An Approximation Algorithm for Alphabet Indexing Problem

Shimozono, Shinichi
Department of Control Engineering and Science, Kyushu Institute of Technology

出版情報：RIFIS Technical Report. 96, 1995-01. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

An Approximation Algorithm for Alphabet Indexing Problem

Shinichi Shimozono
Department of Control Engineering and Science,
Kyushu Institute of Technology
Kawazu 680-4, Iizuka, 820 JAPAN

Abstract

Alphabet Indexing is the problem to find a mapping $f : \Sigma \to \{1, \ldots, K\}$ for alphabet Σ, positive integer K and a pair of disjoint sets of strings $P, Q \subseteq \Sigma^*$ such that f transforms no two strings from P and Q into identical ones. Although Alphabet Indexing problem is NP-complete, we define a combinatorial optimization problem, Max K-Indexing, and propose a simple greedy algorithm for this problem. Then we show that the algorithm achieves the constant error ratio $1/K$ for K-indexing with respect to the number of distinguishable pairs and our problem is MAX SNP-hard.

Keywords: alphabet indexing, approximation algorithm, MAX-SNP, combinatorial optimization

1 Introduction

Given a pair of disjoint sets of strings P and Q over alphabet Σ and positive integer K, Alphabet Indexing is the problem to find a mapping f from Σ to $\Gamma = \{1, \ldots, K\}$ such that f transforms no two strings drawn from P and Q into identical ones in Γ^*.

Alphabet Indexing is named after "Hydropathy Index" for amino acid residues by Kyte and Doolittle [6] in molecular biology, which has been developed to specify the characteristic part of proteins by viewing them as the sequences of numbers representing hydropathy value of the residues. By giving alphabet indexings for string data consisting of a large number of symbols, we can drastically reduce the search domain of machine learning algorithms that require positive and negative training examples. In bioinformatical research field, we have experienced that the processing time is reduced and the domains of machine learning algorithms that computes decision trees over regular patterns are simplified by giving a suitable indexing for amino acid residues over the 20 kinds of symbols [8, 10].

Unfortunately, it is known that Alphabet Indexing Problem is NP-complete, and its polynomial-time local search problem version is PLS-complete [9]. In this paper, to deal with those intractability, we propose a strategy for this problem and introduce a combinatorial optimization problem Max K-Indexing. We present a simple greedy algorithm for

this problem based on that strategy, inspired by the greedy algorithm for Maximum Satisfiability (MAX SAT) developed and analyzed by Johnson [5]. We show our greedy algorithm achieves constant error ratio $1/K$ for K-indexing, with respect to the maximum number of distinguishable pairs. Additionally, we show the problem is MAX SNP-hard [1, 2, 7]. This means that the problem seems to have no polynomial-approximation schemes, i.e., there is a certain constant bound for the error ratio for polynomial-time approximation algorithms.

2 Preliminaries

Let Σ and Γ be finite alphabets. For a string s in Σ^*, we denote the i-th symbol by s_i and the length by $|s|$. We say that two strings s and t are identical (denoted by $s = t$) if $|s| = |t|$ and $s_i = t_i$ for every $1 \leq i \leq |s|$. Let f be a mapping from Σ to Γ. The homomorphism $f : \Sigma^* \to \Gamma^*$ is derived from $f : \Sigma \to \Gamma$ for transforming strings from s in Σ^* to $f(s) = f(s_1) \cdots f(s_{|s|})$ in Γ^*.

A combinatorial optimization problem Π consists of (1) a set of instances L_Π, (2) a finite set of feasible solutions $S_\Pi(\pi)$ of an instance $\pi \in L_\Pi$, and (3) a measure (cost) $\mu_\Pi : S_\Pi(L_\Pi) \to \mathrm{Z}^+$. The problem Π is specified as a maximization or minimization problem, depending on whether the measure of solutions is to be maximized or minimized. The cost of the optimal solution $OPT(\pi)$ for $\pi \in L_\Pi$ is defined by $OPT(\pi) = \mathrm{best}\{\mu_\Pi(s) \mid s \in S_\Pi(\pi)\}$, where best is max (resp. min) for maximization problems (resp. minimization problems). For an approximation algorithm A that solves Π, we define the relative error

$$\epsilon(\Pi, A) = \max_{\pi \in \Pi} \left| \frac{OPT(\pi) - \mu_\Pi(A(\pi))}{\max\{OPT(\pi), \mu_\Pi(A(\pi))\}} \right| ,$$

where $A(\pi)$ is the output of A for π. Also, we say that A is the $\epsilon(\Pi, A)$-approximation for Π. For conveniences, problem and algorithm of ϵ is abbreviated if there is no ambiguity.

Let Π, Φ be optimization problems. We say that Π L-reduces to Φ [7] if there are two polynomial time functions τ, ρ and constants $\alpha, \beta > 0$ such that:

1. Given an instance π of Π, algorithm τ produces an instance $\tau(\pi)$ of Φ such that the measure of the optimum solution of $\tau(\pi)$ is at most $\alpha \cdot OPT(\pi)$, and

2. Given any solution s of $\tau(\pi)$, algorithm ρ produces a solution $\rho(s)$ of π such that $|\mu_\Pi(\rho(s)) - OPT(\pi)| \leq \beta |\mu_\Phi(s) - OPT(\tau(\pi))|$.

This L-reductions compose, and if problem Π L-reduces to problem Φ and Φ can be approximated in polynomial time with error δ and relative error ϵ, then Π can be approximated with error $\alpha\beta\delta$ and relative error $\frac{\alpha\beta\epsilon}{1-\epsilon}$.

A problem is MAX SNP-hard if every MAX SNP problem can be L-reduced to it [7].

3 Alphabet Indexing Problem and An Approximation Algorithm

The Alphabet Indexing problem is formulated as follows [9, 10].

Definition 1. Alphabet Indexing

Instance: Alphabet Σ, disjoint sets of strings $P, Q \subseteq \Sigma^*$, and positive integer K.

Question: Do P,Q have a K-indexing, i.e., is there a function $f : \Sigma \to \{1, \cdots, K\}$ such that $f(p) \neq f(q)$ whenever $p \in P$ and $q \in Q$?

This problem is NP-complete [9]. For dealing with this problem, we consider the following strategy: the number of pairs $(p, q) \in P \times Q$ satisfying $f(p) \neq f(q)$ is maximized to minimize the number of pairs (p', q') with $f(p') = f(q')$. In our approximation scheme, we allow that indexings may have some confusing pairs $(p, q) \in P \times Q$ with $p \neq q$ but $f(p) = f(q)$. According to this strategy, we propose the following combinatorial optimization problem.

Definition 2. Max K-Indexing

Instance: Alphabet Σ, sets of strings $P, Q \subseteq \Sigma^*$, weights w_P and w_Q each maps from P and Q to positive integer.

Object: Find a K-indexing $f : \Sigma \to \{1, \ldots, K\}$ that maximizes the sum of the weight products $w_P(p) \cdot w_Q(q)$ of all pairs $(p, q) \in P \times Q$ that are $|p| = |q|$ but $f(p) \neq f(q)$.

From now on, we concentrate on the pairs consisting of different strings of the same length, *Pairs*(P, Q), defined by

$$\mathit{Pairs}(P, Q) = \{(p, q) \in P \times Q \mid |p| = |q| \wedge p \neq q\} \ .$$

Let p and q be strings in Σ^* and let a and b be symbols in Σ. For a pair of strings (p, q), we say that symbols a and b are *facing* in (p, q) if, for some i with $1 \leq i \leq |p| = |q|$, either $p_i = a \wedge q_i = b$ or $p_i = b \wedge q_i = a$. Notice that, for distinguishing a pair (p, q), there must be at least one facing pair of symbols a and b with different indices $f(a) \neq f(b)$.

Given an instance of Max K-Indexing, we apply the following algorithm Greedy_K.

Algorithm Greedy_K (*input*: Σ, *Pairs*(P, Q))

1. Let *Diff* $:= \emptyset$, *Res* $:=$ *Pairs*(P, Q) and $f(a) := 0$ for all $a \in \Sigma$.

2. For each $a \in \Sigma$ do

   a. For each $i \in \{1, \cdots, K\}$ compute $Saved_i :=$
   $\{(p,q) \in Res \mid a$ and b are facing in (p,q) and satisfying $0 < f(b) \neq i\}$.

   b. Find k that maximizes $\sum_{(p,q) \in Saved_k} w_P(p) \cdot w_Q(q)$.

   c. $Diff := Diff \cup Saved_k$, $Res := Res - Saved_k$ and $f(a) := k$.

3. Return f.

Theorem 1. $\epsilon(\text{Max } K\text{-Indexing}, Greedy_K) = 1/K$.

Proof. Suppose that the algorithm has been applied to the set $Pairs(P,Q)$ of an instance π, and is now going into step 2–(a) of l-th iteration for choosing the best index to $a_l \in \Sigma$. Let $d_i = \{(p,q) \in Res \mid a_l$ and b are facing in (p,q) satisfying $f(b) = i\}$ for $1 \leq i \leq K$, where Res is the remained pairs after the $(l-1)$-th iteration. In another words, d_i is equal to $Saved_k$ and removed from Res in step 2–(b) if index $f(a_l)$ is determined to $k \neq i$. We divide $\bigcup_{i=1}^{K} d_i$ into $K+1$ disjoint parts as follows.

- $D_0 = \bigcup_{i \neq j} d_i \cap d_j$: the set of pairs distinguished by any index for a_l with $f(a_l) > 0$.
- $D_i = d_i - D_0$: the set of pairs distinguished if and only if the index for a_l, $f(a_l)$ is not i.

Also, we define $Ex_i \subseteq D_i$, the set of pairs which exhaust symbols indexed 0 and are not distinguished if $f(a_l)$ is indexed to i. (We ignore the symbols which are still not indexed but facing only the same symbol.) Notice that all of Ex_i will remain in Res and never be saved if i is indexed to a_l, and by the definition, $D_i \cap D_j = \emptyset$ for any $1 \leq i \neq j \leq K$. For choosing $k \in \{1, \ldots, K\}$ to $f(a_l)$, algorithm $Greedy_K$ maximizes the size of $Saved_k = D_0 \cup (\cup_{i \neq k} D_i)$ and thus k must be a number that minimizes $|D_k|$. For an index $f(a_l) = k$ we lose the pairs in Ex_k, so we have

$$|Saved_k| = |D_0| + |\cup_{i \neq k} D_i| \geq |D_0| + (K-1) \cdot |D_k| \geq (K-1) \cdot |Ex_k| \ .$$

The statements discussed above hold at any iteration, and after the algorithm stopped, $Diff$ is the union of all $Saved$'s at each iteration, and Res is also the union of all Ex's. Therefore after the algorithm stopped we have $Greedy_K(\pi) = |Diff| \geq (K-1) \cdot |Res|$, and by $OPT(\pi) \leq |Pairs(P,Q)|$,

$$OPT(\pi) \leq |Pairs(P,Q)| = |Diff| + |Res| \leq |Diff| + \frac{1}{K-1}|Diff| \ .$$

This gives us the error bound of algorithm $Greedy_K$, and we conclude,

$$\epsilon \leq \frac{1}{K} .$$

Now we show the worst case causing the rate $1/K$. Let m be a sufficiently large positive integer. We consider the following instance π' with strings of the length two:

$$\Sigma = \{a_1, \cdots, a_K, b_1, b_2, c_1, \cdots, c_m\}.$$

$$P = \{a_i b_1 \mid 1 \leq i \leq K\}, Q = \{a_i b_2 \mid 1 \leq i \leq K\} \cup \{c_i b_1 \mid 1 \leq i \leq m\}.$$

In *Pairs*, we have $(a_i b_1, a_j b_2)$ for $1 \leq i, j \leq K$ and $(a_i b_1, c_j b_1)$ for $1 \leq i \leq K, 1 \leq j \leq m$. One of the optimum indexings of this instance is, for example,

$$f(\sigma) = \begin{cases} 1 & \text{if } \sigma = a_i \text{ with } 1 \leq i \leq K \\ i & \text{if } \sigma = b_i \\ 2 & \text{if } \sigma = c_i \text{ with } 1 \leq i \leq m \end{cases}$$

and this distinguishes all K^2 of $(a_i b_1, a_j b_2)$'s by $f(b_1) \neq f(b_2)$ and all mK of $(a_i b_1, c_j b_1)$'s by $f(a_i) \neq f(c_j)$. Although the algorithm may assign K different symbols to $a_1, \cdots, a_K$ firstly, with saving $2(i-1)$ pairs at each i-th decision of index, and then different two symbols to b_1, b_2. This results that all pairs of the form $(a_i b_1, a_j b_2)$ to be distinguished, but for any succeeding indexing to each c_j, we can save only $K-1$ pairs of $(a_i b_1, c_j b_1)$'s. For $m \to \infty$, we have the ratio $Greedy_K(\pi')/OPT(\pi')$ goes to the bound

$$\frac{K(K-1) + K + m(K-1)}{K^2 + mK} \to \frac{K-1}{K} ,$$

and thus $\epsilon = 1/K$. □

4 SNP-hardness of Max K-Indexing

Lemma 1. Max K-Indexing is MAX SNP-hard.

Proof. We show a L-reduction from MAX SNP-hard problem MAX CUT [7]. Given a graph $G = (V, E)$, MAX CUT is the problem to find the partition of the vertices S and $\bar{S} = V - S$ that maximizes the number of edges going from S to $\bar{S}$. The algorithm τ builds an instance of Max 2-Indexing for an instance G of MAX CUT as follows. For every i-th vertex v_i of V we have a symbol v_i in Σ. For each edge (v_i, v_j) of E (assume $i < j$) we have two strings $(v_i)^i$ in P and $(v_j)^i$ in Q. Another polynomial-time algorithm ρ is given as follows. If an indexing f of $\tau(G)$ maps v_i to 1, $\rho(f)$ partitions v_i into S, otherwise partitions v_i into $\bar{S}$.

Notice that any pair in $Pairs(P,Q)$ corresponds to the only one edge in E, and the transformation of solutions ρ is one to one and on to mapping. We define the reverse mapping of ρ, say, ρ^{-1}, by mapping v_i to 1 if and only if $v_i \in S$.

Now we show the first condition of L-reduction holds for $\alpha = 1$. From now on, we refer to the problem MAX CUT by $\mathcal{C}$ and to Max K-Indexing by $\mathcal{I}$. Let S' be a partition of G. Any pair $((v_i)^i,(v_j)^i)$ can be distinguished by $f' = \rho^{-1}(S')$ if and only if the corresponding edge (v_i, v_j) goes from S' to $\bar{S'}$ and vice versa. Then S' is optimum if and only if f' is optimum: if there exists a better solution for G, it implies that there also exists a better solution for $\tau(G)$. Therefore $OPT(G) = OPT(\tau(G))$ and the first condition of the L-reduction holds.

Next we show the remaining condition $OPT(G) - \mu_{\mathcal{C}}(G, \rho(f)) \leq \beta(OPT(\tau(G)) - \mu_{\mathrm{I}}(\tau(G), f))$ holds for $\beta = 1$. Let f be an indexing of $\tau(G)$. For each pair in $Pairs(P,Q)$, we have the corresponding edge in E, and if f distinguishes a pair $((v_i)^i, (v_j)^i)$ then $f(v_i) \neq f(v_j)$. By the definition of ρ, we can count the edge (v_i, v_j) as it going from S to $\bar{S}$ for $S = \rho(f)$, and this depends only whether $f(v_i) \neq f(v_j)$. Therefore, we have $\mu_{\mathcal{C}}(G, \rho(f)) = \mu_{\mathcal{I}}(\tau(G), f)$ and the second condition of L-reduction holds for $\beta = 1$. □

5 Conclusion

We have shown the polynomial-time approximation algorithm for Max K-Indexing, and shown the error ratio and the existence of the worst case examples for this algorithm. However, the error ratio $1/K$ seems wrong, especially when a few indices are available, such as $K = 2$. In the recent researches, we have some results on approximation algorithms for MAX K-SAT and MAX CUT with better error ratio [3, 4]. Is there any better approximation algorithm for MAX K-INDEXING? And more, for the result of MAX SNP-hardness, is Max K-Indexing with constant length bounded strings MAX SNP-hard? These are arisen as the open problems.

References

[1] Arora, S., Lund, C., Motwani, R., Sudan, M. and Szegedy, M., "Proof verification and hardness of approximation problems", in *Proc. 33rd Annual Symposium on Foundations of Computer Science*, pp. 14–23, 1992

[2] Blum, A., Jiang, T., Li, M., Tromp, J. and Yannakakis, M., "Linear approximation of shortest superstrings", in *Proc. 23rd Annual ACM Symposium on Theory of Computing*, pp. 328–336, 1991

[3] Goemans, M. X. and Williamson, D. P., "New 3/4-approximation algorithms for max sat", in *Proc. 3rd IPCO*, pp. 313–321, 1993

[4] Goemans, M. X. and Williamson, D. P., ".878-approximation algorithms for max cut and max 2sat", in *Proc. 26th Annual ACM Symposium on Theory of Computing*, pp. 422–431, 1994

[5] Johnson, D. S., "Approximation algorithms for combinatorial problems", *J. Comput. Sys. Sci.*, vol. 9, pp. 256–278, 1974

[6] Kyte, J. and Doolittle, R. F., "A simple method for displaying the hydropathic character of protein", *J. Mol. Biol.*, vol. 157, pp. 105–132, 1982

[7] Papadimitriou, C. H. and Yannakakis, M., "Optimization, approximation, and complexity classes", *J. Comput. Sys. Sci.*, vol. 43, pp. 425–440, 1991

[8] Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S. and Arikawa, S., "Finding alphabet indexing for decision trees over regular patterns", in *Proc. Twenty-Sixth Annual Hawaii International Conference on System Sciences*, pp. 763–772, 1993

[9] Shimozono, S. and Miyano, S., "Complexity of finding alphabet indexing", *IEICE Trans. Inf. Sys.*, vol. 1, 1995

[10] Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S. and Arikawa, S., "Knowledge acquisition from amino acid sequences by machine learning system bonsai", *Trans. Inf. Proc. Soc. Japan*, vol. 35, pp. 2009–2018, 1994