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

Sanjeev Arora* Alan Frieze† Haim Kaplan‡

May 31, 1996

Abstract

We present approximation schemes for "dense" instances of many well-known NP-hard problems, including 0-1 QUADRATIC-ASSIGNMENT, (an optimization formulation of) GRAPH-ISOMORPHISM, MIN-CUT-LINEAR-ARRANGEMENT, MAX-ACYCLIC-SUBGRAPH, BETWEENNESS, and MIN-LINEAR-ARRANGMENT. (A "dense" graph is one in which the number of edges is $\Omega(n^2)$; denseness for the other problems is defined in an analogous way.) Some of our approximation schemes run in $n^{O(\log n/\epsilon^2)}$ time; others run in $n^{O(1/\epsilon^2)}$ time.

Of independent interest is our randomized procedure for rounding fractional solutions of linear programs that represent the assignment problem subject to linear constraints. This extends the well-known LP rounding procedure of Raghavan and Thompson, and also solves an open problem of Luby and Nisan ("Design an NC procedure for converting near-optimum fractional matchings to near-optimum matchings.")

*Computer Science, Princeton University. **E-mail: arora@cs.princeton.edu**

†Department of Mathematics, Carnegie Mellon University, Pittsburgh PA15213. Supported in part by NSF grant CCR9225008. **E-mail: af1p@andrew.cmu.edu**

‡Computer Science, Princeton University. **E-mail: hkl@cs.princeton.edu**

1 Introduction

Computing approximate solutions to NP-hard optimization problems is an important task. Approximation algorithms have been designed for many important problems in the last couple of decades. Designing *polynomial time approximation schemes* or PTAS's has proved much harder. A PTAS is an algorithm that, given an instance of the problem and any fixed $\epsilon > 0$, achieves an approximation ratio $1 + \epsilon$ in time that is polynomial in the input size (the time could depend arbitrarily on ϵ). A PTAS is desirable since it allows us to trade off approximation accuracy for running time.

PTAS's are known for only very 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[A96].) 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 proved for many other problems.

Such inapproximability results raise 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. Arora, Karger and Karpinski [AKK95] showed that many MAX-SNP-hard problems have a PTAS when the instance is "dense" (in case of graphs, this means that each vertex has degree $\Omega(n)$, although some of the algorithms in [AKK95] work also when the average degree is $\Omega(n)$).They also gave a PTAS for BISECTION on dense graphs. Fernandez de la Vega [dlV94] independently gave a PTAS for dense instances of MAX-CUT and some other problems.

In this paper we describe approximation schemes for dense instances of a number of classical optimization problems. These include 0-1 QUADRATIC-ASSIGNMENT, (an optimization formulation of) GRAPH-ISOMORPHISM, MIN-CUT-LINEAR-ARRANGEMENT, MAX-ACYCLIC-SUBGRAPH, LINEAR-ARRANGMENT, and BETWEENNESS. Some of these approximation schemes run in polynomial time; others run in $O(n^{O(\log n/\epsilon^2)})$ time, where n is the input size.

The approach draws some inspiration from [AKK95] but differs in one significant way. In all the problems considered in [AKK95], the set of feasible solutions is the set of $0-1$ vectors with n coordinates, and the objective function is a constant-degree polynomial. In all the problems we consider, the objective function is still a constant-degree polynomial, but the set of feasible solutions are permutations of n elements. (Note: Such permutation-based NP-hard problems are often especially nasty with respect to approximation. Papadimitriou and Yannakakis[PY91] have identified a set of such problems called MAX-π-SNP; BETWEENNESS and MAX-ACYCLIC-SUBRAGPH are complete for this class.) Recall that permutations are represented in mathematical programming via the *assignment constraints* in n^2 variables $\{x_{ij} : i, j = 1, 2, \ldots, n\}$:

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

An integeral solution to these constraints is called a *perfect (bipartite) matching.* A fractional solution is called a *fractional perfect matching.*

A casual examination of the assignment constraints immediately seems to rule out a central technique of [AKK95], namely, the use of Raghavan-Thompson [RT87] rounding to convert fractional solutions to integral "approximate" solutions. Randomized rounding would change a fractional perfect matching (x_{ij}) into a 0-1 vector y by setting $y_{ij} = 1$ with probability x_{ij}. As is well-known, such a y could be far from a matching and leave a constant fraction of the vertices unmatched. (For example, if all $x_{ij} = 1/n$, then the expected number of unmatched vertices in y is n/e.)

The major technique introduced in this paper is a new randomized rounding procedure for fractional matchings. 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." The procedure produces a matching that contains $n - o(n)$ edges, and moreover, this matching "approximately" satisfies, with high probability, any linear inequality that was satisfied by the fractional matching.

We use this rounding procedure to design an algorithm that, given a constant degree polynomial p in the variables (x_{ij}), finds an almost-perfect matching that approximately maximizes p. This algorithm immediately allows us to design the approximation schemes mentioned above. However, our approximation is only additive, and works only if the coefficients of p satisfy certain *smoothness* constraints. That is why the approximation schemes require the problem instance to be *dense*.

Our rounding procedure for fractional matchings is probably of interest beyond its use in such approximation schemes. Assigning n tasks to n people — the canonical use of the assignment constraints— is a basic primitive in many applications. Assigning such tasks so as to minimize a linear *cost* function is the minimum weight perfect matching problem, which is in P. Assigning tasks in presence of more than one linear constraint (while still insisting on integral solutions) is problematic, and leads to NP-hard problems in general. Our rounding procedure suggests a practical approach to dealing with such NP-hard problems: obtain a fractional solution to the integer program, and then round it using our procedure to an integral solution. The integral solution is a matching that matches "almost all" vertices (i.e., assigns all but $o(n)$ jobs to different people), and "approximately" satisfies the original linear constraints. In a practical situation, such an integral solution may well suffice.

Our rounding procedure can be implemented in parallel in depth $O(\log n \log\log n)$, and this also solves the main open problem left open by Luby and Nisan's NC^1 approximation algorithm for fractional packing/covering problems[LN92] They showed how to solve a minimum weight matching problem approximately in NC^1. However, their procedure yields a fractional matching, and they had left it as an open problem whether this fractional matching could be "rounded" to an integral almost-matching in NC.

Comparison with Related Work: 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. On the other hand, we return an almost-perfect matching as a solution, which [ST93] could not.

1.1 The Problems

Now we describe the NP-hard problems that are considered in this paper. Depending upon the problem, we denote a permutation in one of two ways: as a function $\pi: \{1, \ldots, n\} \to \{1, \ldots, n\}$ or

as a vector (x_{ij}) that satisfies the assignment constraints.

Recall that we are only interested in *dense* subcases of these problems. A graph is *a-dense* if the number of edges is at least $a \cdot n^2$. We will mention what denseness means for problems that are not graph problems.

1. 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 $c(x) = \sum_{i,j,k,l} c_{ijkl} x_{ij} x_{kl}$. Well-known inapproximable problems such as CLIQUE and general TSP can be reduced to QUADRATIC-ASSIGNMENT, so the problem has no good approximation algorithms if $P \neq NP$ ([SG76]). We will be interested in the subcase when each $|c_{ijkl}| = O(1)$ and the minimum value of $c(x)$ is $\Omega(n^2)$.

2. GRAPH-ISO: This is an optimization version of the usual decision problem. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ each with n vertices, it seeks the lowest cost embedding of G_1 in G_2. An *embedding of* G_1 *into* G_2 is a permutation (x_{ij}), and its *cost* is the number of uncovered edges, that is,

$$c(x) = \sum_{\{i,j\} \in E_1, \{k,l\} \notin E_2} x_{ik} x_{jl} + \sum_{\{i,j\} \notin E_1, \{k,l\} \in E_2} x_{ik} x_{jl}.$$

   We consider the case when $c(x) = \Omega(n^2)$ for every x. In other words, the graphs are "very nonisomorphic": every embedding of G_1 into G_2 fails to cover $\Omega(n^2)$ edges.

3. MIN-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 n nodes in a straight line, such that the total edge length in the layout is minimized. An $O(\log^2 n)$-factor approximation algorithm is presented in [LR88].

4. d-DIMENSIONAL ARRANGEMENT: The analogue of the linear arrangement problem when the n points are on a d-dimensional grid instead of in a line. "Length" of edge (i, j) is the Manhattan distance from $\pi(i)$ to $\pi(j)$. Here d is a constant.

5. 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. We will restrict attention to dense graphs. Previous work includes an exact algorithm when the graph is a tree [Y85], and a $O(\log^2 n)$-factor approximation on general graphs [LR88]..

6. BETWEENESS: Given a finite set A and a collection 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).

7. MAX-ACYCLIC-SUBGRAPH: Given a digraph $G = (V, E)$, find the largest (with respect to the number of edges) acyclic subgraph in it.

Now we note that 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 MIN-LINEAR-ARRANGEMENT, which we'll deal with separately. Now we formalize this class of problems.

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

$$\begin{array}{ll} max & p(x) \\ s.t. & x \text{ is a permutation} \end{array}$$

where p is a degree d polynomial. The problem could involve minimization instead of maximization.

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

Theorem 1.1 *There is a deterministic 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*

$$p(x) \geq p(x^*) - \epsilon n^d,$$

where x^* is the perfect matching at which p attains its maximum value. The algorithm runs in $n^{O(c^3 d^3 \log n/\epsilon^2)}$ time.

We will indicate a proof of this theorem in Section 3.

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,* BETWEENNESS, MIN-LINEAR-ARRANGEMENT, MIN-CUT-LINEAR-ARRANGMENT, d-DIMENSIONAL ARRANGEMENT, AND MAX-ACYCLIC-SUBGRAPH *have an $n^{O(1/\epsilon^2)}$ time approximation scheme.*

Part of Theorem 1.2 is a corollary to Theorem 1.1. The $n^{O(1/\epsilon^2)}$ time approximation schemes will be treated separately in Section 4. We note that the techniques in that section also allows us to obtain a PTAS for the dense version of the Koopmans-Becker Quadratic Assignment Problem.

1.2 Assignment Problem with Additional Linear Constraints

Let $G = (V_1, V_2, E)$ be a bipartite graph with $|V_1| = |V_2| = n$ and $|E| = m$. The *assignment polytope* of G, denoted $\mathcal{A}_G$ (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.

In particular, minimizing a linear function over this polytope is the well-known *assignment problem,* also called the *minimum cost matching* problem. By incorporating the cost function into a linear constraint in the usual way, the problem is equivalently viewed as a decision problem: find a perfect matching that satisfies one additional linear constraint. We consider a modification of this decision problem in which the number of additional constraints is $K = \text{poly}(n)$. We call this

the *Assignment Problem with Extra Constraints*, or APEC. The problem is to find an integral x such that

$$\begin{aligned} x &\in \mathcal{A}_G \\ a_k^T x &\geq b_k, \qquad k = 1, 2, \ldots, K \end{aligned} \tag{2}$$

Note that the problem contains integer linear programming as a subcase, and hence is NP-hard. One could try to compute "approximate" integer solutions by solving the system as a linear program, and then using Raghavan-Thompson rounding on the resulting fractional solution. This would give a 0-1 vector x that satisfies w.h.p.

$$a_k^T x \geq b_k - \tilde{O}(\sqrt{n} A_k) \qquad k = 1, 2, \ldots, K, \tag{3}$$

where A_k is the largest (in magnitude) coefficient of a_k and $\tilde{O}$ is the usual "soft-Oh" notation that suppresses poly($\log n$) factors. But as already mentioned, such an x may be far from being a permutation.

The following theorem states the existence of an alternative rounding procedure.

Theorem 1.3 *There is a randomized algorithm that, given a fractional solution x^* to the system (2), produces a matching x that contains at least $n - o(n)$ edges and satisfies*

$$a_k^T x \geq (1 - o(1)) b_k - \tilde{O}(\sqrt{n} A_k) \qquad k = 1, 2, \ldots, K. \tag{4}$$

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 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

$$a_k^T x \geq b_k (1 - o(1)) - \tilde{O}(A_k) \qquad k = 1, 2, \ldots, K, \tag{5}$$

(The $-$'s are changed to $+$'s if the inequality involved a $\leq$ instead of a $\geq$.)

2 Rounding Procedure for the Assignment Problem

For ease of exposition, we first describe a simpler rounding procedure that works correctly 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 (and also in solving Luby and Nisan's open problem) are of this type. Furthermore, the simple procedure described here motivates our more general procedure that is described in the appendix. The more general procedure works even when the value of the coefficients is allowed to grow with n.

Let x^* be a fractional perfect matching for the graph $G = (V, E)$, and ϵ be an arbtrarily small constant. 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 with probability at least $1 - 1/n^f$ (where f is any prespecified constant), the matching M satisfies $w(M) \in w(x^*) \pm \tilde{O}(n^{3/4})$. (From

now on, whenever we say "high probability" we mean probability $1 - 1/n^f$ for some large enough f.) Hence it follows that if $K < n^{f-1}$ and $w^1, \ldots, w^K$ are any c-smooth weight functions, then M satisfies with probability at least $1 - 1/n$:

$$w^j(M) \in w^j(x^*) \pm \tilde{O}(n^{3/4}) \qquad \text{for } j = 1, 2, \ldots, K.$$

The procedure consists of two phases. The first, called *decomposition,* produces $\Delta = O(\log^2 n)$ perfect matchings $M_1, M_2, \ldots, M_\Delta$ such that with high probability,

$$\frac{1}{\Delta} \sum_{i=1}^{\Delta} w(M_i) \in w(x^*) \pm \tilde{O}(n^{1/2}). \tag{6}$$

The second phase consists of applying a binary (probabilistic) operator $\odot$ (called *merge*) on matchings. If A and B are two matchings then for any c-smooth function w, the probability is at least $1 - n^{f+1}$ that

$$w(A \odot B) \in \frac{w(A) + w(B)}{2} \pm \tilde{O}\left(n^{3/4}\right) \tag{7}$$

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 (7) implies that with probability $1 - \Delta/n^{f+1}$,

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

Since $\Delta = O(\log^2 n)$, the error term is at most $\tilde{O}(n^{3/4})$. Now it follows from Equation (6) that

$$w(M) \in w(x^*) \pm \tilde{O}(n^{3/4})$$

.

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 w.h.p., the decomposition phase ensures $\frac{1}{\Delta} \sum_i w^0(M_i) \geq n - \tilde{O}(n^{1/2})$. Hence with high probability, the merges ensure that $w^0(M)$ is also at least $n - \tilde{O}(n^{3/4})$; in other words, M has $n - O(n^{3/4})$ edges.

Now we describe the two phases. The following procedure is used in the decomposition phase.

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 $L = \Theta(\log^2 n)$ 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^*.

Claim 1: *With high probability, each vertex in G^* has degree $L \pm \tilde{O}(L^{1/2})$.*
Proof. The degree of vertex i is $\sum_j \xi_{ij}$, and the expectation of this degree is

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

Since $L = \Theta(\log^2 n)$, the Chernoff bound in Lemma 7.1(b) implies that the deviation from the mean is at most $\tilde{O}(L^{1/2})$ w.h.p. ■

The decomposition phase constructs the multigraph G^*. Then, it adds $\tilde{O}(nL^{1/2})$ edges to G^* to make it Δ-regular were $\Delta = O(L) = O(\log^2 n)$. Note that these edges might not be present in G; we are adding them just to simplify the exposition. The procedure decomposes G^* into the disjoint union of Δ perfect matchings, $M_1 \cup M_2 \cup \cdots \cup M_\Delta$. Then it ignores the edges that were not present in G and is left with Δ matchings. Using Claim 1 and the fact that the standard deviation of the random variable $\sum_{i=1}^{\Delta} w(M_i)$ is $O(n^{1/2})$ we can prove

Claim 2: *With high probability,*

$$\frac{1}{\Delta}\sum_{i=1}^{\Delta} w(M_i) \in w(x^*) \pm \tilde{O}(n^{1/2}).$$

This finishes the description of the decomposition phase. Now we describe the merge phase.

MERGE:

Given: Two matchings A and B.

Procedure: $A \cup B$ 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 $\emptyset(n^{1/2})$ *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.

Claim 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 of A in these paths or cycles. Each α_i is at most $cn^{1/2}$ in absolute value. The expected contribution Z_A of A's edges to $w(A \odot B)$ is $\sum_i \alpha_i/2 = w(A)/2$, and furthermore, the m contributions are decided using independent coin tosses. Hence by Lemma 7.4 we obtain that $|Z_A - w(A)/2| = \tilde{O}(n^{3/4})$. ■

Here is a good place to contrast our rounding procedure with Raghavan-Thompson rounding. RT-rounding on $(A+B)/2$ 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.

2.0.1 Solution to Luby and Nisan's problem

Luby and Nisan give an NC^1 approximation algorithm that approximates the linear program for maximum-cardinality matching in unweighted graphs. The program finds a fractional matching of "cardinality" at least $(1-\epsilon)OPT$. However, it does not produce a matching. Our randomized rounding procedure can produce a matching. Furthermore, as mentioned already in Theorem 1.3, the procedure runs in $O(\log n \log\log n)$ time on an EREW PRAM. This relies on a parallel algorithm in [LPV81] that decomposes L-regular multigraphs into L disjoint perfect matchings in $O(\log L \log n)$ parallel time. In our procedure $L = \text{poly}(log(n))$.

We can derandomize our algorithm by derandomizing OVERSAMPLING and MERGE in the standard way. Also, if we don't care about derandomization, then our more sophisticated algorithm of Theorem 1.3 has the same parallel running time but works on weighted graphs so long as the weights are "moderate."

2.1 The General Rounding Procedure

The general procedure is similar to the simple procedure described above: we do an oversampling first, then decompose the resulting multigraph into a disjoint union of perfect matchings, then do a sequence of merges. The important differences is that the merge operation does not break long paths in $A \cup B$ arbitrarily, but at random points.

The proof that the procedure works is more involved, and is given in the appendix.

3 Approximation Algorithm for degree-d arrangement

Now we prove Theorem 1.1

Proof.(Theorem 1.1) For ease of exposition we prove the theorem for $d = 2$. The proof for general d involves a simple induction. Let $\sum_{ijkl} c_{ijkl} x_{ij} x_{kl}$ be the objective function, where each c_{ijkl} is an integer in $[-c, c]$. Let x^* be the perfect matching that maximizes the objective function.

The basic idea is similar to the main proof in [AKK95], except that we always have to deal with the assignment constraints. Let b_{ij} denote $\sum_{kl} c_{ijkl} x^*_{kl}$. Then x^* is the solution to the APEC

$$\begin{array}{ll} \text{maximize} & \sum_{ij} b_{ij} x_{ij} \\ & \sum_{kl} c_{ijkl} x_{kl} = b_{ij} \end{array}$$

The procedure consists of two parts: (i) Use random sampling to estimate b_{ij}'s within an additive error ϵn. Pick a random sample S of $O(c^2 \log n/\epsilon^2)$ vertices. Then enumerate all possible (i.e., $n^{O(c^2 \log n/\epsilon^2)}$) ways in which they can be placed in a permutation. One of these ways is also the way in which they are placed by x^*; we restrict attention to that one. (Of course, the polynomial-time procedure, not knowing x^*, must do the rest of the work for each of the $n^{O(c^2 \log n/\epsilon^2)}$ guesses.) Estimate the sum $\sum_{kl} c_{ijkl} x^*_{kl}$ by looking at the value of x^*_{kl} for each $k \in S$. Let b'_{ij} be this estimate. Chernoff bounds imply that each $b'_{ij} \in b_{ij} \pm \epsilon n$ w.h.p. (ii) Now consider the optimization problem

$$\begin{array}{lll} \text{maximize} & \sum_{ij} b'_{ij} x_{ij} & \\ b'_{ij} - \epsilon n \leq & \sum_{kl} c_{ijkl} x_{kl} & \leq b'_{ij} - \epsilon n \end{array}$$

Use binary search to replace the linear objective function by a linear constraint, thus obtaining an instance of APEC. A solution y to that APEC satisfies

$$|c(y) - c(x^*)| \leq \epsilon n \times n = \epsilon n^2.$$

Of course, we don't know how to solve the APEC exactly. Let us calculate the error due to our approximation of APEC. We scale the objective function by n to make each coefficient a number between $-c$ and c, thus making the APEC c-smooth. Then we solve the APEC as an LP and use our rounding algorithm (actually, even the simple algorithm in Section 2 suffices) to obtain a solution z that is a matching of cardinality $n(1 - o(1))$ and satisfies the given constraints with an additive error of $o(n)$. Hence

$$|c(z) - c(y)| \leq o(n) \times n = o(n^2).$$

Hence we have shown that $|c(z) - c(x^*)| \leq \epsilon n^2$.

■

4 PTASs

We describe PTAS's for MIN-LINEAR-ARRANGEMENT and MIN-CUT-LINEAR-ARRANGEMENT on dense graphs (the PTAS for BETWEENNESS and MAX-ACYCLIC-SUBGRAPH is similar). Recall that MIN-LINEAR-ARRANGEMENT and its d-dimensional version these did not fit the general framework of Theorem 1.1, so we do a direct reduction to an APEC-like problem. BETWEENNESS and MIN-CUT-LINEAR-ARRANGEMENT do fit the framework, but a direct reduction to an APEC-like problem allows a more efficient algorithm.

4.1 Linear Arrangement

We describe a polynomial-time approximation scheme for this problem on dense graphs. Let $G = (V, E)$ be a $2a$-dense graph. First we notice that the optimum value of the cost function is $\geq a^3n^3/16$. The reason is that at least an of its vertices have degree greater than 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 $a^2n^2/8$, which makes the total edge length at least $(an)^3/16$. Hence to obtain a PTAS it suffices to show how to find layouts in which the cost is within ϵan^3 of the optimal, where $\epsilon > 0$ is arbitrary. For convenience we denote $a\epsilon$ by ϵ in the description below.

Partition the interval $[1, n]$ into $t = c/\epsilon$, equal-sized intervals $I_1, \ldots, I_t$ each of size not greater than n/t. For a number $x \in [1, n]$ denote by $I(x)$ the index of the interval to which x belongs. The following lemma shows that the cost of a permutation is essentially decided by how it places vertices into these intervals.

Lemma 4.1 *Let π_1 and π_2 be two permutations that differ only locally; i.e. for every $1 \leq i \leq n$ $I(\pi_1(i)) = I(\pi_2(i))$. Then $|c(\pi_1) - c(\pi_2)| \leq 2n^3/t$*

Proof. The contribution of each edge to $c(\pi_1)$ and its contribution to $c(\pi_2)$ may differ in at most $2n/t$. ■

Hence to obtain a permutation whose cost is whithin an additive error of ϵn^3 of the minimum cost it suffices to discover a "good" assignment of the vertices to the intervals $I_1, \ldots, I_t$. Let $\hat{g}$ be

an assignment of vertices to intervals; i.e. for every $i \in V$, $g(i) = j$ for some j, $1 \leq j \leq t$. An assignment $\hat{g}$ is *proper* if $|\{i \in V | \hat{g}(i) = j\}| = n/t$ for every $1 \leq j \leq t$. We define the *cost* $c(\hat{g})$ of an assignment $\hat{g}$ as $c(\hat{g}) = \sum_{(i,j)\in E} |\hat{g}(i) - \hat{g}(j)|$. We now describe an algorithm to locate a proper assignment g such that $c(\hat{g}) \leq c(\hat{g}^*) + \delta t n^2$ for any $\delta > 0$, where $\hat{g}^*$ is a proper assignment with minimum cost. This algorithm together with Lemma 4.1 will give us the desired result.

Randomly pick with replacement a set S of $f \log n/\delta^2$ vertices, where f is the degree of the desired probability. Let $g : S \to \{1, \ldots, t\}$ be a function assigning each vertex in S to an interval. (One should think of g as a guess of the location of S according to an optimal permutation π^*.) For each possible g we construct a linear program M_g as follows. We compute an estimate e_{ik} of the cost of assigning vertex i to interval I_k in any complete assignment $\hat{g}$ whose restriction to S is g

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

Note that this estimate is well defined no matter what the value of $\hat{g}(i)$. The accuracy of this estimate is specified by the following lemma.

Lemma 4.2 *Let $\hat{g}$ be an assignment that extends a guess $g : S \to \{1, \ldots, t\}$. With high probability,*

$$\sum_{(i,j)\in E} |\hat{g}(j) - k| - \delta n t \leq e_{ik} \leq \sum_{(i,j)\in E} |\hat{g}(j) - k| + \delta n t.$$

Proof. Let X_i be a random variable that equals $|\hat{g}(j) - k|$ if the ith vertex sampled is j. Divide each X_i by t to scale it to the interval $[0, 1]$ and apply Chernoff bounds. ■

Using the estimates e_{ik}, where $1 \leq i \leq n$, $1 \leq k \leq t$ we write the following linear program M_g.

$$\begin{array}{rrcll}
min \ \sum_{i=1}^{n} \sum_{k=1}^{t} e_{ik} x_{ik} & & & & \\
\sum_{i=1}^{n} x_{ik} & = & n/t & \forall & 1 \leq k \leq t \\
\sum_{k=1}^{t} x_{ik} & = & 1 & \forall & 1 \leq i \leq n \\
\sum_{(i,l)\in E} \sum_{j=1}^{t} |j-k| x_{lj} & \leq & e_{ik} + \delta n t & \forall & 1 \leq i \leq n, 1 \leq k \leq t \\
\sum_{(i,l)\in E} \sum_{j=1}^{t} |j-k| x_{lj} & \geq & e_{ik} - \delta n t & \forall & 1 \leq i \leq n, 1 \leq k \leq t \\
0 \leq & x_{ik} & \leq 1 & \forall & 1 \leq i \leq n, 1 \leq k \leq t
\end{array}$$

We use a linear programmimg algorithm to solve M_g for every possible guess g. Let M_h be the linear program with a minimum solution $\hat{x}^h$ among all the linear programs. We use $\hat{x}_h$ to define an assignment $\hat{f}$ of vertices to intervals as follows. For each vertex i we pick the interval I_k with probability $\hat{x}^h_{ik}$. If the assignment $\hat{f}$ is not proper, then we move some vertices between intervals to obtain a proper assignment $\overline{f}$. The following theorem is proved using the same techniques of Section 2.

Theorem 4.3 *With high probability for large enough n $c(\hat{f}) \leq c(g^*) + c\delta t n^2$.* ■

4.2 Minimum Cut Linear Arrangement

The approach is similar to the one for MIN-LINEAR-ARRANGEMENT. As before, we notice that if a graph is a-dense, then the cost of the optimum assignment is $\Omega(n)$. Then we notice (just as in Lemma 4.1) that it suffices to find an assignment that agrees with the optimum assignment on the placement of vertices inside $O(1)$ intervals.

Lemma 4.4 *Let π_1 and π_2 be two permutation that differ only locally; i.e. for every $1 \leq i \leq n$ $I(\pi_1(i)) = I(\pi_2(i))$. Then $|c(\pi_1) - c(\pi_2)| \leq n^2/t$* ■

The estimates e_{ik} we compute are defined now as

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

The value of e_{ik} estimates the contribution of vertex i to a cut right between intervals I_{k-1} and I_j assuming $I(i) \leq (k-1)$.

The linear program M_g that corresponds to a guess g is now the following.

$$\begin{array}{rcll} min \;\; z & & & \\ \sum_{i=1}^{n} x_{ik} & = & n/t & \forall \;\; 1 \leq k \leq t \\ \sum_{k=1}^{t} x_{ik} & = & 1 & \forall \;\; 1 \leq i \leq n \\ \sum_{i=1}^{n} e_{ik}(x_{i1} + \ldots x_{i(k-1)}) & \leq & z & \forall \;\; 1 < k \leq t \\ \left| \sum_{(i,s) \in E, l \geq k} x_{sl} - e_{ik} \right| & \leq & \delta n & \forall \;\; 1 \leq i \leq n, 1 \leq k \leq t \\ 0 \leq & x_{ik} & \leq 1 & \forall \;\; 1 \leq i \leq n, 1 \leq k \leq t \end{array}$$

References

[AFW94] N. Alon, A. Frieze, and D. Welsh. Polynomial time randomized approximation schemes for the tutte polynomial of dense graphs. In *Proc.* 35^{th} *FOCS*, pages 24–35. IEEE, IEEE Computer Society Press, November 1994.

[A96] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. Manuscript, April 1996.

[AKK95] S. Arora, D. Karger and M. Karpinski. Polynomial-time approximation schemes for dense instances of NP-hard optimization problems. In *Proc. ACM STOC'95.*

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc.* 33^{rd} *FOCS*, pages 14–23. IEEE, October 1992.

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

[dlV94] W.F. de la Vega. MAXCUT has a randomized approximation scheme in dense graphs. manuscript, October 1994. To appear in *Random Structures and Algorithms.*

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

[H64] W. Höffding. *Probability inequalities for sums of bounded random variables*, Journal of the American Stastical Association, 1964.

[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. Karmaker and R.M. Karp. An efficient approximation scheme for the one-dimensional bin-packin g problem. In *Proc.* 23^{rd} *FOCS*, pages 312–320. IEEE, 1982.

[LR88] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc.* 29^{th} *FOCS*, pages 422–431. IEEE, October 1988.

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

[KM96] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proc. ACM STOC 1996*, to appear.

[PY91] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *JCSS*, 43:425–440, 1991. Preliminary Version in Proc. ACM STOC, 1988.

[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, 1981.

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

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

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

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

[Y85] M. Yannakakis. A polynomial algorithm for the min cut linear arrangement problem on trees. *JACM*, **32**: pp 950-988, 1985.

Appendix: The general rounding procedure

We now describe the general rounding procedure for APEC (when the linear constraints are not required to be c-smooth). In Section 5 we describe the algorithm and analyze it assuming that the coefficients are all nonnegative. Section 6 indicates the changes in the analysis when the coefficients could be positive or negative. Section 7 gives the details of the tail bounds we use to derive the results.

The following notational conventions will be useful. We will use the letters c and c_1 to denote fixed constants that do not depend on n but on the degree of the desired probability. These constants may take different values in each place they occur unless we explicity link between them. One should think of any other constant as nondecreasing functions of n taking values that are at least one. We will add a subscript n to such a constant whenever we need to emphasis the dependency on n.

5 The Algorithm

In this section we describe our (randomized) algorithm and analyze it for the case in which all the coefficients are nonnegative. The approach is as the one presented in Section 2. We first oversample a multigraph based on the optimal fractional solution. Then we partition the multigraph into matchings. Last we merge the matchings into one matching whose weight is close to the weight of the fractional solution we have started with.

5.1 Sampling a multigraph

Let x^* be the optimal (fractional) solution to (2). As in Section 2 we oversample to construct a multigraph $G^* = (V_1, V_2, E^*)$: For each edge $(i,j) \in E$, toss a biased coin L times, where the coin is biased to come up "Heads" with probability x^*_{ij}. Suppose the coin came up "Heads" ξ_{ij} times. Then E^* contains ξ_{ij} copies of the edge (i,j). Let X^k_{ij} be a random variable taking the value one if the result of the kth coin toss for edge (i,j) is "Heads" and zero otherwise. Here assume $L \geq c\gamma \log^3 n$. Recall that c is a constant that depends on the desired probability and $\gamma = \gamma_n$ is a nondecreasing function of n. First we prove a structural property of G^*.

Lemma 5.1 *With high probability the degree $d_{G^*}(v)$ of any vertex $v \in V_1 \cup V_2$ satisfies $|d_{G^*}(v) - L| < L/(\gamma^{1/2} \log n)$.*

Proof. Assume w.l.o.g. that $v \in V_1$. Note that $d_{G^*}(v) = \sum_{(v,w)\in E} \sum_{k=1}^{L} X^k_{vw}$ is a random variable with mean L. To obtain the claim we apply Lemma 7.1(b) with $\lambda = \sqrt{c \log n}$ to this random variable (c here is the same constant we used to lower bound L). ■

Let $\Delta = \max_{v \in V_1 \cup V_2} d_{G^*}(v)$. Lemma 5.1 implies that $|\Delta - L| \leq L/(\gamma^{1/2} \log n)$ and that G^* is "almost" regular. In order to make it regular one can add no more than $O(Ln/(\gamma^{1/2} \log n))$ edges that will even out all degrees to be Δ.

Here w denote a weight function defined on the edges of G such that $0 \leq w_{ij} \leq B$ for every $(i,j) \in E$. For two vectors x and y we denote their dot product by xy. Note that $E(W) = Lwx^*$. The following lemma shows that the total weight of the edges in G^* averaged over L would be close to the weight of x^*.

Lemma 5.2 *With high probability, $|w\xi/L - wx^*| = O(wx^*/(\gamma^{1/2} \log n) + \gamma B \log^3 n/L)$.*

Proof. Define the following random variables $W^k_{ij} = w_{ij} X^k_{ij}$ for every $(i,j) \in E$, and $W = \sum_{(i,j)\in E} \sum_{k=1}^{L} W^k_{ij}$. Apply Lemma 7.3 to these variables and divide by L. ■

Combining Lemma 5.1 and Lemma 5.2 we obtain the following.

Lemma 5.3 *With high probability, $|w\xi/\Delta - wx^*| = O(wx^*/(\gamma^{1/2} \log n) + \gamma B \log^3 n/L)$.* ■

5.2 Random merging of matchings

Let M_1 and M_2 be two matchings in G given by m_{ij}^1 and m_{ij}^2 respectively. Next, we show how to randomly construct a matching M_3 such that with high probability $w(M_3)$ is "close" to $(w(M_1) + w(M_2))/2$.

M_3 is constructed as follows. First, we put into M_3 every edge in $(M_1 \cap M_2)$, every edge in M_1 that does not share a vertex with any edge in M_2, and every edge in M_2 that does not share a vertex with any edge in M_1. This leaves a union of vertex disjoint alternating paths and cycles denoted by $\mathcal{P}$. Let $k = k_n$ denote a growing function of n to be determined later. Partition the paths and cycles of $\mathcal{P}$ into two sets. One contains those paths/cycles whose length is greater than k and denoted by LP (Long Paths). The other contains the rest of the cycles and denoted by SP (Short Paths). Denote by $V(LP)$ the vertices that occur on cycles in LP and let $q = |V(LP)|$. We randomly pick with repetitions $l = \lfloor q/k \rfloor$ vertices from $V(LP)$ and denote the sampled subset by S. Then we partition each path or cycle $C \in LP$ such that $|C \cap S| \geq 2$ into a set of $|C \cap S|$ paths by breaking it at the vertices in $C \cap S$. Let LP' be the set of paths and cycles obtained after partitioning LP. Note that the paths in LP' are vertex disjoint except possibly for their endpoints. Let $\Psi = LP' \cup SP$. Construct a graph Ψ_s on $V(\mathcal{P})$ as follows. For each $P \in \Psi$ include in Ψ_s either the edges in $P \cap M_1$ or the edges in $P \cap M_2$ with equal probability. Every vertex in Ψ_s has degree at most two. Moreover, the only vertices that could have degree two in Ψ_s are those in S. The construction of M_3 is completed by adding to it the edges in Ψ_s which are not incident with vertices in S and a maximal subset of the edges incident with vertices in S such that M_3 remains a matching.

For a set of vertices X denote by $w(X, M_1)$ and $w(X, M_2)$ the total weight of edges from M_1 and M_2 repectively incident to at least one vertex in X. For a vertex v denote by $w(v, M_i) = w(\{v\}, M_i)$, $i = 1, 2$. The following lemma upper bounds $w(S, M_i)$, $i = 1, 2$.

Lemma 5.4 *With high probability,* $|w(S, M_i) - w(M_i)/k| = O(w(M_i)/(\gamma^{1/2} \log n) + \gamma B \log^3 n)$ *for* $i = 1, 2$.

Proof. Assume $i = 1$. Let X_i, $1 \leq i \leq l$ be a random variable representing the ith choice of a vertex from $V(LP)$. The variable X_i takes each one of the values $\{w_1(v) \mid v \in V(LP)\}$ with probability $1/q$ hence $|X_i| \leq B$. Let $S = \sum_{i=1}^{l} X_i$. Note that $E(S) = l\, w(V(LP), M_1)/q$. We upper bound Y by applying Lemma 7.3 to these random variables. ■

The following lemma upper bounds the length of the paths in Ψ.

Lemma 5.5 *With high probability, the length of any path/cycle in* Ψ *is bounded by* $ck \log n$.

Proof. Let P be path of length $ck \log n$ in a cycle of LP. The probability that none of the vertices on P is in S is $1 - (1 - (ck \log n)/q)^l$ that is greater than $1 - e^{-c \log n}$. ■

For a set of edges A denote by $w(A)$ the sum of the weights of the edges in A. The following lemma shows that the total weight of M_1 edges in Ψ_s is about half the weight of the edges in $(M_1 - M_2) \cap \mathcal{P}$ and the total weight of M_2 edges in Ψ_s is about half the weight of the edges in $(M_2 - M_1) \cap \mathcal{P}$. Hence the total weight of edges in Ψ_s is about half the weight of the edges on the paths/cycles in $(M_1 \oplus M_2) \cap \mathcal{P}$.

Lemma 5.6 *Let $A_i = E(\Psi_s) \cap M_i$, $i = 1, 2$. With high probability,*

$$\begin{aligned} |w(A_1) - w((M_1 - M_2) \cap \mathcal{P})/2| &= O\left(\frac{w((M_1 - M_2) \cap \mathcal{P})}{\gamma^{1/2} \log n} + k\gamma B \log^4 n\right) \\ |w(A_2) - w((M_2 - M_1) \cap \mathcal{P})/2| &= O\left(\frac{w((M_2 - M_1) \cap \mathcal{P})}{\gamma^{1/2} \log n} + k\gamma B \log^4 n\right). \end{aligned}$$

Proof. We prove the first part of the lemma the proof of the second part is analogous. We associate a random variable X_i with each path $P_i \in \Psi$. The variable X_i equals either 0 or the sum of the weights of the M_1 edges on P_i with equal probabilities. Using Lemma 5.5 we obtain that w.h.p. $|X_i| \leq ckB \log n$. Let $S = \sum_{i=1}^{|\Psi|} X_i$ and apply Lemma 7.3. Note that $E(Y) = w((M_1 - M_2) \cap \mathcal{P})/2$. ■

Combining the errors estimated by Lemma 5.4 and Lemma 5.6 we obtain the following

Lemma 5.7 *Let M_3 be the matching constructed from M_1 and M_2 as described above then with high probability*

$$|w(M_3) - (w(M_1) + w(M_2))/2| = O((1/k + 1/(\gamma^{1/2}) \log n)(w(M_1) + w(M_2)) + k\gamma B \log^4 n).$$

Proof. Let Y be the set of edges in Ψ_s that were not added to M_3 due to conflicts at the vertices in S. Clearly $w(Y) < w(S, M_1) + w(S, M_2)$. Note that $|w(M_3) - (w(M_1) + w(M_2))/2| \leq |w(E(\Psi_s)) - w(Y) - w((M_1 \oplus M_2) \cap \mathcal{P})/2|$. The claim follows using the bounds in Lemma 5.4 and 5.6. ■

5.3 From the Multigraph to a Matching

Now we turn again to the multigraph G^* constructed in Section 5.1 and show how to change it into a matching that is approximately correct. Recall that according to Lemma 5.1 G^* can be augmented to be regular by adding at most $2nL/(\gamma^{1/2} \log n)$ edges (Note that these edges may not be edges in G). The new edges constitute a $O(1/(\gamma^{1/2} \log n))$ fraction of the total number of edges. Recall that we denote the degree of the augmented G^* by Δ.

A Δ-regular mutigraph can be decomposed into Δ perfect matchings [LPV81]. After decomposing the augmented G^* one could discard the new edges and remain with Δ matchings (not necessarily perfect). Denote these matchings $M_1, M_2, \ldots, M_\Delta$.

Assume w.l.o.g. that Δ is a power of 2. Combine the Δ matchings into one according to the following procedure. Pair up the D matchings. Merge each pair using the algorithm in Section 5.2. Then pair up the $D/2$ matchings thus obtained, and merge each pair. Continue this way and end up with one matching M, output M. Let w be a weight function on the edges of G, and x^* the matching used to produce G^*. The following theorem shows that the weight of M is close to the weight of x^*. Recall that ξ_{ij} is the multiplicity of the edge $(i, j) \in E$ in G^* before augmenting it to be regular. We denoted by Δ the degree of the augmented G^* hence Δ is also the number of matchings we merge. Note also that $w^T \xi/\Delta = \sum_{i=1}^{\Delta} w(M_i)/\Delta$.

Lemma 5.8 *Let $t = t_n = 1/k + 1/(\gamma^{1/2} \log n)$ and $\epsilon = \epsilon_n = t \log \Delta$. With high probability, $|w(M) - w^T x^*| = O(\epsilon w^T x^* + (1 + \epsilon)\Delta k\gamma B \log^4 n)$.*

Proof. Let $M_1^j, \ldots, M_{\Delta/2^j}^j$ the $\Delta/2^j$ matchings obtained at stage j of the merging procedure where $0 \leq j \leq \log \Delta$. By induction on j using Lemma 5.7 one can prove that

$$(1-t)^j \left(\frac{w^t \xi}{\Delta} - \left(1 - \frac{1}{2^j}\right) \Delta c k \gamma B \log^4 n \right) \leq \frac{\sum_{i=1}^{\Delta/2^j} w(M_i^j)}{\Delta/2^j} \leq (1+t)^j \left(\frac{w^t \xi}{\Delta} + \left(1 - \frac{1}{2^j}\right) \Delta c k \gamma B \log^4 n \right)$$

The statement follows from this inequality for $j = \log \Delta$ together with Lemma 5.3. ■

Note that ϵ_n specified in the previous lemma goes to zero when n goes to infinity when $k = \Omega(\log n)$ and $\Delta = O(\log^a n)$ for some a. The latter requirement could be restated as $L = O(\log^a n)$ since $\Delta = O(L)$.

Lemma 5.9 *With high probability, the number of edges in M is at least $n - o(n)$.*

Proof. Consider a weight function w such that $w_{ij} = 1$ for every $(i,j) \in E$. Apply Lemma 5.8 to this weight function. ■

We summarize with the following theorem whose proof now is immediate.

Theorem 5.10 *Given a fractional solution to the linear program (2) with all coefficients nonnegative. With high probability the rounding procedure we described gives a matching with at least $n - o(n)$ edges that satisfies (5).* ■

6 Positive and Negative Coefficients

In this section we assume that w is a weight function such that $|w_{ij}| \leq B$. The algorithm we apply in this case is identical to the one we described in the previous section. The required changes in the analysis are as follows. Lemma 5.2 changes to

Lemma 6.1 *With high probability, $|w\xi/L - wx^*| = \tilde{O}(n^{1/2} L^{-1/2} B)$.*

Proof. Define the following random variables $W_{ij}^k = w_{ij} X_{ij}^k$ for every $(i,j) \in E$, and $W = \sum_{(i,j)\in E} \sum_{k=1}^{L} W_{ij}^k$. We scale the variables W_{ij}^k by B. Then we apply Lemma 7.2. (We need 7.2 here since W is a sum of a large number of random variables ($n^2 L$) but after scaling by B its variance is relatively small ($\leq nL$).) ■

Hence Lemma 5.3 changes to

Lemma 6.2 *With high probability, $|w\xi/\Delta - wx^*| = \tilde{O}(n^{1/2} L^{-1/2} B)$.* ■

Using Lemma 7.4 we can obtain the following lemma that replaces Lemma 5.4

Lemma 6.3 *With high probability, $|w(S, M_i) - w(M_i)/k| = \tilde{O}(l^{1/2} B)$ for $i = 1, 2$.* ■

Similarly we replace Lemma 5.6 and Lemma 5.7 with the following

Lemma 6.4 *Let $A_i = E(\Psi_s) \cap M_i$, $i = 1, 2$. With high probability,*

$$\begin{aligned} |w(A_1) - w((M_1 - M_2) \cap \mathcal{P})/2| &= \tilde{O}(l^{1/2} k B) \\ |w(A_2) - w((M_2 - M_1) \cap \mathcal{P})/2| &= \tilde{O}(l^{1/2} k B) \end{aligned}$$

■

Lemma 6.5 *Let M_3 be the matching constructed from M_1 and M_2 as described above then with high probability*

$$|w(M_3) - (w(M_1) + w(M_2))/2| = \tilde{O}((w(M_1) + w(M_2))/k + l^{1/2}kB).$$

■

Finally using the same method as in Section 5.3 we obtain

Theorem 6.6 *Given a fractional solution to the linear program (2). With high probability the rounding procedure we described gives a matching with at least $n - o(n)$ edges that satisfies (3).* ■

7 Chernoff Bounds

We use the following Chernoff-like bounds for independent bounded random variables that could be derived from Hoeffding [](Theorem 1) (see also []).

Lemma 7.1 *Let X_i, $1 \leq i \leq k$ be independent random variables such that $0 \leq X_i \leq 1$. Let $S = \sum_{i=1}^{k} X_i$ and $\mu = E(S)$ then*

$$\begin{array}{lllll} (a) & \mathbf{Pr}[S - \mu \geq \lambda\mu] & \leq & (e/(1+\lambda))^{(1+\lambda)\mu}e^{-\mu} & for \quad \lambda \geq 0. \\ (b) & \mathbf{Pr}\left[|S - \mu| \geq \lambda\sqrt{\mu}\right] & \leq & 2e^{-\lambda^2/3} & for \quad \lambda \leq \sqrt{\mu}. \\ (c) & \mathbf{Pr}\left[|S - \mu| \geq \lambda\right] & \leq & 2e^{-2\lambda^2/k} & for \quad \lambda \geq 0 \end{array}$$

■

We will also need the following lemma

Lemma 7.2 *Let X_i, $1 \leq i \leq k$ be independent random variables such that $|X_i| \leq 2$. Let $S = \sum_{i=1}^{k} X_i$, $\mu = E(S)$ and $\sigma^2 = variance(S)$ then*

$$\mathbf{Pr}[|S - \mu| \geq \lambda\sigma] \leq 2e^{\lambda^2/4}$$

whenever $\lambda \leq \sigma$ ■

We use the bounds in Lemma 7.1(a) and (b) to prove the following technical lemma.

Lemma 7.3 *Let X_i, $1 \leq i \leq n$ be independent random variables with $0 \leq X_i \leq U$ and let $S = \sum_{i=1}^{n} X_i$ and $\mu = E(S)$. Then with high probability $|S - \mu| \leq \max\{\mu/(\gamma^{1/2}\log n), c\gamma U \log^3 n\}$ for any $\gamma = \gamma_n \geq 1$.*

Proof. Let $X_i' = X_i/U$, $1 \leq i \leq n$, $S' = \sum_{i=1}^{n} X_i'$ and $\mu' = E(S')$. Clearly $0 \leq X_i' \leq 1$, we now use Lemma 7.1 as follows. If $\mu' \leq c_1\gamma\log^3 n$ then using Lemma 7.1(a) with $\lambda = c_1^2\gamma\log^3 n/\mu'$ we obtain that w.h.p. $|S' - \mu'| \leq c_1^2\gamma\log^2 n$. Multiplying the last inequality by U we obtain the claim. If $\mu' \geq c_1\gamma\log^3 n$ then using Lemma 7.1(b) with $\lambda = \sqrt{c_1\log n}$ we obtain that w.h.p. $|S' - \mu'| \leq \sqrt{c_1(\log n)\mu'}$. Using our assumption that $\mu' \geq c_1\gamma\log^3 n$ the last inequality implies that $|S' - \mu'| \leq \mu'/(\gamma^{1/2}\log n)$. Multiplying the last inequality by U we obtain the claim. ■

If some of the variables are positive and others are negative we get the following

Lemma 7.4 *Let X_i, $1 \le i \le n$ be independent random variables with $0 \le X_i \le U$ for $i \in I$ and $-U \le X_i \le 0$ for $i \notin I$ for some $I \subseteq \{1, 2, \ldots, n\}$. Let $S = \sum_{i=1}^{n} X_i$ and $\mu = E(S)$. Also let $\hat{\mu} = E(\sum_{i=1}^{n} |X_i|)$ Then with high probability $|S - \mu| \le c \max\{(\hat{\mu} U \log n)^{1/2}, U \log^2 n\} = \tilde{O}(n^{1/2} U)$*

Proof. Let $X_i' = X_i/U$ for $i \in I$ and $X_i' = -X_i/U$ for $i \notin I$. Clearly $0 \le X_i' \le 1$. Let $S_1' = \sum_{i \in I} X_i'$ and $S_2' = \sum_{i \notin I} X_i'$. Let $\mu_i' = E(S_i')$ for $i = 1, 2$. We now use Lemma 7.1 as follows. Fix $i = 1$ or 2. If $\mu_i' \le c_1 \log^2 n$ then using Lemma 7.1(a) with $\lambda = c_1^2 \log^2 n / \mu'$ we obtain that w.h.p.

$$|S_i' - \mu_i'| \le c_1^2 \log^2 n. \tag{8}$$

If $\mu_i' \ge c_1 \log^2 n$ then using Lemma 7.1(b) with $\lambda = \sqrt{c_1 \log n}$ we obtain that w.h.p.

$$|S_i' - \mu_i'| \le \sqrt{c_1 (\log n) \mu_i'} \tag{9}$$

Now let $S_i = U S_i'$ and $\mu_i = U \mu_i'$ for $i = 1.2$. Multiplying the inequality (8) or (9) by U we see that w.h.p.

$$|S_i - \mu_i| \le c \max\{(\mu_i U \log n)^{1/2}, U \log^2 n\}.$$

Now $S = S_1 - S_2$ and $\mu = \mu_1 - \mu_2$ and so w.h.p.

$$|S - \mu| \le c \max\{(\log n)^{1/2} U (\sqrt{\mu_1} + \sqrt{\mu_2}), U \log^2 n\}.$$

The lemma now follows from the fact that $\sqrt{\mu_1} + \sqrt{\mu_2} \le \sqrt{2(\mu_1 + \mu_2)} = \sqrt{2\hat{\mu}}$. ■