情報処理学会論文誌　数理モデル化と応用　Vol.7　No.2　27–35 (Nov. 2014)

メタ文字を含む文字列に対する Vantage-Point木を用いた類似文字列検索

森川 浩司[1,a)]　高梨 勝敏[1]　宗形 聡[1]

受付日 2014年2月3日，再受付日 2014年3月20日，
採録日 2014年4月18日

概要：複数の数値から 1 つ選択することを表すメタ文字を含む文字列から，メタ文字を含まない文字列と類似した文字列を高速に抽出する技術を提案する．文字列類似度指標として編集距離を用い，検索の高速化のために Vantage-Point 木を用いる．メタ文字に対応するために数字文字列を単位文字とする編集距離を定義した．木の構築では編集距離として Hausdorff 距離を用い検索では上記定義の編集距離を用いることで検索の高速化を実現した．今回提案する技術は品番がさまざまな仕様の組合せで表現されている産業用製品・部品の品番管理と類似品番検索に有用な技術である．

キーワード：編集距離，ハウスドルフ距離，VP 木，メタ文字列，類似文字列検索

An Approximate String Matching Method Which Uses Vantage-Point Tree for Strings Containing Meta-characters

Kohji Molikawa[1,a)]　Katsutoshi Takanashi[1]　Satoshi Munakata[1]

Received: February 3, 2014, Revised: March 20, 2014,
Accepted: April 18, 2014

Abstract: We propose an approximate string matching method for string containing meta-characters which mean selecting a single number from some choices, when query string does not contain the meta-characters. We use edit distance as similarity measure of strings and Vantage-Point tree for accelerating the search. We use Hausdorff distance as a distance metric for constructing Vantage-Point tree and a distance metric which treats a number string as a unit character for approximate matching of string containing the meta-characters for searching in Vantage-Point tree. The proposed method is valuable for managing and approximate search of industrial materials database whose item numbers are quite similar to each other.

Keywords: edit distance, Hausdorff distance, Vantage-Point tree, approximate string matching, meta-character string

1. はじめに

産業用の製品や部品は多種多様なニーズに応じてさまざまな仕様が設定されている．これらは品番で管理されるが，寸法がわずかに違うシリーズ品やバリエーションも含めると品番の数は膨大になる．そこでメタ文字を使って品番を記述すると品番の数を圧縮できるのでデータベースの管理などが便利になる．たとえば ABC1, ABC1.5, ABC2, . . . , ABC9.5, ABC10 という，数字が 1 から 10 まで 0.5 刻みで連続的に変化しているものについては ABC{1..10(0.5)} と表すことができる．また，XY1Z, XY2Z, XY3Z, XY5Z, XY8Z のように離散的な場合には XY{1|2|3|5|8}Z と表すことができる．アイテム取扱い総数が 500 垓にのぼるのに品番をメタ文字表記することで 200 万種類に抑えている事例もある [1]．産業部品では仕様の差異は数値で表されることが多いのでメタ文字として数値の範囲を表現するものだけを本稿で扱う．このメタ文字表記を数値選択メタ文字表記と呼ぶ．数値選択メタ文字表記と普通の文字表記が混在する文字列を数値選択メタ文字列と呼ぶ．

[1] 株式会社日立ソリューションズ東日本
Hitachi Solutions East Japan, Ltd., Sendai, Miyagi 980–0014, Japan
[a)] koji.morikawa.hz@hitachi-solutions.com

産業用の製品や部品の品番のグループを数値選択メタ文字列で表記し，この数値選択メタ文字列に対する類似文字列検索を実現することで産業界のさまざまなニーズに応えることができるようになる．たとえば製品受注時に注文品が欠品していた場合，少しの仕様の差であれば類似品が注文品の代替品として受け入れられる場合がある．注文品と代替品とで長さなど寸法の差がわずかでその差が許容範囲内である場合や，代替品の方が注文品より強度が高いなど性能面で代替が効く場合などである．旧製品と後継の新製品での代替が効く場合もある．この場合は品番のプレフィックス部分が旧製品のそれとわずかに違っていることが多い．こういった場合は機会損失を防ぐために数値選択メタ文字表記の品番の類似品を検索し代替品として提案したいというニーズがある．製品を在庫させずに注文に応じて生産する注文生産ではできるだけ同じものを多く生産した方が生産コストを低くできる．見込生産と同様に少しの仕様の差であれば類似品が注文品の代替品として受け入れられる場合には，生産コストを下げるために類似品を代替品として提案したいというニーズがある．さらに，保守・修理サービスにおいて緊急に修理が必要なときに必要な部品が在庫せず代替品で応急処置をしたい場合に該当品の品番から類似品をすばやく検索して対応したいというニーズがある．

以上のように数値選択メタ文字列に対する類似文字列検索技術は産業上の価値が高い．

2. 従来技術

メタ文字を含む文字列に関する類似文字列検索技術としては近似正規表現文字列検索（Approximate Regular Expression Matching）[2], [3] が存在する．しかしこれは検索語（クエリ）側にメタ文字が存在し，検索対象側はメタ文字を含まないことを前提としている．たとえば，ユーザが何か分からないことがあって検索を行う場合に正規表現を使ってクエリを表現の幅を持たせて記述し，さらにそのクエリに対して検索結果にも幅を持たせる形になっている．一方，今回提案する技術はクエリにはメタ文字はなく，検索対象側にメタ文字を含む点で近似正規表現文字列検索と異なっている．たとえば，ユーザは明確な意図を持ってその文字を入力したが，数値の幅を持つどの検索対象文字列とも一致しないような場合にクエリに対する類似文字列を提示するものである．両者の違いを図 **1** に示す．

メタ文字表記のような文字集合を文字として含む文字列としては indeterminate string または degenerate string が知られている．Indeterminate string に対する Truncated Generalized Suffix Automaton を用いた索引構造が提案されている [4] が，これは厳密パターン照合しかできない．また versioned documents に対する索引構造 [5] も indeterminate string に適用できるが，同様に厳密パターン照合しかできない．

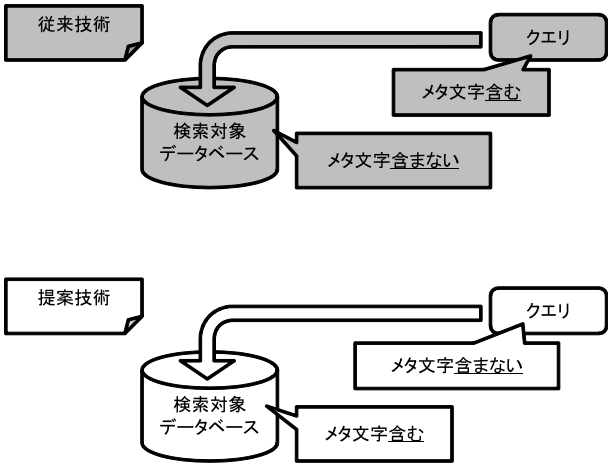

図 1　従来技術と提案技術の違い
Fig. 1　Difference between proposed and conventional methods.

一方，indeterminate string に対する最長共通部分列（Longest Common Subsequence）が提案されている [6] ので，これと先ほどの索引構造を用いて最長共通部分列長が長いほど類似度が高いと考えて類似文字列検索を行う方法も考えられる．しかしこれはクエリ文字列と検索対象文字列との長さの差を考慮していない．この差を含めて 1 つの指標で扱える編集距離の方が今回の産業ニーズに対しては有用性が高い．

今回提案する技術は Google や Yahoo!などの検索エンジンの技術や Amazon や楽天などの EC サイトの検索技術とも異なる．Google や Yahoo!の検索エンジンの技術の詳細は公表されていないがこれらの検索エンジンの検索結果はクエリを部分文字列として含むウェブページやドキュメントであり検索対象にはメタ文字列は含まれず，クエリの表記ゆらぎは辞書登録するなど表記ゆらぎはクエリ側に想定されていると思われる．Google の検索において誤入力などがあった場合は「もしかして」と代替クエリ文字列が提示されるがこれも誤入力と正しいクエリのペアの登録と統計処理によってクエリを修正しようとするものである．Amazon の商品検索についても同様と推測される．また楽天 [7] では文字列の類似度評価に編集距離を用いているが，次章で述べるように編集距離はそのままではメタ文字列に対する類似度評価には使えない．

3. 数値選択メタ文字表記に対する文字列類似度評価指標の不在

2 つの文字列間の類似度を定量的に表す指標としては編集距離や n-gram が代表的である [8]．編集距離は距離の概念を満たすため，Vantage-Point 木（VP 木）など距離の概念に基づくインデックスデータ構造 [9] を用いて検索を高速化できる．一方 n-gram は距離の概念を完全には満たさず，かつ n-gram はその構造のために保持するデータサイ

ズが大きくなる．今回は文字列の数が非常に多いという前提であるため VP 木などの高速なデータ構造を利用でき，かつデータサイズが大きくならない編集距離の適用を検討する．

3.1　編集距離

編集距離は 2 つの文字列がどの程度異なっているかを定量的に表す指標の 1 つである．文字列 s と文字列 t との間の編集距離 $d(s,t)$ は，s を構成する文字に対して以下の操作を施して t を生成する際の操作コストの和の最小値として定義される [10], [11].

- 置換：s を構成する文字の 1 つをそれとは異なる t を構成する 1 つの文字に変える操作
- 削除：s にあって t にない文字を s から 1 つ削除する操作
- 挿入：s になくて t にある文字を s に挿入する操作

これらの操作のコストを等しく 1 とする．これらの操作によって文字列 s から文字列 t を生成する方法はさまざまあり，その方法によって操作コストの和はさまざまな値をとる．編集距離は文字列 s から文字列 t を生成するその方法は問わず，上記操作コストの和の最小値として定義される．編集距離の計算例を付録 A.1 に記した．

編集距離は文字列の構成文字に対する操作で定義されるが，数値選択メタ文字の中の数値の文字数はさまざまである．したがって数値選択メタ文字列についてはこのままでは編集距離を定義できないという課題がある．なお，この数値選択メタ文字列について類似度を評価できないという課題は n-gram 方式においても存在する．

3.2　Vantage-Point 木

編集距離が大きいほど 2 つの文字列は異なっていることになるので，編集距離を用いてクエリと類似した文字列を検索対象文字列群から抽出することができる．検索前に編集距離のしきい値を決めておき，検索文字列に対する編集距離がそのしきい値以下の文字列を検索文字列と類似した文字列として抽出する．

編集距離のように類似度が距離の定義を満たす場合，距離をインデックスとする木構造を用いることで検索を高速化できる．そこで距離をインデックスとする木構造の中で最も一般的な VP 木 [12], [13] を利用して検索を高速化することを検討した．

VP 木は距離をインデックスとする 2 分探索木の 1 つであり，子孫ノードを持つすべてのノードについて以下の 3 点が成り立つ．

- ノードはすべての子孫ノードとの距離の中央値 M を持つ．
- ノードとその左側子孫ノードとの距離はノードが持つ中央値 M 以下である．

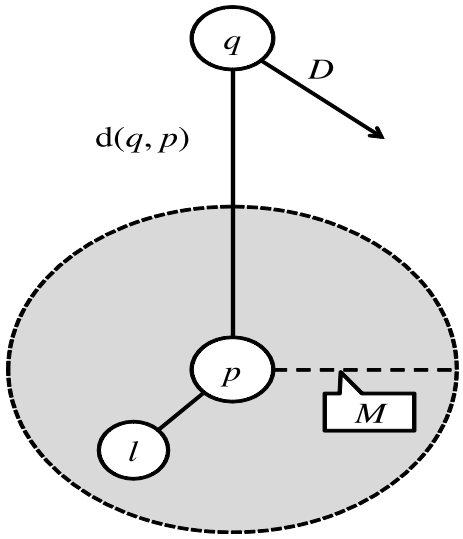

図 2　文字列 q，p，l の距離の関係
Fig. 2　Distance relation among strings q, p, l.

- ノードとその右側子孫ノードとの距離はノードが持つ中央値 M より大きい．

VP 木の例を付録 A.2 に記した．

今，検索文字列を q，VP 木のあるノードの文字列を p，文字列 p のノードの任意の左側子孫ノード文字列を l，文字列 p のノードの任意の右側子孫ノード文字列を r とする．検索文字列との編集距離が D 以下の文字列を VP 木で検索する場合，次のことがいえる．

(1)　$D + M < d(q,p)$ が成り立てばそのノードの左側の枝は検索しなくてよい

これを左側の枝刈りと呼ぶ．$D + M < d(q,p)$ が成り立つとき，距離の 3 角不等式 $d(q,p) \leq d(q,l) + d(l,p)$ から $D < d(q,l)$ となる．これは文字列 p のノードの左側の任意のノード文字列について成り立つ．したがって $D + M < d(q,p)$ が成り立つとき，文字列 p のノードの左側のノード文字列で検索文字列 q との間の編集距離が D 以下になるものは存在しない．**図 2** に q，p，l の関係を示す．文字列 p のノードの左側すべての子孫ノードは文字列 p のノードから半径 M の内側にあるので，$D + M < d(q,p)$ が成り立つときは文字列 p の左側のどの文字列も文字列 q からの距離は D よりも大きくなる．

(2)　$D + d(q,p) \leq M$ が成り立てばそのノードの右側の枝は検索しなくてよい

これを右側の枝刈りと呼ぶ．$D + d(q,p) \leq M$ が成り立つとき，距離の 3 角不等式 $d(p,r) \leq d(p,q) + d(q,r)$ から $D < d(q,r)$ となる．これは文字列 p のノードの右側の任意のノードの文字列について成り立つ．したがって $D + d(q,p) \leq M$ が成り立つとき，文字列 p のノードの右側ノード文字列で検索文字列 q との間の編集距離が D 以下になるものは存在しない．**図 3** に q，p，r の関係を示す．文字列 p のノードの右側のすべての子孫ノードは文字列 p

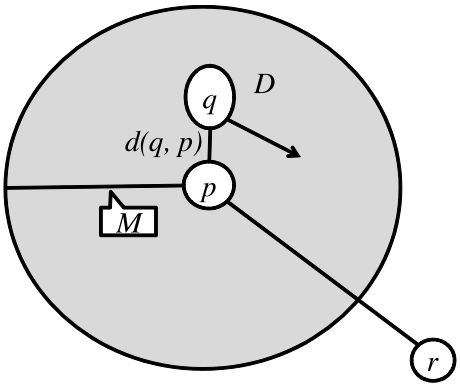

図 3　文字列 q，p，r の距離の関係
Fig. 3　Distance relation among strings q, p, r.

のノードから半径 M の外側にあるので，$D + d(q,p) \leq M$ が成り立つときは文字列 p の右側のどの文字列も文字列 q からの距離は D よりも大きくなる．

これらの枝刈り条件はノード間の距離が数学的な距離の定義を満たしている場合に限り成立するため，数学的な距離の定義を満たしていない文字列類似度の場合には使えないという課題がある．

4. 数字 1 文字編集距離と VP 木における Hausdorff 距離の適用

4.1　数字 1 文字編集距離の導入

編集距離を数値選択メタ文字列に適用するために，数字文字列を単位文字として扱う編集距離を提案する．編集距離の計算に関して以下のルールを設定する．

編集距離計算ルール (1)

数値文字列は単位文字として扱う．以降，これを単位数値文字と呼ぶ．

このルールによって数値選択メタ文字部分にある選択肢の数値はどれも長さ 1 の単位文字となる．なお，数値選択メタ文字部分以外の数値文字列も単位文字扱いとする．たとえば {1|20|300} は「1」という単位文字と「20」という単位文字と「300」という単位文字の中から 1 文字選択することを表す．これにともない以下のルールも設定する．

編集距離計算ルール (2)

数値選択メタ文字部分も単位文字として扱う．

数値選択メタ文字は選択肢の単位数値文字の中から 1 つ選ぶことを表しているので，編集距離ルール (1) からこのルールは自動的に導かれる．以上のルールから，たとえば A{1|20|300}B45 を単位文字に分解する場合「A」「{1|20|300}」「B」「45」となる．

単位数値文字という新しい単位文字を導入したので，これに対応して文字の操作とそのコストとして以下のルールを設定する．

編集距離計算ルール (3)

単位数値文字 n と N 個の単位数値文字からなる数値選択メタ文字 $\{n_1|\ldots|n_N\}$ との置換コストは以下のとおりとする．

- n が $n_1, \ldots, n_N$ のどれかと一致すれば 0.
- n が $n_1, \ldots, n_N$ のどれとも一致しなければ 1.

なお，単位数値文字と他の単位文字との置換・削除・挿入に関しては単位数値文字および数値選択メタ文字部分は単位文字であるので従来の定義の枠内で扱える．

以上のルールを設定すると，数値選択メタ文字を含まない文字列 s と，数値選択メタ文字を含まない文字列 $t_1,\ldots,t_N$ の集合である数値選択メタ文字列 T との距離 $d(s,T)$ は点・集合間距離 $d(s,T) = \min_i \{d(s,t_i)|t_i \in T\}$ となる．したがって，s と T との間の距離については以下が成り立つ．

- s が $t_1,\ldots,t_N$ のどれかに一致する場合，およびそのときに限り距離 0.
- s と T について対称：$d(s,T) = d(T,s) = \min_i \{d(s,t_i)|t_i \in T\}$.
- 数値選択メタ文字を含まない文字列を u とするとき 3 角不等式は $d(s,T) \leq d(s,u) + d(u,T)$

以降，これで定義される編集距離を数字 1 文字編集距離と呼ぶ．

VP 木の構築には数値選択メタ文字列間の編集距離の定義が必要であるので，編集距離計算ルールとして以下も設定する．

編集距離計算ルール (4)

M 個の数値文字からなる数値選択メタ文字 $\{m_1|\ldots|m_M\}$ と N 個の数値文字からなる数値選択メタ文字 $\{n_1|\ldots|n_N\}$ との置換コストは以下のとおりとする．

- 単位数値文字の集合 $\{m_1|\ldots|m_M\}$ と $\{n_1|\ldots|n_N\}$ とが集合として等しければ 0.
- そうでなければ 1.

数値選択メタ文字を含む文字列と含まない文字列とが混在しているデータに対して以上のルールを適用して VP 木を構築して検索を行うと検索漏れが発生した．理由を次に述べる．

4.2　VP 木の構築における Hausdorff 距離の適用

これまで設定したルールに則って作られた VP 木では左右の枝刈りの前提となる距離の 3 角不等式は一般に成り立たない．数値選択メタ文字を含まない文字列を q，s，t，数値選択メタ文字列を U，V とするとき，これまでの定義から成り立つ距離の 3 角不等式は

$$d(q,s) \leq d(q,t) + d(t,s), \quad d(q,U) \leq d(q,s) + d(s,U)$$

だけであり，左側の枝刈り条件成立の前提となる 3 角不等式 $d(q,s) \leq d(q,U) + d(U,s)$ や $d(q,U) \leq d(q,V) + d(V,U)$ また右側の枝刈り条件成立の前提となる 3 角不等式 $d(U,V) \leq d(U,q) + d(q,V)$ は一般には成り立たないからである．また，距離の 3 角不等式が成り立っているノー

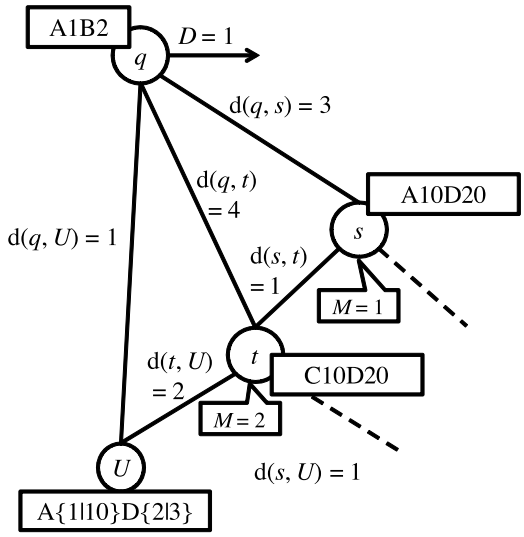

図 4　左側の枝刈り条件が成立しても枝刈りしてはいけない例
Fig. 4　A case in which left nodes are not prunable.

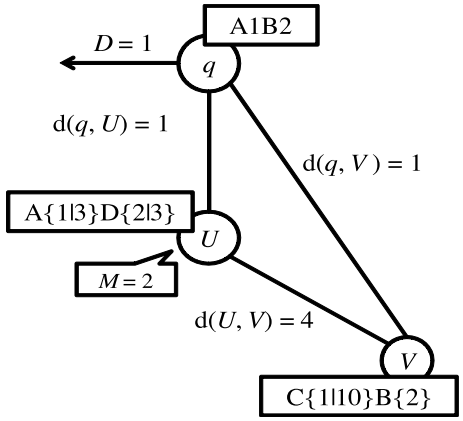

図 5　右側の枝刈り条件が成立しても枝刈りしてはいけない例
Fig. 5　A case in which right nodes are not prunable.

ドでもその子孫ノードでは数値選択メタ文字列と数値選択メタ文字を含まない文字列が混在しているため，枝刈りの前提となる距離の 3 角不等式がすべての子孫ノードで成り立つとは一般にはいえない．図 4 にその例を示す．検索文字列 q（文字列：A1B2）と検索対象ノード s（文字列：A10D20，中央値 $M = 1$），t（文字列：C10D20，中央値 $M = 2$），U（文字列：A{1|10}D{2|3}）についてノード s の左側の枝刈り条件が成り立っているが，ここで枝刈りを行うと抽出されるべき数値選択メタ文字列 U を逃してしまう．したがってこのままでは VP 木では枝刈りしてはいけないことになってしまい，VP 木を利用するメリットが失われる．そこで数字 1 文字編集距離に合わせた VP 木構築方法と検索方法を提案する．まず，VP 木構築において以下のルールを設定する．

VP 木構築ルール (1)

数値選択メタ文字を含む文字列と含まない文字列は分けて別々に VP 木を構築する．

検索文字列は数値選択メタ文字を含まないので，このルールによって数値選択メタ文字を含まない VP 木には通常の編集距離での計算と枝刈り条件が適用できる．

さらに，数値選択メタ文字列の VP 木については次のルールを設定する．

VP 木構築ルール (2)

数値選択メタ文字以外の数値部分をメタ文字で囲む．

このルールによって数値選択メタ文字列間の編集距離の計算時には編集距離ルール (3) ではなく編集距離ルール (4) が適用され，数値選択メタ文字列間の編集距離は Hausdorff 距離となる．その結果メタ文字を含まない文字列 q とメタ文字列 U，V との間の距離の 3 角不等式として

$$d(q,U) \le d(q,V) + d(V,U)$$

が成り立つようになるため，VP 木の左側の枝刈り前提条件となる 3 角不等式が成り立つようになる．したがって数値選択メタ文字列の VP 木でも左側の枝刈り条件が成り立つ場合には左側は枝刈りができる．しかし右側は依然として枝刈りはできない．距離の 3 角不等式 $d(U,V) \le d(U,q) + d(q,V)$ は一般には成り立たないからである．図 5 に例を示す．検索文字列 q（文字列：A1B2）と検索対象ノード U（文字列：A{1|3}D{2|3}，中央値 $M = 2$），V（文字列：C{1|10}B{2}）について，ノード U の右側の枝刈り条件が成り立っているが，ここで枝刈りを行うと抽出されるべき V を逃してしまう．

そこで数値選択メタ文字 VP 木の検索時には以下のルールを設定する．

数値選択メタ文字 VP 木の検索ルール

- 左側の枝刈り条件が成り立てば左側は枝刈りする．
- 右側子ノードはつねに検索する．

このルールによって VP 木全体の右側が枝刈りできなくなるのではなく，各ノードの右側の枝刈り判定をしないだけである．

4.3　数字 1 文字編集距離と Hausdorff 距離の計算量

数値選択メタ文字 $\{n_1|\ldots|n_N\}$ の選択肢の数の，品番データ中の最大値を N_C とする．単位数値文字と上記数値選択メタ文字との文字比較は最大 N_C 回の数値の比較を要する．したがって単位文字数 N_1 の非数値選択メタ文字列と単位文字数 N_2 の数値選択メタ文字列間の数字 1 文字編集距離を動的計画法で計算する場合，計算量は最悪で $O(N_C N_1 N_2)$ である．

数値選択メタ文字は選択肢の最小値を min，最大値を max，刻み幅 step とするときに {min.. max(step)}（以下，範囲型）というように書くこともでき，これは $\{n_1|\ldots|n_N\}$ の形（以下，選択型）に 1 対 1 で書き直すことができる．しかし一般に範囲型を選択型に書き直すと選択肢の数が莫大になるため範囲型は範囲型のままで編集距離計算を行う．この場合は，単位数値文字と範囲型の比較はその単位文字

が範囲型の min 以上 max 以下で step の刻み上にあるか判定すればよいので，通常計算回数は N_C 同程度かそれ以下となることが期待される．

以上から，数字 1 文字編集距離の計算量は最悪で $O(N_C N_1 N_2)$ であるといえる．

同じ単位数値文字集合は範囲型と選択型の両方の表記で書くことをせず必ずどちらか一方のみで表記することにすれば，範囲型と選択型との編集距離計算はメタ文字表記を文字列として比較すればよい．また，数値選択メタ文字における選択肢の数値は必ず昇順で列挙することにし，数値の表記ゆれがないようにすれば，2 つの数値選択メタ文字どうしの比較は文字列として比較すればよい．以上から Hausdorff 距離計算時は数値選択メタ文字表記どうしの比較は単位文字集合の表記が文字列として一致するかで比較すればよい．以上から，数値選択メタ文字表記部分の文字列長の最大値を L_M とすると，単位文字数がそれぞれ N_1 および N_2 の数値選択メタ文字列間の Hausdorff 距離を動的計画法で計算する場合，計算量は最悪で $O(L_M N_1 N_2)$ である．

5. 品番データによる検索性能の検証

ある商品の品番のうち，数値選択メタ文字列となっている品番データ 240 万件とこの品番データにないデータ（非存在品番）1 万件を使って前章の検証を行った．

5.1 検証の環境とデータ

240 万件の数値選択メタ文字列を検索対象文字列として検証を行った．検索文字列となる非存在品番は 5 万件程度あるが，検証用に次のようにして 1 万件を選んだ．

240 万件ある検索対象の数値選択メタ文字列の中からランダムに 15 万件を選び，非存在品番である 5 万件の検索を行った．検索のしきい値は非存在品番の文字列長の 25%とした．なお，小数点以下は切り捨てとした．たとえば，非存在品番の文字列長が 10 であればしきい値は 2 である．この検索で検索結果が 1 個以上あるものの中からランダムに 1 万件を選び，これを検証用検索文字列とした．検証の環境とデータについて**表 1** にまとめた．

データ数の変化による性能の変化を見るためにデータ数を次のように変化させた．上記 15 万件，その 15 万件を含む 30 万件，その 30 万件を含む 60 万件，その 60 万件を含む 120 万件，全体 240 万件の 5 パターンである．

表 1　検証の環境とデータ
Table 1　Data and calculation environment.

検証用 PC	Dell Precision T5600
メモリ	64 GB
CPU	Intel Xeon E5-2650 2.0 GHz
開発言語	Microsoft Visual C
検証対象データ	数値選択メタ文字列 240 万件
検索文字列データ	非存在品番 1 万件

5.2 数字 1 文字編集距離の計算量の測定

数値選択メタ文字列 240 万件から重複を許してランダムに 10 万ペアの文字列を抽出し各ペアの編集距離を計算するというものを 1 セットとして，100 セットの測定を行った．そのペアの文字列の特徴を**表 2** にまとめた．「単位文字数の平均」は 1 文字列あたりの単位文字数の全サンプルの平均値である．「1 文字列あたりのメタ文字個数の平均」は 1 文字列あたりの数値選択メタ文字表記の個数の全サンプルの平均値である．「選択肢の最大値のセット平均」はセットごとの 10 万ペア中の数値選択メタ文字表記の選択肢の最大値のセット平均である．「メタ文字部分文字列長の最大値のセット平均」はセットごとの 10 万ペア中の数値選択メタ文字表記の文字列長の最大値のセット平均である．

このサンプルによる数字 1 文字編集距離および Hausdorff 編集距離の動的計画法による計算時間のセット平均値と標本偏差を**表 3** に記した．なお，比較対象の従来技術編集距離は，サンプルの数値選択メタ文字表記部分を数値選択メタ文字ではない任意の 1 文字と置換して動的計画法による従来技術の編集距離計算を行ったものである．

各編集距離測定間で偏差が異なるので平均値がどの程度有意に異なるかはこの結果から一概にはいえないが，数字 1 文字編集距離および Hausdorff 編集距離の計算時間は従来編集距離に比べて平均して「1 文字列あたりのメタ文字個数の平均」倍程度余計にかかるといえる．

5.3 VP 木における検索漏れ発生率の計測

VP 木構築ルールならびに VP 木検索ルールを適用しない場合と適用した場合で VP 木検索においてどの程度検索漏れが発生するかを計測した．計測結果を**表 4** に示す．1 件の検索で 1 文字列でも漏れが発生した場合にはその検索

表 2　測定数値選択メタ文字列の特徴
Table 2　Properties of the meta-character strings.

	1 つ目の数値選択メタ文字列	2 つ目の数値選択メタ文字列
単位文字数の平均	17.6	17.6
1 文字列あたりのメタ文字個数の平均	4.0	4.0
選択肢の最大値のセット平均	30.4	30.6
メタ文字部分文字列長の最大値のセット平均	98.2	111.2

表 3　数字 1 文字編集距離の平均計算量
Table 3　Mean values of edit distance calculations.

	数字 1 文字編集距離	Hausdorff 編集距離	従来技術編集距離
セット平均時間	0.650	0.553	0.235
セットの偏差	0.029	0.013	0.006

表 4　検索漏れの発生頻度
Table 4　Occurrence frequency of missing.

データ数	構築・検索ルール非適用（比率）	構築・検索ルール適用（比率）
15 万	982（10%）	0（0%）
30 万	1,461（15%）	0（0%）
60 万	2,073（21%）	0（0%）
120 万	2,620（26%）	0（0%）
240 万	3,193（32%）	0（0%）

は検索漏れとカウントし，データ数別に 1 万件の検索で何件漏れが発生したかを示す．なお括弧中のパーセンテージは漏れ発生件数の検索 1 万件に対する比率である．表 4 からデータ数が多いほど漏れの割合も高くなることが分かる．したがって VP 木構築ルールならびに VP 木検索ルールの適用は必須である．

なお，VP 木構築時はすべてのノードで親ノードとなるノードはランダムに選ぶため，同じ非存在品番の検索でも VP 木のどこにどの文字列があるかによって検索性能は異なる．そこで，非存在品番 1 万件を 10 セットに分け，同じ検索対象文字列についてセットごとに VP 木を再作成して検索を行った．本来であれば 1 万件の非存在品番をグループ分けせず，この 1 万件について異なる VP 木で複数回検索を実施して計測すべきであるが検証に要する時間の都合でこのような設定にした．

5.4　検索漏れ対策を適用した VP 木検索の性能測定

VP 木構築ルールならびに VP 木検索ルールを適用した場合の 1 件あたりの VP 木検索時間の平均値をデータ件数別に**図 6** に示す．比較のために VP 木を用いない場合の検索所要時間と VP 木構築ルールならびに VP 木検索ルールを適用しない VP 木検索の検索所要時間も合わせて図 6 に示す．あわせて，各検索所要時間の標本標準偏差（標本偏差）と変動係数を**表 5** に示す．図 6 および表 5 から VP 木検索ルールを適用してもなお，データ数によらず VP 木の枝刈りのメリットを十分享受できていることが分かる．

なお，VP 木を用いない場合の検索は全件検索を行う必要はない．文字列長が L_s である文字列 s と文字列長が L_t である文字列 t との間の編集距離 $d(s,t)$ についてはその定義から $|L_s - L_t| \leq d(s,t) \leq L_s + L_t$ となる．これより $|L_s - d(s,t)| \leq L_t \leq L_s + d(s,t)$ が得られる．したがって検索対象文字列をその長さでグルーピングしておけば，文字列長が L_q である文字列 q に対して編集距離 D 以下の文字列を抽出する際には文字列長さ L が $|L_q - D| \leq L \leq L_q + D$ を満たす文字列についてだけ編集距離を計算すればよい．したがって図 6 の左列ではこの方法で検索し，これを文字列長グループ検索と表記した．

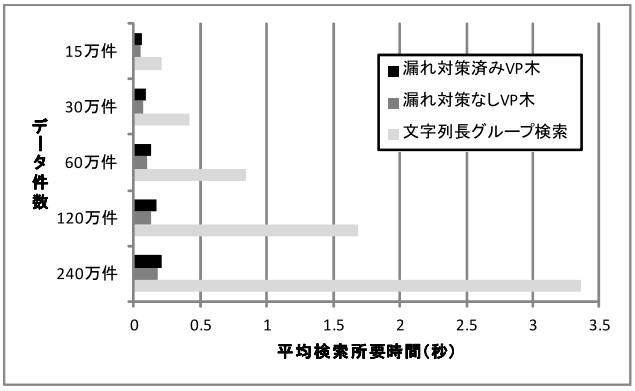

図 6　文字列長グループ検索と VP 木検索の性能比較
Fig. 6　Speed comparison between string length group search and VP tree searches.

表 5　検索時間の平均・標本偏差・変動係数
Table 5　statistical properties of search time.

	データ数	15 万	30 万	60 万	120 万	240 万
漏れ対策	平均（秒）	0.061	0.089	0.126	0.171	0.213
済み	標本偏差	0.034	0.049	0.070	0.094	0.122
VP 木	変動係数	0.56	0.55	0.56	0.55	0.57
漏れ対策	平均（秒）	0.046	0.068	0.096	0.134	0.178
なし	標本偏差	0.034	0.052	0.073	0.102	0.134
VP 木	変動係数	0.74	0.76	0.76	0.76	0.75
文字列長	平均（秒）	0.208	0.418	0.845	1.691	3.36
グループ	標本偏差	0.245	0.491	0.992	1.981	3.928
検索	変動係数	1.18	1.17	1.17	1.17	1.17

5.5　VP 木生成性能

データ数を n とすると VP 生成に要する時間は $O(n\log(n))$ である [12]．**図 7** に各データ数での 10 回の VP 木生成に要した時間の平均を点で，標準偏差をエラーバーで示した．破線はこのデータに対する $n\log(n)$ のフィッ

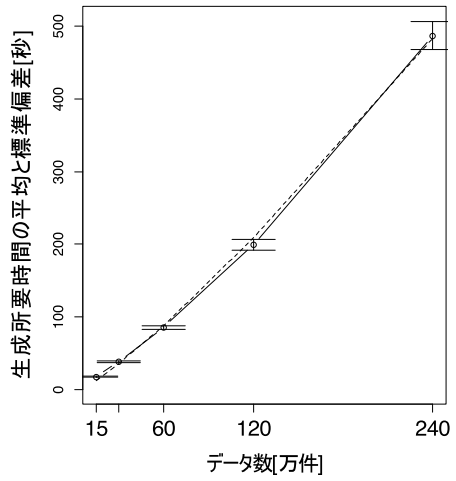

図 7　VP 木生成所要時間
Fig. 7　Elapsed time of VP tree construction.

ティング結果である．VP 木は検索のたびに作る必要はなく，VP 木を構成する文字列の変更がない限りは一度作った VP 木は使い続けることができる．図 7 からはデータ数がたとえば 1,000 万でもデータの更新があった際に夜間バッチで VP 木を再構築できるといえる．

6. おわりに

数値選択メタ文字列に対する類似文字列検索を実現するために，数字文字列を単位文字とする編集距離を定義した．この編集距離定義を用いて VP 木を構築して検索を行うと検索漏れが発生した．そこで VP 木の構築時は編集距離として Hausdorff 距離を用い，検索時は数字文字列を単位文字とする編集距離を用いることで検索漏れを回避できることを示した．この回避策では距離をインデックスとする木構造が持つ枝刈りのメリットが一部失われるが，それでもなお木構造を用いない検索よりも十分高速であることをデータで検証した．

今回提案した技術は，さまざまな仕様の組合せが設定される産業用製品・部品の品番管理と類似品番検索に有用な技術であるがこれまで実現されていなかった技術であり，産業上の価値がきわめて高い．

参考文献

[1] 日経 BP 社：経営新潮流 ミスミグループ本社 三枝匡の経営教室，日経ビジネス，2013 年 4 月 1 日号 (2013).
[2] Myers, E.W. and Miller, W.: Approximate matching of regular expressions, *Bulletin of Mathematical Biology*, Vol.51, pp.7–37 (1989).
[3] Wu, S. et al.: A subquadratic algorithm for approximate regular expression matching, *Journal of Algorithms*, Vol.19, No.3, pp.346–360 (1995).
[4] Flouri, T. et al.: Indexing Factors in DNA/RNA Sequences, *BIRD 2008*, pp.436–445 (2008).
[5] He, J. et al.: Compact full-text indexing of versioned document collections, *Information and Knowledge Management 2009*, pp.415–424 (2009).
[6] Iliopoulos, C.S. et al.: Algorithms for Two Versions of LCS Problem for Indeterminate Strings, *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vo.71, pp.155–172 (2009).
[7] 平手勇宇，竹中孝真，森　正弥：キーワード型検索エンジンにおける修正キーワード候補提示アルゴリズム，*DEIM Forum 2010*, B2-4 (2010).
[8] Navarro, G.: A guided tour to approximate string matching, *ACM Computing Surveys*, Vol.33, No.1, pp.31–88 (2001).
[9] Chavez, E. et al.: Searching in metric spaces, *ACM Computing Surveys*, Vol.33, No.3, pp.273–321 (2001).
[10] Levenshtein, V.I.: Binary codes capable of correcting deletions, *insertions and reversals, Soviet Physics Doklady*, Vol.10, No.8, pp.707–710 (1966).
[11] Wagner, R.A. and Fischer, M.J.: The string to string correction problem, *J. ACM*, Vol.21, pp.168–178 (1974).
[12] Yianilos, P.: Data structures and algorithms for nearest neighbor search in general metric spaces, *Proc. 4th ACM-SIAM Symposium on Discrete Algorithms*, pp.311–321 (1993).
[13] Hjaltason, G.R. and Samet, H.: Index-driven similarity search in metric spaces, *ACM Trans. Database Syst.*, Vol.28, No.4, pp.517–580 (2003).

付　　録

A.1　編集距離の計算例

文字列 solyusion に対して文字操作によって文字列 solutions を生成する例を示し，編集距離がどう計算されるかを示す．

(1) solyusion の 4 文字目の y の削除
　　sol**y**usion → solusion
(2) 新しくできた solusion の 5 文字目の s を t に置換
　　solu**s**ion → solu**t**ion
(3) 新しくできた solution の 9 文字目（最後尾）に s を挿入
　　solution_ → solution**s**

以上の 3 操作が文字列 solyusion を文字列 solutions に変換する最小操作数となるので，文字列 solyusion と文字列 solutions との間の編集距離は 3 である．なお，編集距離を計算する 2 つの文字列の長さは一般に異なってよい．

A.2　VP 木の例

図 **A·1** に 7 つの文字列「ABCDEFG」「ABC4EFG」「ABC45FG」「AB345FG」「A23456G」「A234567」「1234567」からなる VP 木の例を示す．ノード「ABCDEFG」とすべての 6 つの子孫ノード（編集距離の中央値 $M = 3$）について，また「ABC4EFG」とその 2 つの子ノード（編集距離の中央値 $M = 1$）について，そして「A23456G」とその 2 つの子ノード（編集距離の中央値 $M = 1$）について，それぞれ VP 木の性質が成り立っている．

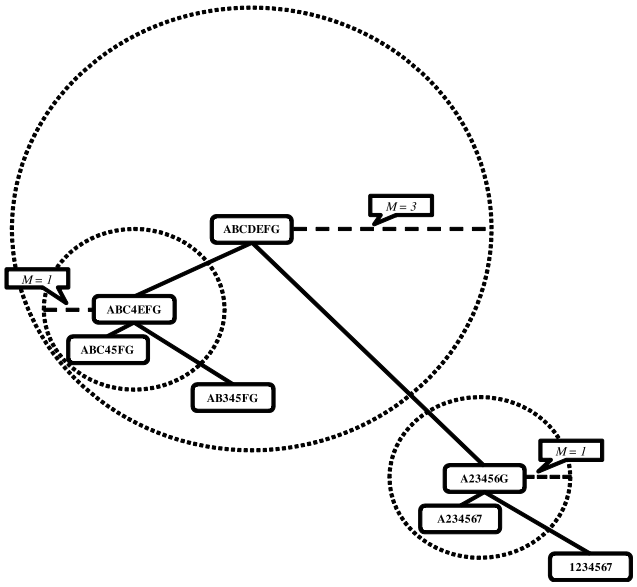

図 **A·1**　7 つの文字列からなる VP 木の例
Fig. A·1　An example of a VP tree composed of seven strings.

森川 浩司

2001 年東北大学大学院理学研究科天文学専攻博士課程修了．同年日立東北ソフトウェア株式会社（現，株式会社日立ソリューションズ東日本）入社．研究開発部所属．博士（理学）．電子情報通信学会会員．

高梨 勝敏　（正会員）

1995 年東北大学大学院理学研究科物理学専攻修士課程修了．同年日立東北ソフトウェア株式会社（現，株式会社日立ソリューションズ東日本）入社．現在，オントロジー工学およびデータアナリティクスの研究と事業化支援に従事．

宗形 聡　（正会員）

2003 年東北大学大学院理学研究科数学専攻修士課程修了．同年株式会社日立東日本ソリューションズ（現，株式会社日立ソリューションズ東日本）入社．製造・流通分野や金融分野での数理モデリング，データ分析技術の研究開発に従事．