A New Rounding Procedure for the Assignment Problem with Applications to Dense Graph Arrangement Problems*

Sanjeev Arora† Alan Frieze‡ Haim Kaplan§

October 21, 2001

Abstract

We present a randomized procedure for rounding fractional perfect matchings to (integral) matchings. If the original fractional matching satisfies any linear inequality, then with high probability, the new matching satisfies that linear inequality in an approximate sense. This extends the well-known LP rounding procedure of Raghavan and Thompson, which is usually used to round fractional solutions of linear programs.

We use our rounding procedure to design an additive approximation algorithm to the Quadratic Assignment Problem. The approximation error of the algorithm is ϵn^2 and it runs in $n^{O(\log n/\epsilon^2)}$ time.

We also describe Polynomial Time Approximation Schemes (PTASs) for dense subcases of many well-known NP-hard arrangement problems, including MINIMUM LINEAR ARRANGEMENT, MINIMUM CUT LINEAR ARRANGEMENT, MAXIMUM ACYCLIC SUBGRAPH, and BETWEENNESS.

*Preliminary version of this paper appeared in the proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS), 1996 [AFK96].

†Computer Science, Princeton University. Supported by NSF CAREER award NSF CCR-9502747 and an Alfred Sloan Fellowship.**E-mail: arora@cs.princeton.edu**.

‡Department of Mathematics, Carnegie Mellon University, Pittsburgh PA15213. Supported in part by NSF grant CCR9225008. **E-mail: alan@random.math.cmu.edu**

§Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Part of this research was done at Princeton University, supported by NSF Grant CCR8920505 and the ONR Contract No. N00014-91-J-1463. **E-mail: haimk@math.tau.ac.il**

1 Introduction

In the *assignment problem*, we wish to minimize a linear cost function over the set of permutations. A permutation on a set of n elements is represented by the *assignment constraints* in n^2 variables $\{x_{ij} : i, j \in [n]\}$.[1]

$$\begin{array}{rcll} \sum_j x_{ij} & = & 1 & \forall i \in [n] \\ \sum_i x_{ij} & = & 1 & \forall j \in [n] \\ x_{ij} & \geq & 0 & \forall (i,j) \end{array} \tag{1}$$

A fractional solution to (1) is called a *fractional perfect (bipartite) matching* and an integral solution to (1) is called a *perfect (bipartite) matching*. We let $\mathcal{A}$ denote the set of integral solutions to (1). Thus we can state the assignment problem as

$$\begin{array}{rl} \text{minimize} & cx \\ \text{s.t.} & x \in \mathcal{A} \end{array}$$

where for two vectors x and y we denote their dot-product by xy. (Sometimes this optimization problem is stated as an equivalent decision problem: find a perfect matching that satisfies a given linear constraint.)

There are a couple of reasons why the assignment problem is so fundamental in operations research. The first is usefulness. Assigning n tasks to n people subject to a linear cost function — a canonical use of the problem— is a basic primitive in many applications. The second is tractability (which, here as in many other situations, goes hand-in-hand with mathematical elegance). This tractability traces to the fact that every vertex of the assignment polytope (1) is integral. Thus linear programming can be used to compute integral solutions.

But one can easily pose variants of the assignment problem that are just as useful, sometimes more so. This paper considers the following variant, in which the perfect matching has to satisfy more than one linear constraint. We call this problem the *Assignment Problem with Extra Constraints*, or APEC. The problem is to find a solution to

$$\begin{array}{rcll} x & \in & \mathcal{A} & \\ a_k x & \geq & b_k, & \forall k \in [K] \end{array} \tag{2}$$

The vertices of the APEC polytope are not necessarily integral any more. Hence solving system (2) as a linear program does not guarantee to produce an integral solution. In fact, we would not expect *any* polynomial-time algorithm to produce integral solutions, since APEC is NP-hard (it contains integer linear programming as a subcase).

[1] By $[n]$ we denote the set of integers $\{1, 2, \ldots, n\}$.

One could try to compute "approximate" integer solutions by solving (2) as a linear program, and then using Raghavan-Thompson rounding [RT87] on the resulting fractional solution. This does not work for the following reason. Given a fractional solution (x_{ij}), Raghavan-Thompson rounding produces a 0/1 vector (y_{ij}) by making $y_{ij} = 1$ with probability x_{ij}. The vector y obtained in this way is quite unlikely to look anything like a perfect matching. For example, if all $x_{ij} = 1/n$, then the expected number of unmatched vertices in y is approximately n/e and the expected number of vertices with two neighbors is $\Theta(n)$. Note however that vector y does satisfy the cost constraints in an approximate sense. A Chernoff-type bound shows that with high probability, y satisfies

$$a_k y \geq b_k - \tilde{O}(\sqrt{n} A_k) \qquad \forall k \in [K], \tag{3}$$

where A_k is the largest (in magnitude) coefficient of a_k and $\tilde{O}$ is the usual "soft-Oh" notation that suppresses polylogarithmic factors. We're assuming that K is not too large, say poly(n). (Note that though the $\sqrt{n}$ factor may appear to come out of the standard Hoeffding bound [H64], it actually requires a more delicate analysis, since the number of variables is n^2 and not n. See Section 7.)

This paper presents a new randomized rounding procedure for fractional matchings (see Theorem 1.3). Instead of rounding all variables independently as in the Raghavan-Thompson technique, the procedure rounds them in a fashion that is "not independent yet independent enough." Our procedure can start with any fractional matching in a bipartite graph. When the initial fractional matching is a perfect fractional matching the procedure produces a matching that contains $n - o(n)$ edges[2]. The integral matching that we produce "approximately" satisfies (in the same sense as (3)), with high probability, any linear inequality that was satisfied by the fractional matching. Although our procedure does not yield a perfect matching ($o(n)$ jobs do not get assigned to any person), it may well prove good enough in many situations.

We use the rounding procedure to derive an additive approximation algorithm for a version of the well-known Quadratic Assignment Problem or QAP (see [BC96, PRW94] for recent surveys on this problem). In this problem, we are given a set of coefficients $\{c_{ijkl} \in \mathcal{Z} : 1 \leq i,j,k,l \leq n\}$ and desire to

$$\begin{array}{ll} \text{minimize} & c(x) = \sum_{i,j,k,l} c_{ijkl} x_{ij} x_{kl} \\ \text{s.t.} & \qquad x \in \mathcal{A} \end{array} \tag{4}$$

We impose the restriction that each $|c_{ijkl}| \leq C$ for some fixed $C > 0$ independent of n. Even with this restriction, no polynomial-time approximation algorithm can approximate QAP within *any* multiplicative factor if $P \neq NP$. This follows from the following reduction from Hamiltonian Cycle to QAP. Given a graph $G = (V, E)$, $V = \{1, 2, \ldots, n\}$ we define a 0-1 cost function c such that there exists $x \in \mathcal{A}$ with $c(x) = 0$ iff G contains

[2]By $o(g(n))$ we mean a function $f(n)$ such that $f(n)/g(n)$ goes to zero as n goes to infinity.

an Hamiltonian Cycle. The cost function c is defined such that $c_{ijkl} = 1$ if $l = 1+j \mod n$ and $(i,k) \notin E$ and $c_{ijkl} = 0$ otherwise. Given an assignment x such that $c(x) = 0$ we define an hamiltonian cycle by making i the jth vertex of the cycle if and only if $x_{ij} = 1$. Similarly, given an hamiltonian cycle we can define a vector x such that $c(x) = 0$.

Using our rounding procedure we describe an algorithm that runs in $n^{O(\log n/\epsilon^2)}$ time and finds an integral x satisfying

$$c(x) \leq c(x^*) + \epsilon n^2,$$

where x^* is the *fractional* optimum.

Our additive approximation for the QAP uses a random-sampling based technique as in Arora, Karger, and Karpinski [AKK95] to first reduce the degree of the optimization problem from 2 to 1. (The reduction is accompanied by some loss of accuracy.) The degree-reduced problem happens to be an instance of APEC. We solve that instance using linear programming and then round the resulting solution using our rounding procedure.

We use our additive approximation for the QAP to design Quasi-Polynomial Time Approximation Schemes for "dense" instances of many NP-hard optimization problems. In case of graphs, "denseness" means that the average degree is $\Omega(n)$. The problems we consider include GRAPH ISOMORPHISM formulated as an optimization problem, MINIMUM LINEAR ARRANGEMENT, MINIMUM CUT LINEAR ARRANGEMENT, MAXIMUM ACYCLIC SUBGRAPH, and BETWEENNESS. Our approximation schemes compute a $(1+\epsilon)$-approximation in $n^{O(\log n/\epsilon^2)}$ time ($n =$ the number of nodes). For some of the problems listed above, by reducing the space on which we do an exhaustive search, we give algorithms that run in $n^{O(1/\epsilon^2)}$ time (and thus are PTASs or Polynomial Time Approximation Schemes). Our algorithms are randomized but, with the exception of the PTAS for BETWEENNESS, we describe how to derandomize them using fairly standard techniques.

Finally, we remark that our rounding procedure can be implemented efficiently on a parallel multiprocessor. For example, it can be made to run on an EREW PRAM in time $O(\log n \log\log n)$ (i.e., it is in the complexity class RNC^2).

1.1 Related Work

Without additional constraints the assignment problem has been intensively studied. It is solvable in polynomial time by linear programming (for more details see [Law76]). Even more interesting is the parallel complexity of this problem. The parallel complexity of the related perfect matching problem (in bipartite graphs) has been intensively studied. In this problem you simply want to find a perfect matching in a given bipartite graph (that admits a perfect matching). One can easily encode this problem into the assignment problem by assigning higher weight to nonexisting edges. To date it is still not known whether this problem is in the class NC (i.e. admits a PRAM algorithm that runs in polylogarithmic time using polynomially many processors). Karp, Upfal and Wigderson [KUW86] showed that the problem is in RNC (see [MVV87] for a simpler and more efficient algorithm). They also showed that finding a perfect matching with maximum weight in an edge-weighted graph is in RNC if the weights are bounded by a polynomial of the number of vertices in the graph.

Approximating the size of the maximum matching in parallel turns out to be easier. Fisher et al [FGHP93] give an NC approximation algorithm for the maximum cardinality matching problem in general graphs. For the case of bipartite graphs, Cohen [Cohen93] developed a more efficient algorithm that finds a bipartite matching of cardinality $(1 - 1/\text{poly}\log(n))$ times the maximum using $O(m)$ processors where m is the number of edges in the graph. Cohen's algorithm uses a previous technique by Goldberg et al [GPST92] to round in NC a fractional matching of value M into a matching of size at least $M - 1/poly(n)$.

Luby and Nisan in [LN92] give an NC^1 approximation algorithm for positive linear programming problems. They observed that one can use their algorithm to approximately find the weight of the maximum matching in a bipartite graph. However, Luby and Nisan did not know how to round their approximate fractional solution to the linear program into a matching except by using Cohen's and Goldberg's techniques [Cohen93, GPST92]. We provide a new rounding technique that achieves this goal and works for general weight functions. When the weights are bounded by a constant then our technique produce a matching whose weight is within an additive error of $o(n)$ from the weight of the fractional matching. In particular when weights are all ones we obtain a matching that contains $o(n)$ edges less than the maximum one. (For general weights the error depends also on the maximum weight.) Furthermore if the original fractional solution satisfies additional linear constraints then the matching that we obtain will approximately satisfy these constraints as well.

Shmoys and Tardos [ST93] consider approximation algorithms for the *generalized assignment problem*, which involves generalized assignment constraints mixed with other linear constraints. However, they don't allow negative coefficients in the constraints. We don't impose this restriction, but then settle for additive approximation instead of multiplicative approximation. We also return an almost-perfect matching as a solution, which [ST93] could not.

Previous work on PTAS's. PTAS's are known for relatively few problems; KNAPSACK [IK75] and BIN-PACKING [FL81, KK82] are the only two well-known examples. Recently, a PTAS has also been discovered for Euclidean TSP (Arora [A96]), for Multiple Knapsack Problem (Chekuri and Khanna [CK00]) and some Scheduling Problems (Afrati et al [A99]). In fact, a result by Arora, Lund, Motwani, Sudan and Szegedy

[ALM+92] shows that if P $\neq$ NP, then PTAS's do not exist for a large body of problems – the so-called MAX-SNP-hard problems. Similar (or stronger) "inapproximability" results have since been proven for many other problems.

Such inapproximability results have raised a very natural question: What subcases of these problems are "easy" with respect to approximation? A paper by Baker [B94] (which actually predated the inapproximability results) showed that many NP-hard graph problems have PTAS's on planar graphs; more recently Khanna and Motwani [KM96] extended her work. Recent work of Arora, Karger and Karpinski [AKK95] and Frieze and Kannan [FK99] shows that many MAX-SNP-hard problems have a PTAS when the instance is "dense" (in the case of graphs, this means that the average degree is $\Omega(n)$). Fernandez de la Vega [FdlV94] independently gave a PTAS for dense instances of MAX-CUT and some other problems.

Since the first appearance of our paper Frieze and Kannan [FK96, FK99] have developed a beautiful theory based on random sampling for approximation algorithms for dense graph problems [FK96, FK99] and the related problem of property testing [GGR96]. The results in [FK96, FK99] rely on decomposing matrices into sums of simple matrices in a way inspired by the regularity lemma of Szemerédi [S78]. Using this technique they obtain a polynomial time approximation scheme for a restricted version of the QAP that contains the other arrangement problems that we consider here as subcases. Asymptotically, the running times of the algorithms of Frieze and Kannan are better than ours. Their approach, however, is more complicated, and their algorithms are impractical. Frieze and Kannan did not consider rounding fractional matchings and the general version of QAP that we deal with here.

1.2 Approximating Dense Instances of Arrangement Problems

In this section we define the arrangement problems that we consider in this paper and state our main approximabity results regarding dense instances of these problems. Depending upon the problem, we denote a permutation in one of two ways: as a bijection $\pi\colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ or as a vector (x_{ij}) that satisfies the assignment constraints. Recall that we are interested in *dense* subcases of graph problems. A graph is *a-dense* if the number of edges is at least $a \cdot n^2$. We will explain what denseness means for problems that are not graph problems.

QUADRATIC ASSIGNMENT: Given a set $\{c_{ijkl} \in \mathcal{Z} : 1 \leq i, j, k, l \leq n\}$, find a permutation (x_{ij}) that minimizes $c(x) = \sum_{i,j,k,l} c_{ijkl} x_{ij} x_{kl}$. We will be interested in the special case when each $|c_{ijkl}| = O(1)$. We call an instance of the quadratic assignment problem dense if the minimum value of $c(x)$ is $\Omega(n^2)$.

GRAPH ISOMORPHISM: The input consists of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$

such that $V_1 = V_2 = [n]$. We define the cost $c(\pi)$ of a permutation π mapping V_1 to V_2 to be the sum of the number of edges $(i, j) \in E^1$ such that $(\pi(i), \pi(j)) \notin E^2$ and the number of edges $(i, j) \in E^2$ such that $(\pi^{-1}(i), \pi^{-1}(j)) \notin E^1$. The problem is to find a permutation π of minimum cost. A dense instance of this problem is a very nonisomorphic pair of graphs. More specifically, we define an instance to be dense if the cost of the best permutation is $\Omega(n^2)$.

Note that this problem is easily reducible to the quadratic assignment problem by setting $c_{ijkl} = 0$ iff $(i, k) \in E^1$ and $(j, l) \in E^2$ and setting $c_{ijkl} = 1$ otherwise. Our results about GRAPH ISOMORPHISM will follow directly from our results about QUADRATIC ASSIGNMENT and this reduction.

MINIMUM LINEAR ARRANGEMENT: Given a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, find a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that minimizes $c(\pi) = \sum_{(i,j)\in E} |\pi(i) - \pi(j)|$. In other words, the goal is to lay out G along the points $1, 2, \ldots, n$ on a straight line, such that the total edge length in the layout is minimized. Leighton and Rao [LR88] gave an $O(\log^2 n)$-factor approximation algorithm for this problem on general graphs.

d-DIMENSIONAL ARRANGEMENT: The analogue of the linear arrangement problem when we have n^d points on a d-dimensional grid instead of n points on a line. We are looking for a mapping π from the vertices of the graph to the points of the grid that minimizes the sum of the lengths of the edges. The *length* of edge (i, j) is the Manhattan distance from $\pi(i)$ to $\pi(j)$. Here d is a constant.

MINIMUM CUT LINEAR ARRANGEMENT: Given a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, find a permutation π that minimizes $c(\pi) = \max_i |\{(k, l) \in E | \pi(k) \leq i < \pi(l)\}|$. The minimum value of $c(\pi)$ is called the *cutwidth* of the graph. Leighton and Rao [LR88] gave an $O(\log^2 n)$-factor approximation algorithm for this problem on general graphs.

BETWEENNESS: Given a finite set A, $|A| = n$ and a collection $\mathcal{C}$ of ordered triples (a, b, c) of distinct elements from A. Find a permutation π of A that maximizes the number of triples (a, b, c) such that either $\pi(a) < \pi(b) < \pi(c)$ or $\pi(c) < \pi(b) < \pi(a)$. A *dense* instance is one in which the number of triples is $\Omega(n^3)$. A constant factor approximation algorithm is trivial (just pick a random permutation).

MAXIMUM ACYCLIC SUBGRAPH: Given a digraph $G = (V, E)$, find the largest (with respect to the number of edges) acyclic subgraph in it. Here, again, a constant-factor approximation algorithm is trivial. We pick a random permutation π of the graph and take as the subgraph the largest among the set of arcs (i, j) such that $\pi(j) > \pi(i)$ and the set of arcs (i, j) such that $\pi(j) < \pi(i)$.

Most problems in the above list involve optimizing an objective function that is a degree d polynomial in (x_{ij}) (In fact, the degree d is 2 for most of the problems, and 3 for BETWEENNESS.) The only problem that doesn't fall into this classification is MINIMUM

CUT LINEAR ARRANGEMENT, which we will deal with separately. Now we formalize this class of problems.

Definition 1 *A* degree-d arrangement problem *is of the type*

$$\begin{aligned} \textit{minimize} \quad & p(x) \\ \textit{s.t.} \quad & x \in \mathcal{A} \end{aligned}$$

where p is a degree d polynomial.

The problem is c-smooth *if each coefficient of p is an integer in $[-c, c]$.*

Remarks: (i) Our results apply also to degree-d arrangement problem that require to maximize a degree-d polynomial rather than minimizing it. We can convert one to another by negating the coefficients of the polynomial.
(ii) For $d = 2$ this problem is the quadratic assignment problem.

We now summarize our results about the problems listed above in the following two theorems.

Theorem 1.1 *There is an algorithm that, given any c-smooth degree-d arrangement problem and an ϵ, finds a matching (x_{ij}) that contains at least $(1-\epsilon)n$ edges, and which satisfies*

$$p(x) \leq p(x^*) + \epsilon n^d,$$

where x^* is the perfect matching at which p attains its minimum value. The algorithm runs in $n^{O(c^2 d^4 \log n/\epsilon^2)}$ time.

Theorem 1.2 *Every problem in the list above has an $n^{O(\log n/\epsilon^2)}$ time approximation scheme on dense instances. In addition* MINIMUM LINEAR ARRANGEMENT, d-DIMENSIONAL ARRANGEMENT, MINIMUM CUT LINEAR ARRANGEMENT, MAXIMUM ACYCLIC SUBGRAPH *, and* BETWEENNESS *have an $n^{O(1/\epsilon^2)}$ time approximation schemes*[3].

Part of Theorem 1.2 is a corollary to Theorem 1.1. The $n^{O(1/\epsilon^2)}$ time approximation schemes are obtained using techniques similar to those of [AKK95] and will be treated separately in Section 4. Alternative strategies for some of these problems can be found in [FK99].

In particular if we apply Theorem 1.1 to GRAPH ISOMORPHISM we obtain an algorithm to decide whether a given pair of graphs is very non-isomorphic. Given some $\delta > 0$ we can use the algorithm specified by Theorem 1.1 for some $\epsilon \ll \delta$ to obtain a procedure that can identify correctly any pair of graphs G_1 and G_2 such that every bijection from V_1 to V_2 would leave δn^2 unmatched edges. Our procedure may classify incorrectly pairs of graphs for which the best bijection leaves between $(\delta - \epsilon)n^2$ and δn^2 unmatched edges.

[3]The algorithm for BETWEENNESS is randomized.

1.3 Assignment Problem with Additional Linear Constraints

The APEC problem that we defined in Equation (2) corresponds to finding a perfect matching in a complete bipartite graph that satisfies a bunch of extra linear constraints. More generally we can define an APEC problem for any bipartite graph as follows. Let $G = (V_1, V_2, E)$ be a bipartite graph with $|V_1| = |V_2| = n$ and $|E| = m$. The *assignment polytope* (or *perfect matching polytope* of G), denoted by $\mathcal{A}_G$ (or just $\mathcal{A}$ when G is understood from context) is the polytope in $\Re^m$ defined by the assignment constraints in (1), and with $x_{ij} = 0$ for all $\{i, j\} \notin E$. As is well-known, every vertex (if one exists) of this polytope is integral.

A *matching* in G is a subset $M \subseteq E$ of pairwise disjoint edges. A matching is *perfect* if it includes every vertex. A solution (not necessarily integral) to the set of equations (1) is called a *fractional perfect matching* and a solution to a set of inequalities similar to (1) in which every equality sign is replaced by a "$\leq$"-sign is called a *fractional matching.* A (perfect) matching M is clearly also a fractional (perfect) matching.

For a given bipartite graph G, we consider a generalization of the decision formulation of the assignment problem in which we are looking for a perfect matching that satisfies $K = \text{poly}(n)$ additional linear constraints. We call this the *Assignment Problem with Extra Constraints*, or APEC. The problem is to find an integral x such that

$$\begin{array}{rcll} x & \in & \mathcal{A}_G & \\ a_k x & \geq & b_k, & \quad k \in [K]. \end{array} \tag{5}$$

(Thus the definition in (2) is for the case $G = K_{n,n}$.)

We show how to round a fractional solution to this linear program to a matching that is "close" to it as stated by the following theorem which we prove in Section 5. Recall that we denote the largest coefficient in a_k by A_k.

Theorem 1.3 *There is an algorithm that, given a fractional solution x^* to the system (5), produces* **whp** *a matching x that contains at least $n - o(n)$ edges and satisfies*

$$a_k x \geq (1 - o(1))b_k - \tilde{O}(\sqrt{n}A_k) \qquad k = 1, 2, \ldots, K. \tag{6}$$

The running time of the algorithm is $\tilde{O}(N)$, where N is the number of nonzero entries in x^*. The algorithm can be implemented in $O(\log n \log\log n)$ time on an EREW PRAM.

Remarks:(i) The polylog() factor hidden in the soft-O (i.e., $\tilde{O}()$) notation is somewhat larger in our result than in Raghavan-Thompson. (ii) As in the Raghavan-Thompson procedure, the error goes down when all coefficients are nonnegative. (LPs of this form are called *fractional packing and covering problems.*) In this case the algorithm guarantees that

$$a_k x \geq (1 - o(1))b_k - \tilde{O}(A_k) \qquad k = 1, 2, \ldots, K, \tag{7}$$

(The $-$'s are changed to $+$'s if the inequality involved a $\leq$ instead of a $\geq$.)
(ii) One can obtain similar results for an APEC problem where the constraints contain the matching polytope rather than the perfect matching polytope of G. I.e. when each equality sign in (1) is replaced by a "$\leq$" sign. The additive error will be as stated above but the matching will not necessarily contain $n - o(n)$ edges.

2 Rounding Procedure for APEC: Small coefficients

For ease of exposition, we first prove the subcase of Theorem 1.3 when the coefficients of the linear constraints are between $-c$ and c, for some constant c independent of n. Let us call such linear constraints *c-smooth.* The constraints encountered while proving Theorem 1.1 are of this type. Furthermore, the simple procedure described here motivates our more general procedure that is described in Section 5. The more general procedure works even when the value of the coefficients is allowed to grow with n.

Notation: If a, b are real numbers we let $[a \pm b]$ denote the interval $[a - b, a + b]$. For positive function $a(n)$ and $f(n)$, we denote by $[a(n) \pm O(f(n))]$ the interval $[a(n) \pm \beta f(n)]$ for some unspecified constant $\beta > 0$. If we have an event $\mathcal{E}_n$ of the form $Z_n \in [a(n) \pm O(f(n))]$, for some random variable Z_n, then we say that this occurs *with high probability* (abbreviated as **whp**) if for every $\alpha > 0$ we can choose the "hidden constant" β large enough so that $\mathbf{Pr}(\mathcal{E}_n) = 1 - O(n^{-\alpha})$. The letter α will be reserved for this exponent and is always assumed to be suitably large.

Let x^* be a fractional perfect matching for the graph $G = (V_1, V_2, E)$. We give a probabilistic procedure to produce a matching M of cardinality $n - o(n)$ in G. Furthermore, if w_{ij} is any c-smooth weight function, then **whp** the matching M satisfies $w(M) \in [wx^* \pm (\xi wx^* + \tilde{O}(n^{3/4}))]$. Here ξ denotes a decreasing function of n which satisfies $\xi = O(1/\log n)$. Now suppose that our APEC contains $K = O(n^\gamma)$ c-smooth constraints $a_1, \ldots, a_K$ and we want a matching M such that with probability $1 - O(n^{-\alpha})$, $a_i M \in [a_i x^* \pm (\xi a_i x^* + \tilde{O}(n^{3/4}))]$ for every $i \in [K]$ where x^* is a fractional matching satisfying the constraints. (Here we think of M as a vector where $M_{ij} = 1$ if the edge $(i, j) \in M$ and $M_{ij} = 0$ otherwise.) To find such matching we apply our rounding procedure looking for a matching that will satisfy any individual constraint with probability $1 - O(n^{-\alpha'})$ where $\alpha' = \alpha + \gamma$. So the constant β hidden by the big-O when we specify the intervals $[w^i x^* \pm (\xi w^i x^* + \tilde{O}(n^{3/4}))]$ is determined by $\alpha = \alpha' + \gamma$.

The procedure consists of two phases. The first, called *decomposition,* produces $\Delta = O(\log^3 n)$ matchings $M_1, M_2, \ldots, M_\Delta$ such that **whp**,

$$\frac{1}{\Delta}\sum_{i=1}^{\Delta} w(M_i) \in [wx^* \pm (\xi wx^* + \tilde{O}(n^{1/2}))]. \tag{8}$$

For convenience we shall assume below that Δ is a power of 2.

The second phase consists of applying a binary (probabilistic) operator $\odot$ (called *merge*) to the matchings. The merge operator is designed such that if we apply it to two matchings A and B we obtain a matching $A \odot B$ which is a subset of the union of A and B such that for any c-smooth function w, **whp**,

$$w(A \odot B) \in \left[\frac{w(A) + w(B)}{2} \pm \tilde{O}\left(n^{3/4}\right)\right]. \tag{9}$$

The second phase proceeds as follows. Partition $M_1, M_2, \ldots, M_\Delta$ into pairs, merge each pair so as to obtain $\Delta/2$ new matchings, and repeat this pairing and merging until we're left with a single matching denoted M. Equation (9) implies that **whp**,

$$w(M) \in \left[\frac{1}{\Delta}\sum_{i=1}^{\Delta} w(M_i) \pm \tilde{O}\left(\Delta n^{3/4}\right)\right].$$

Since $\Delta = O(\log^3 n)$, the error term is at most $\tilde{O}(n^{3/4})$. Now it follows from Equation (8) that **whp**

$$w(M) \in [wx^* \pm (\xi wx^* + \tilde{O}(n^{3/4}))].$$

Finally, let's estimate the cardinality of M. Let w^0 be the linear function that counts the number of edges of G in the matching. Since $w^0 x^*$ is n, we see that **whp**, the decomposition phase ensures that $w^0(M)$ is at least $n - o(n)$; in other words, M has at least $n - o(n)$ edges.

2.1 The decomposition phase

The decomposition step starts with the following OVERSAMPLING procedure.

OVERSAMPLING:

Given: Fractional perfect matching x^* on a bipartite graph $G = (V_1, V_2, E)$.
Procedure: Construct a multigraph $G^* = (V_1, V_2, E^*)$ as follows. For each edge $(i, j) \in E$, toss a biased coin $L = \Theta(\log^3 n)$ times, where the coin is biased to come up "Heads" with probability x^*_{ij}. Suppose the coin came up "Heads" ξ_{ij} times. Then put ξ_{ij} copies of the edge (i, j) in E^*.

Lemma 2.1 *With high probability, each vertex in G^* has degree in $[L \pm O((L \log n)^{1/2})]$.*

Proof. The degree of vertex i is $\sum_j \xi_{ij}$, and the expectation of this degree is

$$E\left[\sum_j \xi_{ij}\right] = \sum_j E[\xi_{ij}] = L \cdot \sum_j x^*_{ij} = L.$$

Now apply Lemma 7.4(b). ■

Let Δ be the maximum degree of a vertex in the multigraph G^*. By Lemma 2.1, $\Delta \in [L \pm O((L \log n)^{1/2})] = O(\log^3 n)$ and the degree of any vertex is not smaller than Δ by more then $\tilde{O}(L^{1/2})$. Therefore by adding $\tilde{O}(nL^{1/2})$ arbitrary edges to G^* we can make it a Δ-regular bipartite graph. Let $\hat{G}^*$ be the resulting Δ-regular bipartite multigraph. It is well-known that a Δ-regular bipartite multigraph can be decomposed into Δ perfect-matchings. (This follows for example from repeatedly applying Hall's Theorem, which guarantees the existence of a perfect matching, see for example [B98].) The decomposition procedure decomposes $\hat{G}^*$ into Δ disjoint perfect matchings, $\hat{M}_1 \cup \hat{M}_2 \cup \cdots \cup \hat{M}_\Delta$. After obtaining the Δ perfect matchings we ignore the edges that were added to make G^* regular. The decomposition phase then ends with Δ matchings $M_1 \cup M_2 \cup \cdots \cup M_\Delta$ such that their union is G^*. We bound the average weight of these Δ matchings in the following lemma.

Lemma 2.2 *With high probability,*

$$\frac{1}{\Delta}\sum_{k=1}^{\Delta} w(M_k) \in wx^* \pm (\xi wx^* + \tilde{O}(n^{1/2})).$$

Proof. Let Y^k_{ij} be a random variable taking the value w_{ij} if the result of the k-th coin toss in OVERSAMPLING for edge (i,j) is "Heads" and zero otherwise. By the definition of OVERSAMPLING, Y^k_{ij} takes the value w_{ij} with probability x^*_{ij}. Let $S = \sum_{i,j,k} Y^k_{ij} = \sum_{k=1}^{\Delta} w(\hat{M}_k)$, and note that $\mathbf{E}[S] = Lwx^*$. Since $\sum_i x^*_{ij} = 1$ it follows that $\hat{\mu} = E[\sum |Y^k_{ij}|] = \tilde{O}(n)$. Furthermore since w is c-smooth $|Y^k_{ij}| \le c$. By applying Lemma 7.2 to S to see that **whp** $S \in [Lwx^* \pm \tilde{O}(n^{1/2})]$. Dividing by Δ and using Lemma 2.1 to bound the ratio L/Δ, gives the lemma. ■

2.2 The merge operator

We use the merge operator in the second phase of our rounding procedure to carefully combine the Δ matchings produced by the decomposition phase into one matching with the desired properties. As we mentioned we perform this combination using a binary *merge* operator $\odot$ that takes two matchings A and B and merges them into one matching $A \odot B$. We merge the matchings in phases where in each phase we pair up the matchings produced by the previous phase and apply the merge operator to each pair thereby reducing the number of matchings by a factor of two. Our final output is the matching produces by the last phase. Here is the description of the merge operator.

MERGE:

Given: Two matchings A and B.
Procedure to construct $A \odot B$: Note that $A \cup B$ is a union of cycles and paths. By deleting $O(n^{1/2})$ edges if necessary, ensure that every cycle/path has length $O(\sqrt{n})$. Consolidate all paths/cycles into $\Theta(n^{1/2})$ *groups* each of size $O(n^{1/2})$.
Probabilistically construct a matching $A \odot B$ as follows: Within all the paths/cycles in a group, pick with equal probability either all the edges of A or all the edges of B. Furthermore, make this decision independently in different groups.

The following lemma estimates the weight of $A \odot B$. The proof technique will be used again in Section 5.

Lemma 2.3 *If weight function w is c-smooth, then with high probability,*

$$w(A \odot B) \in \frac{w(A) + w(B)}{2} \pm \tilde{O}(n^{3/4}).$$

Proof. Let $m = O(n^{1/2})$ be the number of groups of paths or cycles, and let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be the weights of the edges from A in each group. The absolute value of each α_i is $O(n^{1/2})$. Let Z_A be a random variable whose value is the contribution of A's edges to $w(A \odot B)$. The variable Z_A is the sum of m independent random variable one for each group of paths and cycles. The expectation of Z_A is $\sum_i \alpha_i/2 \in [w(A)/2 - O(n^{1/2}), w(A)/2]$. Hence by Lemma 7.2 we obtain that **whp**

$$|Z_A - w(A)/2| = \tilde{O}(n^{3/4}) \ .$$

The same argument shows that $|Z_B - w(B)/2| = \tilde{O}(n^{3/4})$, which together with Equation (2.2) implies the statement of the lemma. ∎

Here is a good place to contrast our rounding procedure with Raghavan-Thompson rounding. If the fractional matching were $(A + B)/2$ then RT-rounding would involve picking every edge of A and B independently with probability $1/2$. Hence the expected weight of the resulting graph is $(w(A) + w(B))/2$, but unfortunately, that graph is not even close to a matching. Our procedure also picks each edge of A and B with probability

1/2 —to be correct, it does this for all edges of A and B except for at most $O(\sqrt{n})$ edges that were deleted. Hence the expected weight of the resulting matching is very close to $(w(A) + w(B))/2$. The crucial difference is that the procedure's decisions for different edges are not independent; in fact they are *very* dependent. Nevertheless, the degree of independence is enough to allow us to get a good upperbound on the probability that $w(A \odot B)$ deviates "significantly" from its expectation.

We end this section with the following two remarks.

Remark: Our rounding procedure is randomized. We use randomization in the OVERSAMPLING procedure and in the merge operator. We can derandomized our algorithm in a standard way using the method of conditional probabilities. For further details about this method see [R88, AS92, RT98].

Remark: We can parallelize our algorithm and implement it on an EREW PRAM (Exclusive Read Exclusive Write Parallel Random Access Machine) model (See e.g. [SG93]). It is easy to implement OVERSAMPLING to run in polylogarithmic time using m processors. To decompose a regular bipartite multigraph into perfect matching we use an algorithm of Lev, Pippenger, and Valiant [LPV81]. This algorithm decomposes a Δ regular bipartite multigraph into Δ disjoint perfect matchings in $O(\log \Delta \log n)$ parallel time using $O(n)$ processors. Recall that in our procedure $\Delta = \text{poly}(\log(n))$.

We can also parallelize the merge operator to run in $O(\log n)$ parallel time using n processors. We do that using standard PRAM techniques (see e.g. [SG93]). We assign a processor to each edge of A and to each edge of B. We assume that A and B are represented such that a processor assigned to an edge e knows the processors assigned to adjacent edges. A processor assigned to an edge e then finds in logarithmic time, using a technique sometimes referred to as pointer jumping [SG93], the index of e on the path or cycle of $A \cup B$ containing it, and the length of this path or cycle. Then processors assigned to edges with indices that are multiples of $\sqrt{n}$ drop the corresponding edges. By assigning a processor to each component we can then group the small components into larger ones such that the size of each group is $O(\sqrt{n})$. Finally (in parallel) a processor flips a coin for each group and using a processor per edge in $A \cup B$ we can delete unselected edges.

3 Approximation Algorithm for degree-d arrangement

In this section we prove Theorem 1.1 by presenting an approximation algorithm for the degree-d arrangement problem. We start by proving the theorem for the case where $d = 2$. Then we extend the proof by induction for general d. The algorithm which we describe is randomized, we later indicate how to derandomize it while keeping its running time quasi-polynomial. The basic idea is to use "exhaustive sampling" as in [AKK95], except that here we always have to care about the assignment constraints. Let $p(x) = \sum_{ijkl} c_{ijkl} x_{ij} x_{kl}$ be the objective function, where each c_{ijkl} is an integer in $[-c, c]$. Let $\overline{x}$ be the perfect matching that maximizes the objective function. Let b_{ij} denote $\sum_{kl} c_{ijkl} \overline{x}_{kl}$. Then $\overline{x}$ is an integral solution to the linear program

$$\begin{array}{cc} \text{minimize} & \sum_{ij} b_{ij} x_{ij} \\ s.t. & x \in \mathcal{A} \\ & \sum_{kl} c_{ijkl} x_{kl} = b_{ij} \end{array}$$

Note that if we knew the b_{ij}'s we could obtain a fractional solution to this linear program and them use our APEC approximation to compute an approximation to the optimum assignment $\overline{x}$. The main idea will be to use the "exhaustive sampling" technique of [AKK95] to obtain reasonable guesses for the b_{ij}'s. Specifically, the procedure is as follows.

(i) Use random sampling to estimate b_{ij}'s within an additive error $\overline{\epsilon} n$, where $\overline{\epsilon}$ is a constant fraction of ϵ that we determine later. Pick a random sample S of $O(c^2 \log n / \overline{\epsilon}^2)$ vertices with replacement and estimate b_{ij} by the sum $b'_{ij} = \frac{n}{|S|} \sum_{k \in S, l} c_{ijkl} \overline{x}_{kl}$. Of course we do not know $\overline{x}$. Therefore we enumerate all $n^{O(c^2 \log n / \overline{\epsilon}^2)}$ positions in which the vertices of S can be placed in a permutation. One of these ways is also the way in which they are placed by $\overline{x}$; we restrict attention to that one. Of course, our algorithm, not knowing $\overline{x}$, must do the rest of the work that we describe next for each of the $n^{O(c^2 \log n / \overline{\epsilon}^2)}$ guesses. Finally we take the best result obtained for any of the guesses. This would give us a solution which is at least as good as the one produced for the correct guess.

(ii) We consider the optimization problem

$$\begin{array}{rcl} \text{minimize} & \sum_{ij} b'_{ij} x_{ij} & \\ s.t. & x \in \mathcal{A} & \\ b'_{ij} - \epsilon n \leq & \sum_{kl} c_{ijkl} x_{kl} & \leq b'_{ij} + \overline{\epsilon} n \end{array} \tag{10}$$

We solve this linear program and denote the value of the optimal solution by ζ. We replace the linear objective function by the linear constraint $\sum_{ij} b'_{ij} x_{ij} \leq \zeta$, thus obtaining an instance of APEC. We round our fractional solution to this APEC problem to an approximate integral solution using the algorithm of Section 2. The approximate integral solution is almost a matching. We arbitrarily complete it to a matching and take the arrangement which it defines as our approximate integral solution to the original degree d arrangement problem. More precisely, we solve an APEC problem as above for each possible guess of where the vertices in S are placed by the optimal solution. Among the solutions to the different APEC problems we take the one whose corresponding arrangement minimizes the objective of the arrangement problem.

We now prove that the algorithm above indeed produces a solution with the desired approximation guarantee. The following lemma establishes the accuracy of the estimates b'_{ij}.

Lemma 3.1 *The estimates b'_{ij}, computed using the placement of S in the optimal solution, are in the interval $[b_{ij} \pm \overline{\epsilon} n]$* **whp***.*

Proof. Fix i and j. Let X_m, $1 \leq m \leq |S|$ be a random variable taking the value c_{ijkl} if the mth sampled vertex is k, $\overline{x}_{kl} = 1$ (the position of vertex k in the optimal solution is l), and c_{ijkl} is nonnegative. Otherwise, X_m takes the value 0. Similarly, let Y_m, $1 \leq m \leq |S|$ be a random variable taking the value c_{ijkl} if the mth sampled vertex is k, $\overline{x}_{kl} = 1$ and c_{ijkl} is nonpositive. Otherwise, Y_m takes the value 0. Clearly, $\frac{n}{|S|} \sum_{m=1}^{|S|}(X_m + Y_m) = b'_{ij}$, and $E[\sum_{m=1}^{|S|}(X_m + Y_m)] = \frac{|S|}{n} b_{ij}$. The result now follows from Lemma 7.4(c) applied separately to the sum of the X's after dividing each by c and to the sum of the Y's after dividing each by a factor of $-c$. In each application of Lemma 7.4(c) we use $\lambda = \overline{\epsilon}|S|/2c$. The constant, say ρ, hidden by the big O when we defined the cardinality of S to be $O(c^2 \log n / \overline{\epsilon}^2)$ determines the error probability to be $\frac{1}{n^\rho}$. ■

Note that our algorithm tries all possible guesses of how S is placed by the optimal solution $\overline{x}$. However the guess we use to define b'_{ij} in Lemma 3.1 is the correct one. In other words b'_{ij} in Lemma 3.1 is a function of S; once S is picked the guess that we use to define b'_{ij} is the restriction of the optimal solution $\overline{x}$ to S.

Lemma 3.1 implies that **whp** $\overline{x}$ is a feasible solution to the linear program (10). Therefore the optimal (fractional) solution y to that linear program satisfies

$$\begin{aligned} \zeta = \sum_{ij} b'_{ij} y_{ij} &\leq \sum_{ij} b'_{ij} \overline{x}_{ij} \\ &\leq \sum_{ij} (\sum_{kl} c_{ijkl} \overline{x}_{kl}) \overline{x}_{ij} + \overline{\epsilon} n^2 \\ &= p(\overline{x}) + \overline{\epsilon} n^2 \ . \end{aligned} \tag{11}$$

We use our rounding algorithm from Section 2 to round y into an approximate matching that approximately solves (10). In order to do that we first make our APEC c-smooth by scaling the constraint corresponding to the objective function of (10) by n to make each coefficient a number between $-c$ and c. Our rounding procedure, applied to the scaled APEC, rounds y into a $\hat{z}$ that is a matching of cardinality $n - o(n)$ and satisfies the given constraints with an additive error of $o(n)$. We then extend $\hat{z}$ to $z \in \mathcal{A}$ arbitrarily by adding $o(n)$ edges to the matching to make it perfect. After scaling back the objective function we obtain that

$$\sum_{ij} b'_{ij} z_{ij} \leq \sum_{ij} b'_{ij} y_{ij} + o(n^2) \ . \tag{12}$$

Furthermore z also satisfies

$$\sum_{kl} c_{ijkl} z_{kl} \leq b'_{ij} + \overline{\epsilon} n + o(n) \ . \tag{13}$$

Combining inequalities (12) and (13) we obtain that

$$p(z) \leq \sum_{ij} b'_{ij} y_{ij} + \overline{\epsilon} n^2 + o(n^2). \tag{14}$$

Combining Equations (11) and (14) and taking $\overline{\epsilon} = \epsilon/3$ we obtain that **whp** $p(z) \leq p(\overline{x}) + \epsilon n^2$ for sufficiently large n.

This completes the proof for $d = 2$. Note that our proof still holds even if the degree d arrangement problem contains additional linear constraints that have to be satisfies up to an additive factor of $o(n)$. Specifically, consider the problem

$$\begin{aligned} \text{minimize} \quad & p(x) && && (15) \\ s.t. \quad x \quad & \in \quad \mathcal{A} && && \\ a_k x \quad & \geq \quad b_k, && k \in [K] && (16) \end{aligned}$$

where the *extra constraints* (16) need only be satisfied approximately, as in APEC. Our proof for $d = 2$ shows how to obtain a solution $z \in \mathcal{A}$ such that if $\overline{x}$ is the optimal solution then $p(z) \leq p(\overline{x}) + \epsilon n^2$ and furthermore z satisfies the constraints (16) within an additive factor of $o(n)$.

Remark: The algorithm which we described is randomized. It uses random bits to pick the multiset S of $O(c^2 \log n/\overline{\epsilon}^2)$ vertices. We can derandomize the algorithm by running the procedure specified above for every possible multiset S of the appropriate size, and returning the best matching. The running time of the resulting deterministic algorithm is larger than the running time of the randomized one by a factor of $n^{|S|}$, and therefore remains quasi-polynomial. This deteministic version has to use a derandomization of the APEC procedure described in Section 2. Similarly, one can derandomize the algorithm for degree-d arrangement problems when $d > 2$, which we describe next.

For $d > 2$ we use induction on d to prove that for a degree d arrangement problem we can obtain a solution z such that $p(z) \leq p(\overline{x}) + (d\overline{\epsilon})n^d$ in time $n^{O(c^2 d^2 \log n/\overline{\epsilon}^2)}$ **whp**. By setting $\overline{\epsilon} = \epsilon/d$, this implies that we can obtain a solution z where $p(z) \leq p(\overline{x}) + \epsilon n^d$ **whp** in $n^{O(c^2 d^4 \log n/\epsilon^2)}$ time. Our proof above establishes the base case for $d = 2$. Assume this claim holds for degree l arrangement problems where $l < d$.

A generic term in the objective of (15) looks like $c_{i_1 j_1 \cdots i_d j_d} x_{i_1 j_1} \cdots x_{i_d j_d}$. For fixed $i_2, j_2, \ldots,$ i_d, j_d we estimate using "exhaustive sampling"

$$b_{i_2 j_2 \cdots i_d j_d} = \sum_{ij} c_{i j i_2 j_2 \cdots i_d j_d} \overline{x}_{ij}$$

by

$$b'_{i_2j_2\cdots i_dj_d} = \frac{n}{|S|}\sum_{k\in S,l} c_{kli_2j_2\cdots i_dj_d}\overline{x}_{kl}$$

for a random sample S. Then we use these estimated values to define the following $(d-1)$-dimensional problem.

$$\begin{aligned}
\text{minimize} \quad & p'(x) = \sum_{i_2j_2\cdots i_dj_d} b'_{i_2j_2\cdots i_dj_d} x_{i_2j_2\cdots i_dj_d} && (17)\\
\text{s.t.} \quad & && \\
x &\in \mathcal{A} && \\
a_k x &\geq b_k && k\in[K]\\
\sum_{kl} c_{kli_2j_2\cdots i_dj_d}x_{kl} &\geq b'_{i_2j_2\cdots i_dj_d} - \overline{\epsilon}n && \forall i_2,j_2,\ldots i_d,j_d\\
\sum_{kl} c_{kli_2j_2\cdots i_dj_d}x_{kl} &\leq b'_{i_2j_2\cdots i_dj_d} + \overline{\epsilon}n && \forall i_2,j_2,\ldots i_d,j_d
\end{aligned}$$

As for the case $d = 2$ since we do not know where the vertices of S are placed by the optimal solution we try all $n^{|S|}$ possibilities. For each such placement we calculate estimates b' and formulate a degree $d-1$ dimensional problem as above. We then divide the objective $p'(x)$ of the degree $d-1$ problem by n to make it c-smooth and solve it recursively. Using our induction hypothesis for each degree $d-1$ problem we obtain an approximation that is within an additive factor of $(d-1)\overline{\epsilon}n^{(d-1)}$ from the optimal solution **whp**. The time it takes us to solve each degree $d-1$ problem is by induction $n^{O(c^2(d-1)^2\log n/\overline{\epsilon}^2)}$. Finally we pick the best among the solutions obtained for different guesses as the solution for our original degree d problem. (This is at least as good as the solution obtained with the correct guess.) It immediately follows that the running time of our algorithm is $n^{|S|} * n^{O(c^2(d-1)^2\log n/\overline{\epsilon}^2)}$.

Similarly to the case of $d = 2$, the size of our random sample is $O(c^2 d\log n/\overline{\epsilon}^2)$ vertices which we pick uniformly at random with replacement. Comparable to the case $d = 2$, we increased the size of S by a factor of d since the number of estimates $b'_{i_2j_2\cdots i_dj_d}$ which we want to be accurate **whp** simultaneously is $O(n^d)$. A lemma analogous to Lemma 3.1 shows that $b'_{i_2j_2\cdots i_dj_d}$ (calculated for the correct guess) is within an additive factor of $\overline{\epsilon}n$ of its true value $b_{i_2j_2\cdots i_dj_d}$ in the optimal solution. Thus the optimal solution $\overline{x}$ is a feasible solution to the degree $d-1$ arrangement problem corresponding to the right guess. Let y be the optimal solution to this degree $d-1$ arrangement problem. It follows from the feasibility of $\overline{x}$ **whp** that

$$p'(y) \leq p'(\overline{x}) \leq p(\overline{x}) + \overline{\epsilon}n^d \tag{18}$$

whp. By induction when we solve the degree $d-1$ arrangement problem (after scaling down the objective) we get a solution z such that **whp**

$$p'(z)/n \leq p(y)/n + (d-1)\overline{\epsilon}n^{d-1}\ . \tag{19}$$

Combining equations (18) and (19) we get that

$$p(z) \leq p(\overline{x}) + d\overline{\epsilon}n^d$$

whp as required.

4 Polynomial Time Approximation Schemes

Our additive approximation procedure for the degree-d arrangement problem runs in $n^{O(\log n)}$ time. This immediately gives a quasi-polynomial time approximation algorithm with additive error for many other problems that are special cases of the degree-d arrangement problem mentioned in Section 1.2. For some problems however we can obtain an approximation algorithm with a similar additive error that runs in polynomial time. As a consequence we obtain a polynomial time approximation scheme (PTAS) for dense instances of these problems. We reduce the running time by shrinking the space of possible placements of the random sample, S, on which we do an exhaustive search. Here is an outline of the rest of this section. Section 4.1 describes a PTAS for MINIMUM LINEAR ARRANGEMENT on dense graphs. Sections 4.2 and 4.3 describe similar PTASs for dense instances of MINIMUM CUT LINEAR ARRANGEMENT and MAXIMUM ACYCLIC SUBGRAPH, respectively. In section 4.4 we describe a PTAS for dense instances of BETWEENNESS.

The PTASs described in Sections 4.1-4.4 are randomized. They have to pick a random multiset of $O(\log n)$ vertices. Here we cannot derandomize these algorithm in polynomial time by trying all possible multisets as there are too many. In Section 4.5 we sketch how to derandomize the PTASs for MINIMUM LINEAR ARRANGEMENT, MINIMUM CUT LINEAR ARRANGEMENT and MAXIMUM ACYCLIC SUBGRAPH in polynomial time.

4.1 Minimum Linear Arrangement

In this section we describe a PTAS for MINIMUM LINEAR ARRANGEMENT on dense graphs. Let $G = (V, E)$ be a $2a$-dense graph. First we notice that the optimum value of the cost function is no smaller than $a^3n^3/8$. The reason is that at least an vertices have degree at least an. Regardless of how the graph is laid out along a line, the total length of the edges incident to any such vertex is at least $(an)^2/4$, which makes the total edge length at least $(an)^3/8$. Hence to obtain a layout of cost $(1 + \gamma)$ times the optimum, where $\gamma > 0$ is arbitrary, it suffices to find a layout whose cost is within an additive factor $\gamma(an)^3/8$ of the optimum. Since $\gamma a^3/4$ is just another constant, we will for convenience denote it by ϵ in the description below.

Let $t = c/\epsilon$ for some suitably large constant $c > 0$ and assume for simplicity that n is a multiple of t. Partition the interval $[1, n]$ into consecutive equal-sized intervals $I_1, \ldots, I_t$ each of size n/t.

Definition 2 *A* placement *is a mapping g from the set of vertices to the set of intervals $I_1, \ldots, I_t$. It is* proper *if it maps n/t vertices to each interval, that is, for every $1 \le j \le t$, $|\{i \in V \mid g(i) = j\}| = n/t$. The* cost *$cp(g)$ of the placement is*

$$cp(g) = \sum_{(i,j) \in E, g(j) > g(i)} (g(j) - g(i)).$$

Note that every permutation (i.e., bijection on vertices) induces a proper placement in the obvious way. Two different permutations can induce the same placement, in which case we think of them as "only differing locally." The following lemma shows that the cost of a permutation is essentially decided (up to an additive factor n^3/t) by the placement it induces.

Lemma 4.1 *If π is a permutation and g is its induced placement then,*

$$|cp(g) - c(\pi)t/n| \le n^2/2.$$

Proof. An edge that contributes x to $c(\pi)$ contributes either $\lfloor xt/n \rfloor$ or $\lfloor xt/n \rfloor + 1$ to $cp(g)$. Therefore, the absolute value of the difference between $cp(g)$ and $c(\pi)t/n$ can be at most the number of edges, that is $\binom{n}{2}$. ■

Let π^* be an optimal linear arrangement, and let g^* be its induced placement. Let g be a placement such that $cp(g) \le cp(g^*) + \epsilon' n^2$, where ϵ' is determined below. Also let π be an arbitrary permutation such that g is the placement induced by π. By Lemma 4.1 we obtain that

$$\begin{aligned} c(\pi) &\le \frac{n * cp(g)}{t} + \frac{n^3}{2t} \\ &\le \frac{n * cp(g^*)}{t} + \frac{(1+\epsilon')n^3}{2t} \\ &\le c(\pi^*) + \frac{(2+\epsilon')n^3}{2t}. \end{aligned}$$

It follows that for $\epsilon' = 2c - 2$, $c(\pi) \le c(\pi^*) + \epsilon n^3$. Therefore to find a linear arrangement close to optimal up to an additive factor of ϵn^3 it suffices to find a placement close to the optimal placement up to an additive error of $\epsilon' n^2$. In the following we show how to find such a placement. To simplify the notation for the rest of this section we use $\epsilon' = \epsilon$.

Let g^* be an optimal placement. We note that the contribution of node i to the cost $cp(g^*)$ is

$$\sum_{(i,j)\in E, g^*(j)>g^*(i)} (g^*(j) - g^*(i)) \tag{20}$$

As in our previous algorithms, we will use exhaustive sampling to estimate terms of the form (20) for each vertex i. For this we randomly pick with replacement a multi-set S of $O(\log n/\delta^2)$ vertices where $\delta = \overline{\epsilon}/t$ and $\overline{\epsilon}$ is a sufficiently small fraction of ϵ which we will determine later. We enumerate all possible functions $h : S \to \{1, \ldots, t\}$ that assign vertices in S to intervals. For each such function we solve a linear program M_h described below and use the (fractional) solution to construct a placement. Among all those placements we pick up one with minimum cost. When our function h is the same as h^*, the restriction of an optimal placement g^* to S, the placement g will satisfy $cp(g) \leq cp(g^*) + \epsilon n^2$ **whp** over the random choices of S.

Now we describe the linear program M_h. For each vertex i and interval I_k, we compute an estimate e_{ik} of the cost of assigning i to I_k in any complete assignment g whose restriction to S is h (see the connection to expression (20)):

$$e_{ik} = \frac{n}{|S|} \sum_{(i,j)\in E, j\in S, h(j)>k} (h(j) - k)$$

Note that this estimate is well defined no matter whether $i \in S$, and no matter what is the value of $h(i)$ in case $i \in S$. Note also that if $(i, j) \in E$, $j \in S$, and $h(j) > k$ then $h(j) - k$ appears as many times in the above sum as j appears in S.

Using the estimates e_{ik}, where $1 \leq i \leq n$, and $1 \leq k \leq t$, we write the following linear program, M_h, which seeks to find the optimum fractional placement in which the cost of assigning vertex i to interval k is $[e_{ik} \pm \overline{\epsilon} n]$.

$$\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i=1}^{n}\sum_{k=1}^{t} e_{ik}x_{ik} \\
\text{s.t.} &
\end{array}$$

$$\begin{array}{rcll}
\displaystyle\sum_{i=1}^{n} x_{ik} & = & n/t & \forall k \in [t] \\
\displaystyle\sum_{k=1}^{t} x_{ik} & = & 1 & \forall i \in [n] \\
\displaystyle\sum_{(i,l)\in E}\sum_{j=k+1}^{t} (j-k)x_{lj} & \leq & e_{ik} + \overline{\epsilon} n & \forall i \in [n], k \in [t] \\
\displaystyle\sum_{(i,l)\in E}\sum_{j=k+1}^{t} (j-k)x_{lj} & \geq & e_{ik} - \overline{\epsilon} n & \forall i \in [n], k \in [t] \\
0 \leq x_{ik} & \leq & 1 & \forall i \in [n], k \in [t]
\end{array}$$

We solve M_h for every possible assignment h of the vertices in S to intervals. Let x^h be the optimal solution for M_h. We round x^h_{ik} using randomized rounding to obtain a placement r'_h as follows. For each vertex i independently we make $r'_h(i) = k$ with probability x^h_{ik}. After the rounding the placement r'_h need not be proper. We construct a proper placement r_h from r'_h by moving vertices from intervals with more than n/t vertices assigned to them to intervals with less than n/t vertices assigned to them arbitrarily. The final answer of our algorithm is the placement r with smallest cost among all placements r_h for every possible h. Let g^* be the optimum placement. We will now show that $cp(r) \leq cp(g^*) + \epsilon n^2$.

Our first lemma specifies the accuracy of the estimate e_{ik}.

Lemma 4.2 *Let g be a placement. Pick uniformly at random with replacement a multiset S of $O(\log n/\delta^2)$ vertices. Let $h : S \to \{1, \ldots, t\}$ be the restriction of g to S. Then with high probability (over the choice of the sample S),*

$$e_{ik} \in \left[\sum_{(i,j)\in E, g(j)>k} (g(j) - k) \pm \delta n t \right].$$

Proof. Let X_l be a random variable that equals $(g(j) - k)$ if the lth vertex sampled is j, $(i, j) \in E$, and $g(j) > k$. Otherwise, the value of $X_l = 0$. Note that $\sum_l X_l = \frac{|S|}{n} e_{ik}$ and $\mathbf{E}[\sum_l X_l] = \frac{|S|}{n} \sum_{(i,j)\in E, g(j)>k} (g(j) - k)$. Divide each X_l by t to scale it to the interval $[0, 1]$ and apply Lemma 7.4(c) to the sum of $X_1, \ldots, X_{|S|}$ after scaling. When applying Lemma 7.4(c), use $\lambda = \delta|S|$, and recall that $|S| = O(\log n/\delta^2)$. We obtain that the statement holds **whp**. The constant hidden behind the big-O when we pick S of size $O(\log n/\delta^2)$ determines the error probability. ■

Let h^* be the restriction of the optimal placement g^* to the sample S. It follows from Lemma 4.2 that **whp** g^* is a feasible solution to M_{h^*}. Let $v(g^*)$ be the value of the objective function of M_{h^*} at g^*. Let x^* be a fractional optimal solution to M_{h^*}, and let $\phi(h^*)$ the value of the objective function at x^*. Since g^* is a feasible solution of M_{h^*} we obtain that

$$\phi(h^*) \leq v(g^*) \leq cp(g^*) + \bar{\epsilon} n^2 , \tag{21}$$

where the second inequality is obtained by substituting the upper bounds on the estimates e_{ik} that follow from the constraints of M_{h^*} that g^* satisfies. Let r^* be the placement which we constructed by rounding the solution to M_{h^*}. We next show that $cp(r^*) \leq \phi(h^*) + \bar{\epsilon} n^2 + o(n^2)$.

Lemma 4.3 *Let r^* be the placement constructed from the optimal fractional solution x^* of M_{h^*}, where h^* is the restriction of an optimal placement g^* to S. Then $cp(r^*) \leq \phi(h^*) + \bar{\epsilon} n^2 + o(n^2)$ where $\phi(h^*)$ is the value of the objective function of M_{h^*} at x^*.*

Proof. Consider first the placement r' obtained after randomized rounding of x^* which need not be proper. Think of this placement as a vector x' such that $x'_{ik} = 1$ if and only if $r'(i) = k$. From the definition of randomized rounding we obtain that $\mathbf{E}[\sum_i \sum_k e_{ik} x'_{ik}] = \phi(h^*)$. Let X_{ik} be the random variable taking the value e_{ik} if $x'_{ik} = 1$ and 0 otherwise. By applying Lemma 7.2 to the sum of the variables X_{ik} we obtain that **whp**

$$\sum_i \sum_k e_{ik} x'_{ik} \in [\phi(h^*) \pm \tilde{O}(n^{3/2})] \ . \tag{22}$$

(When applying Lemma 7.2 note that $e_{ik} \leq nt$ and note that $E(\sum_{i=1}^n \sum_{k=1}^t X_{ik}) \leq (nt)^2$.)

Next we consider the question of how far $\sum_i \sum_k e_{ik} x'_{ik}$ can be from the cost of the placement r'. By definition, the cost of r' is $\sum_i \sum_k f_{ik} x'_{ik}$ where $f_{ik} = \sum_{(i,l)\in E} \sum_{j=k+1}^t (j-k)x'_{lj}$. Let $f^*_{ik} = \sum_{(i,l)\in E} \sum_{j=k+1}^t (j-k)x^*_{lj}$. Since x^* is a feasible solution to M_{h^*} we know that $f^*_{ik} \leq e_{ik} + \overline{\epsilon} n$. For l such that $(i,l) \in E$, and $j > k$, let X_{lj} be the random variable taking the value $(j-k)$ if $x'_{lj} = 1$ and 0 otherwise. By applying Lemma 7.4(c) to the sum of the random variables $Z_{lj} = X_{lj}/t$ we obtain that $f'_{ik} \in [f^*_{ik} + \tilde{O}(\sqrt{n})]$. Combining this with the upper bound on f^*_{ik} we obtain that

$$f'_{ik} \leq e_{ik} + \overline{\epsilon} n + \tilde{O}(\sqrt{n}) \ . \tag{23}$$

Combining Equations (22) and (23) we obtain that $cp(r') \leq \phi(h^*) + \overline{\epsilon} n^2 + \tilde{O}(n^{3/2})$.

To establish the result for the placement r^* obtained by making r' proper we note that for each k, $|\{i : r'(i) \in I_k\}| \in \left[\frac{n}{t} \pm \tilde{O}(n^{1/2})\right]$ **whp**. (One can easily see that by applying Lemma 7.4(c) to the sum of X_{ik} for $1 \leq i \leq n$ and a fixed k). Thus to obtain r from r' we move at most $\tilde{O}(n^{3/2})$ vertices. This can change the cost of the placement by no more than $\tilde{O}(n^{3/2})$. ■

By choosing say $\overline{\epsilon} = \epsilon/3$. We obtain from Equation 21 and Lemma 4.3 that

$$cp(r^*) \leq cp(g^*) + \epsilon n^2 \ .$$

for sufficiently large n. Since r^* is a candidate for our chosen placement r, and we choose the placement with minimum cost, the cost of r is no larger than the cost of r^* and we obtain the desired result.

d-DIMENSIONAL ARRANGEMENT is handled similarly. One defines a partition of the grid into t^d sub-cubes and only define permutations up to placement of vertices into a sub-cube. The argument is very similar to the case $d = 1$ above and is left to the reader.

4.2 Minimum Cut Linear Arrangement

The approach is similar to the one for MINIMUM LINEAR ARRANGEMENT. First we notice that if a graph is a-dense, then the value of the minimum cut in the best arrangement is $\Omega(n^2)$. To see that notice that if the average degree of G is an then G has a subgraph G' in which the minimum degree is an. (We can obtain G' from G by keep eliminating vertices of degree below the average degree.) In any arrangement if we look at the point where $an/2$ of the vertices of G' are on one side of it and the rest of the vertices of G' are on the other side of it then there must be $\Omega(n^2)$ edges crossing that point. Since the value of the optimal solution is $\Omega(n^2)$ to obtain a layout of cost no greater than $(1+\gamma)$ times the optimum it suffices to obtain a solution whose cost is within an additive factor of ϵn^2 for a suitable ϵ. In the rest of this section we show how to obtain such layout.

We partition the interval $[1, n]$ into $t = \frac{c}{\epsilon}$ equal-size intervals $I_1, \ldots, I_t$ for sufficiently large constant c. We define a *placement* and a *proper placement* as in Section 4.1. The cost of a placement g, denoted by $cp(g)$, is defined as

$$cp(g) = \max_{1 \leq i < t} |\{(k, l) \in E | g(k) \leq i < g(l)\}|.$$

The following lemma shows that the cost of an arrangement is determined up to an additive factor of ϵn^2 by the cost of its induced placement.

Lemma 4.4 *Let π be a permutation and g be its induced placement. Then $cp(g) \leq c(\pi) \leq cp(g) + n^2/t$.*

Proof. The lower bound follows from the fact that the cuts considered when we calculate the cost of g are a subset of the cuts considered when we calculate the cost of π. Next we prove the upper bound. Place the vertices according to π and let I be an interval in our partition. The difference between the number of edges crossing a cut defined by a point inside I and the number of edges crossing a cut defined by the point between I and one of its neighboring intervals, is bounded by the maximum possible number of edges between vertices of I and all other vertices. This number is at most n^2/t so the lemma follows. ■

The conclusion from Lemma 4.4 is that it suffices to find a placement g such that $cp(g) \leq cp(g^*) + \epsilon' n^2$, where g^* is the optimal placement, and $\epsilon' = c - 1$. Any arrangement π such that g is its induced placement would then be within an additive factor of ϵn^2 from optimal since by Lemma 4.4

$$\begin{aligned} c(\pi) &\leq cp(g) + \frac{n^2}{t} \\ &\leq cp(g^*) + \frac{(1+\epsilon')n^2}{t} \\ &\leq cp(\pi^*) + \frac{(1+\epsilon')n^2}{t} \end{aligned}$$

where π^* is a min-cut linear arrangement. To simplify our notation we use $\epsilon' = \epsilon$ in the followings.

We find such a placement g in a way analogous to Section 4.1. We sample a (multi)-set S of $O(\log n/\overline{\epsilon}^2)$ vertices where $\overline{\epsilon}$ is a sufficiently small fraction of ϵ which we will determine later. We enumerate all possible functions $h : S \to \{1, \ldots, t\}$ that assign vertices in S to intervals. For each such function we solve a linear program M_h described below and use the (fractional) solution to construct a placement. Among all those placements we pick up one with minimum cost. When our function h is the same as h^*, the restriction of an optimal placement g^* to S, the placement g will satisfy $cp(g) \leq cp(g^*) + \epsilon n^2$ **whp** over the random choices of S.

Now we describe the linear program M_h. For each vertex i and interval I_k, we compute an estimate e_{ik} of the contribution of vertex i to a cut right between intervals I_{k-1} and I_k assuming i is placed in one of the intervals $I_1, \ldots, I_{k-1}$. The estimates e_{ik} are defined now as

$$e_{ik} = \frac{n}{|S|} \left|\{j \in S | (i,j) \in E \text{ and } h(j) \geq k\}\right|.$$

Note that $j \in S$ is counted here as many times as it occurs in S and that this estimate is defined nomatter whether i is in the sample S or not. The linear program M_h that corresponds to a guess h is now the following.

$$\begin{array}{rcll}
\text{minimize} & z & & \\
\text{s.t.} & & & \\
\sum_{i=1}^{n} x_{ik} & = & n/t & \forall k \in [t] \\
\sum_{k=1}^{t} x_{ik} & = & 1 & \forall i \in [n] \\
\sum_{i=1}^{n} e_{ik}(x_{i1} + \ldots x_{i(k-1)}) & \leq & z & \forall k \in [t], k > 1 \\
\left| \sum_{(i,s) \in E, l \geq k} x_{sl} - e_{ik} \right| & \leq & \overline{\epsilon} n & \forall i \in [n], k \in [t] \\
0 \leq & x_{ik} & \leq 1 & \forall i \in [n], k \in [t]
\end{array}$$

Similarly to the algorithm for MINIMUM LINEAR ARRANGEMENT, randomized rounding plus reallocation is applied to the solutions of the M_h's to obtain proper placements. The final answer of our algorithm is the placement r with smallest cost among all placements r_h for every possible h. The proof that r is indeed within an additive factor of ϵn^2 from the optimal placement is similar to the proof for MINIMUM LINEAR ARRANGEMENT in Section 4.1 so we only sketch it below.

Analogously to Lemma 4.2 we have the following lemma that specifies the accuracy of the estimates e_{ik}. We omit the proof which similar to the proof of Lemma 4.2.

Lemma 4.5 *Let g be a placement. Pick uniformly at random with replacement a multiset S of $O(\log n/\overline{\epsilon}^2)$ vertices. Let $h : S \to \{1, \ldots, t\}$ be the restriction of g to S. Then with high probability (over the choice of the sample S).*

$$e_{ik} \in [|\{j \in V \mid (i,j) \in E \text{ and } g(j) \geq k\}| \pm \overline{\epsilon}n] \ .$$

Let $\phi(h^*)$ be the value of the objective function of M_{h^*} at the optimal solution x^*. By Lemma 4.5, g^* is a feasible solution to M_{h^*} **whp**. Let z_{g^*} be the value of the objective function of M_{h^*} at g^*. Clearly we have that $\phi(h^*) \leq z_{g^*}$ **whp**. The constraints of M_{h*} guarantee that the actual contribution of placing i in one of $I_1, \ldots, I_{k-1}$ differ from e_{ik} by at most $\overline{\epsilon}n$. Therefore it follows that $z_{g^*} \leq cp(g^*) + \overline{\epsilon}n^2$ and so

$$\Phi(h^*) \leq cp(g^*) + \overline{\epsilon}n^2 \ . \tag{24}$$

We can also prove that Lemma 4.3 holds with definitions of M_h and cp of this Section. The proof is analogous to the proof of Lemma 4.3 in Section 4.1. To summarize, by choosing say $\overline{\epsilon} = \epsilon/3$ we obtain from Equation (24) and Lemma 4.3 that

$$cp(r^*) \leq cp(g^*) + \epsilon n^2 \ .$$

for sufficiently large n. Since r^* is a candidate for our chosen placement r, and we choose a placement of minimum cost, the cost of r is no larger than the cost of r^* and we obtain the desired result.

4.3 Maximum Acyclic Subgraph

Recall that in the MAXIMUM ACYCLIC SUBGRAPH problem we are given a directed graph G and we look for an acyclic subgraph with maximum number of arcs. We denote an arc directed from v to w by (v, w). First we rephrase the problem as follows. Given a directed graph G and a permutation π of the vertices of G we define the cost, $c(\pi)$, of the permutation, as the number of arcs (v, w) such that $\pi(v) < \pi(w)$. The problem of finding a permutation of maximum cost is exactly the MAXIMUM ACYCLIC SUBGRAPH problem. If G is a-dense, i.e. the average degree (counting both ingoing and outgoing arcs) of a vertex is an, then the size of the maximum acyclic subgraph is $\Omega(n^2)$. To see that notice that the graph contains $\Omega(n^2)$ arcs and if we take an arbitrary permutation π of the vertices then either π or its reversal define an acyclic subgraph with at least half of the arcs in G. Therefore to obtain an acyclic subgraph with at least $(1-\gamma)$ of the arcs it is enough to find a subgraph whose number of arcs is within an additive factor of ϵn^2 from the number of arcs in the largest subgraph for a suitable ϵ.

We use the same technique for finding such subgraph as we used to solve the arrangement problems in Section 4.1 and 4.2. We partition the interval $[1, n]$ into $t = \frac{c}{\epsilon}$ equal-size intervals $I_1, \ldots, I_t$ for sufficiently large constant c. We define a *placement* and a *proper placement* as in Section 4.1. Here the cost of a placement g, denoted by $cp(g)$, is defined as

$$cp(g) = |\{(k,l) \in E | g(k) < g(l)\}| \ .$$

The following Lemma shows that the size of the acyclic subgraph defined by an arrangement is determined up to an additive factor of ϵn^2 by the cost of its induced placement.

Lemma 4.6 *Let π be a permutation and g be its induced placement. Then $cp(g) \leq c(\pi) \leq cp(g) + n^2/t$.*

Proof. The lower bound follows from the fact that the arcs going from an interval to an interval to its right in the placement define an acyclic subgraph. Next we prove the upper bound. Consider the maximum acyclic subgraph defined by π. It is a supergraph of the maximum acyclic subgraph defined by g. The arcs which are in the subgraph defined by π and not in the subgraph defined by g are those arcs directed rightward with both endpoints within one of the intervals $I_1, \ldots, I_t$. Any interval I_i contains at most $(n/t)^2$ such arcs. Summing up over all t intervals we obtain that the total number of such arcs is n^2/t from which the upper bound follows. ∎

So Lemma 4.6 reduces our problem to the problem of finding a placement within an additive factor of ϵn^2 from the optimal placement. As in Sections 4.1 and 4.2 we sample a (multi)-set S of $O(\log n/\overline{\epsilon}^2)$ where $\overline{\epsilon}$ is a sufficiently small fraction of ϵ. We enumerate all possible functions $h : S \rightarrow \{1, \ldots, t\}$ that assign vertices in S to intervals. For each such function we solve a linear program M_h described below and use the (fractional) solution to construct a placement. Among all those placements we pick up one with minimum cost. When our function h is the same as h^*, the restriction of an optimal placement g^* to S, the placement g will satisfy $cp(g) \leq cp(g^*) + \epsilon n^2$ **whp** over the random choices of S. For each vertex i and interval I_k, we compute an estimate e_{ik} on the number of arcs outgoing from vertex i in a maximum acyclic subgraph defined by a placement extending h assuming vertex i is placed in I_k. The estimates e_{ik} are defined as follows

$$e_{ik} = \frac{n}{|S|} |\{j \in S | (i,j) \in E \text{ and } h(j) > k\}| \ .$$

The linear program M_h that corresponds to a guess h is now the following.

$$\begin{array}{lrcll} \text{maximize} & \sum e_{ik}x_{ik} & & & \\ \text{s.t.} & & & & \\ & \sum_{i=1}^{n} x_{ik} & = & n/t & \forall k \in [t] \\ & \sum_{k=1}^{t} x_{ik} & = & 1 & \forall i \in [n] \\ & \left| \sum_{(i,s)\in E, l>k} x_{sl} - e_{ik} \right| & \leq & \bar{\epsilon} n & \forall i \in [n], k \in [t] \\ & 0 \leq & x_{ik} & \leq 1 & \forall i \in [n], k \in [t] \end{array}$$

Similarly to the algorithms in Section 4.1 and 4.2 we use randomized rounding plus reallocation to obtain a proper placement from the solution of M_h. The final answer of our algorithm is the placement r with smallest cost among all placements r_h for every possible h. The proof that r is indeed within an additive factor of ϵn^2 from the optimal placement is analogous to the proofs in Sections 4.1 and 4.2.

4.4 Betweenness

Recall that the set of triples is denoted by $\mathcal{C}$ and the triples contain elements from a set A, where $|A| = n$ (we call these elements vertices). We also recall that we define an instance of BETWEENNESS as dense if the optimal solution satisfies $\Omega(n^3)$ triples. To simplify the presentation we also assume that the set of triples in $\mathcal{C}$ does not contain both a triple and its reversal. So when we write a statement like $(a, b, c) \in \mathcal{C}$ we in fact mean that either $(a, b, c) \in \mathcal{C}$ or $(c, b, a) \in \mathcal{C}$. We will assign vertices into one of $t = c/\epsilon$ consecutive intervals; such an assignment is called a *placement* as in Section 4.1. A triple (a, b, c) is feasible in placement g if either $g(a) > g(b) > g(c)$ or $g(c) > g(b) > g(a)$. Here, $c(\pi)$ denotes the number of *feasible* triples under permutation π and $cp(g)$ denote the number of feasible triples under a placement g. Now we observe that to obtain an additive approximation, it suffices to find a good placement instead of a permutation.

Lemma 4.7 *Let π be a permutation and g be its induced placement. Then $cp(g) \leq c(\pi) \leq cp(g) + n^3/t$.*

Proof. A triple that is feasible in π is feasible in g unless two of its vertices are in the same interval. The number of triples with at least two vertices in the same interval is at most

$$t \times \frac{n}{t}\left(\frac{n}{t} - 1\right) \times (n - 2) \leq \frac{n^3}{t}.$$

This is because there are t possible intervals for the colliding pair, $\frac{n}{t}(\frac{n}{t}-1)$ ordered pairs in each interval and at most $n-2$ possibilities for the third vertex to determine the triple. ■

The algorithm for finding a good placement will choose a random (multi)-set S where $|S| = O(\log n/\bar{\epsilon}^2)$ and $\bar{\epsilon}$ is a fraction of ϵ that we specify later. We enumerate all possible functions $h : S \to [1, t]$ that assign vertices in S to intervals. For each function $h : S \to [1, t]$ we solve a linear program M_h defined below. We round the fractional solutions to placements and take the best placement as our solution. For each pair of vertices a, b and pair of intervals $k, l \in \{1, \ldots, t\}$ we define the following estimate. This estimate estimates the contribution of triples of the form $(a, b, *) \in \mathcal{C}$ to the cost of a placement g extending h assuming vertex a is placed in I_k and vertex b is placed in I_l. The definition and the notation of the estimate depend on whether the interval I_k is to the left of I_l or to the right of I_l.

$$\begin{array}{rrcl} \text{If } k < l \text{ then} & e^{<}_{ak;bl} & = & \dfrac{n}{|S|}|\{c \in S : \ (a, b, c) \in \mathcal{C} \ \text{ and } l < h(c)\}| \\ \text{else} & e^{>}_{ak;bl} & = & \dfrac{n}{|S|}|\{c \in S : \ (a, b, c) \in \mathcal{C} \ \text{ and } l > h(c)\}| \end{array}$$

Furthermore, for each $a \in A$ and an interval $k \in [t]$ we estimate the contribution of triples of the form $(a, *, *)$ to the cost of a placement extending h, assuming element a is placed in interval k. This estimate is defined as follows.

$$e_{ak} \ = \ \frac{n^2}{|S|^2}|\{b, c \in S : \ (a, b, c) \in \mathcal{C} \text{ and } ([k < h(b) < h(c)] \text{ or } [k > h(b) > h(c)])\}|.$$

The linear program M_h is defined using these estimates as follows.

$$
\begin{array}{lrcll}
\text{maximize} & \sum_{a\in A}\sum_{k=1}^{t} e_{ak}x_{ak} & & & \\
\text{s.t.} & & & & \\
& \sum_{a\in A} x_{ak} & = & n/t & \forall k\in[t] \\
& \sum_{k=1}^{t} x_{ak} & = & 1 & \forall a\in A \\
& \left|\left(\sum_{\substack{b\in A\\ k<l}} e^{<}_{ak;bl}x_{bl} + \sum_{\substack{b\in A\\ k>l}} e^{>}_{ak;bl}x_{bl}\right) - e_{ak}\right| & \leq & 2\bar\epsilon n^2 & \forall a\in A, k\in[t] \\
& \left|\sum_{\substack{(a,b,c)\in\mathcal{C}\\ l<m}} x_{cm} - e^{<}_{ak;bl}\right| & \leq & \bar\epsilon n & \forall a,b\in A, k<l\in[t] \\
& \left|\sum_{\substack{(a,b,c)\in\mathcal{C}\\ l>m}} x_{cm} - e^{>}_{ak;bl}\right| & \leq & \bar\epsilon n & \forall a,b\in A, k>l\in[t] \\
& 0\leq & x_{ik} & \leq 1 & \forall i\in[n], k\in[t]
\end{array}
$$

Similarly to the algorithms from Sections 4.1, 4.2, and 4.3, randomized rounding plus reallocation is applied to the solution of each M_h to obtain a placement. Our final solution is the placement with maximum cost among these placements.

We analyze this algorithm in a way similar to the analysis of Sections 4.1, 4.2, and 4.3. The following lemma specify the accuracy of the estimates. Its proof is slightly more complicated than the proof of the corresponding lemmas in previous section and use the Azuma-Hoeffding tail inequality for analyzing the estimate e_{ak}.

Lemma 4.8 *Let g be a placement. Pick uniformly at random with replacement a multiset S of $O(\log n/\bar\epsilon^2)$ elements from A. Let $h : S \to [1,t]$ be the restriction of g to S. Then* **whp**, *for each pair a, b of elements from A and for each pair of intervals $k, l \in [t]$,*

$$
\begin{array}{llll}
(a)\ e^{<}_{ak;bl} & \in & [|\{c\in A:\ (a,b,c)\in\mathcal{C}, l<g(c)\}| \pm \bar\epsilon n] & \textit{when } k<l \\
(b)\ e^{>}_{ak;bl} & \in & [|\{c\in A:\ (a,b,c)\in\mathcal{C}, l>g(c)\}| \pm \bar\epsilon n] & \textit{when } k>l
\end{array}
$$

Furthermore, for each $a \in A$ and $k \in [t]$,

$$(c)\ e_{ak} \ \in\ [|\{b,c\in A:\ (a,b,c)\in\mathcal{C}, \textit{and } k<g(b)<g(c) \textit{ or } k>g(b)>g(c)\}| \pm \bar\epsilon n^2].$$

Proof. Parts (a) and (b) are proved using using Lemma 7.4(c) as in the proof of Lemma 4.2. To prove (c) we use the Azuma-Hoeffding martingale tail inequality – see for example Chapter 7 of Alon and Spencer [AS92] (Theorem 4.2). The random variable $Z = \frac{|S|^2}{n^2} e_{ak}$ depends on $|S|$ independent random variables (the elements of S in their chosen order). Changing the value of one of these variables changes Z by at most $|S|$. So for any $t > 0$

$$\mathbf{Pr}(|Z - \mathbf{E}(Z)| \geq t) \leq 2e^{-2t^2/|S|^3}. \tag{25}$$

Now

$$\mathbf{E}(Z) = |\{(a,b,c) \in \mathcal{C} : \ [k < g(b) < g(c)] \text{ or } [k > g(b) > g(c)]\}|$$

and putting $t = \overline{\epsilon}|S|^2$ into (25) yields the lemma. ■

The rest of the proof is analogous to the proof in Sections 4.1, 4.2, and 4.3. We focus on the guess h^* which corresponds to the optimal placement g^*, and the estimates $e^<_{ak;bl}$, $e^>_{ak;bl}$, and e_{ak} that correspond to h^*. By Lemma 4.8, g^* is a feasible solution to the linear program M_{h^*} **whp**. Let $v(g^*)$ be the value of the objective function of M_{h^*} at g^*. Let x^* be the optimal fractional solution to M_{h^*}, and let $\phi(h^*)$ be the value of the objective function of M_{h^*} at x^*. Clearly we have that $\phi(h^*) \geq v(g^*)$. As a feasible solution to M_{h^*}, g^* satisfies the constraints of M_{h^*}. By substituting the constraints that provide lower bounds on the estimates, into the objective we obtain that

$$\phi(h^*) \geq cp(g^*) - 3\overline{\epsilon}n^3 \ . \tag{26}$$

We also have to bound how far from $\phi(h^*)$ can be the cost of the placement r^*, obtained from x^* by randomized rounding and reallocation of vertices. The following lemma, that is analogous to Lemma 4.3, bounds the error resulting from randomized rounding and reallocation of vertices.

Lemma 4.9 *Let r^* be the placement constructed from the optimal fractional solution x^* of M_{h^*}. Then $cp(r^*) \geq \phi(h^*) - 3\overline{\epsilon}n^3 - o(n^3)$ where $\phi(h^*)$ is the value of the objective function of M_{h^*} at x^*.*

Proof. The proof of this lemma is analogous to the proof of Lemma 4.3 so we just sketch it briefly. By scaling and applying Lemma 7.4(c) we obtain that the placement r^* satisfies the third set of constraints in M_{h^*} with an additive error of $o(n^2)$, the forth and the fifth sets with an additive error of $o(n)$, and its objective value is close to $\phi(h^*)$ up to an additive error of $o(n^3)$. By substituting the lower bounds on the e_{ak} that follow from the constraints that r^* satisfies into the objective we obtain the lemma. ■

To summarize, by choosing say $\overline{\epsilon} = \epsilon/7$ we obtain from Equation (26) and Lemma 4.9 that

$$cp(r^*) \geq cp(g^*) - \epsilon n^3 \ .$$

for sufficiently large n. Since r^* is a candidate for our chosen placement r, and we choose a placement of minimum cost, the cost of r is no larger than the cost of r^* and we obtain the desired result.

4.5 Derandomizing the PTASs

The PTASs described in Sections 4.1-4.4 use randomization to pick the sample set of vertices which we denoted by S. In this section we show how to derandomize the PTASs for MINIMUM LINEAR ARRANGEMENT, MINIMUM CUT LINEAR ARRANGEMENT and MAXIMUM ACYCLIC SUBGRAPH in polynomial time using random walks on a constant degree expander to pick the set S. It is not clear whether the same derandomization technique also works for BETWEENESS. To obtain such result one would need to prove Lemma 4.8 when S is sampled using random walks on expanders as described next.

In Sections 4.1-4.3 we picked the vertices of S uniformly at random with replacement. Still using randomization consider the following alternative way of picking S. Identify the vertices of G with the vertices of a constant degree expander graph E with n vertices (as e.g. in [LPS88]). Pick a random walk of length $|S|$ in E starting at a random vertex. The multiset S that we pick consists of the vertices of G that correspond to the vertices of E that occur on the random walk.

We claim that Lemma 4.2 still holds even with this modified random choice of S. To prove this we partition the vertices of G into $t+1$ groups as folows. Vertex j is in group G_l, for some $1 \leq l \leq t$ if $(i,j) \in E$, $g(j) > k$, and $g(j) - k = l$. Otherwise, vertex j is in the group G_{t+1}. We use the chernoff-like bound for random walks on expanders proved in [Gil98] (Theorem 2.1) to show that the fraction of the vertices of G_i in the sample S, $\rho_S^i = \frac{|G_i \cap S|}{|S|}$, is close to $\rho^i = \frac{|G_i|}{n}$ with high probability. Specifically, if $|S| = O(\log n/\delta^2)$ then from Theorem 2.1 of [Gil98] follows that $|\rho^i - \rho_S^i| = \delta$ with high probability. Therefore, the contribution of each group G_i, to our estimator e_{ik} is off by at most δnt **whp** (We get an error of at most t contributed by at most δn vertices) . Summing over all groups we obtain that

$$e_{ik} \in \left[\sum_{(i,j) \in E, g(j) > k} (g(j) - k) \pm \delta nt * (t+1) \right].$$

Picking $\delta = \bar{\epsilon}/t(t+1)$ rather than $\delta = \bar{\epsilon}/t$ in Section 4.1 we obtain that Lemma 4.2 holds.

Once we establish the validity of the new random choice of S we can derandomize the algorithm by running it for every possible such random walk on the expander. Since there are only polynomially many random walks of length $|S| = O(\log n/\delta^2)$ on a constant degree expander we obtain a deterministic approximation algorithm which runs in polynomial time.

5 The general rounding procedure

In this section we generalize our rounding procedure for c-smooth APEC problems from Section 2 to work for general APEC problems and prove Theorem 1.3. First we assume for simplicity that the coefficients are all nonnegative, and show that we can produce a matching satisfying condition (7). Let x^* be a fractional perfect matching and w be any weight function with positive coefficients, each upperbounded by W. Our procedure with high probability produces a matching M such that for $\delta = O(\log\log n/\log n)$

$$w(M) \in [wx^* \pm (\delta wx^* + \tilde{O}(W))]. \tag{27}$$

(Note that $\delta = o(1)$.) Later in Section 5.3 we analyze the procedure when coefficients could be negative, and prove Theorem 1.3.

The algorithm uses the two phase approach of Section 2, with the following difference. Earlier, the Merge operator on matchings took the union of two matchings, and broke long cycles/paths at arbitrarily-chosen points, so that the resulting paths/cycles are all "small". Now it will choose the breakpoints randomly.

5.1 The decomposition phase

This phase produces $\Delta = O(\log^3 n)$ matchings exactly as in Section 2.1. Instead of Lemma 2.2 we prove the following.

Lemma 5.1 *Let $\xi = O(1/\log n)$. Then with high probability,*

$$\frac{1}{\Delta}\sum_{i=1}^{\Delta} w(M_i) \in [wx^* \pm (\xi wx^* + \tilde{O}(W))].$$

Proof. Mimic the proof of Lemma 2.2 but use Lemma 7.1 instead of Lemma 7.2. ■

5.2 The new merge operator

We define a (probabilistic) binary operator $\otimes$ on matchings with the following property. For every pair (A, B) of matchings in G, $A \otimes B$ is a matching that satisfies with high probability

$$w(A \otimes B) \in \left[\frac{w(A)+w(B)}{2} \pm \left(O\left(\frac{w(A)+w(B)}{k}\right) + \tilde{O}(kW)\right)\right], \tag{28}$$

where $k = \Theta(\log n)$.

$A \otimes B$ is constructed as follows. Note that $A \cup B$ is a union of vertex-disjoint alternating paths and cycles, some of which might have much more than k edges. Let q denote the number of vertices on paths/cycles with more than k edges. Randomly pick (with replacement) a multiset S of $l = \lceil q/k \rceil$ vertices out of them. We call vertices in S *breakpoints*, and we delete the edges incident to each breakpoint. Lemma 5.3 below says that with high probability, none of the remaining paths/cycles has length more than $O(k \log n)$. Now from each remaining path/cycle, randomly (i.e., with equal probability) pick either all the edges of A or all the edges of B, and put them in $A \otimes B$.

To calculate the weight of $w(A \otimes B)$, we first estimate the effect of deleting edges incident to breakpoints.

Lemma 5.2 *Let w_A be the weights of edges in A that are in paths/cycles of length $> k$. Then* **whp** *the weight of edges of A that are incident to S is no greater than $O\left(\frac{w_A}{k}\right) + \tilde{O}(W)$.*

Proof. For a vertex v that is part of a path/cycle of length $> k$, let w_v denote the weight of the edge in A that is incident to v.

Let X_i, $1 \leq i \leq l$ be a random variable representing the weight of A's edge incident to the ith breakpoint. The variable X_i takes each one of the values w_v with probability $1/q$ hence $|X_i| \leq W$. Now apply Lemma 7.1 to $\sum_{i=1}^{l} X_i$. ■

Lemma 5.3 *There is a $c > 0$ such that the following is true* **whp** *(over the choice of breakpoints). After the edges incident to the breakpoints are deleted, the length of each remaining path/cycle in $A \cup B$ is at most $ck \log n$.*

Proof. We partition long paths into (at most $O(n/(k \log n))$ paths of length $ck \log n$ and argue that **whp** each gets a breakpoint. Let P be any path of length at least $ck \log n$. The probability that P doesn't get a breakpoint is at most $(1-(ck \log n)/q)^l \leq exp(-c \log n)$. Hence **whp** each of the paths under consideration gets a breakpoint. ■

Now we show that **whp**, A's contribution to $w(A \otimes B)$ is very close to $w(A)/2$. First we consider A' which is the subset of the edges in A that are not incident to any breakpoint. The following lemma is analogous to Lemma 2.3.

Lemma 5.4 *Let A' denote the edges of A that are not incident to any breakpoint. Then* **whp**

$$w(A' \cap (A \otimes B)) \in [w(A')/2 \pm (O(w(A')/\log n) + \tilde{O}(kW))].$$

Proof. Similar to the proof of Lemma 2.3. Each edge of A' belongs to some path or cycle after removing the edges incident with breakpoints. We define a random variable

X_i for each such path or cycle whose value is the weight of the edges of A' on this path with probability $1/2$ and 0 otherwise. Clearly $|X_i| \leq Wck \log n$. Now we apply Lemma 7.1 to the sum of the variables X_i. ■

By Lemma 5.2 we have that **whp**,

$$w(A) - (O(w(A)/k) + \tilde{O}(W)) \leq w(A') \leq w(A). \tag{29}$$

By combining this equality with Lemma 5.4, and substituting $k = \Theta(\log n)$, we obtain that **whp** the contribution of edges from A to $w(A \otimes B)$ is in

$$[w(A)/2 \pm (O(w(A)/k) + \tilde{O}(kW))].$$

By similarly analyzing the contribution of B to $w(A \otimes B)$, we see that $A \otimes B$ **whp** satisfies condition (28).

Except for using the new merge operator $\otimes$ rather than $\odot$ the merge phase is the same as described in Section 2. We combine the Δ matchings produced by the decomposition phase into one matching A in $\log \Delta$ phases. In each such phase we arbitrarily pair up all the matchings and merge each pair using the $\otimes$ operator. Now we show that M satisfies Equation (27).

Let $M_1^j, \ldots, M_{\Delta/2^j}^j$ denote the $\Delta/2^j$ matchings obtained at round j of the merging procedure where $0 \leq j \leq \log \Delta$. A simple induction on j using the property of the $\otimes$ operator stated in equation (28) shows that the sum of the weights of these matchings is **whp** sandwiched between

$$(1 - O(1/k))^j \left(\frac{\sum_{i=1}^{\Delta} w(M_i)}{2^j} - \Delta \left(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^j} \right) \tilde{O}\left(kW\right) \right)$$

and

$$(1 + O(1/k))^j \left(\frac{\sum_{i=1}^{\Delta} w(M_i)}{2^j} + \Delta \left(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^j} \right) \tilde{O}\left(kW\right) \right).$$

Substituting $j = \log \Delta$ in these expressions, and noting that $(1 \pm O(1/k))^{\log \Delta} \approx 1 \pm O(\log \Delta / k)$ when $\log \Delta \ll k$, we get that $w(M)$ is **whp** sandwiched between

$$(1 - O(\log \Delta / k)) \sum_{i=1}^{\Delta} w(M_i)/\Delta - \tilde{O}(kW\Delta) \tag{30}$$

and

$$(1 + O(\log \Delta / k)) \sum_{i=1}^{\Delta} w(M_i)/\Delta + \tilde{O}(kW\Delta). \tag{31}$$

Finally recall that $k = \Theta(\log n)$, and $\Delta = O(\log^3 n)$, so by combining Equations (30), (31), and Lemma 5.1 we obtain that **whp** $w(M)$ is in $[wx^* \pm (\delta wx^* + \tilde{O}(W))]$, as claimed in Equation (27).

We finish by estimating the number of edges in M. As in Section 2 we consider the weight function $\overline{w}_{ij} = 1$ for every $i, j \in [n]$. We note that a fractional perfect matching x satisfies $\overline{w}x = n$. So by applying Equation (27) to this weight function we obtain that our matching contains $n - o(n)$ edges **whp**.

Remark: There exists a tradeoff between the multiplicative error factor δ and the additive error term $\tilde{O}(kW\Delta)$ that we did not make explicit. As Δ and k increase the multiplicative factor goes to zero faster but the additive error increases.

5.3 Positive and Negative Coefficients

Now we analyze the rounding procedure when each coefficient w_{ij} in the weight function can be positive or negative, and $|w_{ij}| \leq W$. The goal is to prove Theorem 1.3. The only difference from the case when coefficients were positive is that we need a different Chernoff bound, Lemma 7.2, where we earlier used Lemma 7.1. The analogue of Lemma 2.2 is as follows.

Lemma 5.5 *Let $\xi = O(1/\log n)$. Then* **whp** *the matchings produced in the decomposition phase satisfy*

$$\frac{1}{\Delta}\sum_{i=1}^{\Delta} w(M_i) \in [wx^* \pm (\xi wx^* + \tilde{O}(n^{1/2}W))]. \quad \blacksquare$$

Let $k = \Theta(\log n)$ as before. The following lemma replaces Lemma 5.2.

Lemma 5.6 *With high probability, the sum of weights of edges of A that are incident to a breakpoint is in*

$$\left[\frac{w_A}{k} \pm \tilde{O}(n^{1/2}W)\right]. \quad \blacksquare$$

Lemma 5.3 continues to hold. and the following lemma replaces Lemma 5.4

Lemma 5.7 *Let A' denote the edges of A that are not incident to any breakpoint. Then* **whp**

$$w(A' \cap (A \otimes B)) \in [w(A')/2 \pm \tilde{O}(n^{1/2}kW)].$$

From the last two lemmas applied to the edges of A as well as the edges of B we obtain that **whp**

$$w(A \otimes B) \in \left[\frac{w(A) + w(B)}{2} \pm \left(O\left(\frac{w(A) + w(B)}{k}\right) + \tilde{O}(n^{1/2}kW)\right)\right].$$

Now by a simple induction as in the nonnegative case we can show that **whp**

$$w(M) \in [wx^* \pm (\delta wx^* + \tilde{O}(n^{1/2}W))].$$

This finishes the proof of Theorem 1.3.

6 Conclusion

We have presented a rounding procedure for linear programs that contains assignment constraints. In contrast with randomized rounding [RT87] our method produce an integral solution which is an almost perfect assignment. The rounded solution also satisfies each original constraint with some additive error. We use this procedure to design quasi-polynomial time approximation scheme for the quadratic assignment problem. We believe that our rounding scheme may have other applications that are yet to be found.

We can derandomize our algorithm using the standard method of conditional probabilities [R88, AS92]. An interesting question for for further research is whether there is a deterministic parallel version of our rounding scheme that runs in polylogarithmic time (i.e. in NC). One possible approach that may lead to such algorithm (with some penalty in the approximation error) is to use some construction of *polylogarithmic*-wise independent random variables combined with an efficient parallel implementation of the method of conditional probabilities in a way similar to the one used by Berger and Rompel in [BR].

7 Chernoff Bounds

The Chernoff-style tail bounds needed in this paper are a mixture of some standard bounds. Since we haven't found a published source for it, we give brief proofs for it using standard techniques. The goal of this section is to prove the following two lemmas.

Lemma 7.1 *Let $Y_1, \ldots, Y_k$ be independent random variables such that $0 \leq Y_i \leq U$. Let $S = \sum_i Y_i$ and $\mu = E[S]$. Then for every $\alpha > 0$, there exists $c > 0$ such that*

$$\mathbf{Pr}\left[|S - \mu| \geq c \max\{\mu/\log n, U \log^2 n\}\right] < 1/n^{\alpha}.$$

Proof. Let $X_i = Y_i/U$, and apply Corollary 7.6 below to $X_1, \ldots, X_k$. ∎

Lemma 7.2 *Let $Y_1, \ldots, Y_k$ be independent random variables such that for each $i \in [k]$, either $0 \le Y_i \le U$ or $-U \le Y_i \le 0$. Let $S = \sum_i Y_i$, $\mu = E[S]$ and $\hat{\mu} = E[\sum_i |Y_i|]$. Then for every $\alpha > 0$, there exists $c > 0$ such that*

$$\mathbf{Pr}\left[|S-\mu| \ge c\max\{(\hat{\mu} U \log n)^{1/2}, U\log n\}\right] \le 1/n^{\alpha}.$$

Proof. Let $X_i = Y_i/U$, and apply Lemma 7.7 below to $X_1, \ldots, X_k$. ■

In Lemmas 7.3-7.6, $X_1, \ldots X_k$ are independent random variables such that $0 \le X_i \le 1$. Let $S = \sum_i X_i$ and $\mu = E(S)$. The starting point is the following lemma.

Lemma 7.3 *(1) For any $\beta \ge 0$ and any λ,*

$$\mathbf{Pr}[S \ge (1+\lambda)\mu)] \le e^{-\beta(1+\lambda)\mu} e^{\mu(e^{\beta}-1)}$$

(2) For any $\beta \le 0$ and any λ,

$$\mathbf{Pr}[S \le (1-\lambda)\mu)] \le e^{-\beta(1-\lambda)\mu} e^{\mu(e^{\beta}-1)}.$$

Proof. We prove (1); the proof of (2) is analogous. By Markov's inequality for any $\beta \ge 0$

$$\mathbf{Pr}[S \ge (1+\lambda)\mu)] \le E(e^{\beta S}) e^{-\beta(1+\lambda)\mu}.$$

Since $e^{\beta x}$ is a convex function of x for any fixed β we have $e^{\beta x} \le 1 - x + xe^{\beta}$ for $0 \le x \le 1$. So for each i we have

$$\mathbf{E}(e^{\beta X_i}) \le 1 + \mathbf{E}(X_i)(e^{\beta}-1) \le e^{E(X_i)(e^{\beta}-1)}.$$

Hence

$$E(e^{\beta S}) \le \prod_{i=1}^{k} e^{E(X_i)(e^{\beta}-1)}.$$

The result follows by combining the inequalities. ■

Lemma 7.4

$$\begin{array}{llll} (a) & \mathbf{Pr}\left[|S-\mu| \ge \lambda\right] & \le & 2^{1-\lambda} e^{\mu} \\ (b) & \mathbf{Pr}\left[|S-\mu| \ge \lambda\sqrt{\mu}\right] & \le & 2e^{-\lambda^2/3}. \\ (c) & \mathbf{Pr}\left[|S-\mu| \ge \lambda\right] & \le & 2e^{-2\lambda^2/k}. \end{array}$$

Where (a) and (c) hold for any $\lambda \ge 0$ and (b) holds for $\lambda \le \mu^{1/2}$ ■

Proof. To obtain (a) substitute $\beta = \ln 2$ and $\beta = -\ln 2$ in Lemma 7.3 (1) and (2) respectively. To obtain (b) substitute $\beta = \ln(1+\lambda)$ and and $\beta = \ln(1-\lambda)$ in Lemma 7.3 (1) and (2) respectively, and make some approximations. Notice that these two values minimize the right hand sides of Lemma 7.3. To obtain (c) follow the proof of Lemma 7.3 but use the bound $E(e^{\beta S}) \leq e^{\mu\beta+\beta^2 k/8}$ (for a proof see [H64]) instead of the one used here. Then substitute $\beta = 4\lambda\mu/k$ and $\beta = -4\lambda\mu/k$ in (1) and (2) respectively. ∎

Lemma 7.5 *For every $\alpha > 0$ there is a $c > 0$ such that*

$$\mathbf{Pr}\left[|S-\mu| \geq c\max\{(\mu\log n)^{1/2}, \log n\}\right] \leq 1/n^{\alpha}.$$

Proof. If $\mu < c_2 \log n$ use Lemma 7.4 (a) with $\lambda = c_1 \log n$ for a large enough constant c_1. Otherwise, use Lemma 7.4(b) with $\lambda = (c_2 \log n)^{1/2}$. ∎

The following is an easy corollary (It follows since $\max\{(\mu\log n)^{1/2}, \log n\} \leq \max\{\mu/\log n, \log^2 n\}$).

Corollary 7.6 *For every $\alpha > 0$ there is a $c > 0$ such that*

$$\mathbf{Pr}\left[|S-\mu| \geq c\max\{\mu/\log n, \log^2 n\}\right] \leq 1/n^{\alpha}.$$

Now we turn to the case when some variables are negative.

Lemma 7.7 *Let X_i, $1 \leq i \leq k$ be independent random variables such that for each $i \in [k]$, either $0 \leq X_i \leq 1$ or $-1 \leq X_i \leq 0$. Let $S = \sum_{i=1}^n X_i$, $\mu = E(S)$, and $\hat{\mu} = E(\sum_{i=1}^n |X_i|)$. Then for every $\alpha > 0$ there is a $c > 0$ such that:*

$$\mathbf{Pr}\left[|S-\mu| \geq c\max\{(\hat{\mu}\log n)^{1/2}, \log n\}\right] \leq 1/n^{\alpha}.$$

Proof. Let I be the set of i's such that $0 \leq X_i \leq 1$. Let $S_1 = \sum_{i\in I} X_i$, $S_2 = \sum_{i\notin I} X_i$, and $\mu_1 = E(S_1), \mu_2 = E(S_2)$.

Notice that $|S-\mu| \leq |S_1-\mu_1| + |S_2-\mu_2|$. Apply lemma 7.5 to bound $|S_1-\mu_1|$ and $|S_2-\mu_2|$ and use the inequality $\sqrt{x}+\sqrt{y} \leq \sqrt{2(x+y)}$. ∎

Acknowledgements

We thanks the referees for their comments that helped us improve our presentation.

References

[A99] F.Afrati, E.Bampis, C.Chekuri, D.Karger, C.Kenyon, S.Khanna, I.Milis, M.Queyranne, M.Skutella, C.Stein, M.Sviridenko. Approximation schemes for Minimizing Average Weighted Completion Time with Release dates. In *Proceedings of the 40th FOCS*, pages 32–43. IEEE, 1999.

[AS92] N.Alon and J.H.Spencer, The Probabilistic Method. John Wiley and Sons, New York, 1992.

[A96] S. Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *JACM* 45(5): 753-782, 1998.

[AFK96] S. Arora, A. M. Frieze and H. Kaplan. A New Rounding Procedure for the Assignment Problem with Applications to Dense Graph Arrangement Problems. In *Proceedings of the 37th FOCS*, pages 21–30, IEEE, 1996.

[AKK95] S. Arora, D. Karger and M. Karpinski. Polynomial-time approximation schemes for dense instances of NP-hard optimization problems. *JCSS*, 58(1):193-210, 1999.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *JACM* 45(3):501–555, 1999.

[B94] B. S. Baker. Approximation Algorithms for NP-complete problems in planar graphs. *JACM* 41:153-180, 1994

[BR] B. Berger and J. Rompel Simulating $\log^c n$-wise Independence in NC. *JACM* 38(4): 1026–1046, 1991.

[B98] B.Bollobás Modern Graph Theory. Springer Verlag, Graduate Texts in Mathematics 184, New York 1998.

[BC96] R.E.Burkard and E.Cela. Quadratic and Three-Dimensional Assignments: An Annotated Bibliography. In M. Dell'Amico, F.Maffioli and S.Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, Wiley, Chichester, 373-392, 1997.

[CK00] C.Chekuri and S.Khanna. A PTAS for the Multiple Knapsack Problem. In *Proceedings of the 11th SODA*, pages 213–222.

[Cohen93] E. Cohen Approximate max-flow on small depth networks. *Siam J. Comput.* 23(3):579-597, 1995

[FdlV94] W. Fernandez de la Vega. MAXCUT has a randomized approximation scheme in dense graphs. *Random Structures & Algorithms* 8(3):187-198, 1996.

[FL81] W.Fernandez de la Vega and G.S.Lueker Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica* 1(4):349–355, 1981.

[FGHP93] T. Fisher, A. V. Goldberg, D. J. Haglin and S. Plotkin Approximating matchings in parallel. *Information Processing letters* 46:115–118, 1993.

[FK96] A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. In *Proceedings of the 37th FOCS*, pages 12–20. IEEE, 1996.

[FK99] A.M.Frieze and R.Kannan Quick approximation to matrices and applications *Combinatorica* 19:175–220, 1999.

[Gil98] D. Gillman. A Chernoff bound for random walks on expanders. *SIAM Journal on Computing* 27:1203-1220, 1998.

[GPST92] A. V. Goldberg, S. A. Plotkin, D. Shmoys, and E. Tardos Interior-Point Methods in Parallel Computation. *SIAM J. Comput.* 21(1):149–150, 1991.

[GGR96] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. JACM 45(4): 653-750 (1998).

[H64] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Stastical Association* 58:13–30, 1963.

[IK75] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subsets problems. *JACM*, 22(4):463–468, 1975.

[KK82] N. Karmarkar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proc. 23rd FOCS*, pages 312–320. IEEE, 1982.

[KUW86] R.M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. *Combinatorica* 6(1):35–48, 1986.

[KM96] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proc. 28th STOC*, pp. 329–337. ACM, 1996.

[Law76] E. Lawler. Combinatorial Optimization: Networks and Matroids. Holt, Rinehart, and Winston, Fort Worth TX, 1976.

[LR88] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 46(6):787 - 832, 1999.

[LPV81] G. F. Lev, N. Pippenger, and L. Valiant. A Fast Parallel Algorithm for Routing in Permutation Networks. In *IEEE Trans. Computers,* C-30(2):93–100, 1981.

[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261– 277, 1988.

[LN92] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. 25th STOC*, pp 448–457. ACM, 1993.

[MVV87] K. Mulmuley, U. Vazirani and V.V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica* 7(1):105–113, 1987.

[PRW94] P.M.Pardalos, F.Rendl and H.Wolkowicz The quadratic assignment problem: a survey of recent developments. In P.Pardalos and H.Wolkowicz editiors, *Quadratic assignment and related problems*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science Volume 16 pages 1-42, 1994.

[R88] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *JCSS*, 37(2):130–43, 1988.

[RT87] P. Raghavan and C. Thompson. Randomized Rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365-374, 1987.

[RT98] J.D.P. Rolim and L. Trevisan. A Case Study of De-randomization Methods for Combinatorial Approximation Problems. *Journal of Combinatorial Optimization*, 2(3):219-236, 1998.

[SG76] S. Sahni and T. Gonzales. P-complete approximation problems. *JACM*, 23:555–565, 1976.

[S78] E. Szemerédi Regular partitions of graphs. In: Proc. Colloque Inter. CNRS No. 260 (J. C. Bermond, J. C. Fournier, M. Las Vergnas and D. Sotteau eds.), 399-401, 1978.

[ST93] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Math. Programming*, 62:461–474, 1993.

[SG93] P. Spirakis and A. Gibbons PRAM models and fundamental parallel algorithmic techniques: Part I. In *Lectures in Parallel Computation* edited by A. Gibbons and P. Spirakis, Cambridge University Press, 1993.