二次記憶上の大規模グラフの処理に関する研究

1999年3月

進登谷 祥一

二次記憶上の大規模グラフの処理に関する研究

1999 年 3 月

能登谷 淳一

要旨

大規模データの管理と活用を目的とする計算機応用分野は急速に拡大し，各応用分野におけるデータの大規模化，多様化が進んでいる．また，境界的領域の拡大により，既存のさまざまな分野において蓄積されたデータの相互運用が進んでいる．それらの応用分野に対応する計算機システムの構築に際しては，複数の応用間での統合的なデータ利用と，効率的な二次記憶上データの管理が要求される．そのような要求への対応のため，データベース領域の技術を用いた汎用性の高いソフトウェア部品やデータベース管理システムの供給が望まれている．ところが，従来のデータベース分野の研究では，対象とする応用分野を主に事務処理などの限られた領域に限定していたため，それらの分野で扱われることの少なかった多くの問題については，十分な研究がなされていなかった．

そのような，多くの応用分野において重要であるにもかかわらず，従来データベース分野においては十分な研究が行なわれて来なかった問題の例として，グラフ上の問題に帰着する問題群があげられる．

本研究では，大規模グラフ上の問題に帰着可能であるような多くの応用分野の問題に対して有効である，データベース管理システムの機能もしくはソフトウェア部品として利用可能である手法を提案する．

本研究では，二次記憶上の大規模グラフに適した最短路探索アルゴリズムである，DF アルゴリズムと，大規模グラフの操作に適したページ置換手法である，KNC-D ページ置換の 2 つの手法を提案する．

DF アルゴリズムは Dijkstra の最短路探索アルゴリズムと本質的に同一の原理にしたがって動作する単一始点最短路探索アルゴリズムであり，応用分野に固有のデータの性質に関する知識を必要とせずに，Dijkstra の最短路探索アルゴリズムと比較して少ない二次記憶参照数で最短路を計算する．

KNC-D ページ置換手法は，今日データベースの分野で一般に用いられている LRU ページ置換手法に基づく必要時ページ置換手法である．KNC-D ページ置換を用いることにより，グラフに対する多くの演算に共通するデータ参照の特徴を利用した効率的バッファ管理が可能である．

目次

図一覧

表一覧

第 1 章

はじめに

大規模データの管理と活用を目的とする計算機応用分野の拡大と，各応用分野におけるデータの大規模化，多様化が進んでいる．それらの応用分野に対応する計算機システムの構築に際し，複数の応用プログラム間での統合的なデータ利用と，効率的な二次記憶上データ管理機構の実現に要する労力の削減のため，データベース領域の技術を用いた汎用性の高いソフトウェア部品や，データベース管理システムの供給が求められている．

ところが従来型のデータベースの主たる応用分野において対象とされる事が少なかった問題の多くについては，既存のデータベース管理システムなどでは効率的な解法が与えられていない．そのため，応用分野における問題の多様化および複雑化に対し，従来のデータベース管理システムなどによる十分な対応は難しい．

そのような，多くの応用分野において重要であるにも関わらず，従来のデータベース分野においては十分な研究がなされていない問題の例として，グラフ処理に帰着する問題群がある．

本研究ではデータベース管理システムの機能もしくは汎用性の高いソフトウェア部品として利用可能であるような，効率的大規模グラフ処理機構の提供を目的とする．本研究で提案する手法によって，応用分野に依存するデータ固有の性質を利用可能であるか否か，確保可能である主記憶の総量の大小などに関係なく，グラフに

対する処理が可能となる.

1.1 大規模グラフ応用分野における要求の多様化

大規模グラフ問題に帰着するような問題を含む，地理情報処理 [RHDM86, GK93, Suz97] や演繹データベース [AJ90, AJ94] などの応用においても，他の応用分野と同様にデータ処理に対する要求の多様化が進んでいる.

例えば，地理情報処理の重要な応用である高度道路交通システム (ITS) [JHR98] においては，時刻とともに変化する道路網上の流量情報や規制情報，事故情報などをもとに，将来の道路網の状態を予測し，道路利用者や管理者などに最適の経路情報を提供することが必要とされる．このような問題は，グラフ上の最大フロー問題や最短路探索問題，最小全域木導出問題などの組合せに帰着する.

これらの分野においては，従来より大規模グラフに対する処理を支援するシステムが用いられてきた．そのようなシステムの多くにおいては，グラフ処理は応用分野のデータ固有のいくつかの限られた問題を解決する際に用いられ，その処理の効率化のために応用分野固有のデータの性質を利用した発見的手法 [KHI+86, SKC93] や近似解法 [KKT95] が用いられてきた.

大規模グラフ処理に関する発見的手法としては，A* アルゴリズム群を利用する方法などがよく知られている [SKC93] ． A* アルゴリズム群はある評価関数を最小化するような解を求める問題に対して，より容易に評価可能であるような他の関数（パラメータ関数）を最小化するような解から順に探索を進める手法の一つであり，経路長を評価関数とするようなグラフ上の経路探索においては，ユークリッド空間における距離などをパラメータ関数とする A* アルゴリズムなどが多く用いられている.

また，近似解法としては，グラフ上の演算を幾何演算で代用する手法などが現実的な手法として多く用いられている.

しかし， ad-hoc 問合せへの対応など，応用分野における要求の多様化に従い，発

見的手法や近似解法の適用が困難であるような問題の解決が望まれている.

地理情報システムによる道路網上の経路探索において，所要時間など地理的経路長以外の指標を最適化するような経路に対する探索を行なう場合，地理情報システムで従来用いられてきたユークリッド空間上の距離をパラメータ関数とする A* 手法の適用は，正しい解を導出しない.

また，応用分野における問題のある部分問題が，グラフ上の問題に帰着するような，複雑な問題の要求が高まっている．グラフ上の問題を解決する従来の近似解法に関しては，現在までに十分な精度の解析が行なわれておらず，他の問題の入力として与えるためのデータとしては，それらの手法で得た結果を利用することは適切ではない場合があると考えられる.

さらに，複数の応用分野間での統合的なデータ利用の促進に伴い，各応用分野において，それまでとは異なる特徴を持ったデータの処理が要求されている．そのような場合にも，従来用いられてきた発見的手法や近似解法をそのままの形では適用可能ではない.

1.2 新しい応用分野の出現

一方で，従来は大規模グラフに対する処理が必要と考えられていなかった多くの分野において，データマイニング的手法によるデータ利用 [FPSS96, FHS96] の導入のために，大規模グラフ処理への要求が増大している.

データマイニング的手法の一つである，関連ルール [AIS93, BMS97, TUA+98] の導出とその利用においては，対象とする標本空間 Ω と適当に定めた閾値 M_s, M_c に対し，グラフ

$$
\begin{aligned}
G &= (V_G, E_G, W_G) \\
V_G &= \{X | X \subset \Omega, P(X) > M_s\} \\
E_G &= \{(X, Y) | X, Y \in V_G, P(X|Y) > M_c\} \\
W_G(X, Y) &= P(X|Y)
\end{aligned}
$$

の特徴より対象の性質を分析する.

著者らの研究グループでは，キーワード集合を問合せととして受け付ける文献検索システムに与える問合せの作成支援に，文献へのキーワードの出現を標本点とする関係ルールから得られる情報を利用する手法について研究を行なっている [CYN+98, LCYO98a, LCYO98b] .

このようなデータ利用においては，極めて大規模であるデータを単一のグラフと見做してグラフ演算を行なうことにより，データ全体の特徴や傾向を抽出することが可能となる.

従来大規模グラフに対する処理を必要としていなかったような応用分野のデータに対して，新たに大規模グラフ処理を行なう場合，発見的手法や近似解法の適用に有効である応用分野固有の性質の抽出は困難である．特に，データマイニング的手法は，多くの場合データからの知識抽出のために利用される処理であるので，その処理過程において既に応用分野固有の性質に関する情報が存在し，利用可能であると考えるのは，現実的ではない.

1.3 大規模データの管理

今日では，計算機ハードウェアの高速化，大容量化，価格の低下が進み，各応用分野において，従来と比較して豊富な計算機資源が利用可能となっている．それに伴って，一部の応用分野では処理に十分な主記憶を用意することにより二次記憶などの低速な記憶装置を利用せずに大規模データの処理が行なわれている．しかし，一般には，処理の対象となるデータ量は確保可能である計算機資源の量を越える速度で増大しており，その処理に必要となる資源の量を正確に予測することは困難である．特に，汎用のソフトウェア部品やデータベース管理システムなどは，どのような計算機システム上で動作する場合でも，確保可能である資源を常に効果的に使用し，必要とされる処理を実行する必要がある.

本研究では，グラフを表現するデータのうち，主記憶上に配置できない部分に関

しては，二次記憶など，より低速な記憶装置に格納されるものと仮定する．今日の多くのデータベース管理システムなどでは，二次記憶装置などに存在するデータの参照には必要時ページ置換 [Smi78] を用いたバッファ管理が行なわれる．多くの場合，これらのバッファ管理はオペレーティングシステムの管理下を離れ，データベース管理システムにあらかじめ割り当てられた資源を利用し行なわれる．本研究で提案する手法では，データベースシステム分野の既存の技術を利用し，グラフ処理以外のデータ処理と協調して動作することを目的とし，二次記憶上のデータに対する参照方式として，従来のデータベース管理システムと同様の方式を仮定する．その上で，二次記憶への参照が大規模グラフに対する処理全体のボトルネックとなっていることを仮定し，バッファ管理におけるページ置換数を処理の効率の指標と考える．

グラフアルゴリズムなどグラフ処理に関する既存の研究の多くにおいては，グラフを表現するデータは主記憶に入り切るものと仮定し，二次記憶への参照によって生じるボトルネックについては考慮されなかった．しかし，近年になってCPUの速度に関しては毎年 40-60% の向上が見られるのに対し，二次記憶装置の性能向上は，毎年 7-10% に留まっている [RW94] ．したがって，今後大規模グラフ処理においては，二次記憶へのデータ参照が処理全体のボトルネックとなることが予想される．

1.4 研究のアプローチ

本研究では二次記憶上の大規模グラフに対して効率的にグラフ上の問題を解決するアルゴリズムの提案と，グラフに抽象化されるデータに対する処理に必要となるデータ参照を効率的に行なうための機構の提案との二種類のアプローチにより，大規模グラフに対する処理性能の向上を実現する．

大規模グラフに対する処理を必要とする応用分野の問題がグラフ上の問題に直接帰着する場合，グラフ上の問題を効率的に解決するアルゴリズムによって，応用分野の問題の解決に必要となる処理の性能を向上可能である．ここで，グラフ上の問

題とは最小全域木導出やネットワークフロー問題，最短路探索問題など，問題の解決にグラフ全体の構造に関する情報を利用する必要があるような問題を指す．これらの問題を計算機によって解決する場合，一般に同一の問題を解決する複数のアルゴリズムが存在し，使用するアルゴリズムによって処理に必要となる時間や計算機資源の量が異なる．したがって，複数存在するアルゴリズムのうち，想定するシステムの構成などに応じて最も効率的に問題を解決可能であるアルゴリズムを選択可能であれば，グラフ上の問題に帰着する応用分野の問題の効率的解決が可能となる．

一方，応用分野の問題が単純にはグラフ上の問題に帰着しない場合であっても，多くの応用分野において処理の対象となるデータはグラフとして抽象化可能である．そのような場合には，グラフの性質を利用可能であるようなデータ参照の機構を利用することによって応用分野の問題の解決に必要となる処理全体の性能を向上可能である [NGV96]．このような手法は，グラフの特定の頂点や辺に付随する情報のみの参照によって解決可能であるような小規模な問題の繰り返しによって解決可能であるような応用分野の問題や，グラフとして抽象化可能であるようなデータの閲覧など，対話的に解決可能であるような応用分野の問題に対して特に有効であると考えられる．

本研究では二次記憶への参照が大規模グラフ処理全体のボトルネックとなることを仮定し，上記の二種類のアプローチの各々に対して主記憶上のデータのみを利用して実行可能であるような処理を余分に実行することにより，二次記憶への参照数を削減する事を目的とする．

1.5 論文の構成

本論文の 2章以降の構成は以下の通りである．

2章においては，二次記憶上の大規模グラフに対して効率的にグラフ上の問題を解決するアルゴリズムの一例として，二次記憶上の大規模グラフの処理に適した最短路探索手法を提案し，評価を行なう．

2.1節では，従来の最短路探索アルゴリズムを用いて二次記憶上の大規模グラフに対する最短路探索を行なった場合に発生する問題について議論し，二次記憶上の大規模非負重みつき単純有効グラフに対する単一始点最短路探索アルゴリズムとして，DF を提案する．また，DF の単一始点最短路探索アルゴリズムとしての正当性を検証し，二次記憶への参照数という観点からの性能が従来のアルゴリズムと比較して下回らないことを示す．

2.2節では，二次記憶上の大規模非負重みつき単純有効グラフに対する二頂点間最短路探索手法として，DF アルゴリズムとグラフ上の探索空間縮小法とを連携して用いる手法を提案する．

2.3節では，本研究で提案する単一始点最短路探索手法と二頂点間最短路探索手法の双方の性能を実験により確認し，対象となるグラフの性質や処理を実行する計算機環境などが処理の性能に及ぼす影響について検証する．実験においては，既存の最短路探索アルゴリズムを二次記憶上の大規模グラフに対して直接適用した例との比較を行なう．

3章においては，グラフに抽象化されるデータに対する処理に必要となるデータ参照を効率的に行なうための機構の一例として，グラフ処理の際に出現する二次記憶参照パターンに対して有効に動作するページ置換方式によるバッファ管理機構を提案し，評価を行なう．

3.1節では，本研究で対象とする大規模グラフデータ処理におけるデータ参照のパターンの特徴と，その効率的処理に必要なバッファ管理機構の性質について議論する．

3.2節では，第一節で議論した性質を満たすバッファ管理機構として，グラフ処理の際に出現する二次記憶参照パターンに対して有効に動作するページ置換方式である，KNC-D ページ置換方式を提案する．

3.3節では，KNC-D ページ置換方式を用いた，バッファ管理アルゴリズムの提案を行なう．

3.4節では，本研究で提案するバッファ管理アルゴリズムによるバッファ管理性能

を実験により確認する.

4章においては，本研究全体のまとめを行ない，今後の研究課題について議論する.

付録 A では，第二章第一節における DF アルゴリズムの正当性の検証に必要となる各種の補題の証明を行なう.

第 2 章

二次記憶上の大規模グラフのための最短路探索手法

2.1 最短路探索アルゴリズム DF：データ構造とアルゴリズム

2.1.1 グラフと最短路探索問題

本研究では，非負重みつき単純有向グラフに対する単一始点最短路探索問題および二頂点間最短路探索問題を解決する手法を提案する．これらの問題は多くの応用分野の問題が帰着する一般性の高いグラフ上問題である．また，主記憶に格納されたグラフに対してこれらの問題を解決するための手法に関しては十分に研究が進んでおり，Dijkstra による著名なアルゴリズムなど，効率的な解法が多く提案されている．さらに，これまでの研究により，二次記憶上の大規模グラフに対し，これらの問題を解決する場合に，隣接リストの取得 [Jia92] と作業領域の管理 [CGG+95, Arg95, KS96] の性能が処理全体の効率を決定することが知られている．

頂点集合 V_G と有向辺集合 $E_G \subset V_G \times V_G$，重み関数 $W_G : E_G \to \mathbf{R}^+$ からなる非負重み付き単純有向グラフ $G = (V_G, E_G, W_G)$ を考える．$V_P \subset V_G$, $E_P \subset E_G$ とし，$W_G|_{E_P}$ を W_G の E_P による縮小とすると，グラフ G の部分グラフ $P = (V_P, E_P, W_G|_{E_P})$ が $E_P = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{i-1}, v_i), \ldots, (v_{l-1}, v_l)\}$ （但し $|V_P| = l$,

$v_i \neq v_j$) の形をしているとき， P は G 上の経路であるという．このとき v_1 を P の始点と呼び， $\mathrm{src}(P)$ で表す．また， v_l を P の終点と呼び， $\mathrm{dest}(P)$ で表す．$\mathrm{L}(P) = \sum_{e \in E_P} W_G|_{E_P}(e)$ を P の経路長と呼ぶ．

G の全ての経路からなる集合を $\mathcal{P}_G$ と表す．二頂点 $x, y \in V_G$ に対し，

$$\exists P \in \mathcal{P}_G . \mathrm{src}(P) = x \wedge \mathrm{dest}(P) = y$$

のとき， y は x から到達可能であるという．また， $\mathcal{P}_G(x, y)$ によって x を始点とし， y を終点とする全ての経路からなる集合を表す．

ここで，

$$\mathcal{S}_G(x, y) = \{P | \forall Q \in \mathcal{P}_G(x, y) . \mathrm{L}(P) \leq \mathrm{L}(Q)\}$$

によって，グラフ G の頂点 x から y への全ての最短路からなる集合を表す． $P \in \mathcal{S}_G(x, y)$ のとき， P は頂点 x から頂点 y への最短路であるという．

$$\gamma_G(x, y) = \min\{\mathrm{L}(P) | P \in \mathcal{S}_G(x, y)\}$$

を頂点 x から頂点 y への最短路長と呼ぶ．ただし， $\mathcal{S}_G(x, y) = \emptyset$ のときは， $\gamma_G(x, y) = \infty$ とする．

本研究では，単一始点最短路探索問題をグラフ G と一頂点 $s \in V_G$ が与えられたとき， G の各頂点 $v \in V_G$ に対して， s から v への最短路長 $\gamma_G(s, v)$ と，もしあれば最短路の一つ $P \in \mathcal{S}_G(s, v)$ を発見する問題と規定する．

また，二頂点間最短路探索問題をグラフ G と二頂点 $s, d \in V_G$ が与えられたとき， s から d への最短路長 $\gamma_G(s, d)$ と，もしあれば最短路の一つ $P \in \mathcal{S}_G(s, d)$ を発見する問題と規定する．

2.1.2 グラフの二次記憶への格納とドメイン分割

グラフ上の経路探索アルゴリズムの多くは，ある頂点 $u \in V_G$ に対し， u を始点とする辺とその重みからなる対の集合を求める操作を必要とする．この集合を u の

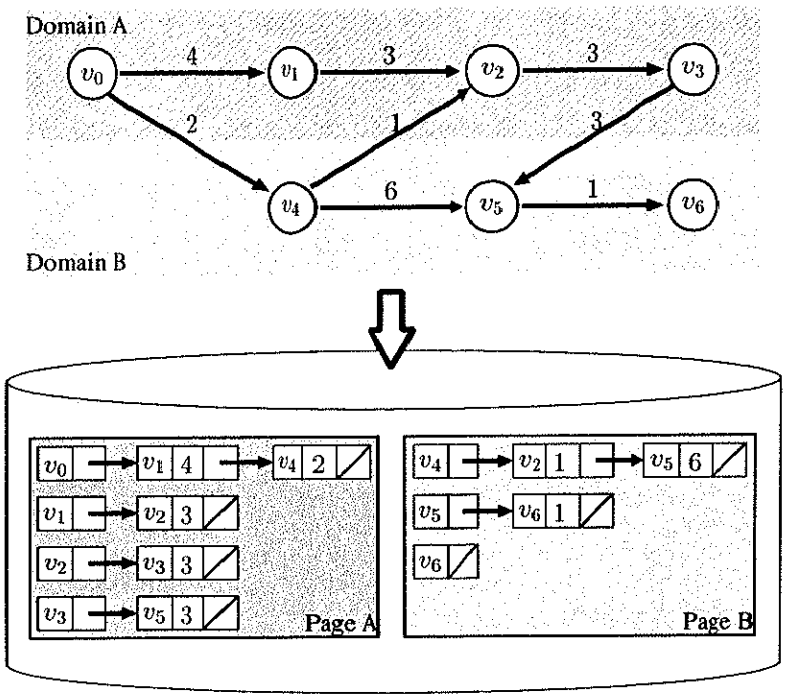


図 2.1: ドメイン分割の例

隣接リストと呼ぶ．グラフが二次記憶上のデータとして格納されている場合，隣接リストの取得には少なくとも一つのページへの参照を必要とする．以降では，議論の単純化のために，任意の $u \in V_G$ に対して，u の隣接リスト取得に必要とされるページ参照数は，常に 1 であると仮定する．

ここで，二頂点 $x, y \in V_G$ について，各々の隣接リスト取得のために参照を必要とするページが互いに等しいとき，$x \sim y$ と表す．同値関係 $\sim$ による V_G の分割 $V_G/\sim$ をグラフ G のドメイン分割と呼び，個々の同値類 $D \in V_G/\sim$ をドメインと呼ぶ．また，頂点をその属するドメインに対応付ける写像 $\mathcal{D} : V_G \to V_G/\sim$ をドメイン関数と呼ぶ．

二次記憶上のグラフデータが，グラフの典型的な表現形態であるグラフの隣接リ

スト表現を用いて格納されている場合について，図2.1にドメイン分割の例を示す．隣接リスト表現では，グラフの各頂点の接続辺とその重みに関する情報を線形リストを用いて表す．そのため，線形リスト構造を単位としたクラスタ化を行なうことにより，個々の頂点に対する隣接リストを少ないページ参照数によって取得可能である．図2.1の例では，グラフ G の頂点 v_0, v_1, v_2, v_3 の隣接リストは同一のページ A に格納され，同様に v_4, v_5, v_6 の隣接リストは同一のページ B に格納されている．このとき，グラフ G は

$$V_G/\sim = \{A, B\}$$
$$A = \{v_0, v_1, v_2, v_3\}$$
$$B = \{v_4, v_5, v_6\}$$

なるドメインに分割されていると考える．

以降では， Fetch(D) によってドメイン $D \in V_G/\sim$ の各頂点の隣接リストからなるページに対する参照を行なう手続きを表す．手続き Fetch(D) は， D の各頂点に対する隣接リストの取得に必要であるページのうち，主記憶上に存在しないものを二次記憶から主記憶へ移動する．実際にページの移動が発生する事を，二次記憶への参照と呼ぶ．

2.1.3 Dijkstra のアルゴリズム

本研究で提案する手法は， Dijkstra の最短路探索アルゴリズム [Dij59] と同一の原理に従って動作する．

Dijkstra の最短路探索アルゴリズムは，応用分野の特性に依存しない単一始点最短路探索アルゴリズムであり，各頂点を各々一回づつ訪問する事で計算を終了する事から，PSP アルゴリズム [GKP85] など，他の単一始点最短路探索アルゴリズムと比較して，二次記憶上のグラフに対しては有効である事が知られている [Jia92] .

Dijkstra の最短路探索アルゴリズム（図2.2）では，始点 $s \in V_G$ から各頂点 $v \in V_G$ への最短路長の上限値を格納する配列 cost[v] と L(P) = cost[v] を満たす s から

```
1  Procedure Dijkstra(G, 𝒟, s)
2  begin
3    InitializeGraph(V_G, s);
4    Explored := ∅;
5    while V_G − Explored ≠ ∅ do
6    begin
7      u := selectmin(cost, V_G − Explored);
8      Explored := Explored ∪ {u};
9      Fetch(𝒟(u));
10     foreach v in{v|(u, v) ∈ E_G} do
11       Relax(u, v, W_G)
12   end
13 end
```

図 2.2: Dijkstra のアルゴリズム

v への経路 $P \in \mathcal{P}_G$ の一つに関してその有向辺集合を格納する配列 Path[v] ，最短路長の計算を終えた頂点からなる集合 Explored を用いて最短路を探索する．

具体的には， V_G − Explored の要素のうち， cost[u] を最小にする u について，u を Explored に追加し，手続き Relax を適用して u に隣接する各頂点 v に対する cost[v] の値の更新を行なう（図 2.3）．これにより， cost[v] の値が cost[u] の値と $W_G(u, v)$ を反映したものとなっている事を示す以下の述語 $\mathcal{V}(u)$ の成立を保証する．

$$\mathcal{V}(u) \Leftrightarrow \forall v \in V_G.(u, v) \in E_G \supset \mathrm{cost}[v] \leq \mathrm{cost}[u] + W_G(u, v)$$

u に隣接する全ての頂点 v に対して手続き Relax(u, v, W_G) を適用することを u の

```
1  Procedure InitializeGraph(V_G, s)
2  begin
3    foreach v in V_G do
4      cost[v] := ∞;
5    cost[s] := 0;
6    Path[s] := ∅
7  end


1  Procedure Relax(u, v, W_G)
2  begin
3    if cost[u] + W_G(u, v) < cost[v] then
4    begin
5      cost[v] := cost[u] + W_G(u, v);
6      Path[v] := Path[u] ∪ {(u, v)}
7    end
8  end
```

図 2.3: 最短路長上限値の初期化と更新

訪問と呼ぶ.

Dijkstra のアルゴリズムでは，グラフの各頂点に対する訪問を繰り返し適用する事により，アルゴリズムの停止時において，各頂点 $v \in V_G$ に対し $\mathrm{cost}[v] = \gamma_G(s, v)$, $\mathrm{Path}[v] \in \mathcal{S}_G(s, v)$ を得る.

2.1.4 DF アルゴリズム

本稿では，二次記憶への参照回数の削減を考え， Dijkstraのアルゴリズムを修正したアルゴリズム DF [NICO99] を提案する．

DF アルゴリズムは，各頂点を始点 s からの距離の小さい順に集合 Explored に追加するという点，および， Explored に追加された頂点 v については，それ以降 cost[v] の値を変更しないという点において， Dijkstraのアルゴリズムと類似の考え方に基づいている．一方， DF アルゴリズムと Dijkstra のアルゴリズムの相違点は，頂点の訪問およびそれに伴う cost の値の変更の回数および順序である．この違いにより， DF アルゴリズムは Dijkstra のアルゴリズムと比較して，少ない二次記憶参照回数によって最短路探索問題を解決可能である．

Dijkstra のアルゴリズムでは， 9 行から 11 行の処理は条件 $\mathcal{V}(u)$ の成立を保証するために実行される．そのため， 9 行の実行の直前において $\mathcal{V}(u)$ が成立する場合，u に対する 9 行から 11 行の処理の実行は必要なく，これにより 9 行における Fetch($\mathcal{D}(u)$) の実行回数を削減可能と考えられる．ところが， $\mathcal{V}(u)$ の評価には， u から接続する各辺の重みの値を必要とするため， u の隣接リストに対する参照を伴う．すなわち， $\mathcal{V}(u)$ の評価の直前には Fetch($\mathcal{D}(u)$) の実行を必要とする．

これを回避するため， $\mathcal{V}(u)$ の十分条件となり，隣接リストの参照を必要としないような条件を考える．

いま，アルゴリズムの進行時点から見て，最後に訪問を受けて以来，他の頂点に対する訪問によって更新を受けていないような頂点からなる集合 Valid を考える．すなわち，

- アルゴリズムの初期状態においては，全ての頂点 $v \in V_G$ について， $\mathcal{V}(v)$ が成立するか否かは不明であるため， Valid $= \emptyset$ とする．
- $u \in V_G$ の訪問時に，周囲の頂点に対する更新を行なった事により，条件 $\mathcal{V}(u)$ が成立した事を示すために， u を Valid に追加する．

- Relax(u, v, W_G) の実行時に，cost$[v]$ が更新される事により，$\mathcal{V}(v)$ が成立しなくなる可能性が生ずるため，v を Valid より削除する．

一方，頂点 u への訪問の直後においては，定義より $\mathcal{V}(u)$ は必ず成立し，それ以降いずれかの $x \in V_G$ に対する Relax(x, u, W_G) の実行があるまで，$\mathcal{V}(u)$ は成立し続ける．なぜならば，いま

$$\text{cost}[v] \leq \text{cost}[u] + W_G(u, v) \tag{2.1}$$

を仮定すると，この後に式 (2.1) が不成立となるためには

1. cost$[v]$ がより大きな値に変更される

2. cost$[u]$ がより小さな値に変更される

のいずれかが必要である．ところが，Dijkstra のアルゴリズムにおいて，任意の $x \in V_G$ に対する cost$[x]$ の値が変更されるのは手続き Relax においてのみであり，手続き Relax では，cost$[x]$ の値は以前より大きな値に変更される事はない．そのため，Relax によって cost$[u]$ の値が変更されない限り，式 (2.1) は成立し続ける．また，同様の議論が u に隣接する全ての v に対し成立する．したがって $u \in$ Valid は $\mathcal{V}(u)$ の十分条件である．

ここで説明の便宜のため，Dijkstra のアルゴリズムに対し集合 Valid を用いた判定を加えたアルゴリズム D1 を考える．D1 アルゴリズムの主手続きを図 2.4に，D1 によって利用される，集合 Valid の管理を伴う頂点訪問手続きを図 2.5に示す．

手続き D1 では，10 行において $u \in$ Valid となる機会は稀であるため，多くの場合，依然 11 行から 14 行の処理を実行する必要がある．ここでは，頂点 $x \in V_G$ に対する訪問によって x が集合 Valid に追加される性質を利用し，以降の処理の過程において x が 8 行で u として選択される場合の，$u \in$ Valid となる機会の増大を試みる．

```
1  Procedure D1(G, D, s)
2  begin
3    InitializeGraph(V_G, s);
4    Explored := ∅;
5    Valid := ∅;
6    while V_G − Explored ≠ ∅ do
7    begin
8      u := selectmin(cost, V_G − Explored);
9      Explored := Explored∪{u};
10     if u ∉ Valid then
11     begin
12       Fetch(D(u));
13       Visit(u, E_G, W_G)
14     end
15   end
16 end
```

図 2.4: D1 アルゴリズム

いま，手続き D1 において，13 行の直後に任意の $u', v' \in V_G$ についての手続き Relax'(u', v', W_G) を追加する事を考える．このとき，以下に示す補題 1 より，手続き Relax' の追加は，アルゴリズムの停止時における最終的な処理の結果には影響を与えない．

補題 1 (文献 [CLR90]) グラフ G に対して任意の頂点 $s \in V_G$ を始点とする手続き InitializeGraph(V_G, s) を実行した後，任意の頂点列 $x_1 x_2 \ldots x_n, y_1 y_2 \ldots y_n$ に対す

```
1  Procedure Visit(u, E_G, W_G)
2  begin
3    Valid := Valid ∪ {u};
4    foreach v in {v|(u, v) ∈ E_G} do
5      Relax'(u, v, W_G)
6  end


1  Procedure Relax'(u, v, W_G)
2  begin
3    if cost[u] + W_G(u, v) < cost[v] then
4    begin
5      cost[v] := cost[u] + W_G(u, v);
6      Path[v] := Path[u] ∪ {(u, v)};
7      Valid := Valid − {v}
8    end
9  end
```

図 2.5: 頂点訪問手続き

る手続き Relax'(x_i, y_i, W_G) の実行列を適用したとする．このとき

$$\forall v \in V_G. \mathrm{cost}[v] \geq \gamma_G(s, v)$$

が成立する．また，任意の $v \in V_G$ について，$\mathrm{cost}[v] = \gamma_G(s, v)$ が成立した時点以降，任意の時点において

$$\mathrm{cost}[v] = \gamma_G(s, v)$$

```
1  Procedure VisitDomain(D, E_G, W_G)
2  begin
3    while D − Valid ≠ ∅ do
4    begin
5      t := selectmin(cost , D − Valid);
6      Visit(t, E_G, W_G)
7    end
8  end
9
```

図 2.6: ドメイン訪問手続き

が成立する.

一方，頂点訪問手続き Visit は手続き Relax' の実行列からなる．そのため，任意の頂点集合 $\{u_i\} \subset V_G$ について，$\{u_i\}$ の各要素に対する手続き Visit は，最終的な処理の結果に影響を与えずに手続き D1 の 13 行の処理の直前へ追加可能である．この追加により，それ以降の処理の過程で 10 行の処理に到達した際の $u \in$ Valid となる機会が増大する．ここで，$\{u_i\} \subset \mathcal{D}(u)$ であるならば，各 Visit(u_i, E_G, W_G) の処理に必要とされる隣接リストは，12 行の Fetch$(\mathcal{D}(u))$ の実行によって主記憶に確保されているページより取得可能である．そのため，Visit(u_i, E_G, W_G) の実行には，Fetch$(\mathcal{D}(u_i))$ の実行を必要としない．

本研究では，$\{u_i\} \subset \mathcal{D}(u)$ なる u_i の系列を生成する手続きとして，図 2.6に示す手続き VisitDomain を考える．手続き VisitDomain(D, E_G, W_G) では，D の各頂点は高々一回訪問され，手続きの終了時には $D \subset$ Valid が成立する．そのため，手続き VisitDomain を手続き D1 の 13 行の直後において実行する事により，D1 の 10 行において $u \in$ Valid となる機会は大幅に増大する．

命題 1 手続き VisitDomain(D, E_G, W_G) の一度の実行に対し，D の各頂点は高々 *1* 回訪問される．

[略証]

いま，VisitDomain において i 番目に訪問される頂点を $t_i \in D$ とすると，t_{i+1} が訪問の対象として選択されるためには，その直前の時点で $t_{i+1} \notin$ Valid である必要がある．すなわち，以下の (1)，(2) のいずれかが成立する．

1. t_i への訪問の直前の時点で $t_{i+1} \notin$ Valid が成立する

2. t_i への訪問によって t_{i+1} が Valid から除かれる

(1) の場合は，t_i を訪問の対象として選択していることから，$\mathrm{cost}[t_i] \leq \mathrm{cost}[t_{i+1}]$ は明らかであり，(2) の場合は，t_i への訪問による更新の結果，$\mathrm{cost}[t_{i+1}] = \mathrm{cost}[t_i] + W_G(t_i, t_i + 1)$ となる．

したがって，t_{i+1} への訪問の直前において

$$\mathrm{cost}[t_i] \leq \mathrm{cost}[t_{i+1}]$$

以下同様に，

$$\mathrm{cost}[t_i] \leq \mathrm{cost}[t_{i+n}]$$

ゆえに，t_i への訪問によって $t_i \in$ Valid が成立し，それ以降 $\mathrm{cost}[t_i]$ は更新を受けないため，$t_i \in$ Valid は手続き VisitDomain の終了時まで成立し続ける．

以上により題意は示された．

Q.E.D.

D1 アルゴリズムの主手続き D1 に対して，手続き VisitDomain(D, E_G, W_G) による訪問を追加したアルゴリズムとして DF アルゴリズムを構成する．図 2.7に二次記憶上のグラフのための最短路探索手法 DF のアルゴリズムを示す．

```
1  Procedure DF(G, 𝒟, s)
2  begin
3    InitializeGraph(V_G, s);
4    Explored := ∅;
5    Valid := ∅;
6    while V_G − Explored ≠ ∅ do
7    begin
8      u := selectmin(cost, V_G − Explored);
9      Explored := Explored ∪ {u};
10     if u ∉ Valid then
11     begin
12       Fetch(𝒟(u));
13       Visit(u, E_G, W_G);
14       VisitDomain(𝒟(u), E_G, W_G)
15     end
16   end
17 end
```

図 2.7: DF アルゴリズム

DF アルゴリズムの主手続き DF は基本的には Dijkstra のアルゴリズムと類似の構造を持つが，8 行で選択された u が Valid に属するかどうかを 10 行で判定する．

ここで，u が Valid に属する場合は，条件 $\mathcal{V}(u)$ がこの時点で成立するものとして，以降の処理を省略する．このとき，もし，省略を行なわず，13 行の手続き Visit を実行したとしても，各 $v \in V_G$ に対する cost[v] の値は変化しない．さらに，14 行の VisitDomain は元来付加的な処理であるため実行する必要はない．これらの省略

により，12行の Fetch の実行回数を削減する．

一方，$u \notin$ Valid である場合，Dijkstra のアルゴリズムの場合と同様に，13行において u への訪問を行なう．このとき，$\mathcal{D}(u)$ に属する頂点に対する訪問に必要とされる手続き Fetch の実行は12行で行なわれているので，$\mathcal{D}(u)$ に属する頂点に対する訪問手続き VisitDomain を実行し，$\mathcal{D}(u)$ に属する頂点を全て Valid に属するようにする．ここで$\mathcal{D}(u)$ に属する頂点が，後に8行で u として選択される場合には，その間にValid から u を除く処理を行なわない限り，12行から14行の処理を省略可能である．

手続き DF の14行において，手続き VisitDomain の引数として与えられる関数 $\mathcal{D}(u)$ の値は，現実の実装においては，u の隣接リストのデータが格納されるページの識別子と考える事ができる．この識別子は，手続きDF の12行において u の隣接リストのデータ取得のために既に計算されているため，$\mathcal{D}(u)$ の値の導出のために二次記憶への余分な参照が発生する事はない．

手続き DF の8行で u が選択される回数および順序は，手続き Dijkstra の7行で u が選択される回数および順序と等しい．そのため，以上の処理の実行により，DF アルゴリズムによる手続き Fetch の実行回数は，Dijkstra のアルゴリズムによる Fetch の実行回数以下となる．

いま，図2.2に示す Dijkstra のアルゴリズムと，図2.7に示す DF アルゴリズムによって，図2.8のグラフ G に対する頂点 v_0 を始点とする最短路探索を各々行なう事を考える．ただし，ドメイン分割は

$$V_G/\sim = \{A, B\}$$
$$A = \{v_0, v_1, v_2, v_3\}$$
$$B = \{v_4, v_5, v_6\}$$

とする．

このときの，各 $u \in V_G$ の Explored への追加の直前における，$v \in V_G$ に対する cost[v] の値と，参照したドメインを表2.1，表2.2 にそれぞれ示す．表において，各

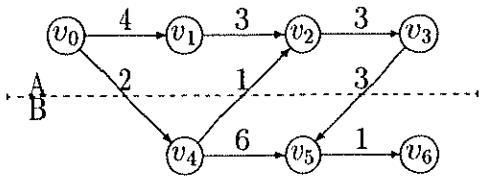


図 2.8: 実行例で用いるグラフ

表 2.1: Dijkstra のアルゴリズムの実行例

i	v_0	v_1	v_2	v_3	v_4	v_5	v_6	Fetch
0	0	∞	∞	∞	∞	∞	∞	
1	↑	4			2			A
2			3		↑	8		B
3			↑	6				A
4		↑						A
5				↑				A
6						↑	9	B
7							↑	B

行は i 個目の頂点の Explored への追加の直後の時点における cost[v] を示し，下線は頂点 $v \in$ Valid を，「↑」は $v \in$ Explored を示す．Fetch の欄は，Explored への追加の後に参照されるドメインを示す．

V_G は二つのドメインに分割されているにも関わらず，Dijkstra のアルゴリズムでは，処理の過程において 7 回のドメインへの参照が必要である．一方，DF アルゴリズムにおいては 3 回のドメインへの参照のみで，Dijkstra のアルゴリズムと同様の結果を導いている．

仮に，バッファ領域が 1 ページに等しい大きさを持つものとすると，Dijkstra の

表 2.2: DF アルゴリズムの実行例

i	v_0	v_1	v_2	v_3	v_4	v_5	v_6	Fetch
0	0	∞	∞	∞	∞	∞	∞	
1	↑	4	7	10	2	13	∞	A
2			3		↑	8	9	B
3			↑	6				A
4		↑						
5				↑				
6						↑		
7							↑	

アルゴリズムでは，4 回の二次記憶への参照を必要とするのに対し， DF アルゴリズムでは，3 回の二次記憶への参照しか必要としていない．

2.1.5 DF アルゴリズムの正当性

本研究で提案する最短路アルゴリズム DF の正当性を検証する．

検証の対象は，

- DF によって最短路および最短路長が計算できること
- DF による二次記憶への参照回数が Dijkstra のアルゴリズムによる参照回数よりも小さくなること

の 2 点である．

以下の議論では，非負重みつき単純有向グラフ $G = (V_G, E_G, W_G)$ と， G 上のドメイン関数 $\mathcal{D}$ について考える．また， $N = |V_G|$ とする．

手続き DF （図 2.7）において，8 行の文

$$u := \text{selectmin}(\text{cost}, V_G - \text{Explored});$$

の $i+1$ 回の実行直後の時点を T_i と表す．また，手続き DF の T_i の時点における変数 u の値を u_i により表す．明らかに $\{u_0, \ldots, u_{N-1}\} = V_G$ である．

述語 $\Delta(i, v)$ を以下のように定義する．

$$\Delta(i, v) \triangleq \mathrm{T}_i \text{ の時点で } \text{cost}[v] = \gamma_G(s, v)$$

以下の補題 2は DF アルゴリズムの正当性を示す定理 1と DF アルゴリズムにおける集合 Explored への頂点の追加順を示す定理 2の証明の中核となるものである．この補題の証明は付録にて行なう．

補題 2 *グラフ G に対し任意の頂点 $s \in V_G$ を始点とする DF アルゴリズムによる最短路探索* $\mathrm{DF}(G, \mathcal{D}, s)$ *を適用したとき，$i \leq j$ なる任意の $i, j \in \{0, \ldots, N-1\}$ に対し，T_i の時点において*

$$\begin{aligned} &\forall k \in \{0, \ldots, i-1\}.\, \Delta(k, u_k) \\ &\quad \supset \text{cost}[u_i] \leq \gamma_G(s, u_j) \end{aligned} \tag{2.2}$$

が成立する．

以下の定理 1によって DF アルゴリズムが最短路探索アルゴリズムとして正当であることが示される．

定理 1 (DF アルゴリズムの正当性) *グラフ G に対し任意の頂点 $s \in V_G$ を始点とする DF アルゴリズムによる最短路探索* $\mathrm{DF}(G, \mathcal{D}, s)$ *を適用したとき，アルゴリズムの終了時に，任意の $v \in V_G$ に対し*

$$\text{cost}[v] = \gamma_G(s, v) \tag{2.3}$$

を得る．

[証明] まず，任意の $i \in \{0,\ldots,N-1\}$ について，ループ不変式 $\Delta(i,u_i)$ が成立することを数学的帰納法により示す．

$i=0$ のときは $u_0=s$ より，T_0 において

$$\mathrm{cost}[s]=\gamma_G(s,s)=0$$

であるので，$\Delta(0,u_0)$ は成立する．

任意 $i \in \{0,\ldots,N-1\}$ に対し，全ての $j \in \{0,\ldots,i-1\}$ について $\Delta(j,u_j)$ の成立を仮定し，このとき $\Delta(i,u_i)$ の成立を示す．補題2を用いると，仮定より T_i において

$$\mathrm{cost}[u_i] \leq \gamma_G(s,u_i)$$

が成立．これと補題1より，任意の $i \in \{1,\ldots,N-1\}$ に対して $\Delta(i,u_i)$ は成立する．

以上と補題1より，アルゴリズム終了時における (2.3) の成立が示された．

Q.E.D.

また，以下の定理2において，DF アルゴリズムにおいては各頂点 $v \in V_G$ は始点 $s \in V_G$ からの最短路長 $\gamma_G(s,v)$ の短いものから順に集合 Explored へ追加される事を示す．これと同様の議論が Dijkstra のアルゴリズムに対しても可能であり，これらより，Dijkstra のアルゴリズムにおいて各頂点が Explored に追加される順番と DF アルゴリズムにおける順番とが等しい事がわかる．Dijkstra のアルゴリズムにおいては，Explored への頂点 $v \in V_G$ の追加に伴って Fetch($\mathcal{D}(v)$) による二次記憶への参照が必ず発生するが，DF アルゴリズムにおいては，v の Explored への追加にも関わらず Fetch($\mathcal{D}(v)$) が発生しない場合が存在する．

ゆえに，DF アルゴリズムによる二次記憶への参照列は，Dijkstra のアルゴリズムによる参照列の部分列となる．これにより，DF による二次記憶への参照回数は Dijkstra のアルゴリズムによる参照回数よりも小さくなる．

定理 2 グラフ G に対し任意の頂点 $s \in V_G$ を始点とする DF アルゴリズムによる最短路探索 $\mathrm{DF}(G, \mathcal{D}, s)$ を適用したとき，任意の $i, j \in \{0, \ldots, N-1\}$ に対し

$$i \leq j \supset \gamma_G(s, u_i) \leq \gamma_G(s, u_j) \tag{2.4}$$

が成立する．

[証明] 補題1と補題2より任意の $i, j \in \{0, \ldots, N-1\}$ に対し

$$\begin{aligned} &\forall k \in \{0, \ldots, i-1\}. \Delta(k, u_k) \wedge i \leq j \\ &\supset \gamma_G(s, u_i) \leq \gamma_G(s, u_j) \end{aligned}$$

が成立する．一方，定理1より任意の $k \in \{0, \ldots, N-1\}$ に対し $\Delta(k, u_k)$ が成立する．

以上により (2.4) が示された．

Q.E.D.

2.2 二頂点間最短路探索への適用

多くの応用では，単一始点最短路探索と同様に，一組の始点と終点との間の最短経路の探索すなわち，二頂点間の最短路探索の効率的な解決が要求される．

一般には，二頂点間最短路探索問題は単一始点最短路探索問題と等価であり，これらの漸近的意味での計算量の上限は等しい．しかし，本研究で問題とする二次記憶への参照について考えると，個々の参照の実行が非常に大きな計算量を伴うと捉えられるため，二次記憶への参照の回数を定数倍削減することにより，処理全体の性能向上につながると考えられる．

Dijkstra のアルゴリズムや，DF アルゴリズムは与えられた終点が集合 Explored に追加された時点で探索を終了する事により，容易に二頂点間最短路探索問題に対応可能である．これにより，探索の始点 s，終点 d に対し，$\gamma_G(s, d) < \gamma_G(s, v)$ な

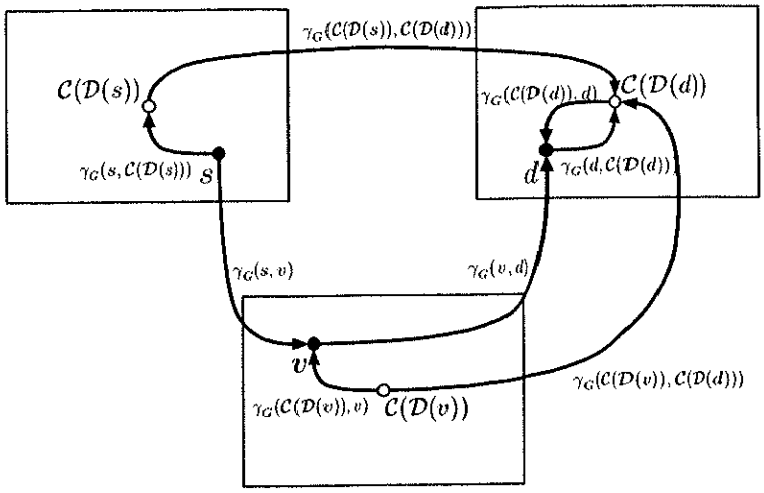


図 2.9: Domain Encoding 法による探索空間の縮小

る任意の $v \in V_G$ について，v の隣接リストを得るための二次記憶への参照を行なわずに二頂点間最短路探索を行なう．

しかし，上記の方法を用いても，実際には s から d への最短路の導出には不要であるような多くの頂点に対する隣接リストの取得と最短路長の計算が行なわれる．本研究では，Agrawal らの提案する，Domain Encoding 法 [AJ94] に基づくにより探索空間の縮小を行ない，DF アルゴリズムによる二頂点間最短路探索の実行に要する隣接リスト参照数の削減を試みる．

2.2.1 Domain Encoding 法

いま，各ドメイン D_i に対し，それぞれ一つの代表点 $c_i \in D_i$ を考える．写像 $\mathcal{C}$: $V_G/\sim \rightarrow V_G$ によってドメインから，そのドメインの代表点への対応を表す．

Agrawal らの提案する Domain Encoding 法では，以下の情報をあらかじめ計算し，二次記憶上に格納して経路探索における探索空間の縮小に利用する．

- 全てのドメインの対 $D_1, D_2 \in V_G/\sim$ について，$\gamma_G(\mathcal{C}(D_1), \mathcal{C}(D_2))$

- 全ての頂点 $v \in V_G$ について，$\gamma_G(\mathcal{C}(\mathcal{D}(v)), v)$ および $\gamma_G(v, \mathcal{C}(\mathcal{D}(v)))$

これらの情報を，最短路探索における部分的結果と呼ぶ．

ここで，任意の頂点 $v \in V_G$ が始点 s から終点 d への最短路上に存在するための必要十分条件は，

$$\gamma_G(s,v) + \gamma_G(v,d) = \gamma_G(s,d) \tag{2.5}$$

である．一方，

$$\begin{aligned} \gamma_G(s,d) \leq\ & \gamma_G(s, \mathcal{C}(\mathcal{D}(s))) \\ & + \gamma_G(\mathcal{C}(\mathcal{D}(s), \mathcal{C}(\mathcal{D}(d))) \\ & + \gamma_G(\mathcal{C}(\mathcal{D}(d)), d) \end{aligned} \tag{2.6}$$

であり，また，

$$\begin{aligned} & \gamma_G(\mathcal{C}(\mathcal{D}(v)), \mathcal{C}(\mathcal{D}(d))) \\ & \quad \leq \gamma_G(\mathcal{C}(\mathcal{D}(v)), v) \\ & \qquad + \gamma_G(v,d) \\ & \qquad + \gamma_G(d, \mathcal{C}(\mathcal{D}(d))) \end{aligned} \tag{2.7}$$

であるので，

条件

$$\begin{aligned} & \psi_1(v) \Leftrightarrow \\ & \gamma_G(s,v) \leq \gamma_G(\mathcal{C}(\mathcal{D}(s)), \mathcal{C}(\mathcal{D}(d))) \\ & \qquad - \gamma_G(\mathcal{C}(\mathcal{D}(v)), \mathcal{C}(\mathcal{D}(d))) \\ & \qquad + \gamma_G(s, \mathcal{C}(\mathcal{D}(s))) \\ & \qquad + \gamma_G(d, \mathcal{C}(\mathcal{D}(d))) \\ & \qquad + \gamma_G(\mathcal{C}(\mathcal{D}(d)), d) \\ & \qquad + \gamma_G(\mathcal{C}(\mathcal{D}(v)), v) \end{aligned} \tag{2.8}$$

は，頂点 v が s から d への最短路上の頂点であるための必要条件である（図 2.9）．

したがって，経路探索の過程において，最短路上の頂点のみに適用が必要であるような操作に関しては，$\psi_1(v)$ を充たす v に関してのみ適用すればよい．

2.2.2 DF アルゴリズムへの探索空間縮小法の適用

Dijkstra のアルゴリズムや DF アルゴリズムを用いた二頂点間最短路探索においては，頂点 u の Explored への追加の直後に実行される u への訪問は，u が s から d への最短路上に存在する場合にのみ適用を必要とする．このとき，$\mathrm{cost}[u] = \gamma_G(s,u)$ であるため，$\psi_1(u)$ は Domain Encoding 法の部分的結果を用いて評価可能である．

一方，手続き VisitDomain の 6 行におけるドメイン内頂点への訪問の適用について考えると，それらの訪問が実行される時点では $\mathrm{cost}[t] = \gamma_G(s,t)$ は必ずしも成立しないため，$\psi_1(t)$ の評価に必要となる $\gamma_G(s,t)$ の値を得ることはできない．そのため，DF アルゴリズムによる二頂点間最短路探索に対し，上記の手法は直接適用可能ではない．

そこで，本研究では，DF アルゴリズムの実行時に取得可能であるような情報のみを利用し，Agrawal らの提案する手法と同様の探索空間の縮小が可能であるような縮小条件を提案する．

DF アルゴリズムにおいて，VisitDomain の 6 行における訪問は，本来必ずしも実行する必要のない付加的な処理であるため，厳密な判定は必要ではない．そのため，この時点で処理を行うかについての判定は，$\psi_1(t)$ の定義の左辺である $\gamma_G(s,v)$ を $\mathrm{cost}[v]$ で置き換えた式による判定で十分である．このような判定の基準を用いた場合，手続き Relax によって $\mathrm{cost}[v]$ は以前より大きな値に書き換えられる事はないため，後に訪問の必要がないと判断される頂点 v が，それ以前に訪問の必要があると判断される事はない．

2.2.3 部分的結果の取得に要する二次記憶参照数の削減

Domain Encoding 法を用いることにより，Dijkstra のアルゴリズムや DF アルゴリズムにおいて，二次記憶上の隣接リストへの参照数の削減が可能となるが，式 (2.8) の判定に必要となる部分的結果の参照のために，余分に二次記憶への参照が発生する可能性がある．文献 [AJ94] では，部分的結果を表すデータ構造を階層的に構成す

ることにより二次記憶への参照数の削減を図っているが，本研究では，DF アルゴリズムと Domain Encoding 法とを組み合わせて使用することを仮定し，より単純な方法によって部分的結果の取得に要する二次記憶参照数の削減を行う．式 (2.8) の右辺各項に示される部分的結果の取得について考えると，第 1 項 $\gamma_G(\mathcal{C}(\mathcal{D}(s)),\mathcal{C}(\mathcal{D}(d)))$，第 3 項 $\gamma_G(s,\mathcal{C}(\mathcal{D}(s)))$，第 4 項 $\gamma_G(d,\mathcal{C}(\mathcal{D}(d)))$，第 5 項 $\gamma_G(\mathcal{C}(\mathcal{D}(d)),d)$ については，アルゴリズムの実行過程において変化することはないため，これらの取得に関するコストは定数であると考えて無視することができる．

一方，$\psi_1(v)$ はアルゴリズムの実行過程において，異なる v に対して何度も評価されるため，第 2 項 $-\gamma_G(\mathcal{C}(\mathcal{D}(v)),\mathcal{C}(\mathcal{D}(d)))$，および第 6 項 $\gamma_G(\mathcal{C}(\mathcal{D}(v)),v)$ は，判定の都度，新たに取得する必要がある．

本研究では，第 2 項 $-\gamma_G(\mathcal{C}(\mathcal{D}(v)),\mathcal{C}(\mathcal{D}(d)))$ の効率的取得のために，各代表点に関する部分的結果 $\gamma_G(c_i,c_j)$ の値を，c_j の値に従ってクラスタ化し，二次記憶へ格納しておくことを考える．すなわち，任意の c_j について，$\{\gamma_G(c,c_j)|c=\mathcal{C}(D)\wedge D\in V_G/\sim\}$ の格納に要するページの数をできる限り少なくする．

DF アルゴリズムの実行過程において d の値は変化しないため，このような二次記憶への格納方法により，第 2 項の値の参照の際に必要な情報が主記憶上に既に存在している可能性が増大する．

また，第 6 項 $\gamma_G(\mathcal{C}(\mathcal{D}(v)),v)$ についても，同様に少数のページに格納することによって二次記憶への参照数の削減を考える．Dijkstra のアルゴリズムや DF アルゴリズムによる二頂点間最短路探索においては，最悪の場合全ての $v\in V_G$ が $\psi_1(v)$ の引数として評価の対象となる．

そこで，本研究では第 6 項 $\gamma_G(\mathcal{C}(\mathcal{D}(v)),v)$ の代替としてドメインの半径 $\mathrm{r}_G(D)=\max\{\gamma_G(\mathcal{C}(D),v)|v\in D\}$ を用いた探索領域縮小条件を考える．

$$
\begin{aligned}
&\psi_2(v) \Leftrightarrow \\
&\mathrm{cost}[v] \leq \gamma_G(\mathcal{C}(\mathcal{D}(s)), \mathcal{C}(\mathcal{D}(d))) \\
&\qquad - \gamma_G(\mathcal{C}(\mathcal{D}(v)), \mathcal{C}(\mathcal{D}(d))) \\
&\qquad + \gamma_G(s, \mathcal{C}(\mathcal{D}(s))) \\
&\qquad + \gamma_G(d, \mathcal{C}(\mathcal{D}(d))) \\
&\qquad + \gamma_G(\mathcal{C}(\mathcal{D}(d)), d) \\
&\qquad + \mathrm{r}_G(\mathcal{D}(v)).
\end{aligned}
\tag{2.9}
$$

ここで，$\mathrm{r}_G(\mathcal{D}(v)) \geq \gamma_G(\mathcal{C}(\mathcal{D}(v)), v)$ であるので，$\mathrm{cost}[v] = \gamma_G(s, v)$ である場合には，ψ_2 は ψ_1 の必要条件である．

これにより，アルゴリズムの実行過程において探索空間縮小に要する部分的結果の取得のために参照するページの数を小さく抑えることが可能となる．

特に DF アルゴリズムにおいては，同一のドメインに属する頂点が連続して ψ_2 による判定の対象となるため，同一のドメインに対する訪問が行われている間は，第 2 項および第 6 項に関する部分的結果を格納するページをバッファ中に保存しておく必要がない．これにより，バッファ領域を有効に活用することが可能となる．

以上の探索空間の縮小法を DF アルゴリズムに採り入れるために，図 2.7の手続き Visit を図 2.10に示す手続きで置き換える．

探索空間の縮小と DF アルゴリズムを組み合わせたアルゴリズムにおいては，$v \in$ Valid は $\psi_2(v) \supset \mathcal{V}(v)$ の必要条件となる．

2.3 評価

DF アルゴリズムが二次記憶上の巨大グラフに対する最短路探索手法として有効である事を示すために，実験を行なった．

表 2.3に，実験で用いたグラフに関する各種パラメータを示す．ここで，ある辺 $e \in E_G$ について，e の始点と終点の属するドメインが等しい場合，e はドメイン内辺で

```
1  Procedure Visit(u, E_G, W_G)
2  begin
3    Valid := Valid ∪ {u};
4    if ψ_2(u) then
5      foreach v in {v|(u, v) ∈ E_G} do
6        Relax(u, v, W_G)
7  end
```

図 2.10: 探索空間の縮小を用いた訪問

あるといい，そうでない場合，e はドメイン間辺であるという．ドメイン内平均次数 ρ_i は，ドメイン内辺が各頂点から平均何本接続しているかを示し，同様に，ドメイン間平均次数 ρ_e は，ドメイン間辺が各頂点から平均何本接続しているかを示す．ドメイン内平均次数とドメイン間平均次数の和は，グラフ G の平均次数となる．ドメイン内辺の重みは標準偏差 p_i の Γ 分布に従うものとし，同様にドメイン間辺の重みは標準偏差 p_e の Γ 分布に従うものとする．

2.3.1 単一始点最短路探索

実験では，表 2.3の 2.11,2.13–2.16 の欄に示される各パラメータに従う有向ハミルトングラフと，2.12の欄に示されるパラメータに従う正則トロイダルグラフ（トーラス面埋め込みグラフ）を，同一パラメータにつき 20 個ずつ生成，Dijkstra のアルゴリズムと我々の提案する DF アルゴリズムとで各々グラフの各頂点を始点とする単一始点最短路探索を行ない，それに要する二次記憶への参照回数の平均を比較した．

バッファ領域としては個々のグラフの表現に要するデータの総量の $B\%$ に相当する大きさの主記憶を確保するものとし，ページの置換は LRU による必要時ページン

表 2.3: グラフのパラメータ

パラメータ	表記	意味
頂点数	N	$N = \|V_G\|$
ドメイン内頂点数	D	$D = \frac{N}{\|V_G/\sim\|}$
ドメイン内平均次数	ρ_i	$\rho_i = \frac{\|\{(u,v)\|(u,v) \in E_G \wedge u \sim v\}\|}{N}$
ドメイン間平均次数	ρ_e	$\rho_e = \frac{\|\{(u,v)\|(u,v) \in E_G \wedge u \not\sim v\}\|}{N}$
重み分布		
ドメイン内辺	p_i	$\frac{\|\{(u,v)\|(u,v) \in E_G \wedge u \sim v \wedge W_G(u,v) \leq x\}\|}{\|\{(u,v)\|(u,v) \in E_G \wedge u \sim v\}\|} = \int_0^\infty \frac{x^{1-p_i}e^x}{\Gamma(p_i)}$
ドメイン間辺	p_e	$\frac{\|\{(u,v)\|(u,v) \in E_G \wedge u \not\sim v \wedge W_G(u,v) \leq x\}\|}{\|\{(u,v)\|(u,v) \in E_G \wedge u \not\sim v\}\|} = \int_0^\infty \frac{x^{1-p_e}e^x}{\Gamma(p_e)}$
バッファサイズ	B	$B = \frac{\text{バッファとして確保可能である主記憶量}}{\text{グラフの二次記憶上における総データ量}} \times 100\%$

グ方式を用いるものとした.

表 2.4に示すパラメータの値の組に対して行なった実験の結果を図 2.11–2.16に示す. 各図において, 縦軸の値は二次記憶への参照回数を表す. また, 図中の縦棒は 95% の信頼区間を示す. 各図において Dijkstra のアルゴリズムによる参照回数の信頼区間と, DF アルゴリズムによる参照回数の信頼区間が十分分離可能である事から, 本実験で得られたデータは, Dijkstra のアルゴリズムと, DF アルゴリズムの性能に関する議論に十分な精度を保持しているといえる.

図 2.11 に, グラフの頂点数が二次記憶への参照回数に及ぼす影響を示す.

図 2.11(a),2.11(b) の横軸は, グラフの頂点数を表す. 今回実験を行なった範囲の任意の規模のグラフに対し, DF アルゴリズムは Dijkstra のアルゴリズムと比較して少ない参照数を示し, その差は頂点数の多いグラフにおいてより顕著である事か

表 2.4: 実験で用いたパラメータの値

図	N	D	ρ_i	ρ_e	$\frac{p_e}{p_i}$	B
2.11(a)	[2000,10000]	100	4	5	1.0	1.0
2.11(b)						10.0
2.12(a)	[2000,10000]	100	4	4	1.0	1.0
2.12(b)						10.0
2.13	10000	100	4	5	1.0	[5.0,100.0]
2.14(a)	10000	100	4	5	[0.0,2.0]	1.0
2.14(b)						10.0
2.15(a)	400	20	19	380	[0.0,5.0]	5.0
2.15(b)						50.0
2.16	10000	100	4	5	1.0	10.0

ら，更に大規模であるグラフに対しても，本実験と同様の傾向が存在するものと考えられる．

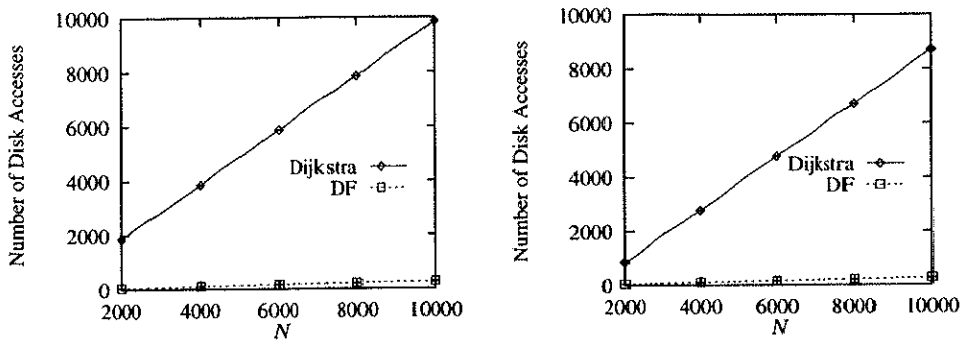
図 2.11と同様の実験を，辺の重みが標準偏差 1.0 の Γ 分布に従う 4 次正則トロイダルグラフに対して行なった結果を，図2.12に示す．

図 2.12(a),2.12(b) の横軸は，グラフの頂点数を表す．実験より，トロイダルグラフに対しても，ハミルトングラフに対する場合と同様の結果を得た．

地理情報システムなど，大規模グラフ処理を必要とするいくつかの応用にとって，平面グラフやトロイダルグラフの効率的処理は特に重要な課題である．

したがって，DF アルゴリズムは，これらの応用に対しても有効であると考えられる．

主記憶上に確保されたバッファ領域の大きさが二次記憶への参照回数に及ぼす影響を図 2.13 に示す．



(a) $B = 1\%$ (b) $B = 10\%$

図 2.11: グラフの頂点数の影響

図 2.13 の横軸の値は，B すなわち，確保可能であるバッファ領域の総量を，グラフの隣接リスト表現全体が二次記憶上において占有する総記憶容量に対する百分率で表す．

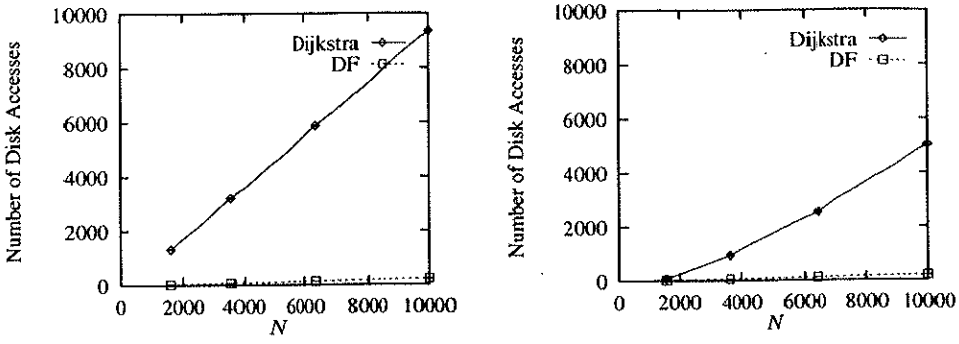
この結果より，確保可能であるバッファ領域の総量の大小に関係なく，DF アルゴリズムが優位であることがわかる．Dijkstra のアルゴリズムでは，二次記憶参照回数はバッファ領域の総量にほぼ線形に影響を受けるのに対し，DF アルゴリズムでは，バッファの総量の大小に関係なく，安定して高い性能を示している．

ドメイン間辺の重みの分布とドメイン内辺の重みの分布の関係が二次記憶への参照回数に及ぼす影響を図 2.14 に示す．

図 2.14(a),2.14(b) の横軸の値は，ドメイン間辺の重みが従う Γ 分布の標準偏差 p_e とドメイン内辺の重みが従う Γ 分布の標準偏差 p_i の比を表す．

これらより，ドメイン間辺に付随する重みが，ドメイン内辺の重みと比較して非常に小さい場合には，DF アルゴリズムの性能の低下が見られることがわかる．ただし，その場合においても，DF アルゴリズムは Dijkstra のアルゴリズムと比較して，十分小さな二次記憶参照回数を示している．

密であるグラフに関して，ドメイン間辺の重みの分布とドメイン内辺の重みの分



(a) $B = 1\%$ (b) $B = 10\%$

図 2.12: グラフの頂点数の影響（4 次正則トロイダルグラフ）

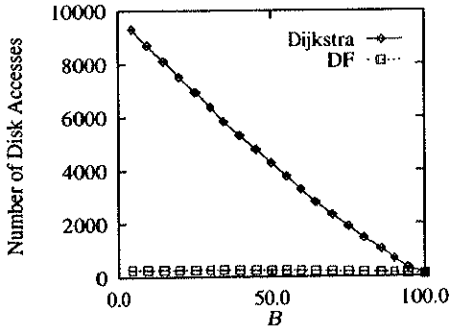


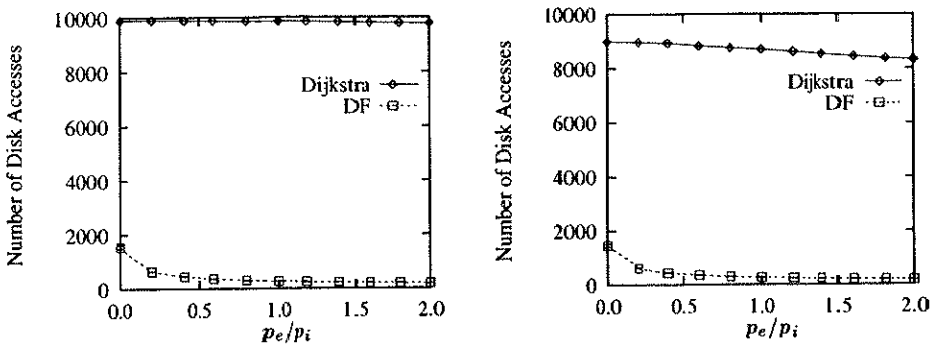
図 2.13: バッファ領域の大きさの影響

布の関係が二次記憶への参照回数に及ぼす影響を図 2.15 に示す．これにより，密であるグラフに対しても，疎グラフに対しての結果と同様の傾向を見る事ができる．

ドメイン間辺の数が二次記憶への参照回数に及ぼす影響を図 2.16 に示す．

図 2.16 の横軸の値は，ドメイン間平均次数 ρ_e を表す．

この結果より，ドメイン間平均次数が Dijkstra のアルゴリズムおよび DF アルゴリズムによる二次記憶参照回数に及ぼす影響は大きくない事がわかる．図 2.16はバッファの総量がグラフの総データ量の 10% である場合の結果を示しているが，他のバッ



(a) $B = 1\%$ (b) $B = 10\%$

図 2.14: 重みの分布の影響（疎グラフ）

ファ量の場合にも，同様の結果を得た．

ドメインに分割された辺の重みと次数の影響が比較的小さいことから，グラフのドメイン分割の方法に関係なく，DF が有効であると推測される．

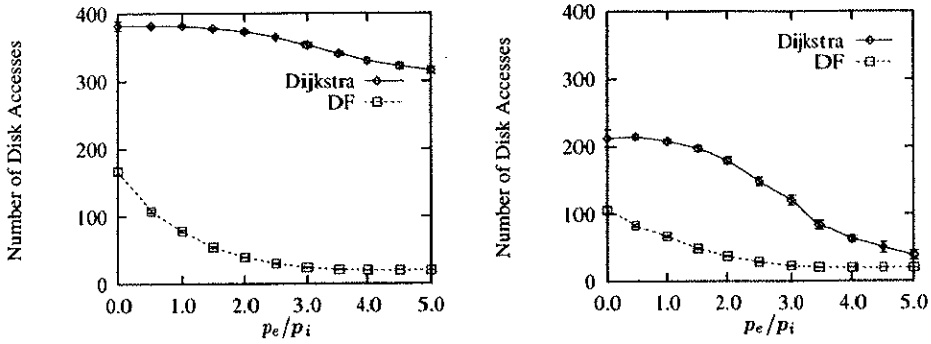
2.3.2 二頂点間最短路探索

実験には，辺の重みが標準偏差 1.0 の Γ 分布に従う，頂点数 $N = 10^4$ の，4 次正則トロイダルグラフを用いた．

比較は，Dijkstra のアルゴリズムと DF アルゴリズム，および各々と Domain Encoding を利用した探索空間の縮小法とを組み合わせた手法との間で行った．

同一条件下で生成した 128 種類のグラフに対し，最短路探索の始点と終点を無作為に定め，二点間の最短路長と，アルゴリズムの実行過程における二次記憶への参照数との関係を評価した．

図 2.17では，横軸に始点と終点との間の最短路長を，縦軸にアルゴリズム実行過程における，隣接リスト，Domain Encoding 法で用いられる部分的結果の参照に要する二次記憶への参照数を示す．



(a) $B = 5\%$ (b) $B = 50\%$

図 2.15: 重みの分布の影響（密グラフ）

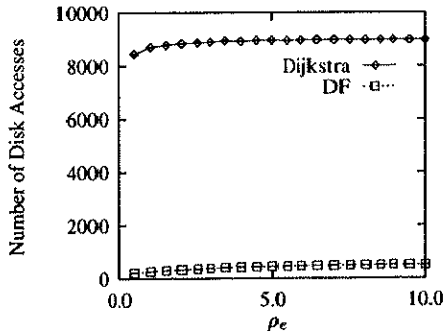
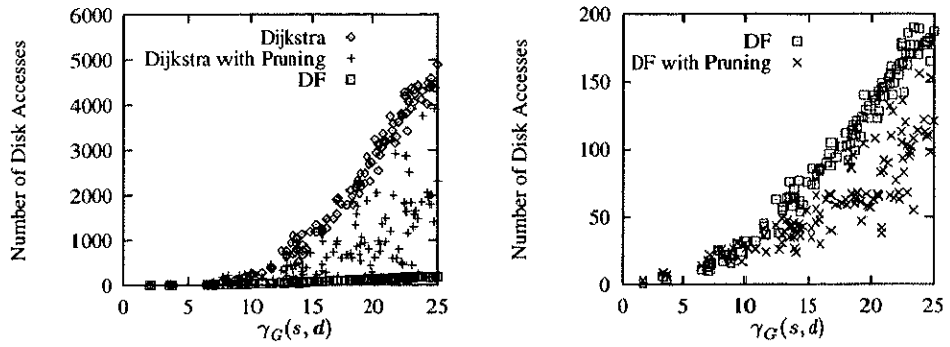


図 2.16: ドメイン間辺の次数の影響

この実験では，バッファ領域としては，個々のグラフの隣接リスト表現に要するデータの総量の 10% に相当する大きさの主記憶を確保するものとし，ページの置換には LRU による必要時ページング方式を用いるものとした．

実験の結果より， DF アルゴリズムは従来二頂点間最短路探索に用いられていた Dijkstra のアルゴリズムや， Dijkstra のアルゴリズムと探索空間の縮小法を組み合わせた手法に対して，より小さい二次記憶参照数で最短路探索問題を解決可能であることがわかる．



(a) DF アルゴリズムによる性能向上　　(b) 探索空間縮小法との組み合わせの影響

図 2.17: 始点・終点間の最短路長の影響

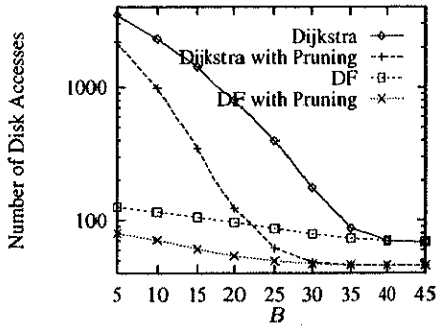


図 2.18: バッファ領域の大きさの影響

また，DF アルゴリズムと本研究で提案する探索空間縮小法とを組み合わせることにより，二次記憶参照数を更に削減可能である．

同様のグラフに対し，最短路長がグラフ全体の辺の重みの標準偏差の 18.0 倍から 20.0 倍となるような二頂点間に対して，確保可能であるバッファ領域の大きさと，二次記憶参照数の関係を評価した．

図 2.18では，横軸に確保可能であるバッファ領域の大きさを，縦軸にアルゴリズ

ム実行過程における，隣接リスト， Domain Encoding 法で用いられる部分的結果の参照に要する二次記憶への参照数を示す．

実験の結果より，単一始点最短路探索の場合と同様に，グラフの規模と比較して相対的に確保可能であるバッファ領域の大きさが小さい場合に， DF アルゴリズムは特に有効であることがわかる．

多くの応用では，プログラム変数など，アルゴリズムの作業領域を主記憶上に確保できない場合が存在する．グラフアルゴリズムの作業領域を二次記憶上で効率的に管理するための手法としては，効率的外部ソート手法 [AV88] の応用として，いくつかの手法が提案されている [Arg95, CGG+95, KS96] .

本研究では，各アルゴリズムの作業領域を二次記憶上で管理し，それらに対する参照，更新を含めた二次記憶への参照数の評価を行なった．作業領域の二次記憶上での管理手法としては，考慮の対象となる各変数が関連する頂点の属するドメイン毎にクラスタ化する単純な方法を用いた．

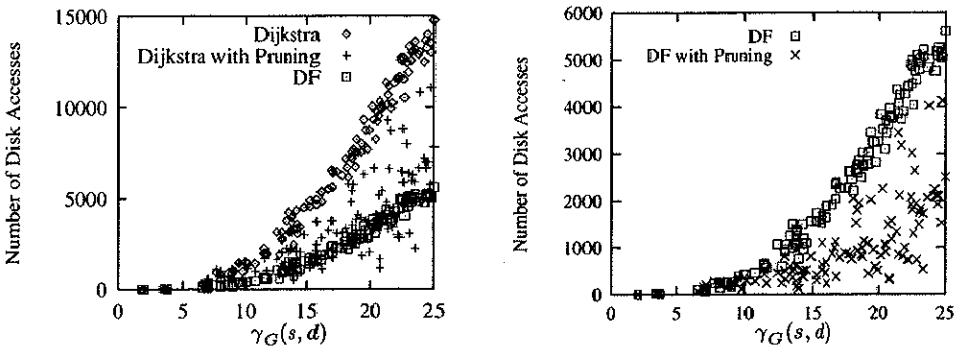
実験では， Dijkstra のアルゴリズムと DF アルゴリズム，および各々と探索空間の縮小法を組み合わせた手法に対し，各頂点 v について $\mathrm{cost}[v]$ の値と， $v \in \mathrm{Explored}$ であるか否かを示すフラグ，さらに， DF アルゴリズムを用いる手法については， $v \in \mathrm{Valid}$ であるか否かを示すフラグを二次記憶上で管理した．

図 2.19では，横軸に始点と終点との間の最短路長を，縦軸にアルゴリズム実行過程における，隣接リスト， Domain Encoding 法で用いられる部分的結果，およびアルゴリズムの作業領域の参照に要する二次記憶への参照数を示す．

また，図 2.20では，横軸に確保可能であるバッファ領域の大きさを，縦軸にアルゴリズム実行過程における，隣接リスト， Domain Encoding 法で用いられる部分的結果およびアルゴリズムの作業領域の参照に要する二次記憶への参照数を示す．

実験の結果より，多くの場合において， DF アルゴリズムを単独で用いた場合であっても， Dijkstra のアルゴリズムと探索空間縮小法を組み合わせた場合と同程度か，より小さい二次記憶参照数で最短路探索問題を解決可能であることがわかる．

Domain Encoding による探索空間の縮小法の利用のためには，部分的結果の導出



(a) DF アルゴリズムによる性能向上

(b) 探索空間縮小法との組み合わせの影響

図 2.19: 始点・終点間の最短路長の影響（作業領域への参照を含む）

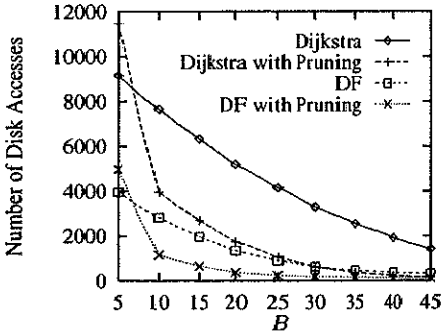


図 2.20: バッファ領域の大きさの影響（作業領域への参照を含む）

のために，あらかじめ $O(|V_G/\sim| \times |V_G|^2)$ の計算を必要とし，またその管理のために，$O(|V_G/\sim|^2)$ に相当する記憶領域を必要とする．これに対し，DF アルゴリズムの利用のためには，Dijkstra のアルゴリズムの場合と比較して，集合 Valid の管理のための，$|V_G|$bit の記憶領域を用意するだけで十分である．したがって，DF アルゴリズムは二頂点間最短路探索手法として有効であると考えられる．

これに加え，Domain Encoding の部分的結果の管理に必要とされる記憶領域の量

や，あらかじめ計算するための計算量が問題とならない場合には，DF アルゴリズムと探索空間の縮小法の組合せによって，さらに少ない二次記憶参照数で最短路探索問題を解決可能であることがわかる．

さらに，作業領域の管理手法として文献 [Arg95, CGG+95, KS96] に示される手法などを用いることで，各アルゴリズムによる二次記憶参照数は図 2.17，図 2.18の結果に近付くと考えられ，その際には DF の有効性はさらに増大すると考えられる．

第 3 章

大規模グラフ処理システムのためのページ置換手法

3.1 大規模データ処理とデータ参照パターン

計算機システムの応用分野には各応用分野に固有の問題や，それらの問題を解決する手段としての処理が存在する．大規模データの管理と利用を必要とするような応用分野においては，個々の問題を解決するための処理に対しても大規模データ全体への参照が必要となる場合が多い．大規模データ全体への参照を一度に行なうことは現実的には困難であるため，多くのシステムではデータを適切な単位に分割し，そのデータ単位 (granule) への参照を順次行なうことで応用上の問題を解決する．

特にグラフ上の問題の繰り返しに帰着するような応用上の問題の解決においては，帰着する個々のグラフ上の問題の解決にはグラフ全体の構造に関する情報を必要としない場合であっても，解決するグラフ上の問題の繰り返しの順番がグラフ全体の構造に大きく依存する場合があると考えられる．多くの場合，応用上の問題はグラフの一部の頂点や辺に付随する情報を参照するグラフ上の問題の繰り返しに帰着し，ある特定の頂点や辺に付随する情報を必要とする問題の次に解決すべき問題には，その頂点や辺に隣接，接続する頂点ないし辺の情報を必要とする．

また，応用上の問題がグラフ上の問題などの抽象度の高い問題に単純には帰着しない場合であっても，データ単位の参照の順番にある一定の規則性を見出すことが可能である場合がある．

例として，ハイパーテキスト文書の閲覧を考える．

ある時点において，利用者によって文書 p への閲覧が行なわれているとする．この時点の次の時点において，p から参照される文書あるいは p への参照を持つ文書への閲覧が発生する可能性は，他の文書への閲覧が発生する可能性よりも高いと考えられる．また，p と類似した内容を示す文書への閲覧が発生する可能性は，他の文書への閲覧が発生する可能性よりも高いと考えられる．

このように，データへの参照のパターンがデータ単位間に存在する「結合の強さ」に依存して決定する場合その参照パターンはマルコフ連鎖により生成されるパターンと同様の特徴を示すものと考えられる．このような参照パターンは，従来のデータベース分野において大規模データの利用の際に発生する典型的なデータ参照のパターンと捉えられてきたパターン [CD85, OOW93] とは明らかに異なる性質を持っている．

データ参照列を生成するマルコフ連鎖の推移確率行列は，データ単位を頂点とし，データ単位間の結合の強さを辺の重みとするグラフの隣接行列を変形した行列によって与えられる．したがって，応用分野上の問題が直接グラフ上の問題に帰着しないような場合でも，応用分野におけるデータをグラフデータと見做すことにより，その参照に関する特性を予測可能であるという利点が生じる．

データ参照のパターンが予測可能である場合，適切なバッファ管理方法を用いることにより，処理性能の向上が実現可能である．システムの性能を単位時間あたりに実行可能であるような処理の数 (throughput) と考えると，一般に他のバッファ管理方式と比較して性能の高いページ置換バッファ管理方式が存在することが知られており [ADU71] ，現在多くのデータベース管理システムなどでページ置換によるバッファ管理機構が用いられている．そこで，本研究ではページ置換によるバッファ管理のみについて議論する．バッファ管理機構はあらかじめ確保した主記憶領域を

グラフデータへの参照を行なう各トランザクションに対し割り当て，トランザクションからは各トランザクションの参照列の特性を反映したパラメータを受け取り，それを利用してバッファの管理を行なう．本研究では議論の単純のため，各トランザクションへのバッファ領域の割り当ては静的に行なわれるものと仮定する．

同一のページ参照列に対して，異なるページ置換方式を用いてページ置換を行なった場合，実際に二次記憶への参照を実行する回数は異なる．したがって，適切なページ置換方式を選択することにより二次記憶への参照の回数を削減可能であり，これにより，処理全体の性能向上を達成できる．ページ参照要求の回数に対し，要求されたページがバッファ中に存在して二次記憶への参照なしにページデータを取得可能であった回数の割合を的中率と呼ぶ．的中率は上記のような観点で考えた性能の指標となる．

ページ置換によるバッファ管理方式に関しては，

> バッファ中のページのうち，次に参照されるまでの時間が最も長くなるようなページを置換の対象として選択する方法が最も効率的である．

という性質 [ADU71] が存在する．

従来，データベースシステムなどの領域においては，ページ置換方式としてLRUページ置換 [EH84] が多く用いられてきた．LRU ページ置換方式は，「あるページが次に参照されるまでに経過する時間は，そのページが最後に参照されてから経過した時間とほぼ等しい」という仮定に基づき，バッファ中のページのうち最後に参照されてからの時間が最も長いページを置換の対象として選択する方式である．ところが，LRU ページ置換方式が置換の対象となるページの選択のための指針として利用する情報は，バッファ中の各ページが最後に参照された時刻に限られている．そのため，LRU ページ置換ではあるページに対する参照が，頻繁に起こる参照の一部として発生した参照であるのか，それとも例外的に発生した参照であるのかを区別することができない．

このような問題に対処するため，LRU-K [OOW93] などのいくつかのページ置換

アルゴリズムが提案されている．しかし，これらの，過去のページ参照の履歴のみを用いて将来のページ参照を予測し，置換の対象となるページを選択する手法では，特に推移的状態を持つようなマルコフ連鎖によって構成される参照列を適切に処理することは困難であると考えられる．

マルコフ連鎖において，ある状態から他の状態への推移に必要とする遷移の回数は，平均最小到達時間 (mean first entrance time) により与えられる．平均最小到達時間は，「次にページが参照されるまでの時間」を反映していると考えられるため，平均最小到達時間を調べることにより，最適なページ置換方式を導出可能である．

しかし，一般にはグラフの隣接行列から推移確率行列を生成することは困難であり，推移確率行列が与えられている場合であっても平均最小到達時間を計算することは容易ではない．そこで，本研究では平均最小到達時間の大まかな近似として，データ単位間の非類似度 (dissimilarity) を利用することを考える．本研究では，二つのページに格納されるデータ単位の非類似度をそれらのページ間の距離と呼ぶ．

多くの応用分野ではデータ単位間の非類似度を考えることが可能であり，データ単位間の「結合の強さ」を反映していると考えられる．

そこで，マルコフ連鎖によって表現可能であるような参照パターンに対するページ置換方式としてページに格納されるデータ単位間の非類似度を用いる方式が有効であると考える．ただし，データ単位間の非類似度が「結合の強さ」を反映していない場合の存在も想定する必要があるため，バッファ中で最後に参照したページとの距離が最大となるようなページを置換の対象として単純に選択するような方法は必ずしも適切ではない．

極端な例として，データ間の非類似度が参照パターンを生成するマルコフ連鎖より得られる平均最小到達時間に対し負の相関を持つ場合を考える．このとき，バッファ中のページのうち最後に参照したページとの間で最大の距離を持つページは，実際には最後に参照したページ自身に次いで高い確率で次回参照すべきページとして選択されるものと考えられる．

このような場合にも極度な的中率の低下を防ぐために，本研究では置換の対象となるページの選択に，距離とともに従来の LRU 置換方式と同様の基準を組合せた手法を提案する．

3.2 ページ置換方式 KNC-D

本研究では，ページ間の距離の概念を利用する必要時ページ置換方式 KNC-D [CNO94, Not95] を提案する．KNC-D の基本的な考え方は，マルコフ連鎖に従う参照列に対する

- 最後に参照したページとの間の距離が，ある閾値 D より大きいページは，それ以降続けて参照される可能性が低い
- そのようなページが複数存在する場合は，それらのうちで，最も長期に及んで参照が行なわれていないものが，以降続けて参照される可能性がより低い

という観察に基づいている．

以上の観察に基づき，KNC-D 置換方式においては，最後に参照されたページが x であるとき，以下の規則に従って置換の対象となるページをバッファ B 中から選択する．ページ間の距離を与える関数 $d()$ と閾値 D はトランザクション毎に適宜与えられるものとする．

- $\{p|p \in B \wedge d(x,p) > D\} \neq \emptyset$ の場合 $\{p|p \in B \wedge d(x,p) > D\}$ 中で Least Recently Used であるページを選択
- $\{p|p \in B \wedge d(x,p) > D\} = \emptyset$ の場合 $\{p|p \in B \wedge d(x,p) \leq D\}$ 中で Least Recently Used であるページを選択

従って，$D = 0$ の場合，KNC-D 置換方式は LRU 置換方式と同一のページを置換対象として選択する．KNC-D置換方式においては，集合 $\{p|p \in B \wedge d(x,p) > D\}$

に属するページを Far Cell と呼び，集合 $\{p|p \in B \wedge d(x,p) \leq D\}$ に属するページを Near Cell と呼ぶ．

3.3 KNC-D ページ置換アルゴリズム

定義に従い KNC-D ページ置換の対象となるページを選択するには，バッファに存在しないページに対する参照が発生する度にバッファ中のすべてのページに対して距離の計算を行ない，各 Far Cell の最終参照時刻を得る必要があるため，現実的ではない．

従来のデータベース管理システムなどにおいては，LRU ページ置換に必要となるページ参照時刻情報の管理のために LRU リストが利用されている．バッファ領域が LRU リストによって管理されている場合，図 3.1に示すアルゴリズムによって，効率的に置換の対象となるページを選択可能である．

図 3.1において，D は KNC-D 置換方式に与えられる閾値を，x は参照の対象となるページを表す．また，線形リスト L は LRU リストを表し，$L.lru$ によってリスト中で Least Recently Used であるような要素を指すポインタが得られるものとする．

LRU リストの利用により，置換対象となるページの選択時にバッファ中の各ページの最終参照時刻の取得が不要となる．さらに，適切な閾値 D を選択することにより，バッファ中の一部のページのみに対するページ間距離の計算で置換の対象となるページを選択可能となる．

3.4 評価

KNC-D ページ置換方式が大規模グラフ処理システムにおけるバッファ管理手法として有効であることを確認するために実験を行なった．

実験は，辺の重みが標準偏差 1.0 の Γ 分布に従う頂点数 $N = 10^6$ の 4 次正則トロ

```
1  Procedure KNC-D(D, x, var L)
2  begin
3    if x ∉ L then
4    begin
5      p := L.lru;
6      while p↑.next ≠ nil and d(x, p↑.key) ≤ D do
7        p := p↑.next;
8      if d(x, p↑.key) > D then
9        delete(p, L)
10   end;
11   LRU(x, L)
12 end
```

図 3.1: KNC-D ページ置換アルゴリズム

イダルグラフ G に対して，辺の重みの逆数に比例する確率で隣接頂点を順次訪問する参照列を生成し，LRU ページ置換および KNC-D ページ置換によるページ置換の適中率を比較した．

すなわち，グラフ G の隣接行列 $\mathbf{P}_G = \{p_{ij}\}$ に対して，推移確率行列 $\mathbf{Q}_G = \{q_{ij}\}$ を持つマルコフ連鎖により生成される列を参照列とした．ただし，

$$q_{ij} = (p_{ij} \sum_{k=1}^{N} \frac{1}{p_{ik}})^{-1}$$

とする．

また，ドメイン間の距離としては，最短路探索問題などの探索空間縮小に用いられる Domain Encoding 法によってあらかじめ計算されたドメイン代表点間距離の値を利用した．

KNC-D 置換方式において，閾値 D の値は辺の重みの標準偏差を単位とし，その 1 倍から 5 倍までの値を用いた．

図 2.13 の横軸の値は，バッファ領域として確保可能である主記憶量をページのサイズを単位として表し，縦軸の値は，全ての参照要求のうち，要求の対象となったページがバッファ中に存在した率を示す．

図中，“optimal” で示される曲線は最適ページ置換が行なわれた場合の適中率を示す．

実験の結果より，KNC-D ページ置換による適中率が LRU ページ置換による適中率と比較して高く，大規模グラフ処理に対するページ置換方式として有効であることが確認された．

また，KNC-D ページ置換において，閾値 D の適切な値は，確保可能であるバッファ領域の大きさに依存して変化することが確認された．

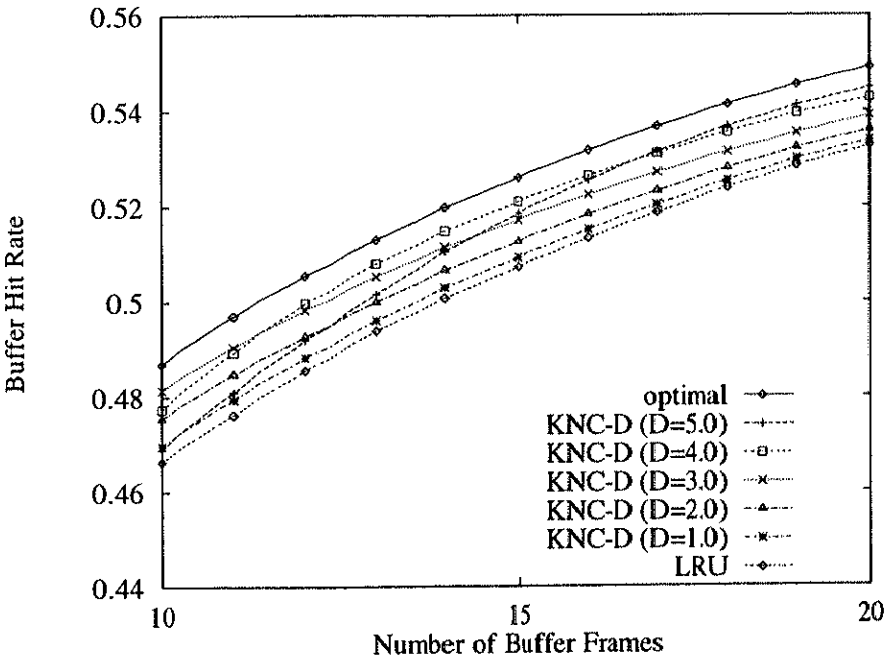


図 3.2: グラフ処理時ページ置換適中率

第 4 章

まとめ

本研究では，二次記憶上の大規模グラフに対する最短路探索に有効である，最短路探索アルゴリズム DF と， DF を二頂点間最短路探索問題に対して適用する際に利用可能である，探索空間縮小法の提案を行ない，各々に対して，実験による評価を行なった．

また，大規模グラフ処理システムに適したページ置換方式である KNC-D を提案し，実験による評価を行なった．

これらにより，二次記憶上の大規模グラフに対して効率的にグラフ上の問題を解決するアルゴリズムの提案と，グラフに抽象化されるデータに対する処理に必要となるデータ参照を効率的に行なうための機構の提案との二種類のアプローチによる，二次記憶上の大規模グラフ処理の効率化が実現された．

グラフ上問題を解決する個々のアルゴリズムの二次記憶参照数に関する性能向上については，今後の研究課題として以下の二点が考えられる．

第一に，最短路探索問題以外のグラフ上の問題に関して，本研究で提案した DF アルゴリズムと同様のアプローチに基づく手法が適用可能であるかについて検討する必要がある．

第二に，グラフ上の各種問題を効率的に解決するための具体的なアルゴリズムの提案と，その実装について議論を進める必要がある．実装に際しては，発見的手法

や近似解法などを含め，従来の研究によって提案されてきた実装上の工夫とも連携し，応用分野固有の特徴に関する情報が利用可能である場合にはそれらを活用し，それらが利用可能でない場合には，汎用性の高い手法によって効率的に問題を解決可能であるような実現手法が望ましい．

二次記憶上の大規模グラフ処理に対応するバッファ管理手法の性能向上については，今後の研究課題として以下の二点が考えられる．

第一に典型的な問題の定式化を行ない，それによってデータ参照パターンのモデル化を行なう必要がある．本研究では，データ参照パターンのモデルとして単純なマルコフ連鎖モデルを用いたが，実際の応用における処理においては，より複雑な参照パターンが生成されるものと考えられる．

第二に，本研究の結果，バッファ管理手法の改善による処理性能の向上は，アルゴリズムの改善による処理性能向上と比較して，大きくは見込めないことが判明した．今後は，このような処理の性能向上のために，二次記憶上でのグラフの表現方法など，他のアプローチからの考察が必要と考えられる．

本研究全体に対する今後の課題として，本研究では二次記憶上のデータ参照がグラフ処理全体のボトルネックであるとし，二次記憶への参照数の削減を目標に，各手法の提案と評価を行なったが，実際のシステムにおいては，他のコスト要因も全体の性能に非常に大きく影響すると考えられる．そこで，それらのコスト要因と二次記憶への参照によって発生するコストとのトレードオフについて議論を進める必要がある．具体的には，本研究で提案した手法を提供するソフトウェア部品の実装と，実際の応用分野における典型的データの特性を反映したワークベンチ，およびそれらを利用する応用プログラム，さらに，応用分野におけるシステムの利用形態に基づいたベンチマークなどを用い，種々の処理に必要とされる時間や資源の量などを含む総合的な効率に関する評価が必要である．

謝辞

本研究を進めるにあたり，主任指導教官として終始御指導頂きました，筑波大学電子・情報工学系 大保信夫教授に深く感謝致します．

加えて，日頃から研究についての重要な示唆と叱咤激励を頂き，さらに貴重な実験機材などを快く使わせて下さいました，筑波大学名誉教授 藤原譲先生，筑波大学名誉教授 鈴木功先生，筑波大学電子・情報工学系教授 北川博之先生，ならびに筑波大学電子・情報工学系講師 宇都宮公訓先生に御礼申し上げます．

本研究の基本的な着想は，NTT データクリエイション株式会社 鈴木尚志氏，ならびにつくば国際大学講師 陳漢雄先生との討論から与えて頂きました．さらに，筑波大学工学研究科 石川雅弘氏との議論は，本研究にとって非常に有意義なものとなりました．

また，茨城大学助手 古瀬一隆先生，神奈川工科大学助手 鈴木孝幸先生，日本学術振興会特別研究員 森嶋厚行氏，ならびに筑波大学電子・情報工学系データベース研究室の皆様には，日頃の研究生活を含め，様々な面で相談に応じて頂きました．

最後に，私の研究活動を支持し，経済面を含む全てにおいて支援して頂いた両親に心から感謝します．

付録 A

補題の証明

本節では 2.1.5節の定理の証明に用いた補題 2の証明を行なう.

以下の補題 3と補題 4を用いて補題 5を示し，それらにより補題 2を証明する.

以下の補題 3は，Dijkstra のアルゴリズムの正当性 [CLR90] を検証するために用いられる補題と同一である.

補題 3 (文献 [CLR90]) $P \in \mathcal{S}_G(s,y)$ とし，$(x,y) \in P$ とする．グラフ G に対して任意の頂点 $s \in V_G$ を始点とする手続き InitializeGraph(V_G, s) を実行した後，任意の頂点列 $x_1x_2 \ldots x_n, y_1y_2 \ldots y_n$ に対する手続き Relax'(x_i, y_i, W_G) の実行列を適用したとする．ただし，ある $1 \leq i \leq n$ が存在し，$(x_i, y_i) = (x, y)$ であるとする．このとき

$$\mathrm{cost}[x] = \gamma_G(s,x) \supset \mathrm{cost}[y] = \gamma_G(s,y)$$

が成立する.

補題 4 グラフ G に対して任意の頂点 $s \in V_G$ を始点とする手続き InitializeGraph(V_G, s) を実行し，集合 Valid を空集合に初期化した後，任意の頂点列 $v_1v_2 \ldots v_n$ に対する手続き Visit(v_i, E_G, W_G) の実行列を適用したとする．このとき，任意の $x, y \in V_G$ に

対し

$$x \in \mathrm{Valid} \wedge (x,y) \in E_G \supset \mathrm{cost}[y] \leq \mathrm{cost}[x] + W_G(x,y) \tag{A.1}$$

が成立する．

［証明］頂点列の長さ n についての帰納法で証明する．$n = 0$ のとき，すなわち初期状態においては

$$\forall x \in V_G . x \notin \mathrm{Valid}$$

より (A.1) は成立する．

$n = k$ のとき (A.1) が成り立つと仮定し，$n = k+1$ の場合に (A.1) が成り立つ事を示す．

手続き $\mathrm{Visit}(v_{k+1}, E_G, W_G)$ の実行の直前において $x \in \mathrm{Valid}$ なる任意の $x \in V_G$ について考える．ここで，

$$(v_{k+1}, x) \in E_G \wedge \mathrm{cost}[v_{k+1}] + W_G(v_{k+1}, x) < \mathrm{cost}[x]$$

ならば，手続き $\mathrm{Visit}(v_{k+1}, E_G, W_G)$ の実行によって

$$x \notin \mathrm{Valid}$$

となる．また，そうでなければ仮定より，手続き $\mathrm{Visit}(v_{k+1}, E_G, W_G)$ の実行の直前において

$$\forall y \in V_G . (x,y) \in E_G \supset \mathrm{cost}[y] \leq \mathrm{cost}[x] + W_G(x,y)$$

が成立する．手続き $\mathrm{Visit}(v_{k+1}, E_G, W_G)$ の実行によって，任意の $y \in V_G$ について $\mathrm{cost}[y]$ の値は増大せず，また，この手続きによって $\mathrm{cost}[x]$ の値は変化しないため，手続き $\mathrm{Visit}(v_{k+1}, E_G, W_G)$ の実行直後において

$$\forall y \in V_G . (x,y) \in E_G \supset \mathrm{cost}[y] \leq \mathrm{cost}[x] + W_G(x,y)$$

が成立する.

一方, 手続き Visit(v_{k+1}, E_G, W_G) の実行の直前において $x \notin$ Valid なる任意の $x \in V_G$ について考えると, $x = v_{k+1}$ ならば, 手続き Visit(v_{k+1}, E_G, W_G) の実行によって

$$\begin{aligned} & x \in \text{Valid} \\ \wedge \forall y \in V_G.(x, y) \in E_G & \\ & \supset \text{cost}[y] \leq \text{cost}[x] + W_G(x, y) \end{aligned}$$

となる. また, $x \neq v_{k+1}$ ならば手続き Visit(v_{k+1}, E_G, W_G) の実行直後においても

$$x \notin \text{Valid}$$

が成立する.

以上より, $n = k + 1$ の場合にも (A.1) が成立する.

Q.E.D.

補題 5 グラフ G に対し任意の頂点 $s \in V_G$ を始点とする *DF* アルゴリズムによる最短路探索 DF$(G, \mathcal{D}, s)$ を適用したとき, 任意の $v, u_i \in V_G$ に対し

$$\begin{aligned} & \Delta(i, u_i) \wedge \exists P \in \mathcal{S}_G(s, v).(u_i, v) \in P \\ & \supset \Delta(i + 1, v) \end{aligned} \tag{A.2}$$

が成立する.

[証明] 式 (A.2) の左辺を仮定する.

T_i の時点において $u_i \in$ Valid ならば, 補題 4 より $\Delta(i, v)$ であるので, 補題 1 より $\Delta(i + 1, v)$ は成立.

T_i の時点において $u_i \notin$ Valid ならば, T_i と T_{i+1} の間において Relax'(u_i, v, W_G) が必ず実行されるので, 補題 1 と補題 3 より $\Delta(i + 1, v)$ は成立.

以上より (A.2) の成立が示された.

Q.E.D.

以上を用いて，補題 2の証明を行なう．

補題 2 (再掲) グラフ G に対し任意の頂点 $s \in V_G$ を始点とする DF アルゴリズムによる最短路探索 $\mathrm{DF}(G, \mathcal{D}, s)$ を適用したとき，$i \leq j$ なる任意の $i, j \in \{0, \ldots, N-1\}$ に対し，T_i の時点において

$$\begin{aligned} \forall k \in \{0, \ldots, i-1\}. \Delta(k, u_k) \\ \supset \mathrm{cost}[u_i] \leq \gamma_G(s, u_j) \end{aligned} \tag{2.2}$$

が成立する．

[証明] $i = 0$ または u_j が s から到達可能でない場合は明らか．

$i > 0$ かつ u_j が s から到達可能である場合，$s = u_0$ から u_j への最短路の一つ $P \in \mathcal{S}_G(s, u_j)$ に対し

$$0 \leq m \leq i-1 \tag{A.3}$$

$$i \leq n \leq N-1 \tag{A.4}$$

なる辺 $(u_m, u_n) \in P$ が必ず存在する．

u_n は P 上の頂点であるので

$$\gamma_G(s, u_n) \leq \gamma_G(s, u_j) \tag{A.5}$$

が成立．また，T_i において

$$\mathrm{selectmin}(\mathrm{cost}, V_G - \mathrm{Explored}) = u_i$$

であるので，(A.4) より

$$\mathrm{cost}[u_i] \leq \mathrm{cost}[u_n] \tag{A.6}$$

である．

いま，(2.2) の左辺を仮定すると，(A.3) より

$$\Delta(m, u_m) \wedge \exists Q \in \mathcal{S}_G(s, u_n).(u_m, u_n) \in Q$$

であるので，補題 5と補題 1より

$$\Delta(i, u_n) \tag{A.7}$$

が成立する．

以上の式 (A.5), (A.6), (A.7) より

$$\mathrm{cost}[u_i] \leq \gamma_G(s, u_j)$$

が示された．

Q.E.D.

参考文献

[ADU71] A. V. Aho, P. J. Denning, and J. D. Ullman. Principles of optimal page replacement. *Journal of the Association for Computing Machinery*, Vol. 18, No. 1, pp. 80–93, January 1971.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Vol. 22(2) of *SIGMOD Record*, pp. 207–216, Washington, D.C., June 1993. ACM.

[AJ90] R. Agrawal and H. V. Jagadish. Hybrid transitive closure algorithms. In *Proceedings of the 16th International Conference on Very Large Data Bases*, pp. 326–334, Brisbane, Queensland, Australia, August 1990. VLDB.

[AJ94] R. Agrawal and H. V. Jagadish. Algorithms for searching massive graphs. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 2, pp. 225–238, April 1994.

[Arg95] L. Arge. The buffer tree: A new technique for optimal I/O-algorithms. In *Proceedings of the 5th International Workshop on Algorithms and*

Data Structures, pp. 334–345, Kingston, Ontario, Canada, August 1995.

[AV88] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, Vol. 31, No. 9, pp. 1116–1127, 1988.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Vol. 26(2) of *SIGMOD Record*, pp. 265–276, Tucson, Arizona, June 1997. ACM.

[CD85] H. Chou and D. J. DeWitt. An evaluation of buffer management strategies for relational database systems. In *Proceedings of the 11th International Conference on Very Large Data Bases*, pp. 127–141, Stockholm, Sweden, August 1985. VLDB.

[CGG+95] Y. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter. External-memory graph algorithms. In *Proceedings of the ACM/SIGACT-SIAM Symposium on Discrete Algorithms*, pp. 139–149, 1995.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*, chapter 6, pp. 514–549. MIT Press, Cambridge, Massachusetts, 1990.

[CNO94] H. Chen, J. Notoya, and N. Ohbo. Keep nearer cells: A buffer allocation for gis. In *Proceedings of the 47th International Federation*

for Information and Documents, FID'94, pp. 43–48, Tokyo, Japan, October 1994.

[CYN+98] H. Chen, J. X. Yu, J. Notoya, Y. Liu, and N. Ohbo. A data mining model for query refinement revisited. In *Proceedings of the 1st International Symposium on Intelligent Data Engineering and Learning, IDEAL'98*, pp. 269–275, Shatin, N.T., Hong Kong, October 1998. Springer-Verlag.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, Vol. 1, pp. 269–271, 1959.

[EH84] W. Effelsberg and T. Haerder. Principles of database buffer management. *ACM Transactions on Database Systems*, Vol. 9, No. 4, pp. 560–595, December 1984.

[FHS96] U. M. Fayyad, D. Haussler, and P. Stolorz. Mining scientific data. *Communications of the ACM*, Vol. 39, No. 11, pp. 51–57, 1996.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. *Advances in Knowledge Discovery and Data Mining*, pp. 1–34, 1996.

[GK93] M. F. Goodchild and K. K. Kemp, editors. *Technical Issues in GIS*. NCGIA Core Curriculum. Univ. of California, July 1993.

[GKP85] F. Glover, D. D. Klingman, and N. V. Phillips. A new polynomially bounded shortest path algorithm. *Oper. Res.*, Vol. 33, pp. 65–73, 1985.

[JHR98] N. Jing, Y. Huang, and E. A. Rundensteiner. Hierarchical encoded path views for path query processing: An optimal model and its per-

formance evaluation. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 3, pp. 409–431, 1998.

[Jia92] B. Jiang. I/O-efficiency of shortest path algorithms: an analysis. In *Proceedings of the 8th International Conference on Data Engineering*, pp. 12–19, Tempe, Arizona, February 1992. IEEE Computer Society.

[KHI+86] R. Kung, E. Hanson, Y. Ioannidis, T. Sellis, L. Shapiro, and M. Stonebraker. Heuristic search in database systems. In *Proceedings of the 1st International Workshop on Expert Database Systems*, pp. 537–548. Benjamin Cummings Publications, 1986.

[KKT95] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the Association for Computing Machinery*, Vol. 42, No. 2, pp. 321–328, March 1995.

[KS96] V. Kumar and E. J. Schwabe. Improved algorithms and data structures for solving graph problems in external memory. In *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing*, New Orleans, Louisiana, October 1996. IEEE Computer Society.

[LCYO98a] Y. Liu, H. Chen, J. X. Yu, and N. Ohbo. A data mining approach for query refinement. In *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD-98*, No. 1394 in LNAI, pp. 394–396, Melbourne, Australia, April 1998. Springer-Verlag.

[LCYO98b] Y. Liu, H. Chen, J. X. Yu, and N. Ohbo. Using stem rules to refine document retrieval queries. In *Proceedings of the 3rd International*

conference on Flexible Query Answering Systems, FQAS'98, No. 1495 in LNAI, pp. 249–260, Roskilde, Denmark, 1998. Springer-Verlag.

[NGV96] M. H. Nodine, M. T. Goodrich, and J. S. Vitter. Blocking for external graph searching. *Algorithmica*, Vol. 16, pp. 181–214, 1996.

[NICO99] 能登谷淳一, 石川雅弘, 陳漢雄, 大保信夫. DF：二次記憶上の大規模グラフに対する最短路探索手法. 電子情報通信学会和文論文誌 D-I, 1999.

[Not95] 能登谷淳一. 地理情報システムのためのバッファ管理手法 KNC-D の研究. 修士論文, 筑波大学大学院工学研究科, 1995.

[OOW93] E. J. O'Neil, P. E. O'Neil, and G. Weilum. The LRU-K page replacement algorithm for database disk buffering. In *In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Vol. 22(2) of *SIGMOD Record*, pp. 197–306, Washington, D.C., June 1993. ACM.

[RHDM86] A. Rosenthal, S. Heiler, U. Dayal, and F. Manola. Traversal recursion: A practical approach to suppoting recursive applications. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, Vol. 15(2) of *SIGMOD Record*, pp. 166–176, Washington, D.C., June 1986. ACM.

[RW94] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, Vol. 27, No. 3, pp. 17–28, 1994.

[SKC93] S. Shekhar, A. Kohli, and M. Coyle. Path computation algorithms for advanced traveller information system (ATIS). In *Proceedings of the 9th International Conference on Data Engineering*, pp. 31–39, Vienna, Austria, April 1993. IEEE Computer Society.

[Smi78] A. J. Smith. Bibliography on paging and related topics. *ACM SIGOPS: Operating Systems Review*, Vol. 12, No. 4, pp. 39–56, October 1978.

[Suz97] 鈴木尚志. 道路地図データベースシステムにおける最短経路探索手法の研究. 修士論文, 筑波大学大学院理工学研究科, 1997.

[TUA+98] D. Tsur, J. D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query flocks: A generalization of association-rule mining. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD Record, pp. 1–12, Seattle, Washington, June 1998. ACM.

本学関係