Vol.2014-MPS-97 No.11
2014/3/3

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† 高梨勝敏† 宗形聡†

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み合わ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．

An Approximate String Matching Method which Uses Vantage-Point Tree for Strings Containing Meta-Characters

KOHJI MOLIKAWA† KATSUTOSHI TAKANASHI†
SATOSHI MUNAKATA†

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.

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等が便利になる．例えば 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 垓に上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]．特に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．

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．

例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極めて高い．

2. 従来技術

数値選択メタ文字列に対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

†(株)日立ソリューションズ東日本
Hitachi Solutions East Japan, Ltd.

Matching）[2]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．例えば，ユーザーが何かわ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も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．例えば，ユーザー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．

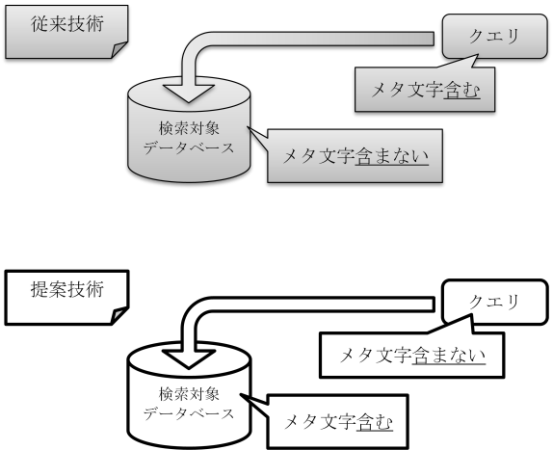

図 1　従来技術と提案技術の違い

また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[3]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[4]．編集距離は距離の概念を満たすため，VP 木など距離の概念に基づくインデックスデータ構造[5]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[6].

- s の 1 文字を異なる t の 1 文字に「置換」する
- t にない文字を s から 1 文字「削除」する
- s になくて t にある文字を s に 1 文字「挿入」する

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 s から t を生成するさまざまな方法の中での操作コストの和の最小値が編集距離として定義される．

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は桁すなわち文字数は様々である．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 VP 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[7]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．以降，子ノードと述べる際には孫以下のノードも含むものとする．

- ノードはそのノードの子ノードすべてとの間の距離の中央値 M を持つ
- ノードとそのノードの左側子ノードとの距離は，ノードが持つ中央値以下である
- ノードとそのノードの右側子ノードとの距離は，ノードが持つ中央値より大きい

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ノード文字列を l，文字

列 p のノードの任意の右側子ノード文字列を 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，次のことが言える．

(1) $D + M < \mathrm{d}(q, p)$ 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．$D + M < \mathrm{d}(q, p)$ が成り立つとき，距離の 3 角不等式 $\mathrm{d}(q, p) \leqq \mathrm{d}(q, l) + \mathrm{d}(l, p)$から $D < \mathrm{d}(q, l)$ 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 $D + M < \mathrm{d}(q, p)$ 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．

(2) $D + \mathrm{d}(q, p) \leqq M$ 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．$D + \mathrm{d}(q, p) \leqq M$ が成り立つとき，距離の 3 角不等式 $\mathrm{d}(p, r) \leqq \mathrm{d}(p, q) + \mathrm{d}(q, r)$ から $D < \mathrm{d}(q, r)$ 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 $D + \mathrm{d}(q, p) \leqq M$ 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．

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

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．例えば{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とになる．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，例えば 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$\{n_1|\cdots|n_N\}$との置換コストは以下のとおりとする．

- n が $n_1, \cdots, n_N$ のどれかと一致すれば 0
- n が $n_1, \cdots, n_N$ 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種との置換・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 $t_1,\cdots,t_N$ の集合である数値選択メタ文字列 T との距離 $\mathrm{d}(s, T)$ は点・集合間距離 $\mathrm{d}(s, T) = \min_i\{\mathrm{d}(s, t_i) \mid t_i \in T\}$ 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 $t_1,\cdots,t_N$ 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：$\mathrm{d}(s, T) = \mathrm{d}(T, s) = \min_i\{\mathrm{d}(s, t_i) \mid t_i \in T\}$
- 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 $\mathrm{d}(s, T) \leqq \mathrm{d}(s, u) + \mathrm{d}(u, T)$

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$\{m_1|\cdots|m_M\}$と N 個の数値文字からなる数値選択メタ文字$\{n_1|\cdots|n_N\}$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$\{m_1|\cdots|m_M\}$と$\{n_1|\cdots|n_N\}$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．理由は以下のとおりである．

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 $\mathrm{d}(q, s) \leqq \mathrm{d}(q, t) + \mathrm{d}(t, s)$ および $\mathrm{d}(q, U) \leqq \mathrm{d}(q, s) + \mathrm{d}(s, U)$ 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 $\mathrm{d}(q, s) \leqq \mathrm{d}(q, U) + \mathrm{d}(U, s)$ や $\mathrm{d}(q, U) \leqq \mathrm{d}(q, V) + \mathrm{d}(V, U)$ ，また右側の枝刈り条件成立の前提となる 3 角不等式 $\mathrm{d}(U, V) \leqq \mathrm{d}(U, q) + \mathrm{d}(q, V)$ 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ドでもその子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 角不等式がそのノードのすべての子ノードで成り立つとは一般には言えない．図 2 にその例を示す．

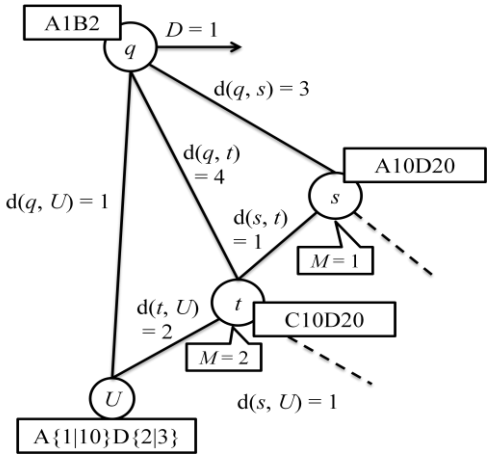

図 2　左側の枝刈り条件が成立しても枝刈りしてはいけない例

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をメタ文字記号で囲む．

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 $d(q, U) \leqq d(q, V) + d(V, U)$ 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 $d(U, V) \leqq d(U, q) + d(q, V)$ は一般には成り立たないからである．図 3 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．

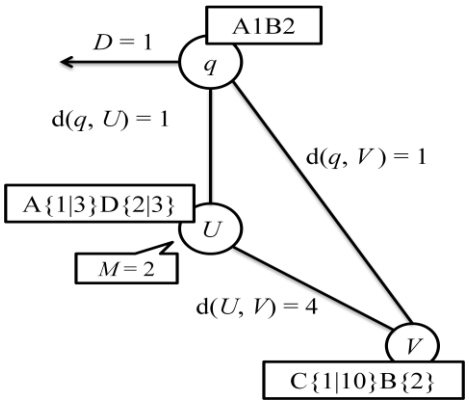

図 3　右側の枝刈り条件が成立しても枝刈りしてはいけない例

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(1)

- 左側の枝刈り条件が成り立てば左側は枝刈りする
- 右側の枝刈り条件が成り立っても右側は枝刈りせず常に検索する

このルールによって VP 木全体の右側が枝刈りしてはいけないことになるのではなく，各ノードの右側を枝刈りしてはいけないだけである．

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．例えば，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．

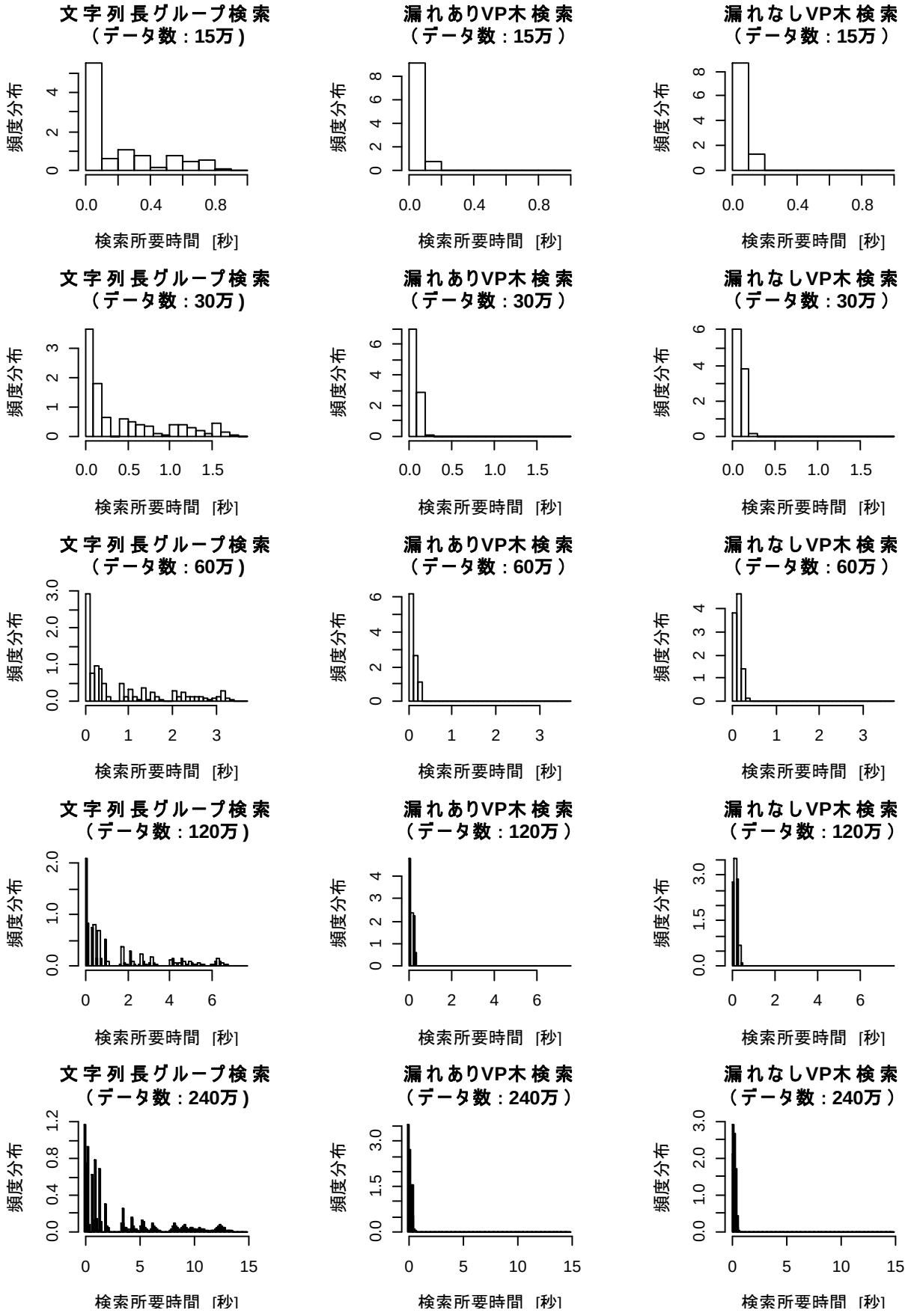

図 4　文字列長グループ検索と VP 木検索の性能比較

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

検証用 PC	Dell Precision T5600
メモリ	64GB
CPU	Intel Xeon E5-2650 2.0GHz
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 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 2 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 2 からデータ数が多いほど漏れの割合も高くなることがわかる．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．

表 2　検索漏れの発生頻度

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	1461（15%）	0（0%）
60 万	2073（21%）	0（0%）
120 万	2620（26%）	0（0%）
240 万	3193（32%）	0（0%）

5.3 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 VP 木検索時間のヒストグラムを図 4 の右列に示す．比較のために VP 木を用いない場合の検索所要時間を図 4 の左列に，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を図 4 の中央に示す．上の行から下の行に向かってデータが増えている．いずれの場合も検証に要する時間の関係で 1 回の測定しかしていない．図 4 から VP 木検索ルールを適用してもなお VP 木の枝刈りのメリットを十分享受できていることがわかる．

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 Ls である文字列 s と文字列長が Lt 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 $|Ls - Lt| \leqq d(s, t) \leqq Ls + Lt$ となる．これより $|Ls - d(s, t)| \leqq Lt \leqq Ls + d(s, t)$ 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 Lq 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 $|Lq - D| \leqq L \leqq Lq + D$ 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 4 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．

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み合わ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，産業上の価値が極めて高い．

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
2)　Myers, E.W. and Miller, W.: Approximate matching of regular expressions, Bulletin of Mathematical Biology, Vol.51, pp.7-37 (1989).
3)　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).
4)　Navarro, G.: A guided tour to approximate string matching, ACM Computing Surveys, Vol.33, No.1, pp.31-88 (2001).
5)　Chavez, E. et al.: Searching in metric spaces, ACM Computing Surveys, Vol.33, No.3, pp.273-321 (2001).
6)　Wagner, R.A. and Fischer, M.J.: The string to string correction problem, Journal of the ACM, Vol.21, pp.168-178 (1974).
7)　Yianilos, P.: Data structures and algorithms for nearest neighbor search in general metric spaces, Proceedings of the Fourth ACM-SIAM Symposium on Discrete Algorithms, pp.311-321 (1993).