Approximation Algorithms and Hardness of Approximation March 5, 2013

Lecture 4-5

Lecturer: Ola Svensson *Scribes: Carsten Moldenhauer*

1 Euclidean TSP

In this lecture we are considering the traveling salesman problem (TSP) again. We already know that no approximation is possible if the distances between cities can be arbitrary. If the distances are metric, we have seen a 2-approximation algorithm by doubling a spanning tree and also heard that a $\frac{3}{2}$-approximation via Christofides' Algorithm [Chr76] in the previous classes. This is the currently best known algorithm for this problem. In fact, it is known that there cannot be a polynomial time approximation scheme (PTAS) for metric TSP (the precise statement is: there cannot be a 220/219-approximation unless P=NP [PV00])

Hence, we have to restrict the input to the problem to gain better algorithms. In this lecture, we will consider euclidean TSP. The problem is as follows

Euclidean TSP:
Given: n points in $\mathbb{R}^2$ with euclidean distances, *i.e.*, $d(x, y) = \|x - y\|_2$.
Find: shortest tour that visits all points.

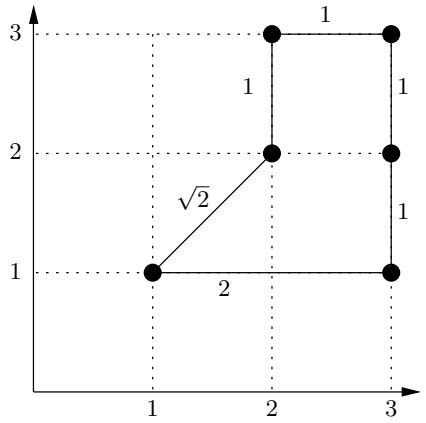

Figure 1: An example of an instance for euclidean TSP with a sample tour of cost $6 + \sqrt{2}$.

An example of an euclidean TSP instance can be found in Figure 1. In this example we can see that the length of a tour might be irrational. An implication of this is that although we know that euclidean TSP is NP-hard we do not know if it is in NP. The reason being that we do not know how to calculate square roots efficiently.

Nevertheless we will derive a PTAS for this problem by employing the following techniques

- rounding the instance
- exploiting the structure of the problem by partitioning the space into squares
- applying dynamic programming to this partition.

Our exposition follows the method first presented by Arora [Aro98] (similar result was independently discovered by Mitchell [Mit99] and both were rewarded the Gödel prize for their discoveries).

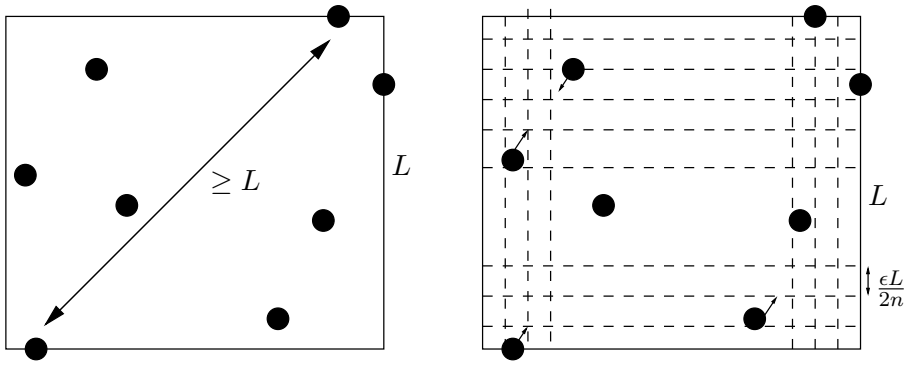

Figure 2: Illustration of the bounding box with length L and the scaled box with fine grid

ϵ-nice instances

Let us first describe how we can transfer an arbitrary instance of euclidean TSP to so-called nice instances.

Definition 1 (ϵ-nice instances) *An instance of euclidean TSP is called* ϵ-nice *if the following two conditions hold.*

1. *Every point has integral coordinates in the interval* $[0, O(\frac{n}{\epsilon})]^2$.
2. *Any two different points have distances at least* 4.

Note that without loss of generality we can translate and scale an input instance as we like, *i.e.*, translate all points by the same vector and scale the coordinates of each point by the same factor. This will change the cost of all tours in the same way since all distances scale by the same factor. But, the approximation ratio is preserved.

In particular, consider the smallest bounding box around the points of the input instance (axis parallel). Let L be the length of the longer side of this box. Without loss of generality we can translate the box such that it is rooted at the origin and scale it such that

$$L = \left\lceil \frac{8n}{\epsilon} \right\rceil$$

Furthermore, for the goal of obtaining a PTAS we can assume that the input instance is an ϵ-nice instance which is the intention of the next lemma.

Lemma 2 *Let I be an arbitrary instance to euclidean TSP. Let* OPT_I *denote the length of the optimum tour in I. We can transform I into an ϵ-nice instance I' such that* $\mathrm{OPT}_{I'} \leq (1+\epsilon)\mathrm{OPT}_I$.

Proof Consider the smallest bounding box around the points of instance I (see Figure 2 for an illustration). The longer of its two sides has length L. Since the box was the smallest bounding box there exist two points, on opposite sides of the box, that are at a distance at least L. Hence, the optimum tour is of size at least L, *i.e.*, $\mathrm{OPT}_I \geq L$. (In fact, we have $OPT_I \geq 2L$ since the tour has to travel forth and back between those two points).

For ease of exposition extend the bounding box such that both sides have length L. Now, to obtain instance I' we draw a fine grid with spacing $\frac{\epsilon L}{2n}$ into the bounding box and map each point to its closest grid point. In fact, you can choose your favorite mapping to the grid points, *e.g.*, truncation or mapping to the closest grid point. However, note that some points in I might map to the same point in I'.

Let us verify that the new instance is indeed ϵ-nice. Clearly, all points lie on integer coordinates in the respective range since $L = \lceil 8n/\epsilon \rceil \in O(n/\epsilon)$. Furthermore the grid spacing is $\frac{\epsilon L}{2n} \geq \frac{\epsilon}{2n}\frac{8n}{\epsilon} = 4$ and therefore the distance between any two different points in I' is at least 4. Hence, I' is ϵ-nice.

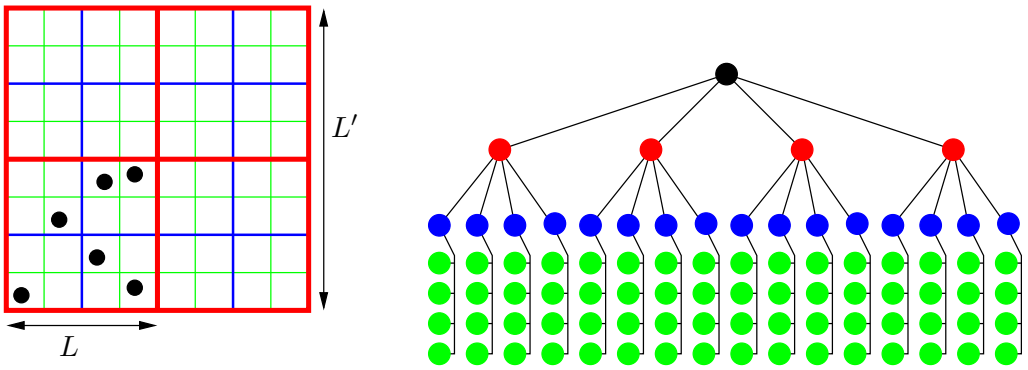

Figure 3: Dissection of the bounding square and the associated quadtree.

It remains to prove that the optimal tours in I and I' differ by at most a factor of $(1+\epsilon)$. Therefore, consider the optimum tour in I. By mapping the points of I to the points in I' we moved every point by at most a distance of $\frac{\epsilon L}{2n}$. Thus, every edge between two points changed by at most $\frac{\epsilon L}{n}$ (these edges are of length 0 if two points in I mapped to the same point in I'). Since there are n edges in any tour, the change in cost is at most ϵL, *i.e.*, we obtained a tour for I' of cost at most $\mathrm{OPT}_I + \epsilon L$. Using that $L \leq \mathrm{OPT}_I$ we obtain

$$\mathrm{OPT}_{I'} \leq \mathrm{OPT}_I + \epsilon L \leq (1+\epsilon)\mathrm{OPT}_I.$$

■

We proceed with the description of the algorithm assuming that the input instance is appropriately scaled and ϵ-nice.

Dissection and quadtree

Extend the bounding box with side length L to a square of side length L and create a new square with side length L' where L' is the smallest power of 2 such that $L' \geq 2L$ (for ease of exposition think of $L' = 2L$). Then, we create a dissection of the new square with a quadtree (see Figure 3). We recursively partition a square into four equal sized squares until the side length of a square is one (note that L' was selected to be a power of 2). In particular, this means there is a line from a square at every integer offset. Note that since the minimum distance between two points is 4 the one-by-one squares at the end of the recursion can contain at most one point.
This induces a quadtree (see Figure 3 for an example) and we will refer to the nodes in this tree as their respective squares in the dissection. The level 0 bounding square is the square of length L', the level 1 squares are all squares of length $L'/2$ and so forth. Since all the points are positioned on integer coordinates and have distance at least 4 from each other we know that the dissection terminates after $O(\log L')$ steps. Hence, the quadtree has height $O(\log L') = O(\log \frac{n}{\epsilon})$. Let k be this height.

The intuition behind the subsequent steps is that we would like to apply dynamic programming to the quadtree, *i.e.*, solve each square that is a leave in the quadtree individually, and then bottom-up combine the solutions to obtain the solutions for squares higher in the quadtree.

Portals

In order to apply dynamic programming, we first consider the problem that an optimum tour can cross the squares at arbitrary places. This will destroy any dynamic programming approach and hence we would like to restrict ourselves to tours that only cross the squares at so-called *portals*. The number

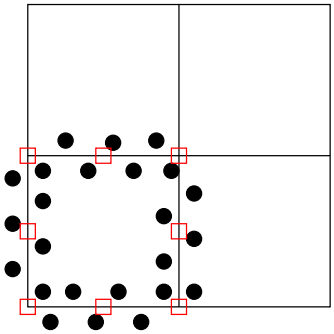

Figure 4: A bad example that shows that the optimum tour and a shortest p-tour can be significantly different.

of portals will influence both, the accuracy of our obtained solution as well as the running time of the dynamic program (the more portals, the better the solution and the slower the algorithm).

Select m to be a power of 2 such that $m \in [\frac{k}{\epsilon}, \frac{2k}{\epsilon}]$. For each square, put a portal into each of the four corners and put $m-1$ equally spaced portals along each side. Thus, each square obtains $4m$ portals. Note that since m is a power of 2 the portals of a square are a subset of the portals of its children.

Definition 3 *A* portal respecting tour *(*p-tour*) is a tour that enters and exists each square only through its portals.*

Now, we would like to relate the optimum tour with the optimum p-tour such that the length of these tours do not differ by a factor larger than $(1+\epsilon)$. However, this is not possible. Consider the example in Figure 4. The red squares indicate the portals. An optimum tour would cross the boundary of the squares many times by "zigzagging". A shortest p-tour however can only cross the boundary of the squares at the portals and therefore has to accept many detours of high cost.

This is where randomization comes into play. We can translate the grid (or alternatively the points themselves) by a random offset of at most $L'/2$ in each coordinate. Then, the points are still guaranteed to stay within the grid, *i.e.*, within the bounding square of length L'. And, we have that with high probability the points are not concentrated around the bounds of squares of low level (which is the problem in the above example)! Higher levels in the dissection have more portals and hence allow for more fine-grained tours. We formalize this in the following

Definition 4 *A* (a,b)-dissection *is the dissection where the origin of the grid is translated by* $(-a,-b)$.

Theorem 5 *Let* a, b *be two integers picked uniformly at random from* $[0, \frac{L'}{2})$. *Then with probability at least* $\frac{1}{2}$ *the* (a,b)*-dissection has a p-tour of cost at most* $(1+4\epsilon)\mathrm{OPT}$*, where* OPT *denotes the length of the optimum tour.*

Proof The general idea is that we massage the optimum tour into a p-tour. That means that we extend the tour whenever it crosses a square so that the new tour respects the portals.

For every vertical or horizontal line ℓ let $t(\ell)$ denote the number of times the optimum tour crosses ℓ. Then, $T = \sum_\ell t(\ell)$ is the number of total crossings.

Claim 6 $T \leq 2\,\mathrm{OPT}$

Let $e = (u,v)$ be an edge of the optimum tour and (x_u, y_u), (x_v, y_v) be the coordinates of u and v, respectively. Let $x = |x_u - x_v|$ and $y = |y_u - y_v|$. Now, e crosses at most $x+1$ vertical and $y+1$ horizontal lines (all lines of the squares are at integer offsets). Hence, e's contribution to T is at most

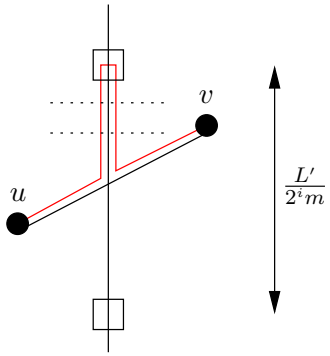

Figure 5: A crossing of a line with an edge of the optimal tour and the new connection (in red).

$x+y+2$. However, e's contribution to OPT is $\sqrt{x^2+y^2}$. Using the inequality $\sqrt{2(a^2+b^2)} \geq a+b$ and that $x, y \geq 4$ (ϵ-niceness) we have

$$x+y+2 \leq \sqrt{2(x^2+y^2)}+2 \leq 2\sqrt{x^2+y^2}.$$

Note that the right hand side is two times e's contribution to OPT which proves the claim.

We continue with the proof of the theorem. We know that the number of crossings is small. Now, we would like to bound the expected length of a detour to respect the portals. Consider a single crossing of a (vertical) line ℓ (see Figure 5). Our p-tour will follow the original edge from u up to the crossing, then along ℓ to the *nearest* portal, back to the crossing along ℓ, and then following the original edge again to v. Note that this detour (hence our connection from u to v) might cross more than $|x_u - x_v| + |y_u - y_v| + 2$ lines since there might be many more lines that are crossed on the part of the detour that lies on ℓ. But, these crossings are (already) at portals (of higher level squares) and hence these crossings do not contribute anything to the prolongation of the p-tour.

The spacing on level i is $\frac{L'}{2^i m}$. Hence, if ℓ is in level i, the detour is of size at most $\frac{L'}{2^i m}$ since the detour goes from the crossing to the nearest portal (forth and back). Now, what is the probability that after the random translation of the grid, the line ℓ that e crosses is at level i. There are $L'/2$ many possible lines that ℓ could be mapped to (we translate by an integer in $[0, L'/2)$). Furthermore, there are at most 2^{i-1} many possible lines of level i that ℓ could be mapped to (in distance less than $L'/2$). Hence, this probability is at most $\frac{2^{i-1}}{L'/2} = \frac{2^i}{L'}$.

Thus, we can now compute an upper bound on the expected length increase of the detour we introduced to circumvent line ℓ:

$$\sum_{i=1}^{k} \frac{L'}{2^i m} \frac{2^i}{L'} = \frac{k}{m} \leq \epsilon.$$

By linearity of expectation and using the above claim we get that the total increase for all detours is at most $2\epsilon\mathrm{OPT}$ in expectation. As a last step, we use Markov's inequality

$$\Pr[\text{total length increase of detours} \geq 4\epsilon\mathrm{OPT}] \leq \frac{2\epsilon\mathrm{OPT}}{4\epsilon\mathrm{OPT}} = \frac{1}{2},$$

which completes the proof. ■

This shows that with reasonable probability the optimal p-tour in the (a,b)-dissection is only an $O(\epsilon)$ factor longer than the optimal tour. Hence, we could simply repeat to assure success of the algorithm with high probability. However, note that a and b are chosen randomly to be integers in the interval $[0, L'/2)$ and that $L' = O\left(\frac{n}{\epsilon}\right)$. Hence, for fixed ϵ we can derandomize the grid shifting by trying

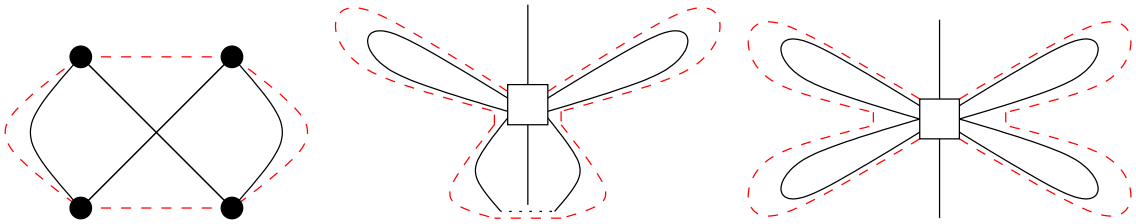

Figure 6: Shortcutting of a p-tour that is not well-behaved or 2-light.

all possibilities for a and b. This will only require $O\left(\frac{n^2}{\epsilon^2}\right)$ time, which is polynomial in n. By the probabilistic method, we know that there is one value for (a, b) where the length of the optimal p-tour is at most $(1 + 4\epsilon)$ times the length of the optimal tour, and by trying out all possible values for (a, b) we are guaranteed to find this p-tour.

Dynamic Programming

What remains to show is how to find an optimal p-tour given a fixed (a, b)-dissection. We will do this by employing dynamic programming. In our dynamic program, we introduce one state for any combination of

- a square; and
- any set of possibible ways of entering and exiting this square.

Why are there only polynomially many states in the dynamic program? First, there are only polynomially many squares $(1+4+4^2+\cdots+L'^2 = \frac{1}{3}(4L'^2-1) = O(\frac{n^2}{\epsilon^2}))$. However, bounding the number of possibilities that a tour could enter and exit a square is more tricky. We will do this subsequently.

Definition 7 *A p-tour is* well-behaved *if it does not cross itself except at portals.*
A p-tour is 2-light *if it goes through a portal at most twice, i.e., enters and exits the portal at most twice.*

Lemma 8 *Without loss of generality a p-tour is well-behaved and 2-light, i.e., for any p-tour we can find a p-tour of at most the same length that is well-behaved and 2-light.*

Proof Idea In fact it is quite easy to convince yourself that this statement is true and we will therefore not give a rigorous proof. If a p-tour does cross itself we can shortcut this crossing and by the triangle inequality obtain a tour that is shorter (note that crossings can only appear inside squares and not at portals).
If a portal is used more than twice, we can shortcut the tour on the inside and outside of the portal's square. Consider Figure 6 for illustrations. ■

The lemma implies that we only need to consider well-behaved 2-light p-tours in the dynamic program. Now, how many possible ways of entering and exiting a square exist for such tours? Every square has $4m$ portals on its sides that can each be used by a (portal-to-portal) path that runs inside the square 0, 1 or 2 times. Hence, the number of configurations of possible portal usages is 3^{4m}. In fact, we only need to consider configurations where the sum of all portal usages is even. Note that 3^{4m} is polynomial for fixed ϵ since $m = O(k/\epsilon) = O((\log L')/\epsilon) = O((\log \frac{n}{\epsilon})/\epsilon)$. Hence, we can try all these combinations.

A fixed combination determines the number of possible (portal-to-portal) paths, *i.e.*, half of the sum of all portal usages. Let us assume that there are r paths for a fixed cofiguration. Note that $r \leq 4m$ since there are at most $4m$ paths (each portal can be used at most twice and there are $4m$ portals). How many possible layouts of r paths are there? Since the p-tour is well-behaved, all these

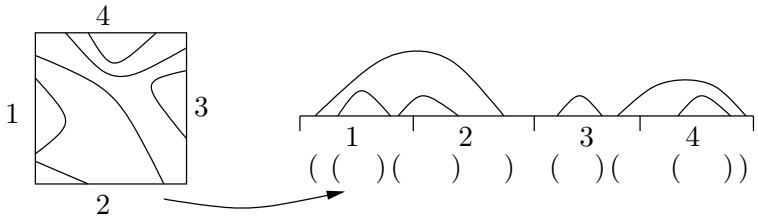

Figure 7: Estimating the number of configurations of r portal-to-portal paths by the rth Catalan number

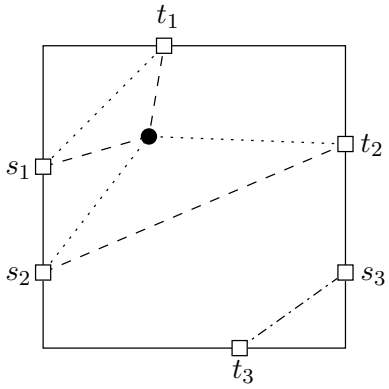

Figure 8: Calculating the value of the table A for the leaves of the quadtree.

paths do not cross. Hence, we can think of these paths as spanning parenthesis. See Figure 7 for an illustration of the idea. This means that we can bound the number of layouts by the rth Catalan number $C_r = \frac{1}{r+1}\binom{2r}{r} = O(2^{2r}) = O(2^{8m})$. From an algorithmic perspective, we can try all settings of parenthesis, translate these into possible layouts of paths and discard the ones that have intersecting paths. In total, we have at most 3^{4m} configurations with $O(2^{8m})$ layouts each. Since $m = O((\log \frac{n}{\epsilon})/\epsilon)$ we have that the number of possible ways of entering and exiting a square is in $O(n^{1/\epsilon})$. It follows, that we have a polynomial number of states in the dynamic program.

It remains to show how the dynamic program computes its values. Let $A[s, (s_1, t_1), \ldots, (s_\ell, t_\ell)]$ be the state that belongs to square s and entry and exit configurations (s_1, t_1) to (s_ℓ, t_ℓ) (the order of these entry and exit points does not matter due to shortcutting arguments). We will fill in the table A in a bottom-up manner. For each point, find the biggest square in the quadtree that contains only this point, say s, and compute $A[s, \ldots]$. Clearly, the single point will lie on only one of the paths from $(s_1, t_1), \ldots, (s_\ell, t_\ell)$ leaving the other ones to straight line distance. We can try all these combinations (see Figure 8) and the cheapest solution sets the value for the corresponding entry in A.

Computing the values of A for a square s knowing all the values for its children in the quadtree can also be done efficiently. Assume you knew through which portals the tour would enter and exit the subsquares of s (consider Figure 9). Then, you could put together the best tour using the values of A for the subsquares (in the example $A[s, (s_1, t_1), (s_2, t_2)] = A[X, (a, t_1)] + A[Y, (b, c)] + A[Z, (s_1, a), (d, b)] + A[Q, (s_2, d), (c, t_2)]$). But, there are only polynomially many possible configurations of entry and exit tuples for each of the four subsquares. Hence, we can try each possible combination and discard the ones where entry and exit portals do not match. Since there are only polynomially many nodes in the quadtree and each computation of the entries in A takes polynomial time, the dynamic program runs in polynomial time. The final solution will then be $A[s]$ (no entry or exit portals) where s is the $L' \times L'$ square.

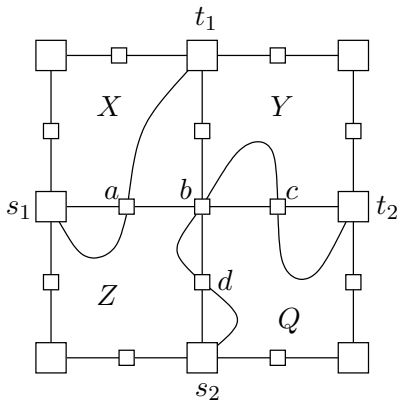

Figure 9: Visualization of the dynamic programming approach.

Remark Note that the computation of the entries of A requires to select the best solution for a given square together with entry and exit portals. However, the cost of such a solution can be irrational, *i.e.*, involve the computation of square roots. However, the value of these square roots can be approximated to any accuracy. Hence, we can compute the values in A to high accuracy and since the height of the quadtree (and hence the recursion in the linear program) is polynomially bounded, we can set this accuracy small enough such that the errors do not accumulate too much.

References

[Aro98] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, September 1998. doi:10.1145/290179.290180.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[Mit99] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. URL: http://epubs.siam.org/doi/abs/10.1137/S0097539796309764, arXiv:http://epubs.siam.org/doi/pdf/10.1137/S0097539796309764, doi:10.1137/S0097539796309764.

[PV00] Christos H. Papadimitriou and Santosh Vempala. On the approximability of the traveling salesman problem (extended abstract). In *Proceedings of the thirty-second annual ACM Symposium on Theory Of Computing*, STOC '00, pages 126–133, New York, NY, USA, 2000. ACM. doi:10.1145/335305.335320.