2者間マッチングにおけるコストベースの最適化手法

濱田貴広[†] 西岡秀一[††]
鬼塚真[††] 山室雅司[††]

バーターや人材仲介等のサービスでは，利用者間を仲介するマッチング機能（ユーザの取引候補を探索する機能）が重要である．これらのサービスを利用するユーザは，相手に要求するモノの条件（クエリ）と，自分が提供するモノに関する内容（データ）のペアとしてモデル化されるため，マッチング機能は入力されたペアを用いて，蓄積済みのペア集合から適合するペアを特定することと定義できる．このマッチングを処理するプランは複数あり，これらの中から最適なプラン（処理時間が最小となるプラン）を選択することが課題である．本稿では，統計値を用いたコストベースの最適化手法を提案する．本手法は，複数の結合演算の最適化技術がマッチング処理に応用できることに着目し，DB 検索処理とフィルタ処理を利用した 2 つのネストループプランと，1 つのマージプランからなる 3 プランを提案し，これらのコストモデルを定義する．このコストモデルについて，事前実験で得た関数，値を用いて定量化を行い，これらについて人工データによる検証実験を行った結果，提案手法は高精度で最適プランを選択することを実証した．

Cost-based Optimization for Pair-matching Problem

Takahiro Hamada,[†] Shuichi Nishioka,[††] Makoto Onizuka[††]
and Masashi Yamamuro[††]

Internet matching services, such as bartering, and recruiting, have grown. Their main function is to detect two-party exchanges from a huge number of users, in which each user's demand is satisfied by the supply of the other user. A user can be modeled as a pair of demand and supply expressed by queries and data, respectively. Thus the pair-matching problem of two-party exchanges is to identify a set of pairs in a pair database whose data and queries crossly match the query and data of an incoming query pair. For this problem, there are several possible execution plans and the optimum plan depends on a query pair and the statistics of the database. In this paper, we propose a cost-based optimization technique for the pair-matching problem. Similar to the optimization of relational join operations, we obtain two nested loop plans and a single merge plan by leveraging the database query processing and stream data processing. We describe a cost model for these plans, derived by investigating the database and stream processing algorithms and by pre-evaluation phase experiments. The experimental results of execution phase verified that our optimization technique could select the optimum plan with very high precision.

1. はじめに

近年，バーター[1),2)] や人材仲介[3)] などのマッチングサービスがインターネット上でさかんに行われている．マッチングサービスは，利用者相互の要求するモノやサービスなどに対する条件（クエリ）と，提供するモノやサービスなどに関する内容（データ）の間で照合を行う．たとえば，金銭の媒介なしにモノやサービスどうしの交換を行うバーターサービスの場合，クエリは利用者が要求するモノやサービスに対する条件であり，利用者が要求する PC に関する CPU スペックの範囲やメモリの最低限の容量などを指定する例があげられる．データは利用者が要求するモノやサービスの対価として提供する物品に関する内容であり，利用者が所有する TV に関するスペック，年式，メーカ名などで記述される例があげられる．この場合，20 インチ以上の薄型 TV を要求する対価として，CPU が 2.0 GHz でメモリが 512 MB である PC の提供を希望する利用者 A と，24 インチの液晶 TV を提供する対価として 1.8 GHz 以上の CPU と 256 MB 以上のメモリで構成される PC を要求する利用者 B の間でマッ

† 日本電信電話株式会社 NTT 情報流通プラットフォーム研究所
NTT Information Sharing Laboratories, NTT Corporation
†† 日本電信電話株式会社 NTT サイバースペース研究所
NTT Cyber Space Laboratories, NTT Corporation

チングが成立する．

バーターシステムは，利用者が登録するデータとクエリの組合せからなるペアを管理しており，利用者が新たなペアを入力したときに，マッチング処理によって，交換対象候補のペアを検出する．

この交換対象候補のペアを検出する手法として，交換対象候補であるペアの連鎖による複数のペアで構成されるサイクルを検出する手法を文献 4) が提案している．文献 4) において，利用者はペアとしてモデル化されており，同時に，入力ペアに適合するペアを検出するマッチング処理に DB 検索処理とフィルタ処理の両方を利用するプロセスモデルを提案している．ここで，DB 検索処理（**図 1** (a)）とは，蓄積済みのデータ集合から，入力されたクエリに適合するデータを特定する処理であり，フィルタ処理（図 1 (b)）とは，蓄積済みのクエリ集合から，入力されたデータに適合するクエリを特定する処理である．

そこで，本稿では，文献 4) と同様に，ペアを検出するマッチング処理に DB 検索処理とフィルタ処理の両方を利用するプロセスモデルを用いる．しかしながら，文献 4) で検出する交換対象候補は複数のペアのサイクルで構成されているが，出会い支援や人材仲介等のアプリケーションでは，入力ペアに適合する単一のペアを検出する 2 者間マッチング（**図 2**）が利用されている．

2 者間マッチングは，文献 4) が示す DB 検索処理の適合結果とフィルタ処理の適合結果をマージする実行プラン（処理手順）のほかにも考えられ，たとえば，**図 3** に示す 2 つのプランがある．ペアがシステムに入力されると，蓄積済みペア集合は次の 2 つのステップで処理される．プラン A は，まず，システムに蓄積済みのデータ集合全体を対象に DB 検索処理を実行し適合結果を取得する（Step1）．そして，前半の適合結果にペアとして対応するクエリを対象にフィルタ処理を実行し適合結果を取得する（Step2）．この適合結果に対応するペアがマッチング処理結果である．しかしながら，前半の処理の適合結果が多い場合，後半の処理性能に対して負担になるため，処理性能は低くなる．図 3 に示す状況において，プラン A は，はじめにフィルタ処理（Step1）を実行し，その適合結果（intermediate results B）を用いて DB 検索処理（Step2）を実行するプラン B よりも，Step1 における適合結果（intermediate results A）が多いため非効率なプランである．つまり，はじめに実行する処理によるマッチング対象の適合結果が多い場合は，次の処理を効率的に行えないため，処理時間が遅くなる．こ

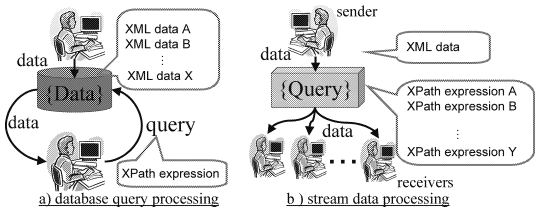

図 1 DB 検索処理とフィルタ処理
Fig. 1 Searching and filtering.

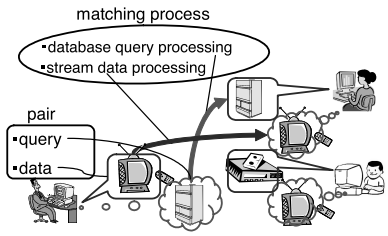

図 2 2 者間マッチング
Fig. 2 Pair-matching.

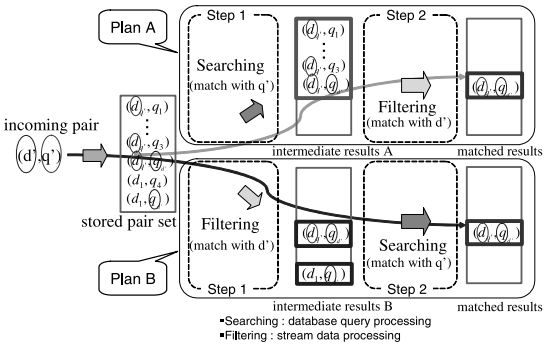

図 3 プランが固定的なマッチング処理
Fig. 3 Matching process in hard-coded order.

のように，ペアどうしの 2 者間マッチングでは，システムがつねに 1 つのプランしか用意していない場合，マッチング処理を効率的に行えないため，入力ペアによって，処理時間が遅くなる問題がある．

本稿では，蓄積済みペア集合から入力ペアに適合するペアを探索する 2 者間マッチングにおいて，複数のプランから処理時間が最小となるプランを選択するための最適化手法を提案する．本稿は，以下のことに寄与する．

- 複数の結合演算（join）を実行する際の実行プランが，2 者間マッチング処理の実行プランと類似していることに着目し，DB 検索処理とフィルタ処理の両方を用いた 2 つのネストループプランと 1 つのマージプランをマッチング処理プランとして提案する．
- これら 3 つのプランは性能特性の違いにより，最適化の余地がある．3 プランの性能比較により，

$$
\begin{aligned}
&pair\text{-}match(P, p(d,q)) \\
&= \{p(d_i,q_i) \in P | q_i \in eval_Q(d,Q), eval(d_i,q) = true\} \qquad (a) \\
&= \{p(d_i,q_i) \in P | d_i \in eval_D(D,q), eval(d,q_i) = true\} \qquad (b) \\
&= \{p(d_i,q_i) \in P | q_i \in eval_Q(d,Q), d_i \in eval_D(D,q)\} \qquad (c)
\end{aligned}
$$

図 4 マッチング処理の導出
Fig. 4 Pair-match rewrites.

最適プランは最も性能の低いプランよりも約 50 倍高速となるケースがあることを示す．このようなケースにおいて最適プランを選択するために，DB 検索処理とフィルタ処理の性能特性を事前実験により調査して，各プランのコストモデルを導出する．そして，このモデルに基づいて，入力ペアと蓄積済みペア集合から，各プランについてコストを事前に各々見積もり，最適プランを選択する手法を提案する．

- 大規模な人工データを用いて最適化手法を評価し，提案手法は非常に高い精度で最適プランを選択することを示す．本稿では，精度の高いコストを見積もるため，事前実験と検証実験において，同一のスキーマにより検証されたデータを使用することを前提とする．

マッチング処理における各処理の順序付け（ordering）によって，2 者間マッチングを実行するプランは複数ありうる．これら複数のプランから最適プランを得るために，個々の処理のコストとそれらを積み上げたプランの総コストを算出して，最小コストとなるプランを選択する．最小コストのプランを選択するという観点において，join ordering [5] の方法と同じであるが，個々の処理は join ではなく，DB 検索処理とフィルタ処理である．特にフィルタ処理のコストについては，過去に議論された論文がないと考えている．

本稿の構成は次のとおりである．2 章でペアモデルとマッチング処理を定義し，3 つのプランについて説明する．3 章でシステム構成を説明し，プロトタイプについて述べる．4 章でコストモデルと各プランの影響因子について説明し，事前実験を 5 章で述べる．6 章で検証実験の測定結果と考察を述べ，7 章でまとめる．

2. マッチング処理モデルとマッチング処理プラン

本章では，ペアモデルとマッチング処理を定義し，この処理を実現するプランについて説明する．

2.1 ペアモデルとマッチング処理

本節では，マッチング処理で扱うペアモデル[4] を定義し，このペアモデルに基づいたマッチング処理を定義する．利用者は提供するモノに関する内容と，要求するモノに対する条件を提示するため，利用者 u はデータ d とクエリ q のペア p の集合でモデル化される☆．

$$u = \{p(d,q)\}$$

ペアどうしの 2 者間マッチングにおいて，システムは蓄積済みペア集合 P に対して入力ペア p を評価して，P から適合するペアを特定する．つまり，この適合したペアのデータとクエリはそれぞれ p のクエリとデータに適合する．したがって，2 者間のマッチング処理は以下のように定義できる．

$$pair\text{-}match(P, p(d,q)) = \{p(d_i,q_i) \in P | eval(d,q_i) = true, eval(d_i,q) = true\}$$

$eval(d,q)$ は，d と q を評価する関数を意味する．

2.2 マッチング処理プラン

本節では，マッチング処理で使用する 3 つのプランと予想される性能特性について説明する．蓄積済みペア集合 P 内のペア $p(d_i,q_i)$ を構成する d_i と q_i の集合をそれぞれ D と Q とし，$eval_D(D,q)$ は D と q を評価し適合結果 $\{d_i\}$ を出力する関数，$eval_Q(d,Q)$ は Q と d を評価し適合結果 $\{q_i\}$ を出力する関数とする．$eval_D(D,q)$ と $eval_Q(d,Q)$ は，それぞれ DB 検索処理とフィルタ処理に対応する．これらの処理を利用して書き換えた *pair-match* 関数を**図 4** に示す．

以後，図 4 における (a)，(b)，(c) をそれぞれ *Filter-Search*（FS）プラン，*Search-Filter*（SF）プラン，*Merge*（MG）プランと呼ぶ．

2.2.1 Filter-Search（FS）プラン

図 5 (a) に示す FS プランの処理手順は以下のとおりである．

① フィルタ処理によって，入力ペアのデータ d で蓄積済みクエリ集合 Q を処理して，適合結果 Q'=$\{q'\}$ を取得する．
② 適合結果 Q' にペアとして対応する D'=$\{d'\}$ を特定する．
③ 入力ペアのクエリ q で D' 内のすべての d' を評

☆ 1 利用者が異なるデータ・クエリで構成されるペアを複数登録することを考慮している．

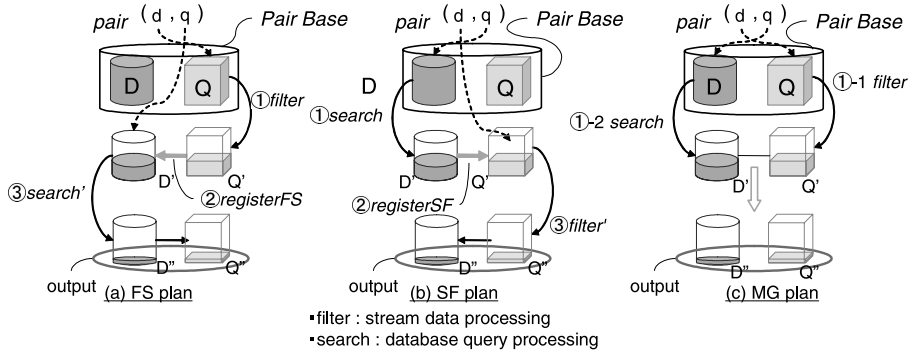

図 5 マッチング処理プラン
Fig. 5 Plans for pair-matching problem.

価する（DB 検索処理）．

このプランの長所は，Q に対して最適化されたオートマトン[6] を用いることで，①を効率的に処理できる点にある．しかし，②，③のコストは①の適合結果数にともない大きくなる短所を持っている．したがって，①の適合結果がより少ないとき，FS プランはより効率的である．

2.2.2 Search-Filter（SF）プラン

SF プランは，FS プランと逆の手順で処理をする．図 5 (b) に示す処理手順は以下のとおりである．

① DB 検索処理によって，入力ペアのクエリ q で蓄積済みデータ集合 D を処理して，適合結果 D'=$\{d'\}$ を取得する．

② 適合結果 D' にペアとして対応する Q'=$\{q'\}$ を特定する．

③ 入力ペアのデータ d で Q' 内のすべての q' を評価する（フィルタ処理）．

このプランの長所は，D に対して B 木等のインデックスを利用できるため，①を効率的に処理できる点にある．しかし，FS プランと同様の短所を持っている．したがって，①の適合結果がより少ないとき，SF プランはより効率的である．

2.2.3 Merge（MG）プラン

図 5 (c) に示す MG プランの処理手順は以下のとおりである．

①-1，2 フィルタ処理と DB 検索処理をそれぞれ独立に実行する．前者は，入力ペアのデータ d で蓄積済みクエリ集合 Q を処理して，適合結果 Q'=$\{q'\}$ を取得する．後者は，入力ペアのクエリ q で蓄積済みデータ集合 D を処理して，適合結果 D'=$\{d'\}$ を取得する．

② 適合結果の Q' と D' の積集合を算出する．このとき，hash 表を用いることで，高速に積集合の算出が可能である．

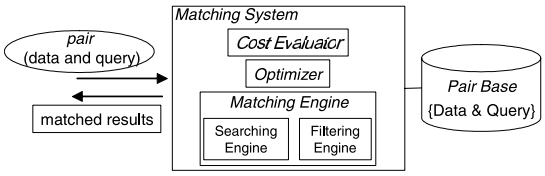

図 6 マッチングシステム
Fig. 6 Matching system.

このプランは，DB 検索処理とフィルタ処理の両方において適合結果が多いときに最適なプランとなることが期待される．

3. システム構成とプロトタイプ

本章では，マッチング処理を実現するシステムの構成について説明し，実験で利用するプロトタイプについて述べる．

3.1 システム概要

本節では，**図 6** に示すシステム概要について説明する．まず事前処理として，処理対象のデータやクエリのスキーマに基づいた小規模のテストセットを用いて，各コスト因子のパラメータを取得し，*Cost Evaluator* に反映する．*pair* がシステムに入力されたとき，*Cost Evaluator* にて各プランのコスト因子の統計値にアクセスし，コスト因子の見積もり値を計算する．次に，*Optimizer* が各プランの処理コストを見積もり，最小コストとなるプランを最適プランとして選択する．選択されたプランは *Matching Engine* で実行される．また，ペア集合は *Pair Base* に蓄積される．

3.2 プロトタイプ

本節では，プロトタイプで利用する XML-DBMS と XML フィルタサーバについて説明する．本稿では，マッチング処理で扱うデータとクエリはそれぞれ XML

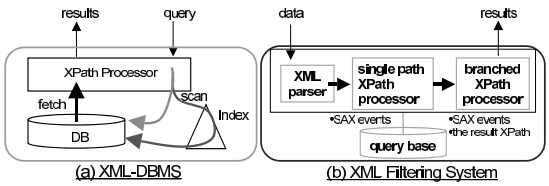

図 7 XML-DBMS と XML フィルタリングシステム
Fig. 7 XML-DBMS and XML-Filtering system.

表 1 各引数の説明
Table 1 Arguments definition.

表記	内容
$hitDB$	DB 検索処理の適合結果件数
$inputData$	入力ペアのデータのサイズ
$hitFilter$	フィルタ処理の適合結果件数
$branchFilter$	フィルタ処理の分岐数
$numXML$	ペアベースに蓄積済みのデータ件数
$numXPath$	ペアベースに蓄積済みのクエリ件数

と XPath 式で表現されることを想定している．したがって，これら XML と XPath 式を扱うため，DB 検索処理とフィルタ処理はそれぞれ XML-DBMS と XML フィルタリングシステムで実行する．

XML-DBMS は，XML スキーマで構造化された XML データを処理する Xmics[7] を利用する．E コマース等で使用されるデータのスキーマは構造化されていることが主なため，定義付けられたスキーマを利用する．Xmics の概要を**図 7** (a) に示す．

XML フィルタリングシステム[6] は，入力 XML データで蓄積済み XPath 式を処理する．システム内部で，蓄積済み XPath 式は決定性有限オートマトン（DFA）に変換される．XFilter[8]，YFilter[9]，Xtrie[10] は XPath 式を非決定性有限オートマトン（NFA）に変換し，NFA を用いて処理する．NFA ベースのシステムの処理性能は，蓄積済み XPath 式の本数に依存して低下するが，DFA ベースのシステムの処理性能は，XPath 式の本数の性能に対する影響は小さいため，DFA ベースの処理を実現する XML フィルタリングシステムを利用する．システムの概要を図 7 (b) に示す．

4. マッチングのコストモデルと因子

本章では，各プランのコストモデルを仮定し，コストの因子を決定する．本稿で提案する最適化手法は，このコストモデルをベースに最適なプランを選択する．

4.1 コストモデル

本節では，各プランのコストモデルについて説明する．2.2 節で述べたように各プランはそれぞれ特徴があるため，入力ペアや蓄積済みペア集合の値の分布を含むあらゆる状況においてマッチング処理を高速に実現する最適なプランを選択し実行しなければならない．このため，各プランに関するコストモデルを導出する．各プランのコストモデルは次のとおりである．

4.1.1 FS プランのコストモデル

2.2.1 項を基に，FS プランの処理コスト $Cost_{FS}$ は，入力ペアのデータでシステムに蓄積済みのクエリ集合を処理するフィルタ処理のコスト $Cost_{filter}$ と，適合結果のクエリにペアとして対応するデータを特定しペアベースに登録する処理のコスト $Cost_{registerFS}$，そして，入力ペアのクエリで登録されたデータを処理する DB 検索処理のコスト $Cost_{search'}$ の 3 つの要素で構成される．

$$Cost_{FS} = Cost_{filter} + Cost_{registerFS} + Cost_{search'} \quad (1)$$

4.1.2 SF プランのコストモデル

2.2.2 項を基に，SF プランの処理コスト $Cost_{SF}$ は，入力ペアのクエリでシステムに蓄積済みのデータ集合を処理する DB 検索処理のコスト $Cost_{search}$ と，適合結果のデータにペアとして対応するクエリを特定しペアベースに登録する処理のコスト $Cost_{registerSF}$，そして，入力ペアのデータで登録されたクエリを処理するフィルタ処理のコスト $Cost_{filter'}$ の 3 つの要素で構成される．

$$Cost_{SF} = Cost_{search} + Cost_{registerSF} + Cost_{filter'} \quad (2)$$

4.1.3 MG プランのコストモデル

2.2.3 項を基に，MG プランの処理コスト $Cost_{MG}$ は，DB 検索処理のコスト $Cost_{search}$ とフィルタ処理のコスト $Cost_{filter}$ の 2 つの要素で構成される．ただし，両処理の結果のマージの処理コストはハッシュ表を利用するので無視できるほど十分小さいため，式には現れない．

$$Cost_{MG} = Cost_{search} + Cost_{filter} \quad (3)$$

4.2 コストモデルにおける因子

本節では，各プランのコストモデル (1)，(2)，(3) における因子を仮定する．各因子の引数は**表 1** に示す．

4.2.1 FS プランの因子

フィルタ処理のコスト $Cost_{filter}$ は，主にデータの入力とパース処理，分岐 XPath 式の処理，適合 XPath 式の特定と出力処理からなる．入力データの SAX イベントによるパース処理コストは，データサイズ $inputData$ に依存する．次に，パース処理されたデータを用いて分岐 XPath 式の処理が行われる．この処理コストは，XPath 式の分岐数 $branchFilter$ が主要因である．入力データに対する適合クエリの特定

において，コストへの影響は小さいが，蓄積済みクエリの件数 $numXPath$ を考慮する．最後に，入力データに適合するクエリが出力される．このコストは，適合結果数 $hitFilter$ に依存する．したがって，フィルタ処理のコストは以下のようになる．

$$Cost_{filter} = f_1(inputData, branchFilter, hitFilter, numXPath) \quad (4)$$

フィルタ処理の適合結果であるクエリにペアとして対応するデータを特定しペアベースに登録するコスト $Cost_{registerFS}$ は，登録件数である適合結果数 $hitFilter$ が影響する．したがって，データ登録処理のコストは以下のようになる．

$$Cost_{registerFS} = f_2(hitFilter) \quad (5)$$

一般的に，DB 検索処理のコストはデータベース内の走査と該当データの取り出しから見積もることができる．FS プランにおける DB 検索処理は，登録されたデータのみを対象とする．このときの DB 検索処理のコスト $Cost_{search'}$ は，走査と取り出しから見積もることができるが，走査についてインデックスの効果がないため，双方ともコストの支配項となる．したがって，DB 検索処理の蓄積済みデータ件数 $numXML$，DB 検索処理対象データ件数 $hitFilter$ と適合結果数から見積もることができる．また，この処理の適合結果数は $hitDB' = hitDB \times (hitFilter/numXPath)$ と予想できる．これは，蓄積済みクエリ集合の全件 $numXPath$ に対するフィルタ処理の適合結果数 $hitFilter$ の割合を利用した予想である．したがって，この DB 検索処理コストは以下のようになる．

$$Cost_{search'} = f_3(hitDB', hitFilter) \quad (6)$$

4.2.2 SF プランの因子

SF プランにおける DB 検索処理のコスト $Cost_{search}$ は，システムに蓄積済みの全データを対象とするため，走査はインデックスの効果より，コストをほぼ一定と見なせ，データ取り出しのコストが支配項となる．したがって，検索処理コストは適合結果数 $hitDB$ から見積もれる．これより，DB 検索処理コストは以下のようになる．

$$Cost_{search} = f_4(hitDB) \quad (7)$$

DB 検索処理の適合結果であるデータにペアとして対応するクエリを特定しペアベースに登録するコスト $Cost_{registerSF}$ は，登録件数である適合結果数 $hitDB$ が影響する．したがって，クエリ登録処理のコストは以下のようになる．

$$Cost_{registerSF} = f_5(hitDB) \quad (8)$$

SF プランにおけるフィルタ処理のコスト $Cost_{filter'}$ は，$Cost_{filter}$ とほぼ同様であるが，処理実行前に対象クエリから DFA を構築する必要がある．対象クエリ件数は，DB 検索処理の適合結果数 $hitDB$ と同じである．つまり，DFA 構築処理のコストは $hitDB$ が影響する．また，この処理の適合結果数と分岐数は，それぞれ $hitFilter' = hitFilter \times (hitDB/numXML)$，$branchFilter' = branchFilter \times (hitDB/numXML)$ と予想できる．これらは，蓄積済みデータ集合の全件 $numXML$ に対する DB 検索処理の適合結果数 $hitDB$ の割合を利用した予想である．したがって，このフィルタ処理のコストは，$Cost_{Filter}$ に加えて登録クエリの事前処理が必要であり，以下のようになる．

$$Cost_{filter'} = f_6(inputData, hitFilter', branchFilter', hitDB) \quad (9)$$

4.2.3 MG プランの因子

MG プランにおける DB 検索処理とフィルタ処理のコストは，それぞれ $Cost_{search}$(7) と $Cost_{filter}$(4) と同じである．

5. 事前実験

本章では，プロトタイプと人工データを用いてコストモデルを構成する因子を導出するために事前実験を行った☆．

各プランのコストモデル（(1)，(2)，(3)）を因子（(4)，...，(9)）に基づいて決定する．各プランのコストは入力ペアとシステム内の蓄積済みペア集合から見積もることができるため，これらのコストモデルを用いることによって，最適なプランを選択することが可能となる．最適プラン選択の精度を高めるために，事前実験と検証実験（6 章）において，同一のスキーマにより検証された XML データを使用した．クエリパターンも，両実験で同じものを使用した．実験環境として，CPU が Intel Xeon 3.40 GHz，メモリが 4.0 GB であり，OS は RedHatLinux である PC を使用した．各実験結果は 10 試行の平均値であり，各グラフの X 軸は関連する因子の値を，Y 軸は処理時間（ミリ秒）を示している．事前実験の結果より，各プランのコスト $Cost_{FS}$，$Cost_{SF}$，$Cost_{MG}$ を求める．実験より求めた各因子の係数（$c_i(i = \{1, 2, \cdots, 10\})$）を**表 2** に示す．

・因子 (7) と因子 (6) のコスト

DB 検索処理のコスト $Cost_{Search}$，$Cost_{Search'}$ は

☆ DB 検索処理を計測するために，1 件あたり 1.2 KByte の登録 XML データと 1 つの述語を含む入力 XPath 式を使用した．フィルタ処理を計測するために，約 250 Byte の入力 XML データと 1 つの述語を含む登録 XPath 式を使用した．

表 2 各因子の係数
Table 2 Weight of factors.

$c_1 = 0.267$	$c_2 = 1.192$	$c_3 = 0.0025$	$c_4 = 0.026$
$c_5 = 0.0318$	$c_6 = 0.0345$	$c_7 = 0.006$	$c_8 = 0.1494$
$c_9 = 0.0151$	$c_{10} = 0.176$		

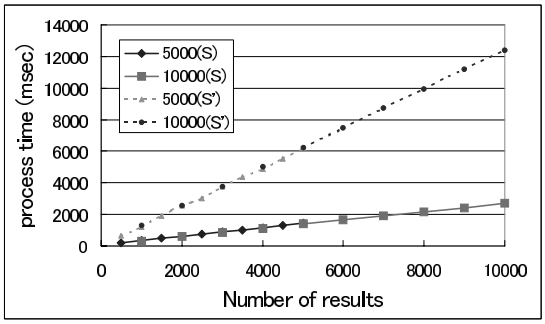

図 8 DB 検索処理コスト
Fig. 8 Database query processing costs.

適合結果数 $hitDB$ と関係がある．5,000，10,000 の登録データ件数において，両コストを測定した結果を**図 8** に示す．図 8 の 5000(S) と 10000(S) は，前者の処理における適合結果数と処理時間の関係を，同図の 5000(S') と 10000(S') は，後者の処理における適合結果数と処理時間の関係を表している．これらは比例関係にあるため因子 (7) と因子 (6) は以下の式になる．

$$Cost_{search} = c_1 \times hitDB \quad (10)$$

$$Cost_{search'} = c_2 \times hitDB \quad (11)$$

・因子 (4) と因子 (9) のコスト

フィルタ処理のコスト $Cost_{Filter}$ は，入力データサイズ $inputData$，分岐数 $branchFilter$，適合結果数 $hitFilter$，蓄積済みクエリの件数 $numXPath$ に関係する．5,000 件の XPath 式を登録したシステムに対して，異なるサイズのデータを入力したときの処理時間を測定した結果を**図 9** に示す．これより，処理時間は入力データサイズに比例することが分かる．次に，蓄積済みクエリ件数が 5000，10000 のときの分岐処理の回数と処理時間の関係を**図 10** に示す．これより，分岐数と処理時間は比例関係にあることが分かる．また，**図 11** は，蓄積済みクエリ件数（1,000〜30,000）と適合クエリの特定処理時間の関係と，登録クエリ件数とこのクエリから DFA を構築するのに要する処理時間の関係を示しており，両者ともに比例関係にあることが分かる．FS プランと MG プランにおいて，DFA は構築済みであるので後者は関係ないが，SF プランでは，フィルタ処理を実行するためにまず登録クエリから DFA を構築しなければならないので，考慮しなければならない．また，**図 12** は，適合結果

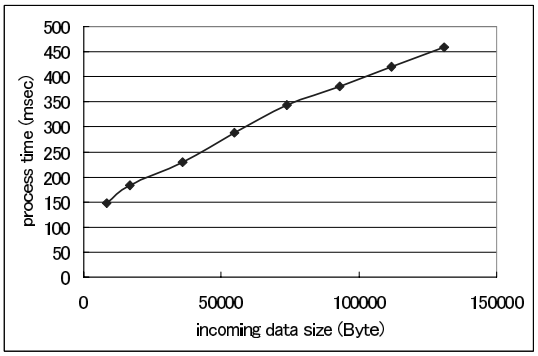

図 9 入力データサイズに対するフィルタ処理コスト
Fig. 9 Stream data processing costs against incoming data size.

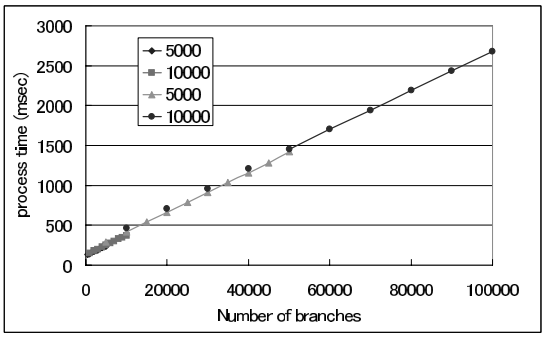

図 10 分岐数に対するフィルタ処理コスト
Fig. 10 Stream data processing costs against the number of branches.

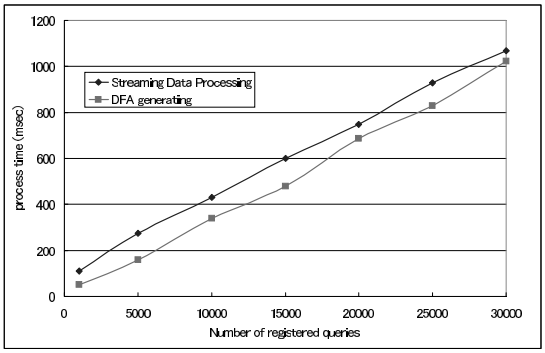

図 11 処理対象クエリ数に対するフィルタ処理コストと DFA 構築コスト
Fig. 11 Stream data processing costs and DFA generating costs against the number of stored queries.

の出力数と処理時間の関係を示しており，比例関係にあることが分かる．以上より，因子 (4) と因子 (9) は以下の式になる．ただし，SF プランにおけるフィルタ処理の分岐数と適合結果数をそれぞれ $hitFilter'$，$branchFilter'$ とする．

$$Cost_{Filter} = c_3 \times inputData$$

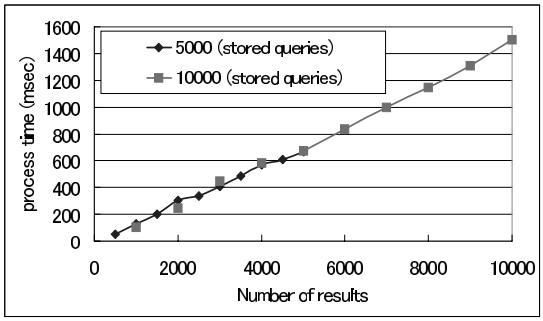

図 12 適合結果数に対するフィルタ処理コスト
Fig. 12 Stream data processing costs against the number of results.

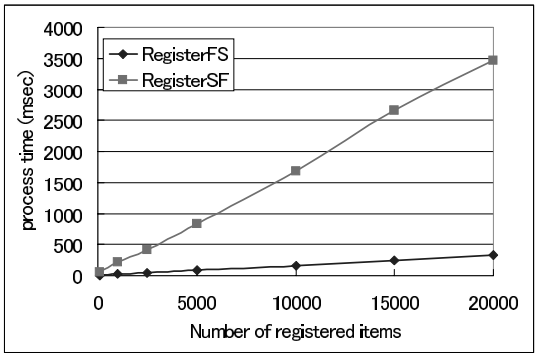

図 13 登録データ数，クエリ数に対する登録処理コスト
Fig. 13 Cost of registering data and queries.

$$+ c_4 \times branchFilter + c_6 \times numXPath + (c_7 + c_8) \times hitFilter \quad (12)$$

$$Cost_{Filter'} = c_3 \times inputData + c_4 \times branchFilter' + c_6 \times hitDB + (c_7 + c_8) \times hitFilter' + c_5 \times hitDB \quad (13)$$

・因子 (5) と因子 (8) のコスト

FS プランと SF プランにおいて，それぞれデータ，クエリをシステムに登録するコスト $Cost_{registerFS}$，$Cost_{registerSF}$ は，ともに登録件数に依存する．これらの件数は，プランのはじめに行う処理の適合結果数である．**図 13** は，登録件数（100〜20,000）と処理時間の関係を示しており，これらは比例関係にあるため因子 (5) と (8) は以下の式になる．

$$Cost_{registerFS} = c_9 \times hitFilter \quad (14)$$

$$Cost_{registerSF} = c_{10} \times hitDB \quad (15)$$

6. 検 証 実 験

本章では，事前実験から導出した因子によるコストベースの最適化手法を 50,000 件のペア集合を用いて検証実験を行い，結果について考察する．

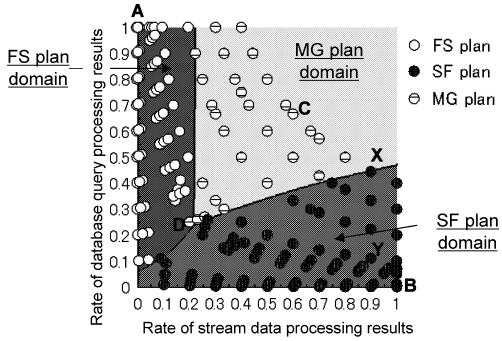

図 14 最適化手法による各プランの適用有効エリアと実際の最適プラン
Fig. 14 Estimated optimum area of 3 plans and the actual optimum plans.

6.1 測 定

本節では，大規模な人工データを用いて，最適化手法の有効性を検証する．実験に先立って，次の事項を前提とする．

(1) ペアは，1 つのデータと 1 つのクエリの構成とし，$numXML = numXPath$ とする．

(2) DB 検索処理の適合結果数は，一度クエリを実行して，値を取得する．

(3) フィルタ処理の適合結果数，分岐処理回数は，事前に得られる既知の情報とする．

50,000 件の蓄積済みペア集合に対して，マッチング結果率が約 0.001，0.005，0.01，0.05，0.06，0.07，0.1，0.2，0.3，0.4 となるペアを入力したときの最適化手法が選択したプランのエリアと，実測結果から最小処理時間であったプランのプロットを重ね合わせた結果を**図 14** に示す．X 軸と Y 軸は，それぞれフィルタ処理の適合結果率と DB 検索処理の適合結果率を示している．図 14 の 3 つのエリアは，最適化手法が求めた各プランの適用有効領域を示している．コストを算出して最適プランを選択するまでにおよそ 200〜300 ミリ秒かかった．188 個のプロットは，それぞれ 3 つのプランの実測結果より得た実際の最適プランを表している．

DB 検索処理の適合結果率，フィルタ処理の適合結果率，マッチング率に特徴のある 4 つのケース A（0.001, 1.0, 0.001），B（1.0, 0.001, 0.001），C（0.6, 0.667, 0.4），D（0.2, 0.25, 0.05）における各プランの性能比較とコストの内訳を**図 15** に示す．この記述は，(フィルタ処理の適合結果率，DB 検索処理の適合結果率，マッチング率）である．各積み重ね棒グラフは，フィルタ処理，DB 検索処理，中間登録処理（registerFS, registerSF）に要した時間を示す．X 軸は各プランを，

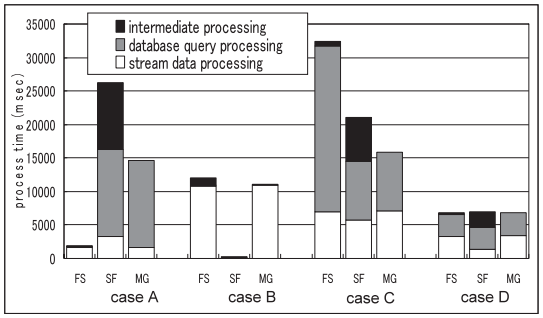

図 15 性能結果の詳細
Fig. 15 Details of performance results.

Y 軸は処理時間（ミリ秒）を示している．たとえば，ケース A は，フィルタ処理と DB 検索処理の適合結果率がそれぞれ 0.001 と 1.0 でマッチング結果率が約 0.001 となるようなケースにおける各プランの処理性能を表している．

6.2 考　　察

本節では，まず，実験結果に対する考察を述べる．次に，最適化手法において使用するパラメータの予測値の精度が性能に及ぼす影響について考察する．

6.2.1 実験結果に関する考察

本項では，検証実験の結果を考察する．検証実験の結果である図 14 から分かるように，実測結果より得た最適プランを示すプロットは，最適化手法から求めた各プランの適用エリアにほとんど対応している．すべての実測プロット数に対して最適化手法で選択したプランが実際に最適であった数の割合を正解率とすると，187/188 ≃ 0.995 であった．これより，提案手法は非常に高い精度であることを示している．

また，図 14 の結果は，各プランの適用有効領域であると考えられる．フィルタ処理と DB 検索処理の性能に大きく影響を与える因子は，それぞれの適合結果数であると想定されるため，これらの結果数からコストを見積もることが有効である．

フィルタ処理の適合結果数が少ないケース A では，図 15 より，FS プランがはじめにフィルタ処理を実行するため適合結果が少なく，次に実行する処理のコストを低減しているため，他のプランに比べて高い性能である．一方で，FS プランと反対の順序で処理する SF プランの性能は最も低い．

DB 検索処理の適合結果数が少ないケース B では，図 15 より，ケース A とは反対に，SF プランがはじめに DB 検索処理を実行するため適合結果が少なく，次に実行する処理のコストを低減していることが分かる．これより，他のプランに比べて高い性能を実現している．一方で，FS プランの性能は最も低い．また，このケースにおいて，最適プランの処理時間は，最も低い性能を実現するプランの処理時間に比べて約 50 倍高速である．

DB 検索処理とフィルタ処理の両方の適合結果数が比較的多いケース C では，図 15 より，MG プランがその他のプランと比較して高い性能を示していることが分かる．FS プランでは，フィルタ処理の適合結果数が多いと，他のプランと比較してコストが高くなる．これは，因子 (6) が著しく大きくなるためである．SF プランは，DB 検索処理の適合結果数が多くなるにつれて，他のプランと比較してコストが高くなる特徴がある．これは，因子 (8) と (7) の著しい増大のためである．MG プランでは，DB 検索処理とフィルタ処理の適合結果数が多くなると，他のプランと比べるとコストが低くなる．これより，各プランの有効適用領域が確立され，本稿で述べた予想（2 章）を支持する結果となった．

ケース D は，唯一最適でないプランが選択された場合であり，最適化手法が FS プランを最適プランとして選択した．しかし，図 15 の実測結果において，最も性能の良いプランは MG プランであった．これは，計算誤差によるものと考えられるが，FS プランと MG プランの性能差はごくわずかであった．最適化手法が選択したプランのコストと実測結果から得た最適プランのコストの比は，6853.9/6794.7 ≃ 1.009 であり，誤差の範囲と見なせる．

マッチング処理のプランが 1 つしかない場合に比べて，200〜300 ミリ秒程度かかるコスト計算時間を上乗せしても，最適化手法が選択する最適プランを実行する方が，残り 2 つのプランを実行するよりもコストが小さいケースが多い．以上より，コストベースの最適化手法に基づいて各プランを選択的に実行するシステムは有効である．

6.2.2 パラメータの精度に関する考察

本項では，コスト見積りで使用するパラメータの精度について考察する．本稿の検証実験は，理想的な予測パラメータを使用したコスト見積りに基づいて最適プランを導いた．コスト見積りに必要なパラメータの予測において，入力ペアのデータサイズと蓄積済みデータおよびクエリの件数に関する正確な値の取得は可能である．FS プラン，SF プランにおける 2 段目の処理のヒット件数 $hitDB'$，$hitFilter'$ の取得は，1 段目の処理のヒット件数 $hitFilter$，$hitDB$ が取得可能ならば，統計的に求めることが可能であり，提案手法はこの統計的に求めた予測値を使用している．DB 検

索処理のヒット件数 *hitDB* は，たとえば，RDB ベースの XML-DBMS を使用する場合，SQL の *count* 関数によって正確な値の取得が可能である．

本稿におけるマッチング処理は，スキーマにより検証された XML データの使用を想定している．これより，このデータに対応するクエリパターンが特定される．コストモデルの因子を取得する事前実験においても，同じスキーマにより検証された XML データとクエリパターンの使用を想定している．これにより，予測するパラメータの精度を高めることができる．

フィルタ処理において，表 2 よりコストに大きく影響し，かつ正確な値の取得が困難なパラメータは，ヒット件数 *hitFilter* である．このパラメータの予測値と実際の値との差が大きい場合，最適化手法の精度に対する影響について述べる．図 14 のケース X において，*hitFilter* を実際の値よりも小さい予測値を取得した場合，最適化手法が選択するプランは，ケース X のプロットを左側へ移動した領域の MG プランとなる．しかし，図 14 のケース Y において，*hitFilter* を実際の値よりも小さい予測値を取得した場合，ケース X のときとは異なり，最適化手法が選択するプランは SF プランのままである．これは，各プランのコストの値に影響するが，最小コストのプラン選択には影響しないことを示している．このように，最適化手法の精度は，各プランが最適プランとして占める領域の境界線に近いケースほど，パラメータの予測精度の影響を受ける．しかし，図 15 のケース D から分かるように，境界線付近のケースにおいて，その境界線に接するプランの性能の差は小さい．したがって，このようなケースにおいて，最適プランの選択を誤ったとしても，多大な性能低下にはならないと考えられる．他のパラメータにおいても，同様の議論を展開することができる．

7. おわりに

本稿では，DB 検索処理とフィルタ処理を利用したマッチング処理において，入力ペアと蓄積済みペア集合から 3 つのプランのコストを見積もり，最適なプランを選択する最適化手法を提案した．実験結果より，提案手法は非常に高い精度で最適プランを選択することを示した．また，最適プランは，最も性能の低いプランよりも約 50 倍高速となるケースがあることを示した．また，コスト見積りに使用するパラメータの予測値の精度が性能に及ぼす影響について議論した．

今回は，フィルタ処理におけるコスト算出に必要な値は既知とした．これらの値の高速な予測と，実サービスによる運用評価，そして，提案した 3 つのプランのほかに，フィルタ処理と DB 検索処理を融合的に実行するプランの実現が今後の課題である．

参考文献

1) ITEX Payment Systems. http://www.itex.com/
2) Recipco. http://www.recipco.com/
3) Yahoo! HotJobs. http://www.hotjobs.com/recruiting
4) Nishioka, S., Yaguchi, Y., Hamada, T., Onizuka, M. and Yamamuro, M.: XML-Based e-Barter System for Circular Supply Exchange, *DEXA*, pp.461–470 (2005).
5) Vance, B. and Maier, D.: Rapid bushy join-order optimization with Cartesian products, *SIGMOD '96: Proc. 1996 ACM SIGMOD international conference on Management of data*, New York, NY, USA, pp.35–46, ACM Press (1996).
6) Green, T.J., Gupta, A., Miklau, G., Onizuka, M. and Suciu, D.: Processing XML streams with deterministic automata and stream indexes, *ACM Trans. Database Syst.*, Vol.29, No.4, pp.752–788 (2004).
7) 春日史朗，坂田哲夫，小谷尚也，芳西 崇：リレーショナルデータベースを用いた XQuery 処理システムの実現，情報処理学会研究報告 2004-DBS-134(II), pp.293–299 (2004).
8) Altinel, M. and Franklin, M.J.: Efficient Filtering of XML Documents for Selective Dissemination of Information, *The VLDB Journal*, pp.53–64 (2000).
9) Diao, Y. and Franklin, M.J.: Query Processing for High-Volume XML Message Brokering., *VLDB*, pp.261–272 (2003).
10) Chan, C.Y., Felber, P., Garofalakis, M.N. and Rastogi, R.: Efficient Filtering of XML Documents with XPath Expressions, *The VLDB Journal*, Vol.11, pp.354–379 (2002).

(平成 17 年 9 月 20 日受付)
(平成 18 年 1 月 6 日採録)

(担当編集委員 金子 邦彦)

濱田　貴広

平成 13 年北海道大学工学部情報工学科卒業．平成 15 年同大学院工学研究科システム情報工学専攻修士課程修了．同年日本電信電話株式会社入社．以来，情報共有システム，XML 処理システムの研究開発に従事．電子情報通信学会会員．

西岡　秀一（正会員）

平成 7 年横浜国立大学工学部電子情報工学科卒業．同年日本電信電話株式会社入社．以来，データベース管理システム，著作権管理システム，XML 処理システムの研究開発に従事．平成 17 年横浜国立大学大学院博士後期課程修了．博士（工学）．日本データベース学会会員．

鬼塚　　真（正会員）

平成 3 年東京工業大学工学部情報工学科卒業．同年日本電信電話株式会社（株）入社．平成 12 年，13 年ワシントン州立大学客員研究員．現在日本電信電話（株）サイバースペース研究所主任研究員．データベース管理システム，XML データ処理の研究に従事．平成 16 年情報処理学会山下記念賞，平成 17 年電子情報通信学会 DEWS2005 優秀論文賞．ACM，電子情報通信学会，日本データベース学会各会員．

山室　雅司（正会員）

昭和 60 年早稲田大学理工学部数学科卒業．昭和 62 年同大学院数学専攻修士課程修了．同年日本電信電話株式会社入社．平成 2 年コロンビア大学大学院電気工学専攻修士課程修了．以来，ネットワークオペレーション情報モデル化，データベース設計，データ視覚化，画像検索システム，XML 処理システムの研究開発に従事．博士（工学）．平成 6 年電子情報通信学会学術奨励賞．電子情報通信学会，日本ソフトウェア科学会，IEEE-CS，日本データベース学会各会員．