

高速な到達可能性判定のための規模耐性の高い索引付け

Scalable Indexing for Fast Reachability Test

中村 有作* 舞田 哲哉† 坂本 比呂志‡

Yusaku NAKAMURA Tetsuya MAITA
Hiroshi SAKAMOTO

本論文では、複雑な有向グラフで表現される半構造データに対して、任意の2ノード間の到達可能性を高速に判定できる規模耐性の高い索引構造を提案する。到達可能性の判定は、有向グラフが木構造の場合、範囲ラベルなどの値をあらかじめ計算しておくことで高速に判定可能である。しかしながら、一般の有向グラフに対しては、このような単純な索引付けが困難である。そこで本研究では、一般の有向グラフに適用可能な到達可能性判定のための索引構造を提案する。また、本研究の手法がこれまでに提案されている他手法よりも、前処理に必要な主記憶量、計算時間および索引サイズにおいて優れていることを示す。

We propose an efficient algorithm for deciding the reachability between any nodes on connected directed graphs. In case of tree structures, this problem is testable in a constant time by preprocessing the range labels of all nodes. However, for the general directed graphs, it is impossible to set a simple range label for the graph even if it is acyclic. In this study, we introduce a new index for the problem of the reachability test on the general graphs. We show that our algorithm answers almost queries in a constant time preserving the space efficiency and a reasonable preprocessing time compared with other proposed methods.

1. はじめに

本研究では、有向グラフで表現されるXMLデータ上の任意の2ノード間の到達可能性を高速に判定するための高い規模耐性を持つ索引構造を提案する。このような技術は、XQueryなどの問い合わせ言語において高速に構造ジョインを行うために有効な手段である。本研究の手法は、これまでに提案されている木構造上の範囲ラベルと有向グラフ上の2Hopラベルを組み合わせた、より規模耐性の高いラベル付けである。そこでまず、これらの関連研究についてまとめる。

1.1 範囲ラベル

定義 1 有向グラフ G の任意のノード v に、ある非負整数のペア $(\ell(v), r(v))$ を割り当てたとき、任意の v, u に対して、“ $\ell(v) < \ell(u)$ ”かつ $r(v) > r(u)$ ”と“ v が u の先祖である”が等価であるとき、この割り当てを G に対する範囲ラベルという。

G が木や平面グラフなど単純な構造であるときには、このような性質を満たす範囲ラベルは高速に計算することができる。例え

*学生会員 九州工業大学 情報工学研究科 博士前期課程
yusaku@donald.ai.kyutech.ac.jp

†九州工業大学 情報工学研究科 博士前期課程
t.maita@donald.ai.kyutech.ac.jp

‡正会員 九州工業大学 情報工学部 知能情報工学科
hiroshi@ai.kyutech.ac.jp

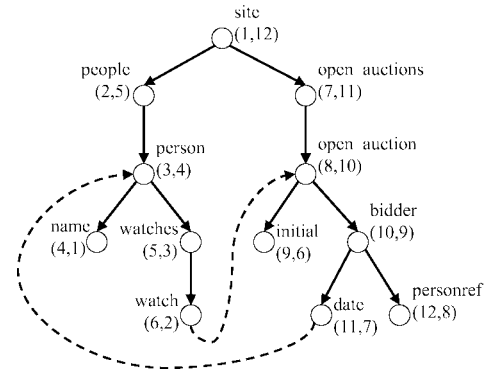


図 1. 有向グラフとその全域木の範囲ラベル

Fig. 1. A range label for a spanning tree

ば木の場合には、そのノード v の前置順と後置順による順位のパラ $(pre(v), post(v))$ を利用する手法がよく知られている。このようにしてラベル付けされた例を図 1 に示す。

より一般には、各 $pre(v)$ や $post(v)$ の間隔を空けて設定することで、範囲ラベルの条件を満たしながらデータの挿入や削除に対応することができる [2, 8, 9, 21]。しかしながら、図 1 の例では、“watch” と “date” の 2 ノードが互いに到達可能であるにも関わらず、これらの範囲ラベルは定義 1 を満足しない。一般には、有向グラフに対する無矛盾な範囲ラベルは存在しないため、有向グラフの到達可能性の判定には別の手法が必要である [14]。

一般の有向グラフに対する到達可能性は、まず与えられたグラフを強連結成分 [5] に分解し、その非巡回グラフ (DAG) に関する到達可能性についてのラベル付けを行えばよい。次に、それらの関連研究について簡単に説明する。

非巡回グラフの範囲ラベル付けとして [4] がある。これは、範囲ラベルを先祖へ伝播させる手法で、判定時間は各ノードに設定される範囲ラベルの最大数 p に依存し、グラフのノード数と辺数をそれぞれ n, m とすると、必要な前処理時間は $O(pm)$ 、記憶領域は $O(pn + m)$ である。また、1 回あたりの判定時間は $O(p)$ となるが、最悪の場合 $p = O(n)$ である。

判定時間を改善する手法として、重なり範囲ラベルと多次元分割による判定法が提案されている [19]。この手法では、1 回あたりの判定時間は多次元分割の個数 q に比例し、この値は [4] の手法における p に比べて比較的小さいことが示されている [19]。この手法における前処理時間は $O(qn^2)$ 、記憶領域は $O(q(n + m))$ である。

一方、前処理時間と判定コストの両方を妥当な範囲に抑えるために、[14] では到達可能性を記録する参照テーブルを分割する方法を提案している。

1.2 2Hop ラベル

定義 2 有向グラフ $G = (E, V)$ の各ノード $v \in V$ に対して、ラベルの集合 $v_{in}, v_{out} \subseteq V$ を決めるとき、 $L(G) = \{(u_{in}, u_{out}) \mid u \in V\}$ を G の 2Hop ラベルという。このとき、任意の $u, v \in V$ に対して、 u から v へ到達可能 $\Leftrightarrow u_{out} \cap v_{in} \neq \emptyset$ であるなら、そのような $L(G)$ を G の 2Hop カバーという。

以降では、2Hop カバーの条件を満たす $L(G)$ のことを単に 2Hop ラベルと呼ぶことにする。任意の連結な有向グラフに対して、2Hop ラベルが存在することはあきらかである。なぜならば任意の $v \in V$ に対して、 v_{in} を v に到達可能なすべてのノードからなる集合とし、 v_{out} を v から到達可能なすべてのノードからなる集合とすればそのラベルは常に定義 2 を満足するからである。各ノードがなるべく小さい数のラベルを持つような 2Hop ラベルを求めることができれば、到達可能性を判定するための索引として都合

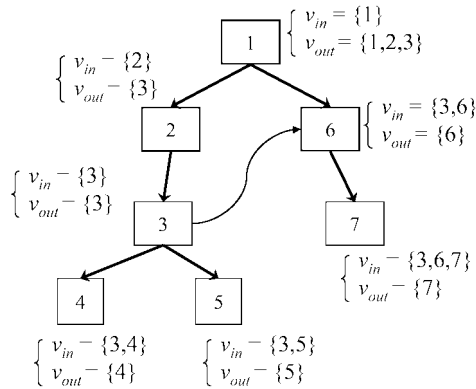


図 2. 自明でない 2Hop ラベル

Fig. 2. A non-trivial 2Hop label

がよい。図 2 にそのような 2Hop ラベルの例を示す。

しかしながら、与えられた一般の有向グラフに対して、最小の 2Hop ラベルを計算する問題は NP-困難である。[7] は、この問題を近似的に解くアルゴリズムを提案し、最適解に対して $O(\log n)$ 倍以内のサイズの 2Hop ラベルを効率的に計算できることを示している。

[15] は、この手法を改良し、大規模なデータに対する実装が可能な HOPI アルゴリズムを提案している。判定時間はラベル数に比例するが、各ノードの平均ラベル数はそれほど大きくならないことが実験によって示されている。ところがこの手法では、分割した結果を統合することで処理時間が増加すること、あらかじめ到達可能なノードの組を候補集合として計算しておかなければならないことなどの主に計算時間と主記憶量に関する欠点がある。

本研究では、このような有向グラフの到達可能性判定問題に対して、範囲ラベルと 2Hop ラベルの利点を組み合わせた索引構造を提案し、前処理時間と主記憶量に関するコストを抑えながら、ノード間の到達可能性を高速に判定できる索引構造を実装する。特に本提案手法では、2Hop ラベルに必要な $O(n^2)$ サイズの候補集合全体をあらかじめ計算する必要がないため規模耐性が高い。

2. 提案手法

本研究が対象とする XML は、次のようにラベル付きグラフで表現される。要素はグラフのノードに対応し、要素間の二項関係がノード間の辺に対応する。すなわち、要素が直近の入れ子関係にあるときや、特定の異なる要素が同じ値を持つときに、そのノード間に辺を定義する。本論文では、要素間の入れ子によって定義される辺を実辺、要素の値によって定義される辺を参照辺と呼ぶ。

参照辺は要素に付加的に定義される特定の属性とその値の組によって定義される。例えば XMark [16] では、ある要素の ID 型属性と別の要素の IDREF 型属性が同じ値を持つことで、この 2 要素間の参照辺を実現している。この参照辺は、XLink で記述される単純リンクに相当する。

本研究で提案するラベル構築アルゴリズムは 3 つのフェーズからなる。第 1 フェーズでは、与えられた有向グラフ $G = (V, E)$ を強連結成分分解し、非巡回有向グラフ G' に変換する。この時間計算量および領域計算量は $O(|E|)$ および $O(|V|)$ である。第 2 フェーズでは、 G' を深さ優先探索し、全域木 T とその範囲ラベルを計算する。もし、 $T = G'$ ならば索引付けは終了する。この計算量は第 1 フェーズと同じである。第 3 フェーズでは、各ノードの 2Hop ラベルを計算する。提案手法は、各ノード v に対して v_{in}, v_{out} が被参照ノードのみを要素として持つ点が、これまでの手法と異なる。通常は被参照ノードは、ノード全体に対してあまり大きくないので、このよう制限することで、構築されたラベル

アルゴリズム $V_{IN}(T)$

入力：連結な非巡回有向グラフ $G = (V, E)$ の全域木 T

出力：任意の $v \in V$ に対するラベル v_{in}

- (1) T を深さ優先探索： $(v$: カレントノード, p : v の親)
 - (1-1) v が被参照ノードならば、 $v_{in} \leftarrow v_{in} + \{v\}$.
 - (1-2) $v_{in} \leftarrow v_{in} + p_{in}$, $v \rightarrow next$.
- (2) $\{v_{in} \mid v \in V\}$ を出力。

図 3. v_{in} 構築アルゴリズム

Fig. 3. Algorithm for v_{in}

アルゴリズム $V_{OUT}(T)$

入力：連結な非巡回有向グラフ $G = (V, E)$ の全域木 T

出力：任意の $v \in V$ に対するラベル集合 v_{out}

- (1) T を深さ優先探索する： $(v$: カレントノード)
 - (1-1) 参照辺 $(v, u) \in E$ があれば、 $V_{OUT}(T_u)$ を呼ぶ。ただし、 T_u は u を根とする T の部分木。
 - (1-2) u が探索済みならば、その親 v へ戻り、
 $(v, u) \in E$ が参照辺ならば、 $v_{out} \leftarrow v_{out} + \{u\}$.
 $(v, u) \in E$ が実辺ならば、 $v_{out} \leftarrow v_{out} \cup \{u\}$.
- (2) v_{out} を出力。

図 4. v_{out} 構築アルゴリズム

Fig. 4. Algorithm for v_{out}

のサイズも抑えられると予想される。

図 3 にラベル v_{in} 構築アルゴリズムを示す。このアルゴリズム $V_{IN}(T)$ は与えられたグラフの全域木の辺のみを深さ優先探索するので、計算時間は $O(\ell|V|)$ である。ただし、 ℓ は被参照ノードの数、すなわち各ノードに継承されるラベルの最大数である。

同様に、図 4 にラベル v_{out} 構築アルゴリズムを示す。このアルゴリズム $V_{OUT}(T)$ は、グラフの辺を高々一度探索するので、計算時間は $O(\ell|E|)$ である。これらのアルゴリズムの動作例を、それぞれ図 5 と図 6 に示す。

直感的には、 v_{in} は v の先祖の被参照ノードの集合を表し、 v_{out} は v の子孫の被参照ノードの集合を表す。したがって、浅く広い構造である XML データのようなものほど、 v_{out} のサイズが極端に大きくなる傾向がある。これは判定時間の増加に直接影響するため、 v_{out} をさらに削減する次のような工夫を導入する。

これまで、アルゴリズム $V_{OUT}(T)$ において、あるノード x から実辺や参照辺を辿って親 y に戻るとき、 x_{out} のラベルをそのまま y_{out} に継承していた(図 6 を参照)。ここで、 y から x_{out} のあるラベルへ到達可能であることが範囲ラベルで判定できれば、そのラベルは y_{out} に継承する必要がない。そこで、あらかじめすべてのノードの範囲ラベルを計算しておくことで、 unnecessary 2Hop ラベルの継承を抑えることができる。以降では、このように改良されたラベル構築アルゴリズムを実装してその有効性を実験によって確認する。

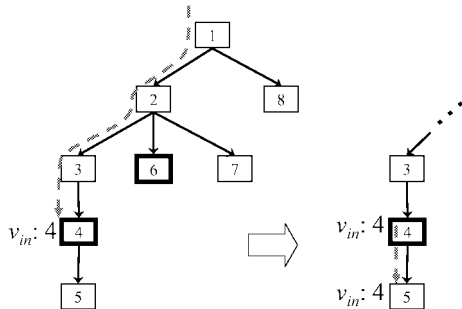


図 5. v_{in} の計算: 参照辺は省略されている

Fig. 5. Computation of v_{in}

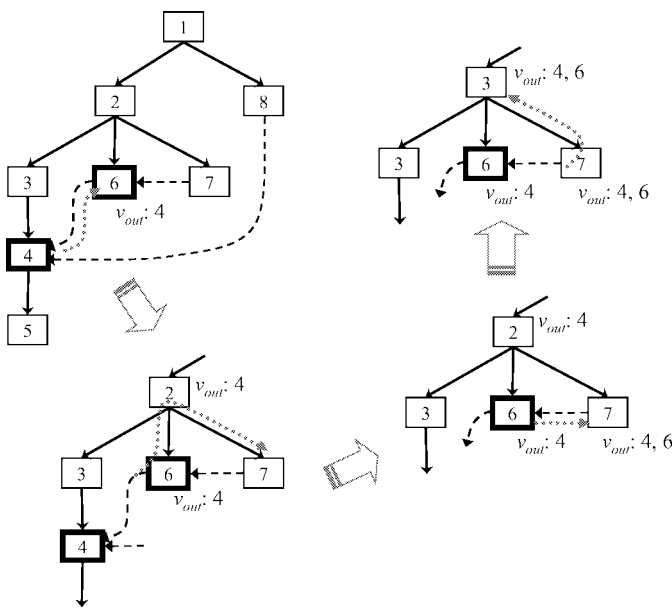


図 6. v_{out} の計算: 破線は参照辺を表す

Fig. 6. Computation of v_{out}

3. 実験

本手法の有効性を実験によって検証する。実験データは、XMark [16] によって生成した XML データ (120KB, 1.2MB, 12MB) を用いた。実験環境は、Celeron 3.2GHz, 992MB メモリ上の WindowsXP である。実験は、本手法における前処理時間および到達可能性を判定する時間を測定し、HOPI [15] との性能比較を行った。

提案手法で得られた 2Hop ラベルのサイズを、1 ノードあたりの平均ラベル数と最大ラベル数について HOPI のものと比較した結果を表 1 に示す。本手法では範囲ラベルも判定に用いるので、実際のノードあたりの平均ラベル数は、この表の値よりもちょうど 2 個大きい。しかしその値を加えても、本手法の方が HOPI よりも平均ラベル数が小さいことが示されている。また平均ラベル数の差はわずかなため、実際の判定時間は、提案手法と HOPI ではそれほど差はないと考えられる。後で示す実際の測定結果もそれを裏付けている。

さらに、表 1 より、データの増加に対して平均ラベル数は、本手法では増加傾向にあるが、HOPI ではほとんど変化していない。このことから、さらに大規模なデータに対する判定時間については、HOPI の方がやや有利であると考えられる。

最大ラベル数は表 1 に示すように、有意な差がみられる。HOPI

表 1. ノードあたりのラベルサイズの比較

Fig. 1. Label size per node

データの種類	サイズ (KB)	平均ラベル数		最大ラベル数	
		提案手法/HOPI	提案手法/HOPI	提案手法/HOPI	提案手法/HOPI
XMark	116	1.2	5.3	23	345
	1155	1.4	5.3	246	2170
	11597	1.9	4.7	2515	7950

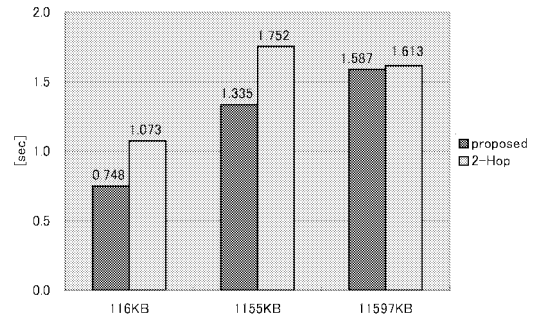


図 7. 判定時間の比較: 100 万回の判定の実行時間

Fig. 7. Time for 1 million testing

は漸的には良い値を示すが、数十 MB 程度のデータでは、提案手法の方が良い結果が得られている。また、用いた実行環境では、HOPI がファイルを分割せずに実行可能なデータサイズは 1MB 程度であり、それ以上のデータは適当にファイルを分割して実験を行っている。本提案手法は、 $O(n^2)$ のテーブルを構築する必要がないため、大規模なデータに対してもファイルを分割することなく直接ラベルを構築できる。このことは、以下に示す前処理時間の比較に現れている。

次に、提案手法と HOPI における前処理時間と到達可能性判定時間の比較結果を示す。図 7 は、ランダムに生成した 100 万組のノードに対する到達可能性判定時間を測定した結果である。これより、実験を行った規模のデータでは提案手法の方がわずかに高速である。しかしながら、判定した回数を考慮に入れるとこれらの判定時間に顕著な違いはない。また、データの増加に対して判定時間の差は縮まっており、前述の平均ラベル数における考察が裏付けられている。

図 8 は提案手法と HOPI の前処理時間の比較結果である。この

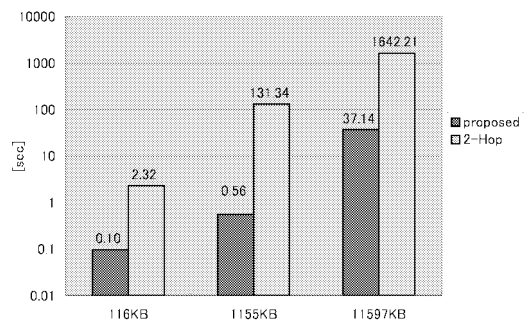


図 8. 前処理時間の比較: 範囲ラベル構築時間を含む

Fig. 8. Preprocessing time: range labeling contained

結果より、本研究で提案する範囲ラベルと 2Hop ラベルを組み合わせたラベル付け手法は、従来の手法に比べて規模耐性が高く十分高速に構築できることが示された。特に、前処理時間は HOPI と比較して最大 230 倍 (1.2MB のデータ) 効率的である。また、その結果得られた索引サイズと判定時間も従来手法と比較して遜色ないものといえる。以上により、本手法の有効性が確かめられた。

4. おわりに

本研究では、木構造よりも複雑な XML データに対して、高速にノード間の到達可能性を判定するための効率的なアルゴリズムを提案し、その効果を実験によって検証した。実験では、最適なラベルに対するよい近似が保証されている 2Hop ラベル構築アルゴリズム [15] と比較して、本提案手法の有効性を示した。本手法は、データサイズに対する規模耐性が高く、特に、他手法との比較において前処理時間の削減率が著しく向上している。しかしながら、比較対象の HOPI [15] では、ラベルの更新法も提案しており、ラベルを効率的に再構築する手法は本研究の今後の課題である。一方で、RDF のようなさらに複雑な構造に対する到達可能性問題へ本手法を応用することも重要な課題である。

【謝辞】

本研究の一部は、平成 18 年度科学研究費補助金 (基盤研究 (A) 課題番号 17200011 課題名 “大規模半構造データからの高速知識発見システムの開発”) の支援を受けて行われた。

【文献】

- [1] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web, Morgan Kaufmann, 2000.
- [2] S. Abiteboul, H. Kaplan, T. Milo, Compact labeling schemes for ancestor queries, In SODA 2001, pp. 547-556, 2001.
- [3] A. Aboulnaga, A. R. Alameldeen, J.F. Naughton, Estimating the Selectivity of XML Path Expressions for Internet Scale Applications, In VLDB 2001, pp. 591-600, 2001.
- [4] R. Agrawal, A. Borgida, H.V. Jagadish, Efficient Management of Transitive Relationships in Large Data and Knowledge Bases, In SIGMOD 1989, pp. 253-262, 1989.
- [5] A. V. Aho, J. E. Hopcroft, J. D. Ullman, Data Structures and Algorithms, Addison-Wesley, 1987.
- [6] L. Chen, A. Gupta, M. E. Kurul, Efficient Algorithms for Pattern Matching on Directed Acyclic Graphs, In ICDE 2005, pp. 384-385, 2005.
- [7] E. Cohen, E. Halperin, H. Kaplan, U. Zwick Reachability and Distance Queries via 2-Hop Labels, In SODA 2002, pp. 937-946, 2002.
- [8] E. Cohen, H. Kaplan, T. Milo, Labeling Dynamic XML Trees, In PODS 2002, pp. 271-281, 2002.
- [9] 江田毅晴, 天笠俊之, 吉川正俊, 植村俊亮, XML 木のための更新に強い節点ラベル付け手法, DBSJ Letters, Vol. 1, No. 1, pp. 35-38, 2002.
- [10] M. F. Fernandez, D. Suciu, Optimizing Regular Path Expressions Using Graph Schemas, In ICDE 1998, pp. 13-23, 1998.
- [11] R. Goldman, J. Widom, DataGuides: Enable Query Formulation and Optimization in Semistructured DataBases, In VLDB 1997, pp. 436-445, 1997.
- [12] T. J. Green, A. Gupta, G. Miklau, M. Onizuka, D. Suciu, Processing XML Streams with Deterministic Automata and Stream Indexes, ACM TODS, vol. 29, no. 4, 2004.
- [13] W.-Y. Lam, W.-N. Peter, M. Levene, XCQ: XML Compression and Querying System, In WWW 2003 (posters), 2003.
- [14] 中村有作, 舞田哲哉, 坂本 比呂志, 参照構造を持つ XML 上の高速な到達可能性判定, 人工知能学会論文誌, Vol. 22, No.2, pp. 191-199, 2007.
- [15] R. Schenkel, A. Theobald, G. Weikum, Creation and Incremental Maintenance of the HOPI Index for Complex XML Document Collections, In ICDE 2005, pp. 360-371, 2005.
- [16] A. Schmidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, R. Busse, Xmark: A benchmark for XML data management, In VLDB 2002, pp. 974-985, 2002.
- [17] T. Schwentick, XPath Query Containment, SIGMOD Record, Vol. 33, No. 1, pp. 101-109, 2004.
- [18] D. Srivastava, S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, Y. Wu, Structural joins: A primitive for efficient XML query pattern matching, In ICDE 2002, pp. 141-152, 2002.
- [19] 鳥井修, 白井智, 金井達徳, 非巡回グラフのための拡張レンジラベリング手法, DBSJ Letters, Vol.5, No.1, pp.157-160, 2006.
- [20] Y. Wu, J. M. Patel, H. V. Jagadish, Structural Join Order Selection for XML Query Optimization, In ICDE 2003, pp. 443-454, 2003.
- [21] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, On Supporting Containment Queries in Relational Database Management Systems, In SIGMOD 2001, pp. 425-436, 2001.

中村 有作 Yusaku NAKAMURA

2006 年九州工業大学情報工学部知能情報工学科卒業。現在、同大学院情報工学研究科情報科学専攻修士課程に在学中。半構造データからのデータマイニング、効率的な索引構造の研究に従事。日本データベース学会学生会員。

舞田 哲哉 Tetsuya MAITA

2005 年九州工業大学情報工学部知能情報工学科卒業。2007 年同大学院情報工学研究科情報科学専攻修士課程修了。高速な XML データベース、大規模な半構造データの索引付けに関する研究に従事。現在、(株)日立製作所に勤務。

坂本 比呂志 Hiroshi SAKAMOTO

九州工業大学情報工学部准教授。1998 年九州大学大学院システム情報科学研究科情報科学専攻博士後期課程修了 (日本学術振興会特別研究員)。博士 (理学)。1999 年から 2003 年 7 月まで九州大学大学院システム情報科学研究科助手。機械学習、半構造データからの知識獲得および半構造データに対する索引付けに関する研究に従事。2006 年度人工知能学会論文賞などを受賞。人工知能学会、情報処理学会、日本データベース学会各会員。