共有ファイルサーバにおけるコミュニティ情報管理ツールの提案

石川　憲一† 　森嶋　厚行†† 　鈴木　勇††† 　杉本　重雄††

† 筑波大学大学院 図書館情報メディア研究科　〒 305–8550 茨城県つくば市春日 1–2
††† 筑波大学 図書館情報専門学群　〒 305–8550 茨城県つくば市春日 1–2
†† 筑波大学大学院 図書館情報メディア研究科/知的コミュニティ基盤研究センター
〒 305–8550 茨城県つくば市春日 1–2
E-mail: †{ishiken,mori,sugimoto}@slis.tsukuba.ac.jp, ††u2002171@ipe.tsukuba.ac.jp

あらまし　研究室などの知的コミュニティが利用する情報システムに共有ファイルサーバを設置し，複数の利用者で情報を共有することが一般的になっている．このようなコミュニティでは，利用者は共通のプロジェクトなどに関わることが多く，それぞれが作成したファイル間に何らかの関連があることが多い．これまでは，これらの関連は利用者同士による情報システム外のコミュニケーションを通じてやりとりされ，情報の共有と管理に暗黙に利用されてきた．しかし，共有ファイルサーバに格納されるファイル数が増加したり，プロジェクトの人員変更が行われたり等すると，このような仕組みを効率よく維持することは難しい．したがって，システムとしてなんらかの支援を行う必要があると考えられる．しかし，既存のファイルシステムでは，このような目的を支援するための機能が不十分であった．本稿では，このような共有ファイルサーバを通じた情報共有と管理を支援するためのツールを提案する．
キーワード　コミュニティ情報管理，共有ファイルサーバ，部分最適，全体最適

Proposal of a Tool to Manage Community Information in Shared File Servers

Kenichi ISHIKAWA†, Atsuyuki MORISHIMA††, Isamu SUZUKI†††, and Shigeo SUGIMOTO††

† Grad. Sch. of Library, Information and Media Studies, Univ. of Tsukuba,　1–2 Kasuga, Tsukuba, Ibaraki, 305–8550 Japan
††† Sch. of Library and Information Science, Univ. of Tsukuba,　1–2 Kasuga, Tsukuba, Ibaraki, 305–8550 Japan
†† Grad. Sch. of Library, Information and Media Studies/ Research Center for Knowledge Communities,
Univ. of Tsukuba.　1–2 Kasuga, Tsukuba, Ibaraki, 305–8550 Japan
E-mail: †{ishiken,mori,sugimoto}@slis.tsukuba.ac.jp, ††u2002171@ipe.tsukuba.ac.jp

Abstract　Today, more and more people in knowledge communities, like research laboratories, are using shared file servers to store and share their information. In such communities, people tend to work together in projects, and their files stored in shared file servers often have relationships with each other. Such relationships are usually exchanged off line and used implicitly to facilitate the management and sharing of information. However, as the number of files stored in the file servers becomes larger and the people in projects change, such system tend not to work. Therefore, it is desirable that software tools provide the function. But the current file systems are insufficient in providing such function. This paper proposes a tool to give advanced functions for the management and sharing of information stored in the shared file servers.
Key words　Community Infomation Management,Shared File Servers,Partial Optimization,Total Optimization

1. はじめに

研究室などの知的コミュニティが利用する情報システムに共有ファイルサーバを設置し，コミュニティ内の利用者が情報を共有することが一般的になっている．

このようなコミュニティでは，複数の人員が共通のプロジェクトなどに関わることが多いため，それぞれが作成し，共有ファイルサーバに置いたファイル間に何らかの関連があることが多い．しかし，これらの関連が明示的にわかるようにファイルが配置されているとは限らない．たとえば，互いに関連するファイルが同じディレクトリに格納されておらず，各作成者のホームディレクトリの下位ディレクトリに分散して格納されているということがしばしばある．

これまでは，これらの関連は，利用者同士による情報システ

ム外でのコミュニケーションを通じてやりとりされ，情報の共有と管理に暗黙に利用されてきた．たとえば，「今度投稿する論文の第 4 章の最新バージョンをもらえないか」「卒業するときには研究に関連する全てのファイルはこのディレクトリの下に整理して置いて欲しい」などのやりとりを通じて，関連ファイルの利用と管理が行われてきた．

しかし，共有ファイルサーバに格納されるファイル数が増加，あるいは，プロジェクトの人員変更などが行われると，利用者間によるコミュニケーションだけではデータの管理を効率よく維持することは難しい．したがって，システムとしてなんらかの支援を行う必要があると考えられる．しかし，既存のファイルシステムでは，単純なディレクトリ作成やファイルのコピー機能などしか提供していないため，このような目的を支援するための機能が不十分であった．

本稿では，このような共有ファイルサーバを通じた情報共有と管理を支援するためのツールを提案する．本ツールは単に関連ファイルを一カ所に置き管理するというものではない．なぜなら，先に書いたように関連ファイルが分散する理由の一つは，各メンバが自分にとって作業がしやすいようにファイルの配置を自分用に「最適化」していると考えられるからである．研究室などの知的コミュニティにおいては，最大化すべきはコミュニティメンバの知的生産性であるため，メンバの作業に不自然な制約をつけることはできるだけ避けたい．一方，各個人の知的生産性の向上を目的に最適化されたものが，コミュニティ全体の情報管理にとっても最適であるとは限らない．なぜなら，上の例にも挙げたとおり，関連する情報が分散されてしまい，かつそれらの関連の発見が難しくなってしまうからである．したがって，本ツールでは，情報間の関連を明示的に表現するための機構を導入する．さらに，各構成員に対しての最適と各プロジェクトに対しての最適など，様々な「X-最適」に向けて，ディレクトリ構造上にビューを構成できるような仕組みを用意する．ここでの問題は，ファイルサーバに格納されている各ファイル間の関係を見つけることと，それらの関係を用いた新しいディレクトリ構成をどのように作成するかということである．

本稿では次を行う．(1) 「X-最適」のためにファイルサーバに分散したファイル群からある目的に沿ったファイルを取り出し新しい木構造のビューを生成するオペレータを提案する．(2) オペレータの動作として，与えられた最適化の条件に応じてどのように新たなディレクトリの木構造を決定するかについて議論し，実現のためのアルゴリズムを提案する．本論文は以下のように構成される．2 章では関連研究を説明する．3 章では実現したい世界を具体的なシナリオを用いて説明する．ここで提案するオペレータの利用法についても説明する．4 章ではオペレータの処理について述べる．5 章では予備実験について説明する．6 章はまとめである．

2. 関 連 研 究

本稿で取り組もうとしている問題の領域を我々は**コミュニティ情報管理** (Community Information Management, CIM) と呼ぶ．図 1 は CIM と個人情報管理 (Personal Information Management,PIM)，Web スケール情報管理 (Web-scale Information Management,WIM) との関連を図示したものである．

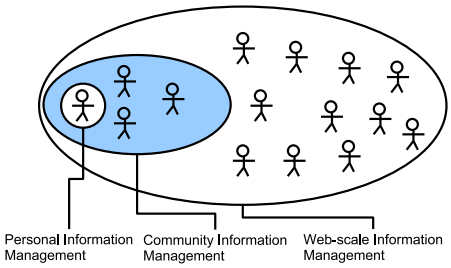

図 1　Community Infomation Management の位置づけ

以下ではこれらの違いについて説明する．まず話を簡単にするため，情報資源を利用するグループ全体に対して情報の管理方法が最適化されていることを「全体最適」と呼び，ある特定のメンバに対して情報の管理が最適化されていることを「部分最適」とする．このとき，PIM では，全体最適=部分最適であるため，前章で説明した CIM で起こるような問題は起きにくい．CIM においては，先に述べたように全体最適と部分最適との折り合いをつける必要がある．WIM では，利用者数が多くかつバラエティに富んでおり，特定のコミュニティを対象とした本研究と違って，全体的に見れば各利用者の関心が共通しておらず，また，お互いの情報が密接に関連しているとは限らない．ただし，Web の中にはそのようなコミュニティが数多く存在していると考えられる．本研究は，そのようなコミュニティの情報管理に焦点を絞り，有用なツールを開発しようとするものである．

実際には，個人と全体だけではなく，様々な「X-最適」が存在する．たとえば，あるコミュニティでは，複数のプロジェクトが立ち上がっており，かつ，それらのプロジェクトは多くのサブプロジェクトを抱えているかもしれない．したがって，これらの様々な「X-最適」に対応する必要がある．しかも，これらのプロジェクトの重要性は時期に応じて変化する可能性があるため，これらの重要度の変化に対しても対応できるような仕組みが必要である．

このような「X-最適」な構造を自動的に作成するための研究は我々の知る限り存在しないが，必要な要素技術に関しては，数多くの関連研究がある．まず，PIM や WIM の領域においても，情報間の関連を発見する研究は数多く行われてきた．たとえば，SEMEX [1] は個人で管理している情報に対して，関連を用いた問合せ (query by associations) を可能にするための PIM ツールである．SEMEX は，あらかじめドメイン毎に定義されたスキーマを入力として与えておくと，自動的にメールの宛先や内容などから人物やイベントなどのオブジェクトを抽出し，それらの関連を発見する．また，WIM に関して言えば，Web ページ間のハイパーリンクの自動生成の研究も数多くある [2] [3]．しかし，これらの研究の焦点はあくまでも関連の発見であり，我々が取り組む問題とは異なる．ただし，関連の発見に関しては，これらで提案されてきた手法が適用可能であると考えられる．

これまでの，コミュニティを対象とした情報管理ツールの代表としては，数多く存在するグループウェアがある．また，複数のソフトウェア開発者で利用する CVS [7] や Subversion [8] のようなバージョン管理システムがある．一般に，これらにおける情報管理は，スケジュールやプロジェクト情報，バージョン間の管理など，定型化された作業を支援することに焦点を置いている．さらに，これらのツールの共通点は，全て「全体最適」に固定されたツールであるということである．したがって，たとえば，バージョン管理を行うためには，中央のサーバの設定や，毎回接続してチェックインやチェックアウトの手続きを行う必要があるなど，利用のハードルが高くなる傾向にある．一般には，このような，定型化された作業でかつ全体最適のみを実現する場合は，固定したデータベーススキーマをもつデータベースを利用すれば良い．我々が扱う問題はこれとは異なるため，新たなアプローチが必要である．

近年，デスクトップ上のデータに対してキーワード検索を提供するシステムとして Google Desktop Search [4]，Spotlight [6] などのシステムが開発されている．これらをファイルサーバ上の情報共有に利用することも考えられる．しかしながら，(1) 検索対象がテキストファイル (もしくはテキストを抽出できるもの) に限定されていること，(2) 他人の作成したファイルを検索するときに適切な検索語を思いつくとは限らないこと，(3) 内容のマッチングによってしか検索できず，他のファイルから参照されているファイルなどの検索が出来ないこと，(4) 検索した結果のファイル集合に対して，さらなる操作を行うことが困難であること，などから，我々の目標とは焦点が異なっている．

近年，コミュニティの非定型的な情報共有を支援しようという試みとして，Wiki [9] や各種コンテンツマネージメントシステム [10] [11] がある．これらと本研究の本質的な違いは，前者が Web アーキテクチャ上に情報共有のためのプリミティブな機能を用意しようとしているのに対し，本研究では既存のファイルシステム上に情報共有のための**より高度な機能**を用意することを目指している事である．

近年，従来型のディレクトリ構造に基づかないファイル管理手法も提案されてきている [13]．本稿では，既存のファイルシステムをベースとした情報共有の枠組みを提案しているが，そこで用いている関連ファイル群の同定手法などは必ずしも既存のファイルシステムに依存していないため，他のファイル管理手法と組合せて利用することも可能と考えている．

3. シナリオ例

ここでは具体的なシナリオを用いて，提案ツールで実現したい機能を説明する．

M 嶋研究室では，共有ファイルサーバを用いて情報共有を行っている．そのディレクトリ構造の一部を図 2 に示す．○はディレクトリ，□はファイルを表す．研究室ではいくつかの研究プロジェクトが動いており，その一つに「CIM」プロジェクトと呼ばれるものがある．研究室のメンバの一人である「ishiken」君は「CIM」プロジェクトの担当者である．CIM プロジェクトに関連するファイルは，それぞれ/project/CIM，/source/CIM，/home/ishiken ディレクトリの下に分散している．

具体的には，次のようにファイルが分散している．/projects/CIM 以下のディレクトリには論文ファイル「論文.tex」，研究計画をまとめた「研究計画.txt」がある． /home/ishiken/ 以下には CIM プロジェクトの打ち合わせのまとめである「打ち合わせ.tex」．そして /source/CIM 以下には CIM プロジェクトのコードが格納されている．

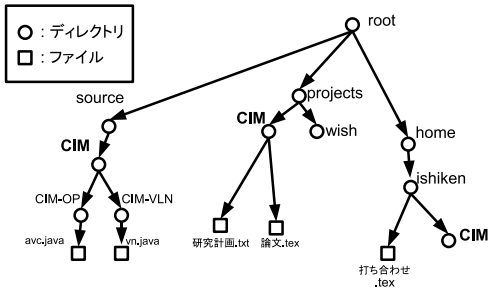

図 2　対象とするディレクトリ構造

CIM プロジェクトは現状では ishiken 君一人が担当しているので，できれば自分のホーム/home/ishiken/の下に CIM 関連ファイルを置きたいと考えている．その理由は，**ファイルシステムの操作の単位はディレクトリ**だからである．たとえば，関連ファイルをまとめてコピーしたりアーカイブするためにも，その単位はディレクトリである．

提案ツールでは次の作業を順に行う事により，/home/ishiken/ 以下に CIM プロジェクトに関連するファイルを移動することができる

（1） CIM プロジェクトの関連ファイルを発見する．

（2） CIM プロジェクトの関連ファイルを/home/ishiken の下に移動する．

本ツールでは，これら二つの作業を，それぞれ行うコマンドが用意されている．詳細は後述するが，前者を実行するコマンドを relate コマンド，後者を実行するコマンドを optimze コマンドと呼ぶ．

まず，(1) を実現するために relate コマンドを実行すると，図 3 に示すように関連を発見する．図 3 における点線はファイルやディレクトリ間の関連を表しており，本研究では **r-link** と呼ぶ．

各 r-link には関連の種類を表す**ラベル**がつく．この例では，/source/CIM,/projects/CIM,/home/ishiken/CIM 間の r-link に対しては，これらが同じ CIM プロジェクトを表すディレクトリであることを表す “same” ラベルが付く．また，コンテンツの類似度が高い/home/ishiken/打ち合わせ.tex と /projects/CIM/論文.tex の間に関連を表す “similar” ラベルが付いた r-link が追加される．

relate コマンドによって生成された r-link は，そのリンクをたどって関連先にたどれるように何らかの形でファイルシステム上に実装される．r-link をどのようにファイルシステム上に実装するかは本質的な問題ではなく，様々な方法が考えられる．たとえば，各ディレクトリに HTML ファイルを置き，r-link の情報をアンカーとして表現させておけば，そのアンカーをたど

ることにより，r-link の先の関連ノードにアクセスできるようになる．

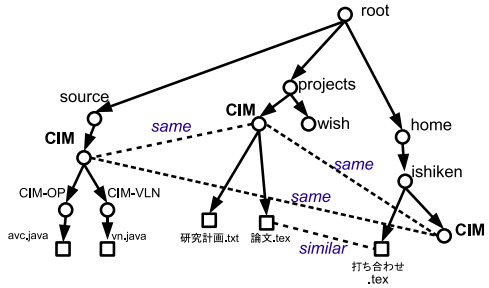

図 3　relate コマンドを実行した結果

次に，(2) を実現するために optimize コマンドを実行して，関連ファイルを /home/ishiken ディレクトリ以下に移動する．optimize コマンドのパラメータは 2 つある．(i) 最適化の中心となるノード (を表すパス式) n と，(ii) 少なくともこの範囲のノードは n と強く関連するという領域を表すパス式 R である．この例では，$n = /home/ishiken/$，$R = (/home/ishiken/*)\mathbf{UNION}(/projects/CIM)$ である．

以上のコマンドを実行する事で，/home/ishiken 以下に CIM 関連のファイルを集める事ができる．図 4 では，ノード (ファイルやディレクトリ) の移動が，グラフの辺の種類 (r-link およびディレクトリを表す有向辺) が変更されることで表現されている．新しい/home/ishiken ディレクトリの下には，指定された/projects/CIM 以下のファイルだけでなく，同じ CIM プロジェクトに属していると思われる /source/CIM 以下のファイルも移動していることがわかる．

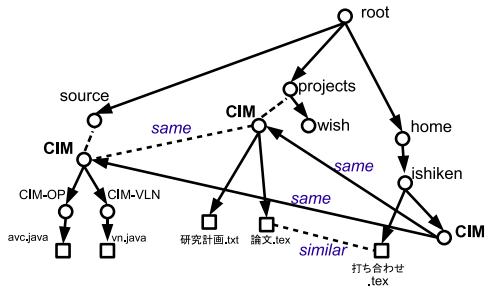

図 4　optimize コマンドを実行した結果

optimize 操作を行っても，各辺の種類が変更されるだけであり，グラフ構造そのものは変更が無いことに注意して欲しい．したがって，/projects/CIM からも (何らかの手段で実装された)r-link 経由ではあるが，依然として CIM 関連ファイルにアクセスすることができる．ただし，関連ファイルが/projects/CIM 以下と異なる場所に分散してしまうため，/projects/CIM からはファイルシステムのディレクトリ操作などを利用して関連ファイル群を操作することができなくなる．

以上は一例であるが，同様の操作を用いて，新人の鈴木君が CIM のサブプロジェクトを担当することになったときに，optimize コマンドを用いて一部を鈴木君のディレクトリの下に置くことができる．また，プロジェクトの管理者が/projects/CIM ディレクトリに関連ファイルを集めるよう optimize コマンドを利用し，その後関連ファイルをまとめてアーカイブしたりと行った操作を行うことも可能になる．

4. X-最適を実現するための機構

本章では 3 章で説明したシナリオを実現するための機構について説明する．シナリオにも書いたとおり，必要な操作は次の二つである．

（1） 関連を見つける

（2） 関連情報に基づき，特定のファイル群を指定されたディレクトリの下に移動する．

4.1 問題の形式化

問題を形式化するため，ディレクトリ構造を次に定義する X-Graph としてモデル化する．

定義 1. X-Graph $G = (V, E)$ は次の条件を満たすグラフである．

- $V = DV \cup FV$ はディレクトリおよびファイルを表すノードの集合である．ここで DV はディレクトリノードの集合，FV はファイルノードの集合である．
- $E = DE \cup XE$ は辺の集合である．ここで，DE はディレクトリ辺 (有向辺) の集合，XE は r-link(無向辺) の集合である．各 r-link はラベルを持つ．
- G の部分グラフ (V, DE) は G の全域木である．

以下の議論では，V のことを $V(G)$，E のことを $E(G)$ と表記することがある．また，(V, DE) が木である事を強調するために，この部分グラフ (V, DE) を $tree(G)$ と表記する．また，$(V, DE) = T_i$ となるようなグラフを G_{T_i} と表記することがある．すなわち，$tree(G_{T_i}) = T_i$ である．

以上のモデルを利用すると，本ツールによる操作は次のように言い換える事ができる．

（1） relate コマンドではディレクトリ構造を表す木 T に r-link を追加し，グラフ G_{T_1} を作成する．

（2） optimize コマンドは，3 章で説明したようにパラメータとして (1) 最適化の中心となる G_{T_1} 中のノード n と，(2) 少なくともこの領域は強く関連するという部分グラフ R (ただし $n \in R$) を受け取り，G_{T_1} と同型であるが辺の種類のみが異なる G_{T_2} を作成する．つまり，$T_1 = tree(G_{T_1})$，$T_2 = tree(G_{T_2})$ であり，$T_1 \neq T_2$ である．さらにこの T_2 は，n を根とし R' 中のノードを全て含む部分木 $T^{(n,R')}$ を持つ．ここで，R' は R に含まれるノードに強く関連する領域を表す部分グラフ (ただし $V(R') \supseteq V(R)$) である (図 5).

次にそれぞれの手順を説明する．

4.2 relate コマンド：ディレクトリ構造を表す木 T_1 に r-link を追加し，G_{T_1} を作成

ファイルやディレクトリ間の関連を発見し，その関連を r-link とそのラベルとして表現する．r-link のラベルは任意のものをつけることができるが，あらかじめ意味が定めされている特別なラベルとして “copy”を用意する．これは，全く内容が一致するファイル間に張られる r-link のラベルである．

r-link をどのように発見するかの議論は本稿の焦点ではないので詳細は省略するが，たとえば，思いつきやすいものとして次のようなものが考えられる．

（1） 内容の類似度を用いて r-link[similar] を発見する．

（2） tex ファイルから eps ファイルへの参照があった場合，r-link[refersTo] を張る．

（3） ディレクトリの名前が同じであれば同じ概念を指していると見なして r-link[same] を張る．

4.3 optimize コマンド：グラフ G_{T_1} と同型の新しいグラフ $G_{T_2'}$ を作成

G_{T_1} の形を変えず辺の種類のみを変更したグラフ G_{T_2} を作る問題は次のように分割できる (図 5).

フェーズ 1 グラフ G_{T_1} から，R を含み R 中のノードに強く関連すると考えられる部分グラフ R' を抽出する．ここで，$V(G_{T_1}) \supseteq V(R') \supseteq V(R)$ である．

フェーズ 2 R' から，n をルートとする全域木である $T^{(n,R')}$ を作成する．

フェーズ 3 R' と同型であり，ディレクトリ辺だけを見たときには $T^{(n,R')}$ に一致するようなグラフ $R'_{T^{(n,R')}}$ を作成する．

フェーズ 4 G の部分グラフ R' を $R'_{T^{(n,R')}}$ で置換したグラフを G_{T_2} とする．

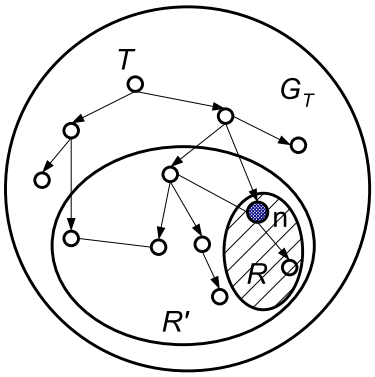

図 5 G_T,n,R,R' の関係

上記フェーズにおいて難しいのは，フェーズ 1 およびフェーズ 2 である．なぜなら，これらのフェーズでは R'，および $R'_{T^{(n,R')}}$ を決定する必要があるからである．どのような基準でこれらを決定するかについてを，次に説明する．

まず，R' は，直感的には，与えられた R に強く関連するノードを含んだ領域を表す部分グラフである．具体的には，次のように定義される．

定義 2. R を G のある部分グラフとする．また，$S \subset V(G)$ となるようなあるノード集合があり，これらを**固定ノード集合**と呼ぶ．このとき，G **と** S **の元での** R **の関連領域** $R'_{[G,S,R]}$ は，次の条件を満たす G の部分グラフである．

（1） $V(R') \supseteq V(R)$

（2） R' を縮約したノード n' を作成したとき，n' から G の他のノードへの辺の数が最も小さい（以下ではこの辺の数の事を R' の**スコア**と呼び，score(R') と表記する）

（3） R' は (2) の条件を満たすもののうち，領域が最大である．

（4） $V(R')$ は $V(R)$ に含まれている以外の固定ノード ($\in S$) を含まない．

（5） R' のノードを始点とし，R' 外のノードを終点とするディレクトリ辺があってはならない．

定義 2 の直感的な説明は次の通りである．まず，(1) は，R' が R を含んでい無ければならないことを表している．(2) は，R' 中のノード間の関連が密であり，R' 中のノードと R 外のノードの関連が疎であることを示している．(3)(4) は指定された固定ノードを含まない範囲で，R' のサイズは可能な限り大きくとられることを示している．ここで，もし固定ノードが指定されなければ，常に $R'_{[G,\{\},R]} = G$ となってしまうことに注意して欲しい．最後に，(5) は，R' にあるノード m が含まれるときは，必ず m をルートとする部分木全体も R' に含まれなければならないことを表している．

以上のように定義される $R'_{[G,S,R]}$ は一意に定まる (証明は省略)．以下では特に断りの無い限り R' が $R'_{[G,S,R]}$ を表すものとする．

このような R' が求まれば，続いて R' と n から $R'_{T^{(n,R')}}$ を計算する必要がある．$T^{(n,R')}$ は，n をルートとする R' の全域木であって，かつ，R' のディレクトリ辺をできるだけ保存するように構築する．具体的には次で計算される．

定義 3. R' が与えられたとき，$T^{(n,R')}$ は n を根とする R' の最小全域木である．ただし，ディレクトリ辺の矢印の向きが，木構造の子から親に向かってはならない．また，最小全域木を計算する際，各辺への重み付けは次のように行う．以下では無向の r-link 辺は，方向の異なる 2 つの有向の r-link 辺が相互に接続されていると考え，全て有向辺であるとして考える．

- ディレクトリ辺の重みは 0 である．
- ディレクトリを始点とする r-link 辺の重みは 1 である．
- ファイルを始点とする r-link 辺の重みは 2 である．

4.4 アルゴリズム

本節では，$R'_{[G,S,R]}$ を導出するフェーズ 1 のアルゴリズムを説明する．フェーズ 2 のアルゴリズムは，前節の定義 3 で用いた辺の重み付けを行った後通常の最小全域木を発見するアルゴリズム [12] を利用すればよいため省略する．また，フェーズ 3,4 のアルゴリズムは自明であるため省略する．

図 6 がフェーズ 1 のアルゴリズムである．関数 phase1 は，入力として，グラフ G, 固定ノードの集合 S, ノード n, 領域 (部分グラフ)R を受け取る．そして，出力として $R'_{[G,S,R]}$ を返す．直感的には次のように動作する．まずグラフ $R'_0 = R$ を構築し，グラフ関数 explore を再帰的に呼び出して，領域を広げていく．ただし，定義 2 の (4)(5) に反したノードへは領域を広げない．領域を広げる過程におけるそれぞれの途中結果 R'_i は，score(R'_i) と供に変数 $candidates$ に保存されていく．次に，得られた $candidates$ に含まれる R'_i の中から最低スコアを持つも

のだけを $minimums$ に保存していく．

関数 phase1 と explore の詳細は次の通りである．

phase1 関数

(5-6 行目)　$R'_0 = R$ からスタートし，徐々に広げていくため explore 関数を呼ぶ．途中過程で得られた各結果は，R' の候補 R'_i として，グローバル変数である $candidates$ に格納される．

(8-17 行目)　$candidates$ に格納された各候補 R_i のうち，最小のスコアを持つもの (一般には複数ある) を $minimums$ に格納する．

(19-24 行目)　$minimums$ に格納されている各候補 R_i のうち，最大の領域を持つものを R' とする．R' は一意に決まる．

explore 関数

(28-29 行目)　R' 中のノードから距離 1 で到達可能な G 中のノードの集合を V_{rest} とする．

(31-35 行目)　R'_i が連結グラフであれば，R'_i から G の他のノードへのリンク (ただし，r-link[same] を除く) 数 $score(R'_i)$ との組 $(R'_i, score(R'_i))$ を candidates に入れる．ただし，R'_i が連結でない場合はエラーを返す．

(37-44 行目)　固定ノードを除いて，R'_i を徐々に広げていく．

4.5　現実のディレクトリ構造へのマッピング

最後に，前節で説明したアルゴリズムによって書き換えられた X-Graph G_{T_2} をファイルシステム中の実際のディレクトリ構造にマッピングする際の問題について議論する (厳密には，変更が生じた $R'_{T^{(n,R')}}$ のみマッピングすればよい)．作成した木の View をファイルシステムに対してマッピングするかどうかは，利用者側がオプションで指定可能とする．各ディレクトリ辺は，ディレクトリの親子構造として実装する．また，各 r-link はリンクを表現するための HTML ファイル等を用意する事により実装できる．しかしながら，必ずしも単純にマッピングできない場合が存在する．それは，ファイルを始めとするディレクトリ辺である．本システムでは，このケースに対処するため，ディレクトリ構造にマッピングする前に，G_{T_2} に**仲介ディレクトリ**を挿入する. 例えば図 7 の場合，$F_1 \rightarrow F_2$ のようにファイルからファイルに向けてディレクトリ辺がある．この場合は仲介ディレクトリ「F1_similar」が F_1 の上位ディレクトリである D_1 の下に作成される．仲介ディレクトリは「ファイル名_辺のラベル名」の名前で作成する．

5.　予 備 実 験

4 章で説明したアルゴリズムが人手による最適化をある程度シュミレートしている事を示すために以下に示す 3 つの異なる方法での最適化を比較した．

（1）　本提案アルゴリズムによる最適化

（2）　人手による最適化

（3）　領域 R' を求めず，コマンドで指定した領域 R のファイルのみを移動する最適化

今回の実験で利用するディレクトリ構造を図 8 に示す．このディレクトリ構造は/home/ishiken ディレクトリの下に「論文.tex」があり,「論文.tex」は/fig ディレクトリにある「fig1.eps」と「fig2.eps」を参照している．又，/home/toshi ディレクトリの

```
Input: グラフ G, 固定ノード集合 S, n, R.  ただし R は G の部分グラフ，また，n in V(R).
output: R'
1     Global Set candidates={}; // R' の全ての候補の集合
2     Global Set minimums={}; // R' の候補のうち最小スコアを持つものの集合
3
4     Phase1(G, S, n, R) { // R' を求める
5       Graph R'_0=R;
6       explore(G, S, R, R'_0); // candidates を求める
7
8       // minimums を求める
9       int min_score= 無限大;
10      for each (R'_i, score(R'_i)) in candidates {
11        if (min_score > score(R'_i)) {
12          min_score=score(R'_i); minimums={ R'_i };
13        }
14        if (min_score==score(R'_i)){
15          minimums =minimums union { R'_i };
16        }
17      }
18
19      // minimums に含まれる R' 候補のうち最大の領域を持つものを探す
20      Graph R'=({}, {}); // 結果
21      for each R'_i in minimums {
22        if (R' is included by R'_i) R'=R'_i;
23      }
24      return R';
25    }
26
27    void explore(G, S, R, R'_i) {
28      V_rest=V(G)-V(R'_i);
29
30      nextN=V_rest 中のノードのうち R'_i 中のノードから距離 1 で到達可能なもの.
31      if (R'_i が連結) {
32        // candidates に R'_i を追加
33        score(R'_i)= R'_i から nextN に含まれるノードへの辺のうち，r-link[copy] 以外の本数;
34        candidates.add((R'_i, score(R'_i)));
35      }
36
37      nextN=(nextN-S) に含まれるノードのうち，そのノードが持つ
          ディレクトリ辺 (DE) が全て R'_i 中を指しているノードの集合;
          //定義 2 (4)(5) に反したノードを除去
38
39      // R'_i を広げる
40      for each n' ∈ nextN {
41        V(R'_i)=V(R'_i) union { n' };
42        E(R'_i)=E(R'_i) union G に含まれる R'_i 中のノードと n' 間の辺の集合;
43        explore(G, S, R, R'_i);
44      }
45      return;
46    }
```

図 6　フェーズ 1：R' を求めるアルゴリズム

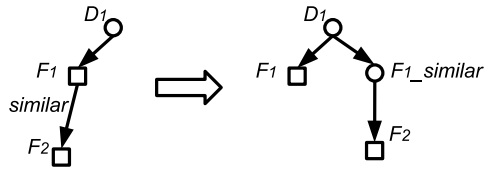

図 7　仲介ディレクトリの作成

下に「要約.tex」があり，このファイルの内容は「論文.tex」と類似度が高い．今回の実験では「ishiken」にとって使いやすくなるように上記の 3 つの手法それぞれで最適化を行った．

まず最初に (1) の本提案アルゴリズムによる最適化を示す．今回は次のようなコマンドを実行した．

$optimize_{/home/ishiken,/home/ishiken/*,/home/toshi/要約.tex}$

ここではルートノード n として/home/ishiken を，n を含む領域 R として/home/ishiken/* と /home/toshi/要約.tex を指定した．

次に 4.4 節で説明した phase1 のアルゴリズムに従い，スコアが最小かつ，領域の大きさが最大となる R' を求める．領域を順番に拡大していくと，次のような領域 R' の候補 R'_0〜R'_3

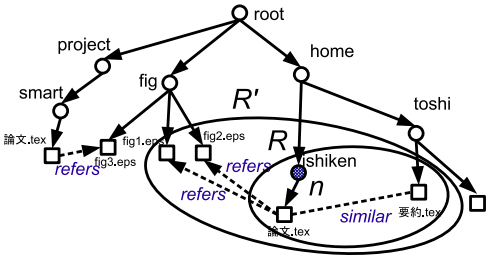

図 8　全域木を求める

が取得できる．$R'_0 = R$ とする．

$V(R'_0) = \{\{n\}, \{\text{論文}.tex\}, \{\text{要約}.tex\}\}$

$V(R'_1) = \{\{n\}, \{\text{論文}.tex\}, \{\text{要約}.tex\}, \{fig1.eps\}\}$

$V(R'_2) = \{\{n\}, \{\text{論文}.tex\}, \{\text{要約}.tex\}, \{fig2.eps\}\}$

$V(R'_3) = \{\{n\}, \{\text{論文}.tex\}, \{\text{要約}.tex\}, \{fig1.eps\}, \{fig2.eps\}\}$

領域 R' の候補として /fig , /home , /home/toshi ノードが含まれ無いのは定義 2 で「R' から出るディレクトリ辺があってはならない」という条件が含まれているためである．R'_0〜R'_3 それぞれの領域の score を計算すると，以下のようになる．

$score(R'_0) = 4$

$score(R'_1) = 4$

$score(R'_2) = 4$

$score(R'_3) = 4$

この中でスコアが最小かつ，領域が最大となるのは R'_3 となる．よって求める R' の領域は図 8 に示したようになる．

次にフェーズ 2 で，領域 R' の辺に対してスコア付けを行い，その中でスコアが最小となるミニマムスパニングツリーを求める．今回の場合「論文.tex」から eps ファイルに対する r-link[refersTo] 辺と「論文.tex」,「要約.tex」の間の r-link[similar] 辺に対してはスコア 3 が付けられ，その他のディレクトリ辺を表す辺にはスコア 0 が付けられる．しかしながら，今回の例ではノード n をルートとするスパニングツリー候補は一つしか存在しない．

求められたスパニングツリーにはファイルからファイルへの辺が存在する．そこで，4．5 節に従い仲介ディレクトリを作成し格納した．結果を図 9 に示す．

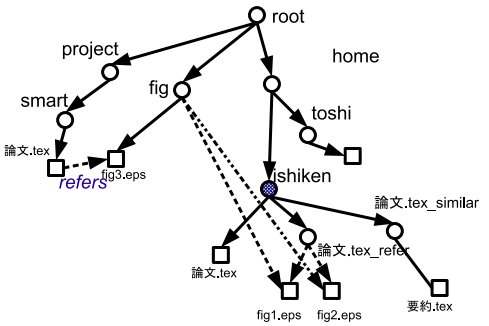

図 9　実験：コマンドを利用した最適化の結果

次に (2) 人手による最適化の結果を示す．

人手による最適化では 6 名の被験者に図 8 を見せ,「このディレクトリ構造を ishiken にとって使いやすい形にしたい」と説明しディレクトリ構造を変換してもらった．その結果，6 名ともが「fig」ディレクトリを/home/ishiken ディレクトリの下に作成し,「fig1.eps」,「fig2.eps」を移動した．又,「要約.tex」を/home/ishiken/ ディレクトリの下に移動した．

次に (3) 領域 R' を求めず，コマンドで指定した領域 R のファイルのみを移動する最適化の結果を示す．今回の場合，領域 R として指定した/home/ishiken/* と /home/toshi/要約.tex を/home/ishiken 以下に移動する事はできるが，R で指定しなかった「fig1.eps」,「fig2.eps」のファイルは移動する事はできなかった．

以上 (1)〜(3) の実験を行った結果，(1) の本提案アルゴリズムによる最適化は (2) の人手による最適化とほぼ一致した．しかし，(3) のアルゴリズムでは「fig」ディレクトリ以下のファイルを/home/ishiken 以下に集める事ができなかった．

以上，今回の予備実験では，4 章で提案した領域 R' の拡張と最小全域木の作成，ファイルシステムへのマッピングを組み合わせたアルゴリズムが，人手による最適化をある程度うまくシュミレートできている事が分かった．

6. ま と め

本稿ではファイルサーバに分散したファイル群からある目的に沿ったファイルを取り出し，新しいディレクトリのビューを生成する機構を提案した．本研究のポイントの一つはファイルシステム上に情報共有のための高度な機能を用意した事である．

今後は本ツールの有効性を評価するためのより詳細な実験やアルゴリズムの改良，コミュニティ情報管理を支援する他の機能の開発を行う予定である．

謝　　辞

ゼミなどでご議論いただきました筑波大学図書館情報メディア研究科田畑孝一教授，阪口哲男助教授，永森光晴講師に感謝いたします．本研究の一部は日本学術振興会科学研究費補助金若手研究 (B)(課題番号 15700108) による．

文　　献

[1] Yuhan Cai, Xin Dong, Alon Y. Halevy, Jing Liu and Jayant Madhavan: Personal Information Management with SEMEX, SIGMOD DEMO 2005

[2] 中谷圭吾，鈴木 優，川越恭二：“ 文書間類似度とキーワードを用いた Web リンク自動生成手法 ”，日本データベース学会 Letters Vol.4, No.1, pp.89-92 2005 年 7 月

[3] Yuhan Cai, Xin Luna Dong, Alon Halevy, Jing Michelle Liu, and Jayant Madhavan: “ Linking in Context ”, Proceedings of the twelfth ACM conference on Hypertext and Hypermedia, pp. 151 - 160 2001

[4] Google Destop Search http://desktop.google.com/

[5] Windows Desktop Search http://toolbar.msn.com/

[6] Spotlight http://www

[7] CVS http://ximbiot.com/cvs/cvshome/

[8] Subversion http://subversion.tigris.org/

[9] wiki http://www.c2.com/cgi/wiki?WelcomeVisitors

[10] xoops http://www.xoops.org/

[11] zope http://www.zope.org/

[12] Shimon Even : Graph ALGORITHMS. Potomac, Md. : Computer Science Press, c1979

[13] Dourish, P., Edwards, W.K., LaMarca, A. and Salisbury, M. (1999). Presto: An Experimental Architecture for Fluid Interactive Document Spaces. ACM Transactions on Computer-Human Interaction,

6(2), 133-161.