Efficient Approximation Algorithms for the Hamming Center Problem

Leszek Gąsieniec [*] Jesper Jansson [†] Andrzej Lingas [‡]

Abstract

The *Hamming center problem* for a set S of k binary strings, each of length n, asks for a binary string β of length n that minimizes the maximum Hamming distance between β and any string in S. The decision version of this problem is known to be NP-complete [6]. We provide several approximation algorithms for the Hamming center problem. Our main result is a randomized $(\frac{4}{3}+\varepsilon)$-approximation algorithm running in polynomial time if the Hamming radius of S is at least superlogarithmic in k. Furthermore, we show how to find in polynomial time a set B of $O(\log k)$ strings of length n such that for each string in S there is at least one string in B within Hamming distance not exceeding the radius of S.

1 Introduction

Let $\mathbb{Z}_2^n$ be the set of all strings of length n over the alphabet $\{0,1\}$. For any $\alpha \in \mathbb{Z}_2^n$ we use the notation $\alpha[i]$ to refer to the symbol placed at the ith position of α, where $i = 1, .., n$, and we let $\alpha[i..j]$ represent the substring of α starting at position i and ending at position j, where $1 \leq i < j \leq n$. The *Hamming distance* between $\alpha_1, \alpha_2 \in \mathbb{Z}_2^n$ is defined as the number of positions in which the strings differ, and is denoted by $d(\alpha_1, \alpha_2)$.

The *Hamming center problem* (HCP) is stated as follows: Given a set S of k binary strings $\alpha_i \in \mathbb{Z}_2^n$, where $i = 1, .., k$, find a string $\beta \in \mathbb{Z}_2^n$ such that the value

$$r = \max_{1 \leq i \leq k} d(\beta, \alpha_i)$$

is minimized.

Such a β is called a *center* of S, and the corresponding value of r is called the *radius* of S. Note that some instances of HCP (in fact most of them) have several centers. Throughout this paper, we will denote approximative solutions to HCP by β'.

[*]Dept. of Computer Science, University of Liverpool, Peach Street, L69 7ZF, UK, leszek@csc.liv.ac.uk
[†]Dept. of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden, Jesper.Jansson@dna.lth.se
[‡]Dept. of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden, Andrzej.Lingas@dna.lth.se

The decision problem version of HCP is: Given a set S of k binary strings $\alpha_i \in \mathbb{Z}_2^n$, where $i = 1,..,k$, and a positive integer r_p, determine whether there exists a string $\beta \in \mathbb{Z}_2^n$ such that

$$\max_{1 \leq i \leq k} d(\beta, \alpha_i) \leq r_p.$$

There is a vast literature on graph and geometric center and multi-center problems modelling canonical facility location problems [15]. Unfortunately, because of the exponential size of $\mathbb{Z}_2^n$, its high dimension, and its discreteness, neither the graph approach nor the geometric approach work directly for HCP.

A variant of HCP has been studied by Berman *et al.* in [2], where they look for all maximal blocks within a set of aligned binary strings, having centers with radius of size at most 1. Their study is motivated by the detection of potentially important regions within closely related DNA sequences. They provide a linear time algorithm for their problem. In fact, there exists a straightforward polynomial-time algorithm for HCP as long as the radius is $O(1)$ (see Section 5). Berman *et al.* left the question open whether there exists an efficient algorithm for the case where the size of the radius is arbitrarily large.

In [6], Frances and Litman showed that the decision version of HCP, by them called the Minimum Radius Problem, as well as the dual Maximum Covering Radius Problem (MCR) are NP-complete; MCR and the decision version of HCP are in fact equivalent [3, 6]. MCR is defined as follows: Given a set $C \subseteq \mathbb{Z}_2^n$, called the set of *code words*, and a positive integer ρ, are all strings in $\mathbb{Z}_2^n$ within Hamming distance ρ of some code word in C? The *covering radius* is the smallest integer ρ that satisfies the above condition. It is a basic geometric parameter of a code which measures its quality. For example, if a code is used for data compression, the covering radius is a measure of the maximum distortion [1], and when it is used in error correction, it gives the maximum weight of a correctable random error [3]. Another related problem is the covering radius of a lattice in Euclidean space, which has applications to quantization and to coding for the Gaussian channel [4, 5].

Motivated by the intractability of HCP, we present here several approximation algorithms.* To start with, we give a very simple 2-approximation algorithm for HCP in Section 2. In Section 3, we provide an integer programming formulation of HCP. It yields polynomial time solutions for HCP if k (the number of input strings) or n (their lengths) is fixed. In the subsequent section, we observe that the method of randomized rounding applied to the linear programming relaxation of the integer programming formulation of HCP yields a solution with the additive error guarantee $O(\sqrt{n \log n})$ with probability greater than $1 - \frac{k}{n^2}$. In Section 5, we present our main result, i.e., the randomized $(\frac{4}{3} + \varepsilon)$-approximation algorithm for HCP running in polynomial time if the radius of set S is at least superlogarithmic in k. Finally, in Section 6 we show how to find in polynomial time a set B of $O(\log k)$ strings of length n such that for each string in S there is at least one string in B within Hamming distance not exceeding the radius of S.

*A preliminary version of this paper appeared in [7].

2 A Simple Heuristic

We start with the following heuristic, showing that one can get a constant approximation ratio instantly: Set the approximative solution β' to α_l, where l is chosen arbitrarily from $\{1,..,k\}$.

The Hamming distance obeys the triangle inequality ([9], p. 424):

$$d(x,y) \leq d(x,z) + d(z,y), \quad \text{for all } x,y,z \in \mathbb{Z}_2^n. \tag{1}$$

Theorem 2.1 *The simple heuristic always yields a solution that is within a factor 2 of the optimal.*

Proof: Let β be an optimal solution. Due to inequality (1), $d(\beta',\alpha_i) \leq d(\beta',\beta) + d(\beta,\alpha_i)$ for every i in $\{1,..,k\}$. Next, $d(\beta',\beta) + d(\beta,\alpha_i) = d(\alpha_l,\beta) + d(\beta,\alpha_i) \leq r + r = 2r$ by the definition of r. □

3 Integer Programming Formulation

The Hamming center problem is equivalent to a special case of the integer programming problem. Any given instance $(\alpha_1,..,\alpha_k,r_p)$ of the decision version of HCP, where $\alpha_i \in \mathbb{Z}_2^n$ for $1 \leq i \leq k$ and $r_p \in \mathbb{N}$, can be expressed as a system of k linear inequalities.

Let $x = (x_1,..,x_n) \in \{0,1\}^n$ represent a vector of 0-1-variables and let the ith inequality be

$$\sum_{\substack{\alpha_i[m]=0 \\ 1\leq m\leq n}} x_m + \sum_{\substack{\alpha_i[m]=1 \\ 1\leq m\leq n}} (1-x_m) \quad \leq \quad r_p$$

The left-hand side is equal to the Hamming distance between α_i and x. (For each position m, if $\alpha_i[m] = 0$ then the sum is incremented by one if and only if $x_m = 1$, and conversely, if $\alpha_i[m] = 1$ then the sum is incremented by one if and only if $x_m = 0$.) The constraint '$\leq r_p$' ensures that the value of $d(\alpha_i,x)$ is smaller than or equal to the proposed radius.

A 0-1-solution x that is compatible with the system of inequalities corresponds directly to a solution β for the supplied instance, as can be seen by setting $\beta[m] = x_m$ for $1 \leq m \leq n$.

A system of k inequalities in n variables can be transformed into a system of k equalities in $n+k$ variables. The integer programming problem for a system of linear equalities on non-negative variables with coefficients unrestricted in sign admits pseudo-polynomial-time solutions when the number of equalities is constant [13]; this means that the running time of such an algorithm is bounded by a polynomial in n and the maximum among all the absolute values of coefficients involved in the equalities. Thus, HCP for $k = O(1)$ is solvable in time polynomial in its input length.

On the other hand, Lenstra, Lenstra, and Lovász [11] have shown that there is a polynomial (in k) time algorithm for testing whether a system of k linear inequalities with a fixed number of variables has an integer solution. So, HCP for $n = O(1)$ can also be solved in polynomial time.

Lemma 3.1 *HCP restricted to instances with $O(1)$ strings as well as HCP restricted to instances with strings of length $O(1)$ are solvable in polynomial time.*

4 Randomized Rounding

By relaxing the integer constraints on x in the 0-1-integer programming formulation of HCP, thus allowing each component x_m to be a real number in the interval $[0, 1]$, we get a linear programming problem that can be solved in polynomial time by known methods [14]. Randomization can then be applied to the solution of the relaxed problem to get an approximative 0-1-solution β' to the original problem.

We use the following randomized rounding scheme: Let $\hat{x}_m$ be the value assigned to x_m in the solution to the linear program. For each m, where $1 \leq m \leq n$, set $\beta'[m]$ to 1 with probability $\hat{x}_m$ and to 0 with probability $1 - \hat{x}_m$.

To analyze this method, we can proceed analogously as in a standard analysis of the lattice approximation problem (see pp. 449–450 in [8]). We consider each inequality separately, and flip the probability from p_i to $1 - p_i$ if $\alpha_i[m] = 1$. By using two variants of Chernoff's bounds (see pp. 449–450 in [8]) and defining the *generalized distance* between any $z_1, z_2 \in [0, 1]^n$ as $d_G(z_1, z_2) = \sum_{m=1}^{n} |z_1[m] - z_2[m]|$, we obtain:

Lemma 4.1 *For an instance of HCP with k strings of length n, let r be the radius of the instance, and let β' be the approximative solution found by randomized rounding. Then the maximum Hamming distance between β' and any string in the instance is at most*

1. *$r + O(\sqrt{n \log n})$ with probability $\geq 1 - \frac{k}{n^2}$, and*
2. *$r(1 + \varepsilon)$ with probability $\geq 1 - k \cdot 2 \exp(-0.38\varepsilon^2 w)$, where w is the minimum generalized distance between a string in the instance and an optimal solution to the relaxed version of the instance.*

Corollary 4.1

1. *If the radius of the instance is larger than $q\sqrt{n \log n}$ then the maximum distance between β' and any string in the instance is at most $r + O(q^{-1})$ with probability $\geq 1 - \frac{k}{n^2}$.*
2. *If the minimum generalized distance w between a string in the instance and an optimal relaxed solution is at least $\frac{\ln(4k)}{0.38\varepsilon^2}$ then the maximum distance between β' and any string in the instance is at most $(1 + \varepsilon)r$ with probability $\geq \frac{1}{2}$.*

In other words, the method of randomized rounding is likely to yield nearly optimal solutions if the radius r is substantially larger than $\sqrt{n \log n}$ and k is substantially smaller than n^2, or if the minimum generalized distance w is sufficiently large.

5 A Randomized $(\frac{4}{3} + \varepsilon)$-Approximation Algorithm

In this section, we give a randomized algorithm for HCP that with high probability yields answers which are within a factor of $(\frac{4}{3} + \varepsilon)$ of the optimum, where $\varepsilon > 0$ can be made arbitrarily small. The algorithm runs in polynomial time, provided the radius of the input instance is not very small.

Recall that r is defined as

$$r = \min_{\beta \in \mathbb{Z}_2^n} (\max_{1 \le i \le k} d(\beta, \alpha_i))$$

where β is an optimal solution. The approximative solution found by our algorithm is called β', and we denote the value of $\max_{1 \le i \le k} d(\beta', \alpha_i)$ by r'. We first describe the algorithm and then prove that $r' \le (\frac{4}{3} + \varepsilon)\, r$ with probability $\ge \frac{1}{2}$.

Description of the Algorithm

First of all, calculate the *diameter* f of the instance, where f is defined to be the maximum of all Hamming distances between any two of the given binary strings, i.e.,

$$f \;=\; \max_{1 \le j \le k} (\max_{1 \le i \le k} d(\alpha_i, \alpha_j)).$$

Next, compute $\frac{12.5 \ln(4k)}{\varepsilon^3}$ using the specified value of ε.

If $f < \frac{12.5 \ln(4k)}{\varepsilon^3}$ then let the algorithm branch to Case 1 below to obtain β'; if $f \ge \frac{12.5 \ln(4k)}{\varepsilon^3}$ then obtain β' from Case 2.

Case 1: $f < \frac{12.5 \ln(4k)}{\varepsilon^3}$

Note that $r \le f$ (since $r \le \min_{1 \le j \le k} (\max_{1 \le i \le k} d(\alpha_i, \alpha_j)) \le f$). Find an exact solution by brute force, i.e., by trying all of the $\sum_{j=0}^{f} \binom{n}{j} = O(n^f)$ strings in $\mathbb{Z}_2^n$ within Hamming distance f of α_l, where l is chosen arbitrarily from $\{1, .., k\}$. Each such string can be evaluated in $O(kn)$ time; therefore, the time required is $O(k \cdot n^{f+1})$.

Case 2: $f \ge \frac{12.5 \ln(4k)}{\varepsilon^3}$

The second case is divided into two subcases: $f \le \frac{4}{3} r$ and $f > \frac{4}{3} r$.

The algorithm can't know beforehand which subcase holds since r is still unknown. To get around this, the algorithm runs both procedures below and chooses the better one of the two solutions.

Subcase 2a: $f \le \frac{4}{3} r$
Set $\beta' = \alpha_l$, where l is chosen arbitrarily from $\{1, .., k\}$.

Subcase 2b: $f > \frac{4}{3}r$
Rearrange the α_i's so that $d(\alpha_1, \alpha_k) = f$. Then, normalize the strings as follows:

old_$\alpha_1 := \alpha_1$
for $m := 1$ **to** n **do**
  if $\alpha_1[m] = 1$ **then**
    for $i := 1$ **to** k **do** $\alpha_i[m] := 1 - \alpha_i[m]$

This doesn't change any of the Hamming distances because whenever some bit in a string is changed, the corresponding bit in every other string is changed as well. Next, let some permutation σ act on the columns so that the f positions of α_k that contain 1s are located at $\alpha_k[1], .., \alpha_k[f]$ (see Figure 1). Note that this operation does not affect the Hamming distances either.

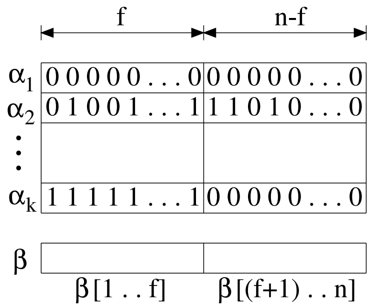

Figure 1: After normalization, the two strings that are the farthest apart are α_1 and α_k. They differ at precisely positions $1..f$.

Consider the 0-1-integer programming problem corresponding to the normalized instance with the integer constraints relaxed as described in Section 4. Add constraints which force the last $n - f$ positions of a solution to be 000..0. Let ρ be an optimal solution to this problem, and set γ to that $\alpha_i[1..f]\,0^{n-f}$, $i = 1, .., k$, which minimizes the generalized distance between γ and ρ. Next, apply the randomized rounding scheme from Section 4, and call the solution obtained μ. Set β' equal to the one of the two strings γ, μ with the smallest maximum distance to a string α_i, $i = 1, .., k$.

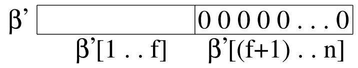

Figure 2: For Subcase 2b, the last $n-f$ positions of β' will always be 0.

Finally, to transform β' back to the original instance, apply σ^{-1} to β' and perform:

for $m := 1$ **to** n **do**
  if old_$\alpha_1[m] = 1$ **then**
    $\beta'[m] := 1 - \beta'[m]$

and Subcase 2b is done.

Performance Analysis

Now we analyze the algorithm's approximation ratio.

Case 1: $f < \frac{12.5\ln(4k)}{\varepsilon^3}$

The algorithm finds an exact solution, so $r' = r$.

Subcase 2a: $f \geq \frac{12.5\ln(4k)}{\varepsilon^3}$ and $f \leq \frac{4}{3}r$

Here $r' \leq f \leq \frac{4}{3}r$.

Subcase 2b: $f \geq \frac{12.5\ln(4k)}{\varepsilon^3}$ and $f > \frac{4}{3}r$

First, we observe that an optimal solution β can't have a lot of ones in the last $n-f$ positions.

Lemma 5.1 *$\beta[(f+1)..n]$ contains less than $\frac{1}{3}r$ ones.*

Proof: Suppose $\beta[(f+1)..n]$ contains $\geq \frac{1}{3}r$ ones. Since $d(\beta,\alpha_1) \leq r$, $\beta[1..f]$ must contain $\leq \frac{2}{3}r$ ones. Similarly, $d(\beta,\alpha_k) \leq r$ implies that $\beta[1..f]$ contains $\leq \frac{2}{3}r$ zeros. But this would mean that $f \leq \frac{2}{3}r + \frac{2}{3}r = \frac{4}{3}r$, which is a contradiction since $f > \frac{4}{3}r$. $\square$

Next, define ζ to be a string in $\mathbb{Z}_2^n$ with zeros on the last $n-f$ positions that minimizes the value of $\max\limits_{1\leq i\leq k} d(\zeta,\alpha_i)$.

Lemma 5.2 $\max\limits_{1\leq i\leq k} d(\zeta,\alpha_i) \leq \frac{4r}{3}$.

Proof: Let ξ be the concatenation of the first f symbols of β and the string consisting of $n-f$ zeros, i.e., $\xi = \beta[1..f]\,0^{n-f}$. From the definitions and Lemma 5.1, we see that

$$\max_{1\leq i\leq k} d(\zeta,\alpha_i) \;\leq\; \max_{1\leq i\leq k} d(\xi,\alpha_i) \;\leq\; \max_{1\leq i\leq k}(d(\xi,\beta)+d(\beta,\alpha_i)) \;\leq\; \frac{1}{3}r + r \;=\; \frac{4r}{3}.$$

$\square$

Lemma 5.3 *If $f \geq \frac{12.5\ln(4k)}{\varepsilon^3}$ then $r' \leq (1+\frac{3\varepsilon}{4})\cdot\max\limits_{1\leq i\leq k} d(\zeta,\alpha_i)$ with probability at least $\frac{1}{2}$.*

Proof: We have

$$r' \;=\; \max_{1\leq i\leq k} d(\beta',\alpha_i) \;\leq\; \max_{1\leq i\leq k}(d_G(\beta',\rho)+d_G(\rho,\alpha_i)) \;\leq\; d_G(\beta',\rho) + \max_{1\leq i\leq k} d(\zeta,\alpha_i).$$

Let w be the minimum generalized distance between ρ and $\alpha_i[1..f]\,0^{n-f}$, $i = 1,..,k$.
If w is at most $\frac{3\varepsilon}{8}f$, then

$$d_G(\beta',\rho) \;\leq\; d_G(\gamma,\rho) \;\leq\; \frac{3\varepsilon}{8}f \;\leq\; \frac{3\varepsilon}{4}r \;=\; \frac{3\varepsilon}{4}\cdot\max_{1\leq i\leq k} d(\beta,\alpha_i) \;\leq\; \frac{3\varepsilon}{4}\cdot\max_{1\leq i\leq k} d(\zeta,\alpha_i)$$

since $f = d(\alpha_1, \alpha_k) \leq d(\alpha_1, \beta) + d(\beta, \alpha_k) \leq 2r$.

Otherwise, w satisfies $w > \frac{3\varepsilon}{8} f > \frac{\ln(4k)}{0.38(\frac{3\varepsilon}{4})^2}$. In this case, the lemma follows from the relation $\max_{1 \leq i \leq k} d(\beta', \alpha_i) \leq \max_{1 \leq i \leq k} d(\mu, \alpha_i)$ along with a straightforward generalization of the second part of Corollary 4.1 to account for the constraint requiring zeros on the last $n - f$ solution positions, with ε set to $\frac{3\varepsilon}{4}$. □

Now, because $f \geq \frac{12.5 \ln(4k)}{\varepsilon^3}$, it follows from Lemma 5.2 and Lemma 5.3 that

$$r' \leq (1 + \frac{3\varepsilon}{4}) \cdot \frac{4r}{3} = (\frac{4}{3} + \varepsilon) r$$

with probability $\geq \frac{1}{2}$.

This concludes Subcase 2b. We have thus proved the main theorem of this section.

Theorem 5.1 *The approximative center β' found by the algorithm for an instance of HCP is within distance $(\frac{4}{3} + \varepsilon) r$ of every α_i, $i = 1, .., k$, with probability $\geq \frac{1}{2}$. If the diameter f of the input instance satisfies $f \geq \frac{12.5 \ln(4k)}{\varepsilon^3}$ then β' will be constructed in polynomial time. If $f < \frac{12.5 \ln(4k)}{\varepsilon^3}$, then β' is actually an optimal solution, and will be found in $O(kn^{\frac{12.5 \ln(4k)}{\varepsilon^3} + 1})$ time.*

Note that as long as f is superlogarithmic in k, the algorithm is guaranteed to run in polynomial time.

6 Another Type of Approximation

HCP can be approximated in other ways than described above. One is to find a set of strings B such that every given α_i, $i = 1, .., k$, is at a distance of at most r from some element in B, where r is the optimal radius. So, instead of looking for *one* optimal center β with radius r, we look for *a small set of* approximative centers that together are as good as β is by itself.

We can implement this idea as follows. Transform the given instance of HCP to its integer programming formulation as described in Section 3. By considering the linear programming relaxation of the formulation and using randomized rounding, we can find in polynomial time an approximative solution β'_1 that satisfies at least a quarter of the inequalities with probability greater than $\frac{1}{3}$. (To see this, let Z_1 be a random variable representing the number of inequalities that are not satisfied by β'_1. $\mathbf{E}[Z_1] = \frac{k}{2}$, so by the Markov inequality, $\mathbf{Pr}[Z_1 < \frac{3}{4}k] > \frac{1}{3}$.) After that, we remove all inequalities satisfied by β'_1, and iterate our method for the remaining inequalities to obtain a solution β'_2, remove all inequalities satisfied by β'_2, and so on. After K iterations, where $K = O(\log k)$, every inequality is satisfied by some β'_j ($j \in \{1, .., K\}$) with probability $\geq n^{-O(1)}$. Clearly, the set $B = \{\beta'_1, .., \beta'_K\}$ consists of a logarithmic (in k) number of strings that together fulfill the criterion $d(\beta'_j, \alpha_i) \leq r$, for some $j \in \{1, .., K\}$ for every $i \in \{1, .., k\}$.

By performing independent iterations of our algorithm until such a set B of cardinality $O(\log k)$ is found, we obtain the following theorem.

Theorem 6.1 *For an instance of HCP with binary strings α_i, $i = 1, .., k$, each of length n, we can find, in expected time polynomial in $n + k$, a set of binary strings β'_j, $j = 1, .., K$ with $K = O(\log k)$, such that for each α_i, $i = 1, .., k$, there is a $j \in \{1, .., K\}$ for which the Hamming distance between α_i and β'_j doesn't exceed the radius of the instance.*

7 Final Remarks

Special thanks go to Mike Paterson for pointing out the relation between the Hamming center problem and coding theory.

The randomized algorithm in Section 5 can be derandomized by using the method of conditional probabilities (see [12]).

Using techniques similar to the ones in this paper, Lanctot *et al.* [10] have independently found a $(\frac{4}{3} + \varepsilon)$-approximation algorithm for HCP that runs in polynomial time for all Hamming radii. By incorporating a simple modification inspired by their paper, our algorithm can also be made to run in polynomial time for all Hamming radii.

References

[1] T. Berger. *Rate Distortion Theory*. Prentice Hall, Englewood Cliffs, NJ, 1971.

[2] P. Berman, D. Gumucio, R. Hardison, W. Miller, and N. Stojanovic. A Linear-Time Algorithm for the 1-Mismatch Problem. *Proc. of the 5th Workshop on Algorithms and Data Structures* (WADS'97), 1997.

[3] G.D. Cohen, M.G. Karpovsky, H.F. Mattson, and J.R. Schatz. Covering Radius – Survey and Recent Results. *IEEE Trans. Information Theory*, Vol. IT-31, pp. 328–343, 1985.

[4] J.H. Conway and J.A. Sloane. Voronoi regions of lattices, second moments of polytopes, and quantization. *IEEE Trans. Information Theory*, vol. IT-28, pp. 211–226, 1982.

[5] J.H. Conway and J.A. Sloane. Fast quantizing and decoding algorithms for lattice quantizers and codes. *IEEE Trans. Information Theory*, vol. IT-28, pp. 227–232, 1982.

[6] M. Frances and A. Litman. On Covering Problems of Codes. *Theory of Computing Systems* **30**, pp. 113–119, 1997.

[7] L. Gąsieniec, J. Jansson, and A. Lingas. Efficient Approximation Algorithms for the Hamming Center Problem. *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'99), pp. S905–S906, 1999.

[8] D.S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, 1996.

[9] B. Kolman, R. Busby, and S. Ross. *Discrete Mathematical Structures* [3rd ed.]. Prentice Hall, New Jersey, 1996.

[10] J.K. Lanctot, M. Li, B. Ma, S. Wang and L. Zhang. Distinguishing String Selection Problems. *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'99), pp. 633–642, 1999.

[11] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.* 261, pp. 515–534, 1982.

[12] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[13] C. Papadimitriou. On the Complexity of Integer Programming. *Journal of the ACM*, Vol. 28, No. 4, pp. 765–768, 1981.

[14] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[15] S. Vishwanathan. An $O(\log^* n)$ Approximation Algorithm for the Asymmetric p-Center Problem. *Proc. of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'96), pp. 1–5, 1996.