCONSTRUCT: for Main Phase round t, construct all sets S_m^ℓ

1: Input: sets A, B, and C of assigned 1-, 2-, and 3-indices.
2: Let $X = A$ and $x = n - |X|$.
3: "Equipartition" $\bar{B}$ into $S_1^{d+1} \cup S_3^{d+1} \cup \cdots \cup S_{2^{d+1}-1}^{d+1}$ each of size $\lfloor x/2^d \rfloor$, discarding the $x - 2^d \lfloor x/2^d \rfloor$ remaining elements of $\bar{B}$
4: Likewise, equipartition $\bar{C}$ into $S_2^{d+1} \cup S_4^{d+1} \cup \cdots \cup S_{2^{d+1}}^{d+1}$
5: **for** $\ell = d, \ldots, 1$ **do**
6:     **for** $m = 1, \ldots, 2^\ell$ **do**
7:         Let $((S^\star)_m^\ell, S_m^\ell, X) = $ TRIPLES$(X, S_{2m-1}^{\ell+1}, S_{2m}^{\ell+1})$
8: Return all sets S_m^ℓ and $(S^\star)_m^\ell$ defined above

---

Algorithm 4, MAKETREE, is as shown earlier except that ℓ ranges from 1 to d, and of course the names J and K are to be disregarded.

We will use an augmentation output by MAKETREE just as before, adding all positive triples to the assignment and removing all negative triples from it. Instead of adding 7 triples and removing 6, as in (7) for the $d = 2$ special case, we now add $2^{d+1} - 1$ triples and subtract $2^{d+1} - 2$.

Properties 5 and 6 hold for the same reasons as before.

Algorithm 5, POISONPROPAGATE, again has no changes except that in Line 2 the poisoned leaves are $s_1^{d+1}, \ldots, s_{2^{d+1}}^{d+1}$, and that Line 3 iterates through $\ell = d$ to 1.

Algorithm 6, MAINPHASE is unchanged but for interpreting J as S_1^1 and K as S_2^1, and likewise their starred variants.

Property 7 holds for the same reason as before.

This establishes that MAINPHASE is correct, and we need only show that q.s. it does not fail. To this end we will prove generalizations of Claims 9, 10, and 11, starting with Claim 9's determination of the sizes of the sets S_m^ℓ.

Letting $n' = n - 2n^{1-\eta/10}$, in analogy with (9) we define nominal sizes of level-ℓ sets by $\sigma_{d+1} = \lfloor x/2^d \rfloor$ and, for $d \geq \ell \geq 1$,

$$\sigma_\ell = \rho n' \sigma_{\ell+1}^2, \tag{19}$$

a recurrence leading easily to

$$\sigma_\ell = (\rho n')^{2^{d+1-\ell}-1}(\sigma_{d+1})^{2^{d+1-\ell}} = \frac{1}{\rho n'}(\rho n' \sigma_{d+1})^{2^{d+1-\ell}}. \tag{20}$$

As in the depth-2 case, we will show that the sizes of each set S_m^ℓ is approximated by the nominal size σ_ℓ up to a factor $(1 \pm \delta)^{2^{d+1-\ell}}$, with

$$\delta = n^{-\eta^2/3}. \tag{21}$$

We note that these bounds are good, by assertion 0 of the following claim.

Claim 13. *For each depth $\ell \in \{1, \ldots, d+1\}$ and each $m \in \{1, \ldots, 2^l\}$:*

0. For δ as defined in (21), $(1 \pm \delta)^{2^d} = 1 + o(1)$.

1. The nominal sizes in (20) satisfy $\sigma_1 \gg \cdots \gg \sigma_{d+1} = \Theta(x/2^d)$.

2. Q.s., $\left| S_m^\ell \right| = (1 \pm \delta)^{2^{d+1-\ell}-1} \sigma_\ell$.

3. For every pair of sets $S' = S_{2m-1}^\ell$, $S'' = S_{2m}^\ell$, with $S^\star = (S^\star)_m^{\ell-1}$, q.s. every $s' \in S'$ gives $|S^\star(\cdot, s', \cdot)| = (1 \pm \delta)\rho n' |S''|$ and symmetrically every $s'' \in S''$ gives $|S^\star(\cdot, \cdot, s'')| = (1 \pm \delta)\rho n' |S'|$.

Proof. The proof mirrors that of Claim 9 but with more computation. We start with some observations we will use repeatedly. By definition,

$$\eta = \Theta(2^{-d}) = \Omega(\log^{-e} n).$$

Then

$$n^{\eta^2} = \exp(\log n \, \Omega(\log^{-2\varepsilon} n)) = \exp(\Omega(\log^{1-2\varepsilon} n)). \tag{22}$$

(This is the reason we require $\varepsilon < 1/2$.)

We first prove part 0 (which was not worth stating for the $d = 2$ case). First,

$$(1 + \delta)^{2^{d+1}} \le \exp(\delta 2^{d+1}) = \exp(n^{-\eta^2/3}\Theta(1/\eta)).$$

By (22), the argument of the exponential satisfies

$$n^{-\eta^2/3}\Theta(1/\eta) = \exp(\Omega(\log^{1-2\varepsilon} n))O(\log^\varepsilon n) = o(1), \tag{23}$$

where the last step uses $\varepsilon < 1/2$ and follows from the general observation that the exponential of a logarithm to any positive power dominates any polylogarithmic quantity. (Taking logarithms, log to any positive power dominates log log.) Thus,

$$(1 + \delta)^{2^{d+1}} = \exp(o(1)) = 1 + o(1),$$

establishing assertion 0.

The first assertion's last equality follows from $x \ge 2^d$ (so that $\lfloor x/2^d \rfloor$ is not 0). Proving the rest of the first assertion inductively on $\ell = d, \ldots, 1$, suppose that $\sigma_{\ell+1} \ge \sigma_{d+1}$; the base case $\ell = d$ is trivial. Then, recalling that $x \le n^{1-\eta}$, we see that

$$\rho n' \sigma_{\ell+1} \ge \rho n' \sigma_{d+1} = \Omega(\rho n x \, 2^{-d}) = \Omega((n/x)^\eta \log n \, \log^{-\varepsilon} n) = \Omega(n^{\eta^2} \log^{1/2} n)$$
$$= \exp(\Omega(\log^{1-2\varepsilon} n)) \log^{1/2} n = \omega(1), \quad (24)$$

20

where the last line makes use of (22) and the previous observation on domination. From (24) and (19), $\rho n' \sigma_{\ell+1}{}^2 = (\rho n' \sigma_{\ell+1}) \sigma_{\ell+1} \gg \sigma_{\ell+1}$, establishing the first assertion.

We prove the rest of the claim by induction on ℓ, starting with $\ell = d+1$ and working backwards to $\ell = 1$. We equate $S' = S_{2m-1}^\ell$, $S'' = S_{2m}^\ell$, and $S^\star = (S^\star)_m^{\ell-1}$. For $\ell = d+1$, the second statement is immediate. For any $\ell > 1$, the first two statements imply the third (which is empty when $\ell = 1$). To see this, Property 4 implies that given S'', distributionally,

$$\left| S^\star(\cdot, s', \cdot) \right| = \sum_{S''} \mathrm{Bin}(n', \rho) = \mathrm{Bin}(\left| S'' \right| n', \rho). \tag{25}$$

We wish to apply the Chernoff bound to this. From assertion 0 and the inductive hypothesis, $|S''| = (1 + o(1))\sigma_{\ell+1} = \Omega \sigma_{d+1}$, and thus the Chernoff bound is exponentially small in

$$\begin{aligned}
\delta^2 \left| S'' \right| n' \rho &\geq (n^{-\eta^2/3})^2 (1 + o(1)) \Theta(2^{-d} x) n (x^{-1-\eta} n^{\eta-1}) \\
&= (n^{-2\eta^2/3}) \Theta(\eta)(n/x)^\eta \quad \text{which, reasoning as in (24), is} \\
&= \Omega(n^{+\eta^2/3}\eta) = \exp(\tfrac{1}{2}\Omega(\log^{1-2\varepsilon} n)) \, \Omega(\log^{-\varepsilon} n) = \omega(\log n).
\end{aligned}$$

Thus the Chernoff bound is smaller than any polynomial, so the binomial's value is q.s. almost exactly equal to its expectation. Thus, q.s. for all $s' \in S'$, $|S^\star(\cdot, s', \cdot)| = (1 \pm \delta)\,|S''|\,n'\rho$, with the symmetric statement holding for all $s'' \in S''$. This gives the claim's third assertion for ℓ.

Then, simply by summing over all $s' \in S'$, $|S^\star| = (1 \pm \delta)\,|S'|\,|S''|\,n'\rho$. By the inductive hypothesis,

$$|S^\star| = (1 \pm \delta) \left[(1 \pm \delta)^{2^{(d+1)-(\ell)}-1} \sigma_\ell \right]^2 n' \rho = (1 \pm \delta)^{2^{d+1-(\ell-1)}-1} \sigma_{\ell-1}.$$

This establishes the claim's second assertion for $\ell - 1$, concluding the induction and the proof. $\qquad\blacksquare$

The next claim parallels Claim 10 and is proved by identical reasoning.

Claim 14. *There is q.s. no failure in* MainPhase.

Proof. Claim 13's first assertion establishes the sizes of the sets S_1^1 and S_2^1, and we now consider the sizes of their healthy subsets. After $\lambda \sigma_{d+1}$ iterations within a round, by definition a λ fraction of S_1^{d+1} is poisoned. Reasoning as in the proof of Claim 10, Claim 13's second assertion says that each element in the child set is responsible for an equal number of elements of the parent, to within a factor $(1 \pm \delta)^2$, and thus the poisoned λ fraction of S_1^{d+1} poisons a fraction $(1 \pm \delta)^2 \lambda$ of S_1^d. This in turn poisons a fraction $(1 \pm \delta)^4 \lambda$ of S_1^{d-1} and so on, finally poisoning a fraction $(1 \pm \delta)^{2d}$ of S_1^1. Each leaf set S_m^{d+1} poisons this same fraction of S_1^1, so altogether, the fraction of S_1^1 poisoned is at most $2^d (1 + \delta)^{2d} \lambda$. (Half the leaves lead to S_1^1; the other half act on S_2^1.)

We wish to ensure that at most half of S_1^1 is poisoned. This will be so if $2^d (1 + \delta)^{2d} \lambda \leq 1/2$. Claim 13's assertion 0 tells us that $(1 + \delta)^{2d} = 1 + o(1)$, so it suffices to ensure that $\lambda \leq \frac{1}{3} 2^{-d}$. The round runs for $\lceil \alpha x \rceil$ iterations and we are not interested in the state after the last iteration, so $\lambda \sigma_{d+1} \leq \lfloor \alpha x \rfloor \leq \alpha x$. Thus it suffices to take $\alpha x / \sigma_{d+1} \leq \frac{1}{3} 2^{-d}$, or equivalently $\alpha \leq (\sigma_{d+1}/x) \frac{1}{3} 2^{-d}$. The right-hand side of this inequality is equal to $(\lfloor x/2^d \rfloor / x) \frac{1}{3} 2^{-d} \geq (\frac{1}{2} 2^{-d})(\frac{1}{3} 2^{-d}) = \frac{1}{6} 2^{-2d}$, so it suffices to ensure that $\alpha \leq \frac{1}{6} 2^{-2d}$; this is done by its definition in (17).

From Claim 13's assertion 2 and then assertion 0, $\left| S_1^1 \right|, \left| S_2^1 \right| = (1 \pm \delta)^{2^{d+1}} \sigma_1 = (1 + o(1))\sigma_1$, so the product of the sizes of their healthy subsets is $\geq \frac{1}{5} \sigma_1{}^2$, and $\left| G_t \cap \{i\} \times S_1^1 \times S_2^1 \right| \succsim \mathrm{Bin}(\sigma_1{}^2/5, \rho)$, where the symbol $\succsim$ denotes stochastic domination. By Chernoff, this is q.s. nonzero if the expected value, $\rho \sigma_1{}^2/5$, is $\omega(\log n)$, as we now verify.

21

First, recalling that $2^d = \Theta(1/\eta)$, we note that

$$(n'/n)^{2^d} = (1 - n^{-\eta/10})^{\Theta(1/\eta)} = 1 - \Theta(n^{-\eta/10} \cdot 1/\eta)) = 1 - o(1),$$

where the last step uses

$$n^{-\eta/10} \cdot 1/\eta = \exp(-\log^{1-\varepsilon} n)O(\log^\varepsilon n) = o(1),$$

as comes from reasoning like that for (22) and (23). By (20) and using $\sigma_{d+1} > \frac{1}{2}x/2^d$, the expectation $\rho\sigma_1{}^2/5$ is at least one fifth of

$$\rho\frac{1}{(\rho n')^2}(\rho n'\sigma_{d+1})^{2^{d+1}} \geq \frac{1}{n^2\rho}\left(\frac{\rho nx}{2^{d+1}}\right)^{2^{d+1}}$$

$$= \left(\frac{x}{n}\right)^{1+\eta}\frac{1}{\log n}\left(\left(\frac{n}{x}\right)^\eta\frac{\log n}{2^{d+1}}\right)^{2^{d+1}}$$

$$= \frac{1}{\log n}\left(\frac{\log n}{2^{d+1}}\right)^{2^{d+1}} \quad (\text{using } (2^{d+1})\eta = 1 + \eta)$$

$$\geq \frac{1}{\log n}\left(\frac{\log n}{\log^\varepsilon n}\right)^{2^{d+1}}$$

$$\geq (\log n)^{2^d - 1}.$$

That is, the expectation is of order $\omega(\log n)$ for any $d > 1$ (quick inspection shows it is also so for $d = 1$), whereupon the Chernoff bound shows that q.s. every set $G_t \cap \{i\} \times S_1^1 \times S_2^1$ is nonempty, and there is no failure in MainPhase. $\square$

The next claim parallels Claim 11 and is proved by identical reasoning.

Claim 15. *There is q.s. no failure in* Triples *(Algorithm 2), in Lines 2 or 9.*

Proof. By analogy with the proof of the $d = 2$ special case, it suffices to show that the sum of the sizes of all sets S_m^ℓ produced by SetConstruct (calling Triples) is $o(n^{1-\eta/10})$, so by Claim 13 it suffices to show that the sum of all the nominal sizes is this small. From Claim 13's assertion 1, each set's nominal size dominates the sum of the sizes of its two children, so the sum of all the sets' nominal sizes is less than twice σ_1. We now verify that this is $\ll n^{1-\eta/10}$. We use that $2^d - 1 < 2^d - 1/2 = 1/(2\eta)$. First,

$$\sigma_1 = \frac{1}{\rho n'}(\rho n'\sigma_{d+1})^{2^d} \leq \frac{1}{\rho n'}(\rho nx)^{2^d} = x(\rho n'x)^{2^d - 1} = x((n/x)^\eta \log n)^{1/(2\eta)} = (nx)^{1/2}(\log n)^{1/(2\eta)}.$$

This is increasing with x, and $x \leq x_1 = n^{1-\eta}$, so

$$\sigma_1 \leq n^{1-\eta/2}(\log n)^{1/(2\eta)} = \exp((1 - \eta/2)\log n + 1/(2\eta)\log\log n).$$

Recalling that $\eta = \Omega(\log^{-\varepsilon} n)$, the term $(\eta/2)\log n$ is $\Omega(\log^{1-\varepsilon} n)$ while the final term is $O(\log^\varepsilon n \log\log n)$, so we conclude that

$$\sigma_1 \leq \exp((1 - \eta/3)\log n) = n^{1-\eta/3} = o(n^{1-\eta/10}). \tag{26}$$

$\square$

5.4 Final Phase

We execute the Main Phase so long as the number of unassigned elements is at least 2^d. In the Final Phase, then, we must add $2^d - 1$ triples, and as in Section 4.3, we will proceed in rounds, each round adding just one triple. In a round, there are $x < 2^d$ unassigned elements. We assume that the assigned first indices are $A = [n - x]$, and that the partial assignment consists of the set of triples $T = \{(i, i, i) : i \in A\}$. We simplify the notation of Section 5.2 and let

$$\rho = n^{\eta - 1} \log n, \quad w = -\log(1 - \rho) = \rho(1 + o(1)), \tag{27}$$

with $\eta = 1/(2^{d+1} - 1)$ as in Sections 5.1 and 5.2.

Because each round adds just one triple, if successful, the cost of this phase is

$$O(2^d(W_\tau + 2^d w)) = O(2^{2d} n^{\eta - 1} \log n).$$

To see this, the $2^{2d} w$ on the left clearly matches the bound. Recalling $W_\tau = w_0 + \sum_{t=1}^\tau w_t$, the first term's contribution $2^d w_0 = 2^d n^{2(\eta - 1)} \log n$ is negligible. Then, $\sum_{t=1}^\tau w_t = n^{\eta - 1} \log n \sum_{t=1}^\tau x_t^{-1 - \eta}$ is summed as in (18) to give $n^{\eta - 1} \log n \frac{(2^{-d})^{1+\eta}}{1 - \beta^{1+\eta}}$. Still as in (18), $(2^{-d})^{1+\eta} = \Theta(2^{-d})$ while $\beta^{1+\eta} = 1 - \Omega(\alpha(1 + \eta)) = 1 - \Omega(\alpha)$. So this term, with its 2^d prefactor, contributes $2^d n^{\eta - 1} \log n \frac{O(2^{-d})}{\Omega(2^{-2d})}$, again matching the bound.

A certain function ϕ will play a key role in defining a general set of augmenting triples, analogous to the $d = 2$ case illustrated in Figure 3. The function ϕ may be thought of as taking as input a binary string (interpretable as a binary number), along with its length, and dropping the final all-0 or all-1 block (of maximum length). That is, we define

$$\phi(\ell, b_1 \ldots b_\ell) = (i, b_1 \ldots b_i) \quad \text{where } b_i \neq b_{i+1} \text{ but } b_{i+1} = \cdots = b_\ell.$$

If $b_1 = \cdots = b_\ell$ then $\phi(\ell, b_1 \ldots b_\ell) = (0, \$)$ where $\$$ denotes the empty string.

Consider a complete binary tree $\mathcal{T}$ on levels $0, \ldots, d$, with nodes (ℓ, m) for $\ell = 0, \ldots, d$ and $m = 0, \ldots, 2^\ell - 1$, where the children of (ℓ, m) are $(\ell + 1, 2m)$ and $(\ell + 1, 2m + 1)$.[4] Make the natural correspondence between a number and its binary representation, with the empty string $\$$ having value 0. Then ϕ establishes a map from the nodes at any level of $\mathcal{T}$ (notably the leaves) to nodes at a higher level,

$$\phi(\ell, m) = (i, m'),$$

where we interpret m as its ℓ-bit binary representation $b_1 \ldots b_\ell$, and m' is the number with i-bit binary representation $b_1 \ldots b_i$. For a given level ℓ, this map from level-ℓ nodes to shallower nodes is injective except for the two nodes $(\ell, 0)$ and $(\ell, 2^\ell - 1)$ mapping to the root. This follows because, except at the root, ϕ is invertible, with $\phi^{-1}(i, b_1 \ldots b_i)) = (\ell, (b_1 \ldots b_i(1 - b_i) \ldots (1 - b_i))$, where the final binary number has ℓ bits.

The mapping ϕ will be used to define a generic set of triples that can be used to augment an assignment, as illustrated in Figure 4 for $d = 3$. It is analogous to the triples illustrated in Figure 3 for $d = 2$, with the indices $1, \ldots, 15$ here playing a role similar to that of the variables $n, j, k, \ldots, s$ there. Each node $u = (\ell, m)$ of $\mathcal{T}$ is given a triple

$$(i_u, j_u, k_u).$$

---

[4] We take the range of m to be $0, \ldots, 2^\ell - 1$ ("index origin" 0 rather than 1) because this is notationally better given our use of the binary representation of m.

The first index i_u is given by

$$i_u \equiv i_{\ell,m} := 2^\ell + m, \quad \text{and for the root,} \quad i_{(0,\$)} = 1. \tag{28}$$

(Any unique labeling will do, though we will capitalize on the parities of the leaves.) For a non-leaf u with left child v and right child w we complete the triple as (i_u, i_v, i_w). A leaf $u = (d, m)$ with m even is given triple $(i_u, i_{\phi(u)}, i_u)$, and one with m odd is given triple $(i_u, i_u, i_{\phi(u)})$. (The role of ϕ is critical, as we will explain shortly.)
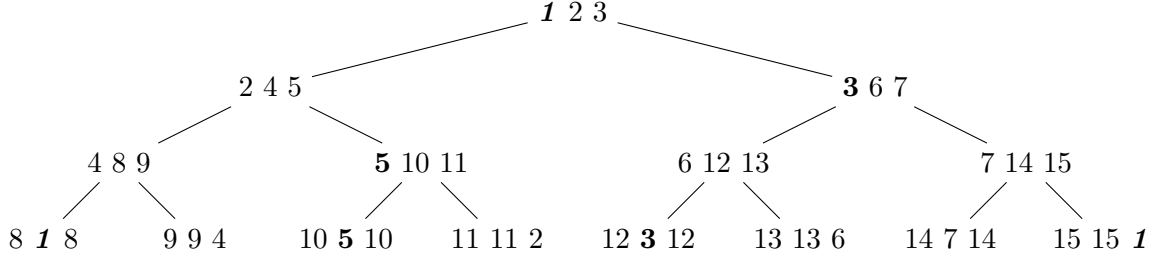

Figure 4: For $d = 3$, construction, based on ϕ, of a generic tree of triples (i, j, k) to augment the assignment.

The figure shows in bold font linkages between various leaves and internal nodes via ϕ. The first and last leaves, with $m = 0$ and $m = 7$, have binary representations 000 and 111, both mapping to the root, and thus their triples both include the root's value $i_u = 1$. The third leaf maps to the second level-2 node, since $\phi(3, 010) = (2, 01)$, so its triple $(10, 5, 10)$ includes that node's value $i_u = 5$. Likewise, the fifth leaf maps to the second level-1 node since $\phi(3, 100) = (1, 1)$, so its triple $(12, 3, 12)$ includes that node's value $i_u = 3$.

As in the $d = 2$ special case of the Final Phase (Section 4.3), to avoid dependencies we equipartition the assigned first indices $[n - x]$ into sets $N_{\ell,m}$ of size

$$n_0 = \frac{n - x}{2^{d+1} - 2}, \tag{29}$$

for $1 \le \ell \le d$ and $0 \le m \le 2^\ell - 1$ (both x and 2^{d+1} are of order $o(n)$, so we can and shall ignore integrality), and we define a singleton set $N_{0,0} = \{n\}$: we will, without loss of generality, discuss the addition of first index n to the assignment in this round.

We instantiate the generic tree of triples via a mapping

$$\pi \colon \left\{1, \ldots, 2^{d+1} - 1\right\} \to \{1, \ldots, n\} \quad \text{with} \quad \pi(i_{\ell,m}) \in N_{\ell,m}. \tag{30}$$

The function π plays the same role as the assignment of values to the variables $n, j, k, \ldots, s$ of Figure 3, just in a more general notation. For example where before we assigned a value to $s_1^1 = j$, now we choose a value $\pi(i_{1,0}) = \pi(2)$; another important example is that our choice of $N_{0,0}$ implies that $\pi(i_{0,0}) = \pi(1) = n$. To augment the assignment, for every triple (i, j, k) in the tree we add $(\pi(i), \pi(j), \pi(k))$ to the assignment, and, for every i in $2, 3, \ldots, 2^{d+1} - 1$, we delete the old assignment triple $(\pi(i), \pi(i), \pi(i))$. We will seek a mapping π such that the added triples are all of low cost. We denote the augmentation corresponding to π by A_π.

Claim 16. *The augmentation A_π described above yields a proper assignment.*

24

Proof. Let us ignore π for the moment, imagining it to be the identity. We first show that each of the values $r = 1, \ldots, 2^d - 1$ occurs exactly three times in the triples, once each as a first, second, and third element.

First consider $r = 1$. This value appears as the first index $i_u = 1$ of the root triple, the second index in the triple at leaf $(d, 0)$, and the third index in the triple at leaf $(d, 2^d - 1)$.

Next consider $2^\ell \leq r < 2^{\ell+1}$ with $1 \leq \ell < d$. Writing $r = 2^\ell + m$, r occurs as the first index i_u of a node $u = (\ell, m)$ which is neither the root nor a leaf. (For example, $r = 3$ in node $(1, 1)$ in Figure 4.) If m is even, then u is the left child of its parent $v = (\ell - 1, m/2)$ and so r appears as the second index j_v of the triple at v. Let $w = (d, m')$ be the unique leaf for which $\phi(w) = u$. The definition of ϕ implies that m' is odd, and thus $r = i_u = i_{\phi(w)}$ appears as the third index k_w of the triple at w. The case m odd is handled similarly.

Finally, consider $2^d \leq r < 2^{d+1}$. Write $r = 2^d + m$. Assume first that m is even. Then r appears as the first and third indices of the triple at (d, m) and the second index of its parent $(d - 1, m/2)$. (For example, $r = 10$ in the leaf $(3, 2)$ and its parent in Figure 4.) If m is odd then r appears as the first and second indices of the triple at (d, m) and the third index of its parent $(d - 1, (m - 1)/2)$.

By counting, these three occurrences of each value r account for all elements of all triples, so none occurs anywhere other than in the positions described.

Now consider the instantiated triples, of the form $(\pi(i_u), \pi(j_u), \pi(k_u))$. By definition (30), π is injective, so each value $\pi(r)$ occurs exactly once in each coordinate, including $\pi(1) = n$. Also by (30), for each $r = 2, \ldots, 2^{d+1} - 1$, $(\pi(r), \pi(r), \pi(r))$ was previously in the assignment, but $(\pi(1), \pi(1), \pi(1)) = (n, n, n)$ was not. Thus, after the augmentation A_π adds the first set of triples and deletes the second, the assignment is augmented by (n, n, n). $\qquad\square$

Recalling that $\pi(1) = n$, our aim now is to find a set of values $\pi(2), \ldots, \pi(2^{d+1} - 1)$ respecting (30) and making all the added triples cheap. For the subtree $\mathcal{T}_u$ rooted at u, let π' be a π-like map but on a domain restricted to indices occurring in triples outside the subtree $\mathcal{T}_u$. (Critically, some of these indices may also occur in $\mathcal{T}_u$.) We say that an extension to $\mathcal{T}_u$ (possibly a completion to all of $\mathcal{T}$) π of π' is good if, for every $v \in \mathcal{T}_u$, $(\pi(i_v), \pi(j_v), \pi(k_v)) \in G$. (The definition does not consider the cost of triples outside $\mathcal{T}_u$, something determined entirely by π', though of course we will seek to ensure that they too are of low cost.)

Claim 17. *For any non-root vertex u, the only indices occurring both in triples in $\mathcal{T}_u$ and outside it are i_u (occurring twice inside $\mathcal{T}_u$ and once outside) and $i_{\phi(u)}$ (occurring once inside and twice outside). Given the cost matrix, whether π' has a good extension to $\mathcal{T}_u$ depends solely on the values of $\pi'(i_u)$ and $\pi'(i_{\phi(u)})$.*

Proof. We prove the claim by induction on the depth of u, from the leaves to the root. We first establish the base cases. If u is a leaf, i_u occurs twice in u's triple (thus once outside it), and $i_{\phi(u)}$ occurs once in u's triple (thus twice outside it), and of course there are no other indices in $\mathcal{T}_u$. Since no index occurs exclusively in $\mathcal{T}_u$, $\pi = \pi'$. For even u, π is good on $\mathcal{T}_u$ iff $(\pi'(i_u), \pi'(i_{\phi(u)}), \pi'(i_u)) \in G$, a function of $\pi'(i_u)$ and $\pi'(i_{\phi(u)})$ alone; for odd u a similar argument applies.

Now consider u with left child v and right child w, starting with the case that u is even $(u = (\ell, m)$ with m even). The triple at u is $(i_u, j_u, k_u) = (i_u, i_v, i_w)$. By definition of ϕ, $\phi(v) = \phi(u)$, $\phi(w) = u$. Also, the four nodes $\phi(v) = \phi(u)$, $\phi(w) = u$, v, and w are all distinct (using that u is not the root) as are their corresponding indices i. What indices can occur both within and outside $\mathcal{T}_u$? The only candidates are the new index i_u and, inductively from the two subtrees, i_v, $i_{\phi(v)}$, i_w, and $i_{\phi(w)}$. Inductively, i_v makes only one appearance outside $\mathcal{T}_v$, and that is accounted for by its appearance as the second element of the triple at u. The same goes for i_w, appearing as the third element of the triple at u. Two appearances of $i_{\phi(v)}$ lie outside $\mathcal{T}_v$, and since $\phi(v) = \phi(u)$

25

is an ancestor of u (again using that u is not the root), lying outside $\mathcal{T}_u$, there are (as desired) two appearances of $i_{\phi(u)}$ outside $\mathcal{T}_u$. Finally, two appearances of $i_{\phi(w)}$ lie outside $\mathcal{T}_w$, and since $\phi(w) = u$, one of these appearances is the first element i_u of the triple at u, leaving (as desired) one appearance of i_u outside $\mathcal{T}_u$. A similar argument applies to u odd, with the roles of v and w swapped.

Since i_u and $i_{\phi(u)}$ are the only indices appearing both outside $\mathcal{T}_u$ (thus in the domain of π') and inside it, and since for any other index in $\mathcal{T}_u$ we are free to choose the value of π as we like (subject only to the prescription in (30) that $\pi(u) \in N_u$) it is immediate that whether π' has a good extension π depends only on the evaluation of π' at these two points. $\square$

Taking advantage of Claim 17, we define a pair (a, b) to be good for $\mathcal{T}_u$ if, with $\pi'(i_{\phi(u)}) = a$ and $\pi'(i_u) = b$, π' has a good extension π to $\mathcal{T}_u$.

Remark 18. *We can recursively, bottom-up, construct all good pairs for every non-root node of $\mathcal{T}$. For a leaf u, (a, b) is good iff $(b, a, b) \in G$ (for even u) or $(b, b, a) \in G$ (for odd u). For a non-root, non-leaf even u with left child v and right child w, let $\pi'(i_{\phi(u)}) = \pi'(i_{\phi(v)}) = a \in N_{\phi(u)}$ and $\pi'(i_u) = \pi'(i_{\phi(w)}) = b \in N_u$. Then (a, b) is good for $\mathcal{T}_u$, i.e., π' has a good extension π to $\mathcal{T}_u$, iff there exist values $\pi(i_v) = c \in N_v$ and $\pi(i_w) = d \in N_w$ such that the triple at u is good, i.e., $(b, c, d) \in G$, and the subtrees at v and w have good extensions, which by the inductive hypothesis is to say that (a, c) is good for $\mathcal{T}_v$, and (b, d) is good for $\mathcal{T}_w$. A similar calculation works for u odd.*

For a node $u = (\ell, m)$, with $a \in N_{\phi(u)}$, it proves convenient to define

$$S_m^\ell(a) = \{b \in N_u : (a, b) \text{ is good for } \mathcal{T}_u\}$$

Algorithm 9 simply implements the leaf-up iterative application of the goodness test described in Remark 18.[5] It generalizes Algorithm 7 (FINALPHASE for $d = 2$); it may also be seen as playing a role analogous to Algorithm 8 though it is entirely different in its details.

---

[5]This iterative construction can be viewed as a dynamic program. Specifically, it is a dynamic program on a tree decomposition of a hypergraph, of a sort described by Scott and Sorkin [SS09, Theorem 6 and its proof]. We cannot exploit [SS09] because we need more details to do our probabilistic counting, but we will state the connection. Consider a hypergraph whose vertices are the indices of $\mathcal{T}$ and whose hyperedges are its triples. The critical aspect of the connection (and of our function ϕ) is that $\mathcal{T}$ gives a bounded-treewidth tree decomposition of the hypergraph: the "bag" at a node u consists of the three indices i, j, k in its triple and (for internal nodes) the (unique) index $\phi(i)$ that occurs both below and above u.

Let $f(\pi)$ be a function on labelings which is obtained as the product over edges e of functions f_e of the labeling of edge e. [SS09] shows how to efficiently solve the problem of finding a labeling maximizing f, and indeed related counting problems, in the general setting of graphs of small treewidth; an extension to hypergraphs is merely notational. In this case we take the function for each hyperedge to have value 1 if the labeled triple is good (low cost), 0 otherwise, so that the product function f of the whole labeling is 1 iff the labeling makes all triples good, and the Scott-Sorkin algorithm will find a good labeling if one exists.

Algorithm 9 Final Phase SetConstruct: construct sets used to augment the assignment by one
___

1: **for** $m = 1, \ldots, 2^d - 1$ **do**
2:    **for** $a \in N_{\phi(d,m)}$ **do**
3:       Let

$$S_m^d(a) = \begin{cases} \{b \in N_{d,m} : (b, a, b) \in G\} & \text{for } m \text{ even,} \\ \{b \in N_{d,m} : (b, b, a) \in G\} & \text{for } m \text{ odd.} \end{cases}$$

4: **for** $\ell = d - 1, \ldots, 1$ **do**
5:    **for** $m = 0, \ldots, 2^\ell - 1$ **do**
6:       **for** $a \in N_{\phi(\ell,m)}$ **do**
7:          Let

$$S_m^\ell(a) = \begin{cases} \left\{ b \in N_{\ell,m} : \exists c \in S_{2m}^{\ell+1}(a),\ d \in S_{2m+1}^{\ell+1}(b),\ \text{s.t.}(b, c, d) \in G \right\} & \text{for } m \text{ even,} \\ \left\{ b \in N_{\ell,m} : \exists c \in S_{2m}^{\ell+1}(b),\ d \in S_{2m+1}^{\ell+1}(a),\ \text{s.t.}(b, c, d) \in G \right\} & \text{for } m \text{ odd.} \end{cases}$$

8: Let $S_0^0 = \left(\{n\} \times S_0^1(n) \times S_1^1(n) \right) \cap G$.
___

After Algorithm 9 completes, the round of Final Phase completes as follows. If $S_0^0 = \emptyset$, FAIL. Otherwise, choose any triple $(n, a, b) \in S_0^0$ and define a partial mapping by $\pi'(0,0) = n$ (as always), $\pi'(1,0) = a$, and $\pi'(1,1) = b$. By construction, $(n, a, b) \in G$ is good, $a \in S_0^1(n)$ means that (n, a) is good for $\mathcal{T}_{1,0}$ so π' has a good extension π'' to this subtree, and $b \in S_1^1(n)$ means that (n, b) is good for $\mathcal{T}_{1,1}$ and thus π'' has a completion π which is good for this subtree and thus for all of $\mathcal{T}$. That is, if S_0^0 is nonempty, a good set of labels for all nodes can be found by "unwinding" Algorithm 9 back down to the leaves. We will not bother to give the procedure explicitly. This gives a valid augmentation, as guaranteed by Claim 17.

5.5 Probability of success

It remains now to show that Algorithm 9 succeeds w.h.p., producing a nonempty set S_0^0. Let us first point out that $N_{0,0}$, the odd set with cardinality 1 rather than n_0 (defined in (29)), only ever occurs in Algorithm 9's Lines 2 and 6, affecting the number of choices for a and thus the number of sets $S_u(a)$ defined, but never in Lines 3 and 7 determining the sizes of these sets.

Let

$$\sigma_\ell = (\rho n_0)^{2^{d-\ell+1}-1} \text{ for } 0 \leq \ell \leq d, \tag{31}$$

Note that

$$\sigma_\ell = \rho n_0 \sigma_{\ell+1}^2, \quad \rho n_0 \geq n^\eta, \quad \text{and so} \quad \sigma_1 \gg \sigma_2 \gg \cdots \gg \sigma_d. \tag{32}$$

Claim 19. *Quite surely, for all* $\ell \in \{1, \ldots, d\}$, $m \in \{0, \ldots, 2^d - 1\}$, *and* $a \in S_m^\ell$,

$$\left| S_m^\ell(a) \right| \geq (1 - \delta)^{2^{d-\ell+1}-1} \sigma_\ell \tag{33}$$

where σ_ℓ is as in (31) and $\delta = 1/\log^2 n$.

Proof. The proof is by induction, proceeding through $\ell = d, \ldots, 1$ and $m = 0, \ldots, 2^\ell - 1$ in the same sequence as Algorithm 9. At each step in the analysis we will condition upon a successful history, with no failures (i.e., (33) holds); the final success probability is the product of the conditional probabilities, which is at least 1 minus the sum of the conditional failure probabilities. Conditioning will assure us that the previous set sizes are as desired, but, as we will see, all the triples considered are disjoint and thus the conditioning has no other effect. For $\ell = d$, from Algorithm 9 Line 3, for any m and any $a \in N_{\ell,m}$,

$$\left| S_m^d(a) \right| \sim \mathrm{Bin}(n_0, \rho).$$

The sizes of the various sets are independent because the trials for one value of u are distinct from those for any other, as the first elements of their triples come from distinct sets $N_{\ell,m}$ in the partition. Since $\left| S_m^d(a) \right| \sim \mathrm{Bin}(n_0, \rho)$, we have $\mathbf{E}(|S_m^d|) = \sigma_d$, and the Chernoff bounds imply

$$\mathbf{Pr}\left(\left| S_m^d(a) \right| < (1 - \delta)\sigma_d \right) \leq \exp\left\{ -\delta^2 \sigma_d / 3 \right\}. \tag{34}$$

The analysis of the size of a set $S_m^\ell(a)$ in Algorithm 9 Line 7 for an internal node, with $\ell < d$, is slightly more complicated. For m even, a given b belongs to $S_m^\ell(a)$ if any of the $\left| S_{2m}^{\ell+1}(a) \right| \left| S_{2m+1}^{\ell+1}(b) \right|$ candidates for c and d leads to a triple $(b, c, d) \in G$. Conditioning upon success up to this point, $S_{2m}^{\ell+1}(a)$ and $S_{2m+1}^{\ell+1}(b)$ each have cardinality satisfying (33), so the number of trials for b is at least

$$N = \left((1 - \delta)^{2^{d-\ell}-1} \sigma_{\ell+1} \right)^2. \tag{35}$$

The same value N results for m odd, so the two cases now continue as one. These trials for $(b, c, d) \in G$ are independent of all those in the history: the first index b rotates through distinct choices within node ℓ, m, and $b \in N_{\ell,m}$ assures that first index b was not explored for any previous node. By the independence, each trial succeeds with probability ρ. Thus, for a given b, at least one trial is successful with probability at least

$$1 - (1 - \rho)^N \geq 1 - \exp(-\rho N) \geq \rho N (1 - \rho N),$$

where the second inequality relies on $\rho N \leq 1$ (in fact $\rho N = o(1)$, see (39)).

Considering the n_0 candidates for b yields that

$$\left| S_m^\ell(a) \right| \succsim \mathrm{Bin}(n_0, \rho N(1 - \rho N)). \tag{36}$$

We will shortly need the results of some calculations. Recall from the hypothesis of Theorem 2 that $d \leq \varepsilon \log_2 \log n$, with $\varepsilon < 1/2$ so that

$$2^d \leq \log^\varepsilon n = o(\log n).$$

Since ℓ is playing the role of inductive index here, we use λ for a generic equivalent in this paragraph. In general,

$$\sigma_\lambda = (\rho n_0)^{2^{d-\lambda+1}-1} = \left((n^{\eta-1} \log n) \frac{n(1 - x/n)}{2^{d+1} - 2} \right)^{2^{d-\lambda+1}-1} \leq (\log n \, n^\eta)^{2^{d-\lambda+1}-1}, \tag{37}$$

and in particular, using $\eta(2^d - 1) = (2^d - 1)/(2^{d+1} - 1) \leq 1/2$,

$$\sigma_1 \leq (\log n)^{2^d-1} n^{\eta(2^d-1)} \leq (\log n)^{\log^\varepsilon n} n^{1/2} = n^{1/2+o(1)}. \tag{38}$$

From (35), $N \le \sigma_{\ell+1}^2$ and, using (32) and (38),

$$\rho N \le \rho\sigma_{\ell+1}^2 = \frac{1}{n_0}\left(n_0\rho\sigma_{\ell+1}^2\right) \le \frac{1}{n_0}\sigma_\ell \le \frac{1}{n_0}\sigma_1 \le n^{-1/2+o(1)}. \tag{39}$$

Returning now to (36) and the first part of (32), the expectation of the binomial in the RHS is

$$n_0\rho N(1 - \rho N) = (1 - \rho N)(1 - \delta)^{2^{d-\ell+1}-2}\sigma_\ell,$$

and (33) holds if the binomial's value is at least $(1 - \delta)^{2^{d-\ell+1}-1}\sigma_\ell$, i.e., if it is at least $1 - \zeta :=$ $(1 - \delta)/(1 - \rho N)$ times its expectation.

Now (39) gives $\rho N = n^{-1/2+o(1)} \ll 1/\log^2 n = \delta$ and implies that $\zeta = \delta - o(\delta)$, implying $\zeta^2/3 > \delta^2/4$. Then the Chernoff bound gives us

$$\mathbf{Pr}\left(\left|S_m^\ell(a)\right| < (1 - \delta)^{2^{d-\ell+1}-1}\sigma_\ell\right) \le \exp\left\{-\zeta^2(1 - \delta)^{2^{d-\ell+1}-2}\sigma_\ell/3\right\}$$

$$\le \exp\left\{-\delta^2(1 - \delta)^{2^{d-\ell+1}-2}\sigma_\ell/4\right\} \le \exp\left\{-\delta^2(1 - \delta)^{2^d}\sigma_\ell/4\right\}. \tag{40}$$

The failure probabilities in (34) and (40) are both $\le \exp\left\{-\delta^2(1 - \delta)^{2^d}\sigma_d/4\right\}$. Recall that $2^d = o(\log n)$, so

$$\sigma_d = \rho n_0 \gg n^\eta \ge n^{1/\log^{1/2} n} \ge (\log n)^K$$

for any fixed K. Using this and $(1 - \delta)^{2^d} \ge 1 - \delta 2^d = 1 - o(1)$, the failure probability is at most

$$\exp\left\{-\frac{\delta^2\sigma_d}{4 + o(1)}\right\} \le \exp\left\{-\Omega(\delta^2\sigma_d)\right\} \le \exp\left\{-\Omega((\log n)^{K-4})\right\} \le \exp\left\{-\log^2 n\right\}.$$

The sum of all failure probabilities is thus $\le 2^{d+1}\exp(-\log^2 n)$, and q.s. there is no failure. $\square$

Claim 19 establishes that, q.s., $\left|S_1^1(n)\right|, \left|S_2^1(n)\right| \ge (1 - o(1))\sigma_1$. Since S_0^0 simply consists of triples $(n, a, b) \in G$ with $a \in S_0^1(n)$ and $b \in S_1^1(n)$, and no triple with first index n has previously been considered, conditioned on successful progress of the algorithm to this point, S_0^0 is empty with probability at most $(1 - \rho)^{\sigma_1^2/2} \le \exp\left(-\rho\sigma_1^2/2\right)$. But

$$\rho\sigma_1^2 = \frac{1}{n_0}\sigma_0 \ge \frac{2^{d+1} - 2}{n(1 - x/n)}\left((n^{\eta-1}\log n)\frac{n(1 - x/n)}{2^{d+1} - 2}\right)^{2^{d+1}-1}$$

$$\ge n^{-1+\eta(2^{d+1}-1)}(2^{d+1} - 2)\left(\frac{\log n}{2^{d+1} - 1}\right)^{2^{d+1}-1} = (2^{d+1} - 2)\left(\frac{\log n}{2^{d+1} - 1}\right)^{2^{d+1}-1}$$

$$\ge 6(\log n/7)^7 \gg \log n.$$

The penultimate inequality comes from noting that the function $(x - 1)(a/x)^x$ is log-concave for $x > 0$ and so, over an interval, is minimized at one of the endpoints. Taking $a = \log n$, putting $d = 2$ gives $x = 7$ for a value of $6(\log n/7)^7$, while putting $d = \frac{1}{2}\log_2 \log n$ (the right endpoint of a larger interval than is allowed) gives a larger value.

This shows that the final phase succeeds q.s.

5.6   Execution time

It is fairly clear that the algorithm runs in polynomial time, but we will show that it runs in linear time (remembering that the input is of size n^3). Specifically we will show that GREEDYPHASE takes time $\Theta(n^3)$, while the other two phases take slightly less: MAINPHASE runs in time $O(n^{3-\frac{5}{4}\eta})$, and FINALPHASE in time only $O(n^{\frac{5}{2}+o(1)})$.

The aspects of each round of MAINPHASE that are potentially time-critical are SETCONSTRUCT, and, for each iteration: finding a healthy element, MAKETREE, and POISONPROPAGATE. The last two cannot dominate. MAKETREE merely repeatedly finds, for a (non-poisoned) triple, the (unique) two elements that generated it in SETCONSTRUCT; thus if we create pointers during SET-CONSTRUCT, this takes time linear in the size of the output. Since in all iterations of MAKETREE all the output elements are distinct, the total time is bounded by the size of SETCONSTRUCT's output and thus by its running time. Similarly, in POISONPROPAGATE, the elements poisoned by a given element are a direct function of the output of SETCONSTRUCT and can be driven by pointers, so each poisoning takes unit time. We contrived to poison at most half the output of SETCONSTRUCT, so again the time is bounded by SETCONSTRUCT's time. (Some elements may be poisoned repeatedly, but in our bound on the sizes of the poisoned sets we made the pessimistic assumption that all poisonings were distinct, so the bound still applies.)

In each round, the time to find a valid element, in Algorithm 6 (MAINPHASE) Line 10, is $O(\sigma_1^2) = O(n^2)$. We will shortly see that this is dominated by the time for SETCONSTRUCT. The running time for one round of SETCONSTRUCT is of order $\sum_{\ell=1}^{d} \sum_{m=1}^{2^\ell} n\sigma_{\ell+1}^2 = O(n\sigma_2^2)$, since $\sigma_\ell \gg \sigma_{\ell+1}$ (see Claim 13) means that the time for each node (ℓ, m) dominates the sum of the times for its two children, so the $\ell = 1$ terms dominate the sum. By (19), $n\sigma_2^2 = (1 + o(1))\frac{1}{\rho}\sigma_1$, and by (26), $\sigma_1 \le n^{1-\eta/3}$. Using $x \le n^{1-\eta}$ we have $1/\rho = x^{1+\eta}n^{1-\eta}/\log n \le n^{2-\eta-\eta^2}$, and thus $\frac{1}{\rho}\sigma_1 \le n^{3-\frac{4}{3}\eta}n$. In all there are $O(\log^2 n)$ rounds, so by familiar calculations the total time for all calls to SETCONSTRUCT is $O(n^{3-\frac{5}{4}\eta})$.

We reason similarly for FINALPHASE. The time per round to find a valid element (Algorithm 7 Line 10) is $O(\sigma_1^2) = O(n^2)$, which we shall shortly see is dominated by the time per round for set construction. The set construction time (Algorithm 9) is of order $\sum_{\ell=1}^{d} \sum_{m=0}^{2^\ell-1} n_0^2 \sigma_{\ell+1}^2 = O(n_0^2\sigma_2^2) = O(\frac{n_0}{\rho}\sigma_1)$. By (27) and (38), this is $O\left(\frac{n}{n^{\eta-1}}n^{1/2+o(1)}\right)$. There are $2^d - 1 \le \log^\varepsilon n$ rounds, so the total time is $n^{\frac{5}{2}-\eta+o(1)}$.

This completes the proof of Theorem 2.

6   Difficulties with Axial assignment in dimensions $D > 3$

Neither MAINPHASE nor FINALPHASE successfully carries over to $D > 3$. Consider MAINPHASE with depth $d = 2$. Our algorithm was complicated in order to cope with conditioning, but ultimately an augmentation, for $D = 3$, relied on choosing $2 + 4 = 6$ indices $j, \dots, s$ from A and 8 indices $\xi_1, \dots, \xi_8$ from $\bar{A}$ to make $1 + 2 + 4 = 7$ triples cheap. If "cheap" means $\le \rho$, by the first-moment method this can succeed with high probability only if $n^6 x^8 \rho^7 = \Omega(1)$, forcing $\rho = \Omega(n^{-6/7}x^{-8/7})$. When the number of unassigned elements x is $\Theta(1)$, this means that a single augmentation costs $\Omega(n^{-6/7})$: more than the $O(n^{2-D}\log n)$ bound on the *total* assignment cost from Theorem 1, but not vastly more. The equivalent for $D = 4$, with a ternary tree, is to choose $3 + 9 = 12$ indices from A and 27 from $\bar{A}$ to make $1 + 3 + 9 = 13$ triples cheap, requiring $\rho = \Omega(n^{-12/13}x^{-27/13})$. Here, $x = \Theta(1)$ gives $p = \Omega(n^{-12/13})$, which — still stuck above n^{-1} — is now vastly worse than the

$O(n^{2-D} \log n)$ upper bound.

FINALPHASE, in its last round, performs a replacement of some $k = 2^d - 2$ assignment D-tuples — which, up to symmetries, we take to be $\{(i, i, \ldots, i), 1 \le i \le k\}$ — with $k + 1$ D-tuples from $([k] \cup \{n\})^D$. When k is a constant, the probability that there is any new such assignment of cost $\le \rho$ is at most $\binom{n}{k}(k!)^{D-1}\rho^k = O((n\rho)^k)$, so for the algorithm to have any hope of succeeding requires $\rho = \Omega(n^{-1})$. Again this is satisfactory for $D = 3$, but for $D > 3$ it is vastly worse than the $O(n^{2-D} \log n)$ upper bound.

7 Multi-dimensional Planar assignment

7.1 Main theorem

Here we give our main theorem for Planar assignment.

Theorem 3. *The optimal solution value $Z_{D,n}^P$ satisfies the following:*

(a) $Z_{D,n}^P = \Omega(n^{D-2})$ *w.h.p. for $D \ge 3$.*

(b) *When $D = 3$ there is a polynomial-time algorithm that finds a solution with cost Z where $Z = O(n \log n)$ w.h.p.*

The theorem is proved in the next two sections.

7.2 Lower bound

From the defining equation (2) for Planar assignment, just attending to constraints of the first type, for each choice $i_1, \ldots, i_{D-1}$ of the first $D - 1$ coordinates, the cheapest element in the line (ranging over the last coordinate) has distribution $\mathrm{Exp}(n)$, with expectation $1/n$. Summing over all n^{D-1} such lines, the total expected cost is $\ge n^{D-2}$. Each line, independently, has cost $\ge 1/n$ with probability $1/e$, so there are $\mathrm{Bin}(n^{D-1}, 1/e)$ lines for which this is so, by a Chernoff bound q.s. at least $n^{D-1}/(2e)$ lines have this property, and thus q.s. the total cost is $n^{D-2}/(2e)$.

7.3 Upper bound for $D = 3$

For the upper bound we need a result of Dyer, Frieze and McDiarmid [DFM86]. We will not state it in full generality, instead tailoring its statement to precisely what is needed here. Suppose that we have a linear program

$$\text{LP:} \quad \text{Minimize } c^T x \text{ subject to } Ax = b,\ x \ge 0.$$

Here A is an $m \times n$ matrix and the cost vector $c = (c_1, \ldots, c_n)$ is a sequence of independent copies of $\mathrm{Exp}(1)$. Let Z_{LP} denote the minimum of this linear program. Note that Z_{LP} is a random variable. Next let y be *any* feasible solution to LP.

Theorem 4 ([DFM86])**.**

$$\mathbf{E}(Z_{\mathrm{LP}}) \le m \max_{j=1,\ldots,n} y_j. \tag{41}$$

Furthermore, Z_{LP} is at most $1 + o(1)$ times the RHS of (41), w.h.p.

31

Our algorithm is based on a characterization of a 3-dimensional Planar assignment as a collection of 2-dimensional Planar assignments. (We think of these as $(D-1)$-dimensional Planar assignments, but in 2 dimensions Planar and Axial assignment are the same.) for For generalization to dimensions $D > 3$ we think of these for purposes o Fix a 1-coordinate i. Then for every 2-coordinate j, by the first constraint type of (2), the line $(i, j, *)$ must contain exactly one assignment element, and by the third constraint type, the same is true for every 3-coordinate k and line $(i, *, k)$. Thus, the assignment elements in plane $(i, *, *)$ define a 2-dimensional assignment.

Of course there are constraints among these 2-dimensional assignments. By the second constraint type, for $i \neq i'$, the elements (i, j, k) and (i', j, k) cannot both be selected; that is,

$$(\forall j, k) \sum_i X_{i,j,k} \leq 1. \tag{42}$$

This condition is sufficient as well as necessary: any collection of 2-dimensional assignments satisfying it is a 3-dimensional assignment. This is clear because the collection of 2-dimensional assignments has n^2 elements, thus $\sum_{j,k} \sum_i X_{i,j,k} = n^2$, thus if (42) is always satisfied, it must in every case be satisfied with equality. In that case the selection satisfies the second constraint type of (2) as well as the first and third, and is a 3-dimensional assignment.

Now consider the following greedy-type algorithm. First, find a minimum 2-dimensional assignment for 1-plane $i = 1$. For each element (j, k) selected, remove (j, k) from consideration for 2-dimensional assignments for all 1-planes i' with $i' > i$. Find a minimum assignment for 1-plane $i = 2$ with this restriction, remove these elements from consideration, and so on.

Algorithm 10 GREEDYPLANAR3D: construct a 3-dimensional Planar assignment

1: **for** $i = 1, \ldots, n$ **do**
2:     Let $H_i = K_{n,n} \setminus (\Xi_1 \cup \cdots \cup \Xi_{i-1})$
3:     Give each edge $(j, k) \in E(H_i)$ cost $M_{i,j,k}$
4:     Let Ξ_i be a minimum cost matching of H_i
5: Return $\Xi_1, \ldots, \Xi_n$

The output $\Xi_1, \ldots, \Xi_n$ defines a set of triples $T = \{(i, j, k) : (j, k) \in \Xi_i\}$ that by the previous discussion is a 3-dimensional Planar assignment. Writing Z_i for the cost of matching Ξ_i, we claim that

$$\mathbf{E}(Z_i) \leq \frac{2n}{n - i + 1}. \tag{43}$$

For this we apply Theorem 4 to the following linear program, which we note always has an integer optimum solution:

$$\text{Minimize} \sum_{(j,k) \in E(H_i)} M_{i,j,k} \, x_{j,k} \quad \text{subject to}$$

$$\sum_{k:\,(j,k) \in E(H_i)} x_{j,k} = 1, \quad j = 1, \ldots, n$$

$$\sum_{j:\,(j,k) \in E(H_i)} x_{j,k} = 1, \quad k = 1, \ldots, n$$

$$x_{j,k} \geq 0, \quad j, k = 1, \ldots, n.$$

We note that there are $2n$ constraints and that $x_{j,k} = 1/(n - i + 1)$ is a feasible solution. By Theorem 4, this implies (43) and the upper bound in Theorem 3 for the case $D = 3$.

8 Difficulties with Planar assignment in dimensions $D > 3$

The previous section's characterization of a 3-dimensional assignment as a collection of n 2-dimensional assignments extends: a D-dimensional assignment is a collection of n $(D-1)$-dimensional assignments, one for each 1-index i_1, satisfying the property that if $(i_2, \ldots, i_n)$ belongs to assignment i_1 then it belongs to no other assignment i_1', i.e., $(\forall i_2, \ldots, i_n) \sum_{i_1} X_{i_1, \ldots, i_n} \leq 1$. As before, this constraint is necessary (by the defining constraint on lines for coordinate 1) and also sufficient: by the same counting argument as before, if the constraint is always satisfied then it is always satisfied with equality, which thus implies the assignment-defining constraint for lines along coordinate 1. The constraints for lines along other coordinates are satisfied since each is included in some $(D-1)$-dimensional assignment.

The corresponding algorithm we have in mind is a generalization of Algorithm 10: repeatedly solve $(D-1)$-dimensional instances and remove their elements from future consideration. Each lower-dimensional instance is solved by a recursive call to the same algorithm, until dimension 2 is reached and the instance is solved as an LP. Note that all but the first of the $(D-1)$-dimensional instances has only a subset of the elements available for assignment: it is a matching problem on a $(D-1)$-partite hypergraph, but not the complete $(D-1)$-partite hypergraph. That is, the generalized algorithm should solve "matching" problems, not just "assignment" problems.

The difficulty is that such problems do not always admit a solution, and even when we start with the complete hypergraph — an assignment problem, which does always have a solution — we may generate subinstances that do not.

A small example is shown below, with $D = 4$ and $n = 4$. Suppose that in the first 3-dimensional sub-array considered (shown as four 2-dimensional arrays), the elements selected are those indicated by 1s in the table below; this will happen, for example, if the corresponding cost elements are small and the others are large.

```
0 1 0 0    0 0 0 1    0 0 1 0    1 0 0 0
0 0 1 0    1 0 0 0    0 1 0 0    0 0 0 1
1 0 0 0    0 1 0 0    0 0 0 1    0 0 1 0
0 0 0 1    0 0 1 0    1 0 0 0    0 1 0 0
```

Then, in the second 3-dimensional sub-array, perhaps the following selection gives the cheapest assignments for the first three 2-dimensional instances solved.

```
0 0 1 0    0 1 0 0    0 0 0 1
0 1 0 0    0 0 1 0    1 0 0 0
0 0 0 1    1 0 0 0    0 1 0 0
1 0 0 0    0 0 0 1    0 0 1 0
```

There is no way to complete this 3-dimensional array: the last 2-dimensional instance has no solution. We check that it is impossible to select any element in its first row. The first row's first element is blocked by the 2-dimensional array above it (the 4th 2-dimensional matching comprising the first 3-dimensional matching) and the row's remaining elements are blocked by the 2-dimensional arrays to the left of it (respectively, by the 2nd, 1st, and 3rd 2-dimensional matchings for the second 3-dimensional matching problem).

This example shows that in dimensions $D > 3$, the obvious generalization of our 3-dimensional Planar assignment algorithm may fail to return any solution, regardless of cost.

9 Conclusions

For the 2-dimensional random assignment problem, we know the limiting expected cost, and a given instance can be solved in polynomial time. As noted in the introduction, much less is known about multi-dimensional assignment problems, and as far as we are aware, very little was known about polynomial-time algorithms solving these problems well on average, especially for random costs with a density $f(x)$ that is close to a constant for x close to zero.

For the 3-dimensional Axial assignment problem, we give an upper bound within $n^{o(1)}$ of the obvious $\Omega(1/n)$ lower bound, as a trivial application of an extension of a result of Johansson, Kahn and Vu [JKV08]. Our main result is an algorithm that constructs w.h.p. constructs a solution of cost $n^{o(1)}$, although the cost bound here is not as small as that coming from [JKV08].

For the 3-dimensional Planar assignment problem, we prove an upper bound within a logarithmic factor of the obvious $\Omega(n)$ lower bound (likely the true answer), by analyzing a simple and fast greedy algorithm.

Neither result extends to $D > 3$. We are left with open questions including these:

P1 What are the growth rates of $\mathbf{E}[Z_{D,n}^A]$ and $\mathbf{E}[Z_{D,n}^P]$ for $D \geq 3$?

P2 Are there asymptotically optimal, polynomial-time algorithms for solving these problems when $D \geq 3$?

P3 For $D > 3$, are there polynomial-time algorithms yielding solutions within logarithmic or $n^{o(1)}$ factors for Planar and Axial assignment problems (as we have given for $D = 3$)?

References

[Ald92] D. Aldous, *Asymptotics in the random assignment problem*, Probab. Theory Related Fields **93** (1992), no. 4, 507–534.

[Ald01] D. J. Aldous, *The $\zeta(2)$ limit in the random assignment problem*, Random Structures Algorithms **18** (2001), no. 4, 381–418.

[AS99] N. Alon and J. Spencer, *The probabilistic method*, third, Wiley, 1999.

[BCR02] M. W. Buck, C. S. Chan, and D. P. Robbins, *On the expected value of the minimum assignment*, Random Struct. Algorithms **21** (2002), no. 1, 33-58.

[BDM09] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

[CS99] D. Coppersmith and G. B. Sorkin, *Constructive bounds and exact expectations for the random assignment problem*, Random Structures Algorithms **15** (1999), no. 2, 113–144.

[DFM86] M. E. Dyer, A. M. F. Frieze, and C. J. H. McDiarmid, *On linear programs with random costs*, Mathematical Programming **35** (1986), 3–16.

[DK07] S. A. Dichkovskaya and M. K. Kravtsov, *Investigation of polynomial algorithms for solving the multicriteria three-index planar assignment problem*, Computational Mathematics and Mathematical Physics **47** (2007), 1029–1038.

[Don69] W. E. Donath, *Algorithm and Average-Value Bounds for Assignment Problems*, IBM Journal of Research and Development **13** (1969), no. 4, 380–386.

[Fri83] A. M. Frieze, *Complexity of a 3-dimensional assignment problem*, European J. Oper. Res. **13** (1983), no. 2, 161–164.

[Fri85] _____, *On the value of a random minimum spanning tree problem*, Discrete Appl. Math. **10** (1985), no. 1, 47–56.

[GM13] S. Gerke and A. McDowell, *Non-Vertex-Balanced Factors in Random Graphs*, 2013. http://arxiv.org/abs/1304.3000.

[JKV08] A. Johansson, J. Kahn, and V. Vu, *Factors in random graphs*, Random Structures Algorithms **33** (2008), no. 1, 1–28.

[Kar72] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972), 1972, pp. 85–103.

[KGP07] P. Krokhmal, D. Grundel, and P. Pardalos, *Asymptotic behavior of the expected value of the random multidimensional assignment problems*, Mathematical Programming Series B **109** (2007), 525–551.

[Kra05] V. M. Kravtsov, *Polynomial algorithms for finding the asymptotically optimal plan of the multiindex assignment problem*, Cybernetics and Systems Analysis **41** (2005), no. 6, 940–944.

[LW04] S. Linusson and J. Wästlund, *A proof of Parisi's conjecture on the random assignment problem*, Probab. Theory Related Fields **128** (2004), no. 3, 419–440.

[MMR04] O. C. Martin, M. Mézard, and O. Rivoire, *A frozen glass phase in the multi-index matching problem*, posted on 2004, DOI 10.1103/PhysRevLett.93.217205, (to appear in print), available at `cond-mat/0407623`.

[MMR05] _____, *Random multi-index matching problems*, J. Stat. Mech. Theory Exp. **9** (2005), P09006, 36 pp. (electronic).

[MP85] M. Mézard and G. Parisi, *Replicas and Optimization*, J. Physique Lettres **46** (1985), 771–778.

[MP87] _____, *On the Solution of the Random Link Matching Problems*, J. Physique Lettres **48** (1987), 1451–1459.

[NPS05] C. Nair, B. Prabhakar, and M. Sharma, *Proofs of the Parisi and Coppersmith-Sorkin random assignment conjectures*, Random Structures Algorithms **27** (2005), no. 4, 413–444, DOI 10.1002/rsa.20084.

[Par98] G. Parisi, *A Conjecture on Random Bipartite Matching*, 1998.

[SPS85] J. Schmidt-Pruzan and E. Shamir, *Component structure in the evolution of random hypergraphs*, Combinatorica **5** (1985), no. 1, 81–94, DOI 10.1007/BF02579445.

[SS09] A. D. Scott and G. B. Sorkin, *Polynomial constraint satisfaction problems, graph bisection, and the Ising partition function*, ACM Trans. Algorithms **5** (2009), no. 4, Art. 45, 1–27, DOI 10.1145/1597036.1597049.

[Wäs05] J. Wästlund, *A simple proof of the Parisi and Coppersmith-Sorkin formulas for the random assignment problem*, Technical Report 6, Linköping Studies in Mathematics, 2005.

[Wäs09] _____, *An easy proof of the $\zeta(2)$ limit in the random assignment problem*, Electron. Commun. Probab. **14** (2009), 261–269, DOI 10.1214/ECP.v14-1475.