EFFICIENT SPEAKER SEARCH OVER LARGE POPULATIONS USING KERNELIZED LOCALITY-SENSITIVE HASHING

Woojay Jeon

Samsung Electronics
Suwon, South Korea

Yan-Ming Cheng

Motorola Solutions
Schaumburg, U.S.A.

ABSTRACT

We propose a novel method of efficiently searching very large populations of speakers, tens of thousands or more, using an utterance comparison model proposed in a previous work. The model allows much more efficient comparison of utterances compared to the traditional Gaussian Mixture Model(GMM)-based approach because of its computational simplicity while maintaining high accuracy. Furthermore, efficiency can be drastically improved when approximating searches using kernelized locality-sensitive hashing (KLSH). From a speaker's utterance, a set of statistics are extracted according to the utterance comparison model and converted to a set of hash key bits. An Approximate Nearest Neighbor search using the Hamming Distance can be done to find candidate matches with the query speaker, which are then rank-ordered by linearly comparing them with the query using the utterance comparison model. Compared to GMM-based speaker identification and some of its variants that have been proposed to increase its efficiency, the proposed KLSH-based method is orders of magnitude faster while compromising a negligible amount of accuracy for sufficiently long query utterances. At a more fundamental level, we also discuss how our speaker matching framework differs from the traditional Bayesian decision rule used for speaker identification.

Index Terms— speaker identification, speaker search, kernelized locality-sensitive hashing, lsh

1. INTRODUCTION

Speaker identification is the classification of a given speaker into one out of n known speakers. It is well known that Gaussian Mixture Models (GMMs) allow this to be done with high accuracy under clean recording conditions. When the population n of known speakers is large (at least tens of thousands), however, the amount of computation required to calculate the likelihoods of all n GMMs for a given speech query can be overwhelming [1]. Various strategies have been proposed in the past to make this process faster while minimizing the loss of accuracy, including reducing the number of acoustic features involved in the GMM decoding [2], approximating the likelihood calculations with an approximate cross entropy measure [3], and using speaker model clusters to hierarchically reduce the search space [1]. However, the computational gains of these methods are not enough for population sizes of more than one thousand.

Recently, locality-sensitive hashing (LSH) has been successfully applied as an elegant framework for highly efficient searches over large databases of music, text, and images. In particular, *kernelized* LSH [4] has been proposed for use with distance measures that are more general than the Euclidean distance.

In previous work [5], we proposed an utterance comparison model based on factor analysis and eigenvoices that computes the probability that any two arbitrary speech utterances originated from the same speaker, and showed the effectiveness of the model when used as a distance metric for speaker clustering in a speaker diarizer. In this paper, we will show that the utterance comparison model allows very efficient matching of speakers due to its computational simplicity. Furthermore, we recognize that the utterance comparison model is also a kernel function, and apply the model in a kernelized LSH system to achieve highly efficient approximate speaker identification in matched conditions over large speaker populations with minimal compensation of accuracy for sufficiently long utterances. In addition to the fact that the resulting search is orders of magnitude faster than traditional speaker identification, it also enjoys all the benefits of LSH, including good scalability and ease in adapting to a distributed computing environment.

Although the problem we are attacking is similar to "speaker identification(ID)," we term our problem "speaker search" to emphasize the fact that the matching is done over a large population using information retrieval techniques with explicit consideration for *approximate* matches. Also, the mathematical framework is not the Bayesian decision framework used in speaker ID, and we will discuss how the two methods theoretically relate to each other.

2. UTTERANCE COMPARISON MODEL

In this section, we give a brief description of the utterance comparison model that was proposed in the previous study [5].

Assume two arbitrary speech utterances, X_a of length A and X_b of length B, each utterance defined as a sequence of acoustic feature vectors originating from exactly one speaker.

$$X_a = \{\mathbf{x}_{a,1}, \mathbf{x}_{a,2}, \cdots, \mathbf{x}_{a,A}\}, X_b = \{\mathbf{x}_{b,1}, \mathbf{x}_{b,2}, \cdots, \mathbf{x}_{b,B}\} \quad (1)$$

We begin by defining the hypothesis H_1 that X_a and X_b were uttered by the same speaker. We define the utterance comparison function as the posterior probability of H_1. Assuming we can obtain the posterior $P(w_i|X)$ for every speaker w_i in the world for any given utterance X, an exact formula for the probability can be given:

$$P(H_1|X_a, X_b) = \sum_{i=1}^{W} P(w_i|X_a) P(w_i|X_b) \quad (2)$$

where W is the population of the world. Of course, it is completely impractical to try to directly solve this equation, so we turn to eigenvoice theory and factor analysis, which allow us to approximate the GMM mean "supervector" (the mean vectors for all mixtures stacked onto a single vector) $\mathbf{s}$ of a speaker model as [6]

$$\mathbf{s} = \mathbf{m} + V\mathbf{y}, \;\; \mathbf{y} \sim N[0, I] \quad (3)$$

where $\mathbf{m}$ contains the mean parameters of a universal background model (UBM), V is an eigenvoice matrix, and $\mathbf{y}$ is a speaker factor vector with a unit Gaussian distribution. Now, if we assumed each speaker w_i is mapped to a unique v-dimensional speaker factor vector $\mathbf{y}_i$, the summation in (2) can be rewritten as

$$P(H_1|X_a, X_b) = \sum_{i=1}^{W} P(\mathbf{y}_i|X_a)\, P(\mathbf{y}_i|X_b) \tag{4}$$

This equation is still impractical, but by breaking it down into Riemann summation form and using probability theory, we can mold it into the following analytical form [5]:

$$\begin{aligned} P(H_1|X_a, X_b) &\approx \frac{1}{W}\int_{-\infty}^{\infty} \frac{p(\mathbf{y}|X_a)\, p(\mathbf{y}|X_b)}{p(\mathbf{y})} d\mathbf{y} \\ &= \frac{1}{W}\frac{1}{p(X_a)}\frac{1}{p(X_b)}\int_{-\infty}^{\infty} p(X_a|\mathbf{y})\, p(X_b|\mathbf{y})\, p(\mathbf{y})\, d\mathbf{y} \end{aligned} \tag{5}$$

Using factor analysis, we can derive closed-form solutions for $p(X)$ and $p(X|\mathbf{y})$. For an utterance X with A feature vectors, we have

$$p(X|\mathbf{y}) = \prod_{t=1}^{A} p(\mathbf{x}_t|\mathbf{y}) = \prod_{t=1}^{A}\sum_{k=1}^{M} c_k N(\mathbf{x}_t; \mathbf{m}_k + V_k\mathbf{y}, C_k) \tag{6}$$

where c_k, m_k, and C_k are the weight, mean vector, and covariance matrix, respectively, of the k'th Gaussian in the GMM model admitted by $\mathbf{y}$, M is the total number of Gaussians, and V_k is the submatrix of the eigenvoice matrix V in (3) that corresponds to the k'th Gaussian. Since (6) is too difficult to manage analytically, we assume each observation is "generated" by only one Gaussian, i.e.,

$$p(X|\mathbf{y}) = \prod_{t=1}^{A} N(\mathbf{x}_t; \mathbf{m}_t + V_t\mathbf{y}, C_t) = \prod_{t=1}^{A} N(V_t\mathbf{y}; \mathbf{x}_t - \mathbf{m}_t, C_t) \tag{7}$$

where $\mathbf{m}_t$, V_t, and C_t are the $d \times 1$ mean vector, $d \times v$ eigenvoice matrix, and $d \times d$ covariance matrix pertaining to the Gaussian that "generated" $\mathbf{x}_t$, respectively. There can be a number of ways to decide which Gaussian to use for each $\mathbf{x}_t$. One way is to obtain the speaker factors for X via maximum likelihood estimation using known methods [6], then for each $\mathbf{x}_t$ finding the Gaussian with the maximum "occupation" probability, $\arg\max_m \gamma_m$.

Using this simplified form, it is then possible to obtain closed-form expressions for $p(X)$ and $p(X|\mathbf{y})$, which then result in a closed-form expression for (5):

$$\begin{aligned} P(H_1|X_a, X_b) &= \frac{1}{W}\left(\frac{|J_A|\,|J_B|}{|D|}\right)^{-\frac{1}{2}} \\ &\cdot \exp\left[-\frac{1}{2}\left\{-\mathbf{d}^T D\mathbf{d} + \mathbf{d}_A{}^T J_A \mathbf{d}_A + \mathbf{d}_B{}^T J_B \mathbf{d}_B\right\}\right] \end{aligned} \tag{8}$$

where (omitting D_B, $\mathbf{d}_B$, J_B, for which equivalent expressions can be easily obtained)

$$\begin{aligned} &D_A{}^{-1} = \sum_{t=1}^{A} V_{a,t}{}^T C_{a,t}{}^{-1} V_{a,t}\ ,\ J_A = \left(I + D_A{}^{-1}\right)^{-1} \\ &\mathbf{d}_A = \sum_{t=1}^{A} V_{a,t}{}^T C_{a,t}{}^{-1}(\mathbf{x}_{a,t} - \mathbf{m}_{a,t}) \\ &D = \left(J_A{}^{-1} + J_B{}^{-1} - I\right)^{-1},\ \mathbf{d} = \mathbf{d}_A + \mathbf{d}_B \end{aligned} \tag{9}$$

3. SPEAKER MATCHING USING THE UTTERANCE COMPARISON MODEL

The utterance comparison equation (8) is, by definition, a likelihood of how two given utterances X_a and X_b are from the same speaker. Hence, given an utterance X_q from a "query speaker" and a set $\{X_{t1}, X_{t2}, \cdots, X_{tN}\}$ of utterances from a set of corresponding "target speakers" (or test speakers) $\{z_1, z_2, \cdots, z_N\}$, we can perform a linear search for the best-matching target speaker as follows:

$$\arg\max_{1\le k\le N} P(H_1|X_q, X_{tk}) \tag{10}$$

Before we begin such an endeavor, however, it is necessary to consider how exactly this formulation differs from the well-known Bayesian decision rule for speaker identification:

$$\arg\max_{1\le k\le N} P(z_k|X_q) = \arg\max_{1\le k\le N} p(X_q|z_k)\, P(z_k) \tag{11}$$

where z_k is the k'th target speaker. To see the relation between (10) and (11), let us rewrite the comparison function in (10) using the original definition of the utterance comparison model in (2):

$$P(H_1|X_q, X_{tk}) = \sum_{i=1}^{W} P(w_i|X_q)\, P(w_i|X_{tk}) \tag{12}$$

Note that each speaker w_i in (12) is purportedly one of *all* the speakers in the world, which, in practice, is actually a (much smaller) set of *training speakers* (used to train the UBM and eigenvoice parameters in (9)). In contrast, the speaker z_k in (10) and (11) is a *test speaker* that, in practice, is not present in the set of training speakers.

Now, if we assume that each test speaker z_k also exists in the set of training speakers such that test speaker z_k is the same as training speaker w_k, then we have

$$P(w_i|X_{tk}) = P(z_i|X_{tk}) = \delta(i-k) \tag{13}$$

This is because we already know that z_k generated X_{tk} and therefore $P(w_i|X_{tk})$ must be 1 for $i = k$. We also know $P(w_i|X_{tk})$ is 0 for $i \neq k$ because any $w_i \neq w_k$ will also satisfy $w_i \neq z_k$. Applying (13) in (12), we have

$$P(H_1|X_q, X_{tk}) = P(z_k|X_q) \tag{14}$$

in which case the matching rule in (10) is now exactly equivalent to the Bayesian decision rule in (11).

Conceptually speaking, the matching rule in (10) does not assume any knowledge of which speaker in the world produced the test utterance X_{tk}, but rather, considers all the possible known speakers in the world using prior distributions of speaker parameters in a Bayesian framework to compute a match probability. In contrast, the Bayesian decision rule assumes that the speaker who created X_{tk} has a deterministic set of model parameters and does not care about any speakers in the world that are not part of the set of test speakers. In principle, the matching rule in (10) does not require the estimation of model parameters of the test speakers, whereas the Bayesian decision rule in (11) does. One may argue that the computation of the sufficient statistics in (9) required by the matching rule is a type of model parameter estimation, but fundamentally these sufficient statistics are auxiliary variables for strictly computational purposes, not statistical model parameters for class representation.

4. SPEAKER SEARCH USING KLSH

The matching rule in (10) can be performed fairly quickly because (8) depends only on the set of statistics J_A and $\mathbf{d}_A$ in (9). The matrices $V_k {C_k}^{-1}$ and ${V_k}^T {C_k}^{-1} V_k$ pertaining to the k'th Gaussian can be precomputed offline for $k = 1, 2, \cdots, M$. For each target utterance, we can use ML estimation of the speaker factors using the method in [6] to obtain a "mixture sequence," i.e., a sequence of indices indicating which Gaussian "generated" each feature vector, and then use the precomputed matrices to quickly compute J_A and $\mathbf{d}_A$ in (9), which are all that is needed to compute (8). Hence, in a memory constrained environment, one may discard the utterances and keep only the J_A's and $\mathbf{d}_A$'s. During testing, we do the same for a given query utterance and then compute the probability in (8), which can be done much faster than computing GMM output likelihoods. However, this is still an exhaustive *linear search*, where the probability must be computed for *all* target utterances. Assuming the computation of the probability is done in constant time for each pair of utterances, the runtime overhead of the search algorithm is $O(N)$ where N is the number of target speakers. This is unacceptable for information retrieval from extremely large databases.

4.1. A brief overview of KLSH

Locality-Sensitive Hashing (LSH) searches a large database efficiently by reducing the search space via a hash function that projects each data item to a set of bits. Each target vector in the database of size N is mapped to a b-bit vector where $b \ll N$, called a *hash key*. The same is done for a given query vector $\mathbf{q}$. Similar vectors tend to be mapped to similar hash keys, so only those targets that were mapped to the same (or approximately the same) hash key as the query need to be linearly compared with the query. LSH methods for Euclidean and inner-product distances have already been developed and applied. More recently, *kernelized* LSH [4] was proposed where any distance metric that qualifies as a kernel can be used without knowledge of the embedding function $\phi(X)$:

$$\kappa\left(X_a, X_b\right) = \phi(X_a)^T \phi\left(X_b\right) \tag{15}$$

To perform KLSH [4], we first randomly choose p target data points to create a $p \times p$ kernel matrix K, where each element $K_{i,j} = \kappa(X_i, X_j)$. We then construct b hash functions, $h_i\left(\phi\left(X\right)\right)$ where $1 \leq i \leq b$. Each h_i is constructed via the following steps [4]:

1. Select t indices at random from $[1, \cdots, p]$ to form the $p \times 1$ vector $\mathbf{e}_{S,i}$ containing ones at the locations corresponding to the t indices and zeros elsewhere.
2. Form $\mathbf{w}_i = K^{-1/2}\left(\frac{1}{t}\mathbf{e}_{S,i} - \frac{1}{p}\mathbf{e}\right)$ where $\mathbf{e}$ is a vector of 1's.
3. $h_i\left(\phi\left(X\right)\right) = \text{sign}\left\{\sum_j \mathbf{w}_i\left(j\right)\kappa\left(X, X_j\right)\right\}$

The b hash functions map every vector to a bit sequence of length b, which is the vector's hash key. In some cases where b is very long and the hash table is sparsely populated, hash collisions may be very few or not even exist. In such a case, Approximate Nearest Neighbor (ANN) searches must be done to also visit hash bins with keys that are close to the query hash key in terms of the Hamming Distance.

4.2. Speaker search using KLSH

It is already well-known that equation (5) is also a kernel [7]. Hence, to perform the search in (10) more efficiently, we can simply replace $k(X_a, X_b)$ in the KLSH algorithm above with our model (8). Two types of searches are possible. One is also a linear search, but using the Hamming Distance on the query and target *hash keys* instead of using the utterance comparison equation (8). The other is the full KLSH search where ANN is used to look up the query's neighboring keys in the hash table to find an approximate set of matches which are then re-ranked by linear comparison. For our implementation, we used the ANN algorithm in [8]. This algorithm requires choosing a set of random permutations of the hash key bits and maintaining a sorted order of the target hash keys according to each permutation. For a given query hash key, each permutation is performed on it and a binary search of the permuted query hash key is done on the corresponding sorted order of target hash keys. By examining the neighboring target hash keys of the binary search result for all such permutations, an approximate set of nearest hash keys according to some set Hamming distance range can be efficiently obtained. The greater the number of permutations, the more target data items visited, and therefore the closer the ANN is to exhaustive linear search.

Table 1. MRR and classification rate for varying length l of voiced parts of query utterances ($b = 300$, $p = 300$, $t = 30$, $e = 80$). LINH=linear Hamming Distance search; LIN=linear search using the kernel function.

l (seconds)		4	6	10	20	30	60
KLSH	MRR	0.774	0.830	0.877	0.925	0.952	0.983
	Rate	0.765	0.825	0.876	0.924	0.952	0.983
LINH	MRR	0.957	0.982	0.992	0.997	0.999	0.999
	Rate	0.938	0.972	0.988	0.996	0.998	0.999
LIN	MRR	0.957	0.982	0.992	0.997	0.999	0.999
	Rate	0.938	0.972	0.988	0.996	0.998	0.999

Table 2. MRR and classification rate for varying number of hash key bits b ($l = 60$, $p = 300$, $t = 30$, $e = 80$)

b		50	150	250	300
KLSH	MRR	0.922	0.947	0.971	0.983
	Rate	0.922	0.947	0.971	0.983
LINH	MRR	0.999	0.999	0.999	0.999
	Rate	0.999	0.999	0.999	0.999

5. EXPERIMENT

We obtained 9,422 distinct speakers from the SPEECON database for 17 languages. From this pool, we set aside 1,500 speakers as training data to obtain a UBM with 256 Gaussian components, and a set of 20 eigenvoices via Principal Component Analysis(PCA) on MAP-adapted (mean only) speaker-dependent models. The remaining 7,922 speakers were used as test data. The set of speech utterances from each test speaker was split into a query utterance set and a target utterance set with no overlap (all query utterances and target utterances were distinct). The task was to iterate over all 7,922 queries to see how well they matched up with their corresponding targets. The acoustic features were 12 MFCC coefficients with energy and delta coefficients, resulting in a total 26 features, and a harmonicity-based Voice Activity Detector was used to retain only voiced frames. The objective of this work was to test search efficiency and effectiveness under *matched* conditions, so only close-talk microphone recordings were used for all speech utterances.

To measure search performance, we use the Mean Reciprocal Rank (MRR), which is defined as $\frac{1}{N}\sum_{i=1}^{N} 1/r_i$ where N is the

number of queries and r_i is the rank of the correct target in the list of returned results for the i'th query. We also compute the traditional classification rate (or "top-1 rate"), $\frac{1}{N}\sum_{i=1}^{N}\delta(r_i-1)$. Tab.1 shows the MRR and classification rate from three types of searches – linear search using our kernel function (8) (LIN), linear search using the Hamming Distance on the query and target hash keys (LINH), and KLSH using our kernel (8) and the aforementioned ANN algorithm – for varying lengths l (seconds) of the voiced parts of the query while the KLSH parameters b, p, t (explained in Sec.4.1), and e (the number of permutations in the ANN) remain fixed. While LIN and LINH are highly accurate even for queries as short as 4 seconds, it is evident that the KLSH takes a performance hit because of the higher mismatch between query and target for short queries, resulting in a higher miss rate in the ANN. Note, however, that we can always use a larger number of bit permutations (see our discussion regarding Tab.3) if we wish to make the KLSH approach LINH. Also worth mentioning is that the MRR and classification rate tend to be identical when using KLSH. This is because KLSH-based search tends to be a "hit-or-miss" affair where the correct target either appears at the top of the result list or does not appear at all, because of the randomness inherent in the approximate nearest neighbor mapping. In this case, we have either $r_i = 0$ or $r_i \approx \infty$, which results in the MRR being equal to the classification rate. Note that we did not perform the same experiment using GMM's, simply because it would take an impossibly long time to do so (see our discussion on Tab.4).

Tab.2 shows the MRR for KLSH and LINH for a varying number of hash key bits. For $l = 60$ (seconds), accurate search can already be done with only 50 bits. As the length decreases, however, more bits are expected to be needed. Tab.3 illustrates the dependence of KLSH on the accuracy and speed of the ANN. The average search time per query in this table represents the time spent on search only, not the time spent on extracting the query utterance statistics (9) (about 2.75s for a voiced length of 10 seconds). All times were measured by running searches on a single 3.2GHz CPU core.

Finally, in Tab.4, we make a comparison between the average search time per query, with a fixed query length 10 seconds. The time for the GMM-based speaker ID is the time purely spent on computing the likelihoods to carry out the Bayesian decision rule (assuming uniform priors) $\arg\max_{1\le k\le N} p(X_q|\lambda_k)$ where λ_k is the set of GMM parameters (MAP-adapted from UBM) of the k'th target utterance X_{tk}. The time spent on computing $p(X_q|\lambda_k)$ for all $N = 7922$ target models is extrapolated from measurements on a smaller set of data (based on this extrapolation, one can see that the entire experiment using all queries would take about 566 days on a single CPU core if we were to actually attempt it). The times for LIN, LINH, and KLSH include the time spent on extracting the query's statistics in (9) (note that the time consumed by the KLSH search itself is independent of the query length). It is evident that the KLSH-based method is orders of magnitude faster than the baseline GMM. Other studies [2, 3, 1] reported speed-up rates as high as 310 compared to the GMM baseline, which is much lower than that achieved in the proposed system. Furthermore, the number of speakers in the experiments conducted in these studies were only in the hundreds, not thousands. To the best of our knowledge, no other competing method experimented with a speaker set as large as the one used in this paper. Although LINH is slower than LIN in this experiment, for lower numbers of hash key bits it is expected to be faster because the Hamming Distance is easier to compute than the kernel function. Also, although we used 10-second queries for Tab.4, the KLSH speed-up ratio is roughly the same for longer or shorter queries, since both GMM computation time and speaker factor ML estimation time (which comprises most of the time required by KLSH in Tab.4) are linearly proportional to the query length.

Table 3. MRR, classification rate, and average search time (excluding computation of utterance statistics) per query of KLSH for varying number of bit permutations e ($l=60$, $b=300$, $p=300$, $t=30$)

e	10	20	40	60	80
MRR	0.868	0.937	0.969	0.980	0.983
Rate	0.868	0.937	0.969	0.979	0.983
Time(s)	0.186	0.211	0.259	0.298	0.337

Table 4. Average search time per query ($l = 10$, $b = 300$, $p = 300$, $t = 30$, $e = 80$) on database of 7,922 speakers. For LIN, LINH, and KLSH, time includes 2.75s average time spent on extracting query utterance statistics. *Time for GMM is extrapolated from measurements on a smaller set of data.

	GMM	LIN	LINH	KLSH
Avg. Time(s)	6176*	6.778	8.782	3.093
Speed Improvement	1×	911×	703×	1997×

6. CONCLUSION

In this paper, we proposed a method of efficiently searching large populations of speakers using kernelized LSH with an utterance comparison model proposed in previous work. In an experiment with a database of 7,922 speakers, we were able to achieve search speeds that were orders of magnitude higher than the traditional GMM-based method with negligible loss of accuracy for sufficiently long query lengths. Even without KLSH, a linear comparison over the database using our kernel proved to be much faster than GMMs and highly accurate for very short queries. Further studies will be done on how to make the KLSH more robust, particularly to mismatched conditions where environmental noise is present.

7. REFERENCES

[1] V. R. Apsingekar and P. L. De Leon, "Speaker model clustering for efficient speaker identification in large population applications," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, no. 4, pp. 848–853, May 2009.

[2] T. Kinnunen, E. Karpov, and P. Franti, "Real-time speaker identification and verification," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 277–288, 2006.

[3] H. Aronowitz and D. Burshtein, "Efficient speaker recognition using approximated cross entropy (ace)," *IEEE Trans. ASLP*, vol. 15, no. 7, pp. 2033–2043, 2007.

[4] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *IEEE Proc. 12th Int. Conf. on Computer Vision*, Sept. 2009.

[5] W. Jeon, C. Ma, and D. Macho, "An utterance comparison model for speaker clustering using factor analysis," in *IEEE Int. Conf. on Acoustics, Signal and Speech Proc.*, 2011.

[6] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 8, no. 6, Nov. 2000.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*, pp. 291–299, Springer, 2006.

[8] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. ACM Symposium on Theory of Computing*, 2002, pp. 380–388.