論文 / 著書情報
Article / Book Information

論題(和文)	アクセス履歴を用いたユーザの作業に対応する仮想ディレクトリの生成
Title(English)	Access-Log based Virtual Directory Creation to Restore User ' s Works
著者(和文)	小田切健一, 渡辺陽介, 横田治夫
Authors(English)	Kenichi OTAGIRI, Yousuke WATANABE, Haruo YOKOTA
掲載誌(和文)	DEIM2009論文集
Citation(English)	Proceeding of DEIM2009
Vol, no, pages	, ,
発行日 / Pub. date	2009, 3

DEIM Forum 2009 i1-27

アクセス履歴を用いたユーザの作業に対応する仮想ディレクトリの生成

小田切健一† 渡辺 陽介†† 横田 治夫†,††

† 東京工業大学院情報理工学研究科計算工学専攻 〒 152-8552 東京都目黒区大岡山 2-12-1
†† 東京工業大学学術国際情報センター 〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: †{otagiri,watanabe}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし 近年のファイル数の増加に伴い、大量のファイル管理が問題となっている。ファイル数急増に伴いディレクトリ数も急激に増え、同一作業に関わる一連のファイル群が分散してしまうとそれらの配置を把握することが困難となっている。本研究では、そのような分散してしまったファイル群を発見し、集約して仮想的なディレクトリとしてユーザに提示することを目的とする。同一作業に関わるファイルは同時に使用されることが多いと考え、アクセス履歴から抽出されるファイルアクセスの重複時間・重複回数などの尺度を用いて同一作業に関わる可能性が高い 2 ファイルの組み合わせを発見し、更にクラスタリング手法により同一作業に関わる可能性が高いファイル集合を発見する。本稿では、アクセス履歴からファイル同士が同一作業に関わる可能性を適切に算出する尺度や、同一作業に関わるファイル集合発見に適するクラスタリング手法について検討する。

キーワード ファイル管理, アクセス履歴, クラスタリング, 情報爆発, 仮想ディレクトリ, ファイル間関連度, 同一作業指数

Access-Log based Virtual Directory Creation to Restore User's Works

Kenichi OTAGIRI†, Yousuke WATANABE††, and Haruo YOKOTA†,††

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
†† Global Scientific Information and Computing Center, Tokyo Intitute of Technology 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
E-mail: †{otagiri,watanabe}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract Today the file management is a problem with the increase of files. When related files disperse over directories, it's difficult to find these files because of so many directories. This research proposes the method that finds such dispersed related files and presents these as a virtual directory. Assuming that user uses related files at the same time, we can estimate the strength of relation between files by the measure of the total time of file-use-overlap or by the measure of the total frequency of file-use-overlap or by other measures. And we can find gathers of strongly-related-files by clustering methods. This paper ingestigates which relation measure and which clustering method are effective for finding dispersed related files.

Key words file management, access log, clustering, information explosion, virtual directory, relationship between files, same work index

1. はじめに

過去に何かの作業で用いたファイル群は、作業の再開、新しい作業のひな形、参考資料、などに有効に利用することができる。しかしユーザは場面に応じて、ファイル種別ごとのディレクトリや、時期ごとのディレクトリ、テンポラリディレクトリなど、様々な分類方法によるディレクトリを作成する。そのため、同一作業に用いられたファイル群であっても、あるファイルは「画像ディレクトリ」、あるファイルは「○○年度ディレクトリ」といったように分散しがちである。さらに、近年のストレージの大容量化に伴いユーザが扱うデータ量が急増しているため、ファイル数やディレクトリ数も急増している。大量のファイルが大量のディレクトリに一貫性のない置き方をされ、しかも異なるディレクトリに分散しているために、過去の作業に用いられたファイル群を発見し再利用することが非常に困難となっている。

本研究では、過去のファイルを有効に再利用するために、過去に同一作業に用いられていたが異なるディレクトリに分散してしまったファイル群を発見するための手法の提案を行う。アイディアとして、デスクトップ検索でキーワード非含有ファイルを検索可能にした研究 FRIDAL [1] で確認された「アクセスが重複するファイル同士は同一作業に用いられていることが多い」という性質を利用する。

FRIDAL では、デスクトップ検索において、全文検索結果のファイル群と同時アクセスが高い頻度で起こったファイル群を全文検索結果に追加することで、全文検索単体では見つけられなかったファイルを検索可能にした。この研究では、同一作業に用いられたファイル同士はファイルアクセスの重複があると仮定しており、実際にそれを裏付ける結果が得られた。

本研究では、その性質を検索ではなくファイル整理に利用することを考えた [2]。FRIDAL では 2 ファイル間の関連を発見するだけであったが、互いにアクセス重複が多いファイル群を発見すればそのファイル群の全てのファイルが同一作業に属している可能性が高いと考えられる。2 要素間の関係から、関係の強い大きな集合を発見する方法は、クラスタリングと呼ばれており、確立した手法が多数存在する。実際に同一作業に属しているファイル群を発見できれば、それを仮想的なディレクトリとして提示でき、ユーザはそれを作業ごとに整理された使いやすいディレクトリとして扱うことができる。

また、本研究はファイルのアクセスの重複に着目しているため、ファイルアクセス履歴さえあればよく、実際のファイルの置かれ方やファイルの種別や内容を利用しない。そのため、ファイル種別ごとに個別対応をしたり、ファイルからテキストを抽出すると言った作業が不要であるという利点がある。

本稿では、提案手法が実際に分散してしまった同一作業に用いられたファイル群を発見できるかの評価を行う。その際ファイルアクセスの重複に着目するが、重複時間・重複回数といったいくつかの着目方法がある。また、手法の中でクラスタリングを行うが、クラスタリングにはノイズを許して大きなクラスタを作ろうとする物や、ノイズを極力排したクラスタを作ろうとする物など、様々な性質を持つ複数の手法がある。同一作業に用いられたファイル群を発見する際に、ファイルアクセスの重複のどこに着目するべきか、どのような性質を持つクラスタリング手法を使用するべきかは現時点では分かっていない。本稿ではそれを調べるために、実際のアクセス履歴を用いて評価実験を行う。

2. 仮想ディレクトリ生成手法概要

図 1 に提案手法の流れを示す。まず、ファイルへのアクセス履歴が記録されたログファイルがある。そのログファイルには「時刻・ファイル名・開いた／閉じた」という情報が格納されている。しかし、ユーザがそのファイルを使った時間と、ログ上の開いた・閉じたが対応していない事があるため、補完処理を行う必要がある。補完処理の詳細は 2.1 節で述べる。

次にファイルアクセスの重複から各 2 ファイルの組み合わせが同一作業に属する可能性を計算する。FRIDAL ではファイルアクセスの重複から計算される値を「ファイル間関連度」と呼んでいたが、本研究では同一作業に属するかの指標として用いるため「同一作業指数」と呼ぶ。ファイルアクセスの重複から計算される同一作業指数が大きいほどそのファイル同士が同一作業に属している可能性が高いと考える。同一作業指数を計算する方法として、ファイルアクセスの重複回数が多い物の得点が高いとするのか、ファイルアクセスの重複時間が多い物の得点が高いとするのか、様々な計算方法が考えられる。今回は、FRIDAL で提案された 4 つの尺度をそれぞれ単体で同一作業指数の計算方法として利用する。各尺度の詳細は 2.2 節で述べる。

最後に、各 2 ファイル間の同一作業指数から互いに同一作業指数が高いファイルの集合をクラスタリング手法を用いて発見する。今回は階層的クラスタリング [3] というクラスタリング手法を用いた。階層的クラスタリングは、内部で用いる距離の定義を変えることで性質を変えることができる。詳しくは 2.3 節で述べる。

2.1 ファイル使用時間の推定

本研究ではファイルサーバーに Samba [4] を用いた。Samba のアクセスログにはファイル名・開かれた/閉じられた・時刻といった情報が記録されている。しかし、「ユーザがファイルを開いたまま席を立つことがある」「ファイルを開いてメモリに読み込んだ後にすぐに閉じてしまうアプリケーションがある」といった理由により、ログ上の開かれていた時間とユーザが使用した時間が一致しない。そのため FRIDAL で用いられていた手法 [5] を用いてユーザが使用していた時間を推測する。まず、30 分区間ごとにファイルアクセスの有無からユーザの活動の有無を調べる。活動がなかった時間では、ファイルが開いた状態でも未使用として扱う。また、活動期間中に複数回のアクセスがあった場合、最初のアクセスと最後のアクセスの間をファイルを使用していた時間として扱う。更に、一分以下のファイル使用をゴミとして除去し、CLOSE がない場合の補完といった処理も行う。これらにより、ユーザが実際にファイルを使用していた時間が推定される。

2.2 同一作業指数の計算

クラスタリングに先立って、各 2 ファイル間のファイル使用時間の重複から、その 2 ファイル間が同一作業に属する可能性を表す同一作業指数を計算する。今回は、FRIDAL で提案された 4 つの尺度をそれぞれ単体で用いて同一作業指数を計算する。以下の数式中の x, y と図 2 中の x, y はそれぞれファイルを表している。

2.2.1 共起時間 T

2 ファイルの使用時間が重複していた合計時間に着目した尺度である。合計時間が大きければ同一作業に属している可能性が高いとする。t_i はそれぞれの重複時間を表しており、図 2 の t_1, t_2, t_3 に対応している。

$$T(x, y) = \sum_{i=1}^{n} t_i \tag{1}$$

2.2.2 共起回数 C

2 ファイルの使用時間が重複した回数に着目した尺度である。

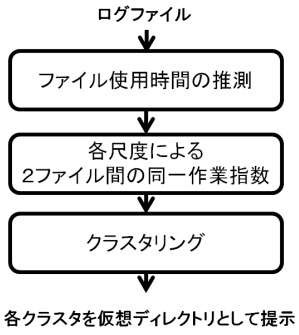

図 1　仮想ディレクトリ生成の流れ図

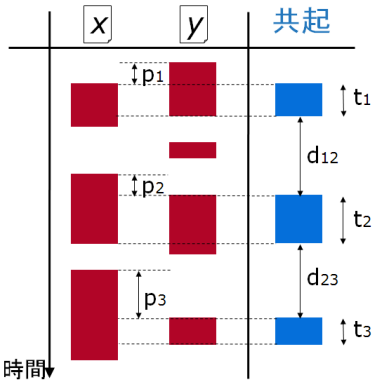

図 2　ファイル使用時間と尺度の関係

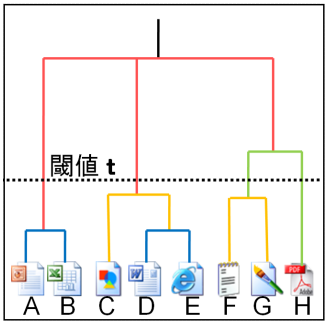

図 3　デンドログラム

この回数が大きければ同一作業に属する可能性が高いとする。図 2 では $n = 3$ である。

$$C(x, y) = n \tag{2}$$

2.2.3　使用開始の類似度 P

2 ファイルの使用時間が重複するとき、お互いのファイルの使用開始時刻が近いかどうかの尺度である。毎回の使用開始時刻が近いほど同一作業に属している可能性が高いとする。p_i はそれぞれの使用開始時刻のずれを表しており、図 2 の p_1, p_2, p_3 に対応している。

$$P(x, y) = (\sum_{i=1}^{n} p_i)^{-1} \tag{3}$$

2.2.4　共起間隔 D

2 ファイルの使用時間の重複がどれくらいの期間で起きているかの尺度である。同じ n 回の重複であっても、それが一時的であれば同一作業に属している可能性は低く、長い期間に渡っていれば同一作業に属している可能性が高いとする。$d_{i(i+1)}$ は i 番目の重複の終了から (i+1) 番目の重複の開始までの時間を表しており、図 2 の d_{12}, d_{23} に対応している。

$$D(x, y) = \sum_{i=1}^{n-1} d_{i(i+1)} \tag{4}$$

2.3　正　規　化

クラスタリングの前に以下のように、0 から 1 の範囲へ正規化を行う。$R(x, y)$ は実際には $T(x, y), C(x, y), P(x, y), D(x, y)$ のいずれかである。

$$R'(x, y) = \frac{\log R(x, y)}{\log \max_{x,y \in AllFiles} R(x, y)} \tag{5}$$

また、T,C,P,D は同一作業に属する可能性が高いほど高い指数をつけるが、今回用いるクラスタリング手法は距離の概念を用いるため強い結びつきほど値が小さい必要がある。そのため、以下の式で大小関係を逆転させる

$$d(x, y) = 1 - R'(x, y) \tag{6}$$

2.4　クラスタリング手法

クラスタリングは、大きな集合を共通の特徴をもつ部分集合へと切り分ける手法の総称である。要素そのものに特徴があり特徴が似ている要素を切り出す k-means のような手法と、要素そのものに特徴がなく要素間の関係のみを用いて関係の強い集合を発見する手法がある。本研究ではファイルの特徴ではなく、ファイル間の関係のみを用いる。そのため、前者の手法は用いることができず、後者のクラスタリング手法の一つである階層的クラスタリング [3] を用いた。

2.4.1　階層的クラスタリング

階層的クラスタリングは、最初にすべての要素を大きさ 1 のクラスタとし、距離が最小のクラスタ同士を連結していくことで徐々に大きなクラスタを求めるクラスタリング手法である。その際、クラスタ間の距離を測る必要があるが、本来要素間の距離しか与えられていないため、クラスタに含まれる要素間の距離を元にクラスタ同士の距離を定義する。距離の定義の仕方によりクラスタ間の距離が変わるため、距離の定義の仕方で要素が結合されていく順番や最終的なクラスタの大きさが異なる。

2.4.2　クラスタ間の距離の計算方法

本稿ではクラスタ間の距離の定義に以下の 4 種類を用いた。クラスタ A,B の要素と距離を以下のように表記した場合のそれぞれの定義を示す。

クラスタ $A = \{a_1, a_2, ..., a_m\}$
クラスタ $B = \{b_1, b_2, ..., b_n\}$
$D(A, B)$ クラスタ A-B 間の距離
$d(a, b)$ 要素間の距離

1) 単連結法

$D(A, B) = \min_{a_i \in A, b_j \in B} d(a_i, b_j)$

一番近い要素間の距離をクラスタの距離とする。そのため、全体としては A と B の要素の距離が大きくとも、一つでも距離が近い要素があればクラスタ間の距離が小さいと見なされる。そのため、クラスタが結合されやすく大きな仮想ディレクトリが期待できるが、不適切なクラスタも結合されやすくゴミを多く含む仮想ディレクトリも生成されやすいと推測される。

2) 完全連結法

$D(A, B) = \max_{a_i \in A, b_j \in B} d(a_i, b_j)$

一番遠い要素間の距離をクラスタ間の距離とする。そのため、全体としては A と B の要素の距離が小さくとも、一つでも距離が大きい要素があればクラスタ間の距離は大きいと見なされ

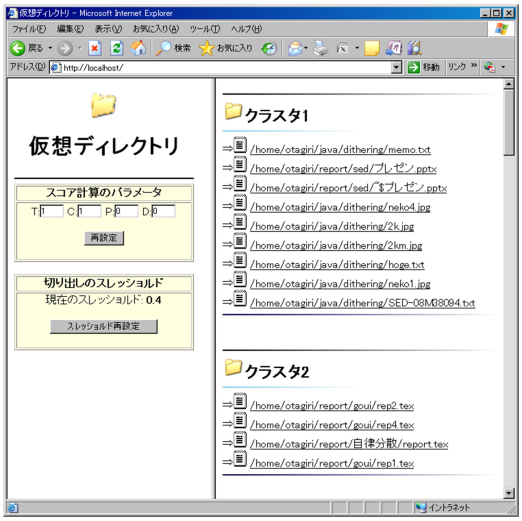

図 4　プロトタイプシステム

る。それゆえ、一つでも不適切なファイルが含まれたクラスタは結合されにくく、ゴミを含まない仮想ディレクトリが期待できる。

3) 群平均法

$D(A,B) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} d(a_i, b_j)$

クラスタそれぞれの要素の間の平均距離をクラスタ間の距離とする。そのため、単連結法と完全連結法の中間の性質を持つ仮想ディレクトリを生成すると考えられる。

4) ウォード法

クラスタ内の平方和の増加分が最小のクラスタ同士を結合する手法。ここでは詳しい解説は行わない。一般に階層的クラスタリングでもっとも精度がよいと言われている [3] ため使用した。

2.4.3　デンドログラム

階層的クラスタリングの結果を樹形図で表した物をデンドログラムという（図 3）。近い距離で結合されたクラスタ同士は図の下方で繋がるように描かれ、遠い距離で結合されたクラスタ同士は上方で繋がるように描かれる。図 3 では、「A と B」「D と E」の距離が非常に近く（同一作業指数が大きく）、「AB のクラスタ」「CDE のクラスタ」「FGH のクラスタ」間は距離が大きい（同一作業指数が小さい）ことを表している。

2.4.4　クラスタの生成

階層的クラスタリングの結果生成されたデンドログラムを任意の高さの水平線で切断することで、任意の大きさのクラスタを得ることができる。図 3 で閾値 t の水平線を用いて切断すると、「ファイル A,B」「ファイル C,D,E」「ファイル F,G」「ファイル H」というクラスタが生成される。これらのクラスタから大きさ 1 のクラスタをのぞいた、「AB」「CDE」「FG」の三つのクラスタが三つの仮想ディレクトリとして提示される。

3.　プロトタイプシステム

実験に先立ち、提案手法で仮想ディレクトリを生成し Web ブラウザ上に提示するシステムを作成した。

前処理として Perl プログラムで Samba のログからアプリケーションディレクトリへアクセスを削除し、Java プログラムでファイル使用時間の推定（2.1 節）と各尺度による同一作業指数を計算する（2.2 節）。

ユーザは図 4 に示される Web ベースのインターフェースにアクセスする。インターフェース上で尺度の選択や、デンドログラムを見ながら閾値 t の選択を行うことができる。尺度や閾値 t の選択を行うと、システムは前処理で算出された各尺度による同一作業指数を用いてクラスタリングを行い、サイズ 1 のクラスタを除去したクラスタ群を仮想ディレクトリとして列挙する（図の右フレーム：プロトタイプのため図中では仮想ディレクトリがクラスタと表記されている）。Web インターフェース部は Perl で記述された CGI で実現されている。階層的クラスタリングと閾値 t によるデンドログラムの切断は、Perl から R [6] を外部プログラムとして呼び出すことで実現している。

4.　実　　験

4.1　実験データ

Samba サーバーに個人データが置かれている環境において、同時に複数の作業を行うことが多いユーザ X の 2008/4/8〜2008/11/6 の約 7ヵ月間のアクセスログと、一つの作業に専念することが多いユーザ Y の 2008/8/8〜2008/12/24 の約 5ヵ月間のアクセスログを用いた。これ以降、同時に複数の作業を行うユーザのことを「並行作業型」と呼び、一つの作業に専念するユーザのことを「単一作業型」と呼ぶ。

アプリケーションが置かれているディレクトリへのアクセスログはあらかじめ除去し、ユーザが作成したファイル群へのアクセスログのみを用いた。そのユーザが過去に用いたファイル群の中で、同一作業で用いられたが二つのディレクトリにファイルが分散している物を評価対象とした（表 1,2）。表内の数字はそのディレクトリに存在する正解ファイル数を表している。

4.2　評 価 手 法

提案手法を実装したシステムが、それぞれの評価セットの両方のディレクトリのファイルを含む仮想ディレクトリを生成できたかについて評価を行う。

評価セットを含まない仮想ディレクトリについては無視する。生成された仮想ディレクトリの中で、実際の文書やソースコードと言った正解ファイル数と、明らかに関係がない別の作業に属する不正解ファイルの数に注目する。テンポラリファイルと

表 1　評価セット（ユーザ X）

	ディレクトリ 1	ディレクトリ 2
セット S_1	課題のプログラム (42)	課題のレポート (7)
セット S_2	PowerPoint(1)	素材ファイル (5)
セット S_3	課題のプログラム (11)	課題のレポート (4)
セット S_4	PowerPoint(1)	素材ファイル (1)

表 2　評価セット（ユーザ Y）

	ディレクトリ 1	ディレクトリ 2
セット S_5	データファイル (6)	設定ファイル (1)
セット S_6	自作 CGI 用 HTML(7)	自作 CGI マニュアル (1)

いった無関係ではないが重要ではないファイルについては無視する。評価セットの片方のディレクトリのファイルのみを含む仮想ディレクトリしか生成できない場合は、発見をなしとする。しかし、セット S_4 のような極端に小さな評価セットの場合を考慮して、「ディレクトリ 1 から正解ファイル」「ディレクトリ 2 から正解ファイルのテンポラリファイル（.bak など）」についても評価を行う（この場合正解ファイル数 1 と評価される）。また、両ディレクトリのファイルを含む仮想ディレクトリが複数できた場合は、一番正解ファイルが多い仮想ディレクトリを結果に利用する。

4.3 閾値 t の設定

仮想ディレクトリのサイズを左右する閾値 t はユーザごとに設定を行い、クラスタリング手法と尺度の組み合わせごとに適切な値に設定する。つまり、表内の 1 つのセルに記述される S_1, S_2, S_3, S_4（あるいは S_5, S_6）の間では閾値 t は同一である。これは、ユーザは探したいファイルごとに閾値 t を選択して仮想ディレクトリを作成し直すといった使い方は好まず、適切な閾値 t で仮想ディレクトリが生成されたらどんなファイルもその仮想ディレクトリ上で探す、と考えたからである。

閾値 t はできるだけ大きな仮想ディレクトリを生成するように選択するが、その際ゴミの数が過剰にならないように「不正解ファイルの数が正解ファイルの数を超えるセットが存在しない」「不正解ファイルが 10 を超えるセットが存在しない」という制限を付ける。例えばユーザ X の「群平均法・P」の組み合わせの場合、閾値 t をあげて大きな仮想ディレクトリを生成させるとセット S_1 の正解ファイルは増えるがセット S_4 の不正解ファイル数が正解ファイル数以上になってしまう。そのため、閾値 t をこれ以上上げることができない。

4.4 表の見方

各評価セットに対する結果を表 3,4 に示す。○は正解ファイル数を示し、×は不正解ファイル数を示している。順位はそれぞれの評価セットの正解ファイル数がクラスタリング手法 4 種×尺度 4 種＝ 16 通りの中で何位であるかを示している。正解ファイル数が同一のものは同順位とする。また、正解ファイル数 0 の場合が多い評価セットでは 0 が高順位になってしまう可能性があるので、正解ファイル数 0 は必ず最下位の 16 位としている。その下の行には順位を平均した平均順位を表示している。各評価セットについて議論する場合は正解ファイル数を用いて比較を行い、手法同士の比較を行う場合は平均順位を用いて比較を行うものとする。

4.5 実験結果の傾向について

全ての評価セットについて、少なくとも 2 正解ファイル以上を発見している手法が必ず存在する。これにより、今回提案した手法が異なるディレクトリに分散したファイルを発見できることを確認できた。

次に同一作業指数算出の違いに着目してみると、共起間隔 D を用いた場合正解ファイルを発見できずに「なし」と表記されてる場合が多いことが分かる。これは、共起間隔 D が長期にわたり重複が出現した方が同一作業の可能性が高いと判定するが、実際には同一作業のファイル群でもそれほど長期間に渡った重複が起こりにくいためと考えられる。表 3 の平均順位に着目すると、単連結法では T が最良で、完全連結法では P が最良で、群平均法では C、ウォード法では T,P が最良であり、クラスタリング手法に寄らず最良な尺度は存在しない。表 4 では T と P が C,D よりも良い評価である。

表 3　実験結果（ユーザ X）

	共起時間 T	共起回数 C	使用開始の類似度 P	共起間隔 D
単連結法	S_1:○ 14 × 7 S_2:○ 6 S_3:○ 2 S_4:○ 2 順位 : 1/1/4/1 平均順位:1.75	S_1:○ 11 × 5 S_2:○ 5 S_3:なし S_4:○ 2 順位:2/9/16/1 平均順位:7.0	S_1:○ 9 × 6 S_2:○ 2 S_3:なし S_4:○ 1 順位:6/13/16/6 平均順位:10.25	S_1:○ 11 × 6 S_2:なし S_3:なし S_4:なし 順位:2/16/16/16 平均順位:12.5
完全連結法	S_1:なし S_2:○ 6 S_3:○ 2 S_4:なし 順位:6/1/4/16 平均順位:9.25	S_1:○ 3 S_2:○ 5 S_3:なし S_4:なし 順位:9/9/16/16 平均順位:12.5	S_1:○ 3 S_2:○ 6 S_3:○ 3 S_4:○ 1 × 1 順位:9/1/1/6 平均順位:4.25	S_1:なし S_2:○ 2 S_3:なし S_4:なし 順位:16/13/16/16 平均順位:15.25
群平均法	S_1:○ 3 S_2:○ 6 S_3:○ 2 S_4:なし 順位:9/1/4/16 平均順位:7.5	S_1:○ 11 × 5 S_2:○ 6 S_3:○ 2 S_4:○ 2 順位:2/1/4/1 平均順位:2.0	S_1:○ 2 S_2:○ 6 S_3:○ 3 S_4:○ 1 × 1 順位:14/1/1/6 平均順位:5.5	S_1:○ 11 × 7 S_2:○ 4 S_3:なし S_4:なし 順位:2/12/16/16 平均順位:11.5
ウォード法	S_1:○ 3 S_2:○ 6 S_3:○ 2 S_4:○ 2 順位:9/1/4/1 平均順位:3.75	S_1:○ 4 S_2:○ 5 S_3:なし S_4:○ 2 順位:8/9/16/1 平均順位:8.5	S_1:○ 5 S_2:○ 6 S_3:○ 3 S_4:○ 1 順位:7/1/1/6 平均順位:3.75	S_1:○ 3 × 1 S_2:なし S_3:なし S_4:なし 順位:9/16/16/16 平均順位:14.25

表 4　実験結果（ユーザ Y）

	共起時間 T	共起回数 C	使用開始の類似度 P	共起間隔 D
単連結法	S_5:○ 7 × 1 S_6:○ 8 順位:1/1 平均順位:1.0	S_5:○ 7 × 1 S_6:○ 6 順位:1/6 平均順位:3.5	S_5:○ 7 × 2 S_6:○ 8 順位:1/1 平均順位:1.0	S_5:なし S_6:○ 6 順位:16/6 平均順位:11.0
完全連結法	S_5:なし S_6:○ 6 順位:16/6 平均順位:11.0	S_5:なし S_6:なし 順位:16/16 平均順位:16.0	S_5:なし S_6:○ 6 順位:16/6 平均順位:11.0	S_5:なし S_6:なし 順位:16/16 平均順位:16.0
群平均法	S_5:○ 7 × 2 S_6:○ 8 順位:1/1 平均順位:1.0	S_5:○ 6 × 1 S_6:○ 6 順位:7/6 平均順位:6.5	S_5:○ 7 × 2 S_6:○ 8 順位:1/1 平均順位:1.0	S_5:なし S_6:○ 6 順位:16/6 平均順位:11.0
ウォード法	S_5:○ 7 × 1 S_6:○ 6 順位:1/6 平均順位:3.5	S_5:なし S_6:○ 6 順位:16/6 平均順位:11.0	S_5:なし S_6:○ 7 順位:16/5 平均順位:10.5	S_5:なし S_6:○ 6 順位:16/6 平均順位:11.0

クラスタリング手法に着目した場合、ユーザ Y では完全連結法が全体的に悪くなっているが、ユーザ X では尺度との組み合わせで変化する。単連結法は大きなクラスタができやすく、ユーザ X の評価セット S_1 では正解ファイルを発見しているが、反面不適切な物を結合しやすく不正解ファイルも含みやすいことを確認した。完全連結法はゴミを含まない小さなクラスタを作る傾向があり、不正解ファイルを含みにくいが正解ファイル発見数も少ない。

4.6 ユーザのファイル使用方法の違いについて

並行作業型のユーザ X ではクラスタリング手法や同一作業指数の定義方法により大きく結果が異なるが、単一作業型のユーザ Y ではクラスタリング方法・同一作業指数を変更しても結果の変化が比較的小さい。特に単連結法と群平均法を比較した場合、ユーザ X では尺度 T,C,P,D 全ての組み合わせで結果が変化しているが、ユーザ Y では共起回数 C のみ正解ファイル数が 1 異なるが T,P,D 全てで結果が同一になっている。並行作業型の場合、ゴミを入れやすい／入れにくいクラスタリング手法の選定やファイルアクセスの重複のどの要素に着目するかの選定で別々の作業の重複による悪影響の度合いが大きく変わるが、単一作業型の場合、別々の作業の重複とその悪影響が無いために手法を変えても変化が少ないのだと考えられる。

4.7 最適な組み合わせについて

表 3・表 4 両方で平均順位が一番高い組み合わせを調べると、「単連結法・T」であることがわかる。これにより、並行作業型ユーザと単一作業型ユーザ両方で有効な手法は「単連結法・T」であると判明した。

5. 関 連 研 究

Gifford らによる Semantic File Systems [7] では、各ファイルから属性を抽出し、その属性を用いて管理をするファイルシステムを提案している。しかし、属性を抽出するためにファイルタイプごとに属性抽出プログラムを書く必要があり、ファイル形式に依存しない本研究と異なる。

暦本による Time Machine Computing [8] では、ファイルは全てデスクトップに置かれ時間とともに消えていくシステムを提案している。ファイルを探したい場合は、過去のデスクトップに遡ってそのときデスクトップに置かれていたファイル群と一緒に発見できる。時間の概念を利用している点では本研究と近いが、ユーザが明示的にファイルをデスクトップに置く必要がある点と、本研究のディレクトリのような明確な分類を行わない点が異なっている。

大澤らによる俺デスク [9] では、ブラウザや Word の動作を記録し、その操作履歴をタイムライン表示することでユーザが「Word で企画書を作成した時に参照していた Web ページ」といった物を検索できるようにしている。イベントの共起をタイムラインで表示するが、本研究のように自動で共起したものを集約する機能がない点が異なる。

Soules らによる Connections [10] は、FRIDAL と同様のアプローチであるが、「N 秒以内にアクセスがあったファイル同士を関連付ける」といった FRIDAL とは異なる尺度を提案している。また、井ノ口らの研究 [11] では、ファイルアクセスの順序やアクセスの密度に着目した尺度を提案している。FRIDAL の尺度が仮想ディレクトリのために提案された訳ではないが本研究に有効であったように、これらの尺度も同様に本研究に応用可能であると考えられる。

6. まとめと今後の課題

提案した手法で複数のディレクトリに分散してしまった同一作業で使用されたファイル群を発見できることを確認した。また、ファイルアクセスの重複から同一作業指数を計算する尺度として「共起時間 T」を使用し、クラスタリング手法として「単連結法」を使用した組み合わせが並行作業型のユーザと単一作業型のユーザで共に有効であることが判明した。

今後の課題として、FRIDAL 以外のファイルアクセスの重複を評価する尺度を利用した場合の評価、複数の尺度を組み合わせたときの評価、今回よい結果を出した共起時間 T の改良、閾値 t の自動選択といった事が考えられる。

謝辞 本研究の一部は，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST，および文部科学省科学研究費補助金特定領域研究 (#19024028) の助成により行なわれた．

文　献

[1] 渡部徹太郎, 小林隆志, 横田治夫. キーワード非含有ファイルを検索可能とするファイル間関連度を用いた検索手法の評価, 2008. 第 19 回データ工学ワークショップ (DEWS2008).

[2] 小田切健一, 渡辺陽介, 横田治夫. アクセス履歴に基づくファイル間関連度を用いたデスクトップ情報管理ツールの開発 (poster presentation). 信学技報, 第 108 巻 of *DE2008-72*, p. 49, 12 月 2008.

[3] 新納浩幸. R で学ぶクラスタ解析. オーム社, 2007.

[4] *Samba*. http://us3.samba.org/samba/.

[5] 渡部徹太郎, 小林隆志, 横田治夫. ファイル検索におけるアクセスログから抽出した関連度の利用 (情報抽出, 夏のデータベースワークショップ 2007(データ工学, 一般)). 電子情報通信学会技術研究報告. DE, データ工学, Vol. 107, No. 131, pp. 503–508, 20070625.

[6] R Development Core Team (2008). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

[7] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, James W. O'Toole. *Semantic File Systems*, 1991. 13th ACM Symposium on Operating Systems Principles.

[8] Jun Rekimoto. Time-machine computing: a time-centric approach for the information environment. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pp. 45–54, New York, NY, USA, 1999. ACM.

[9] 大澤亮, 高汐一紀, 徳田英幸. 俺デスク:ユーザ操作履歴に基づく情報想起支援ツール, 2006. 第 47 回プログラミング・シンポジウム報告集，pp.15-21.

[10] Craig A. N. Soules and Gregory R. Ganger. Connections: using context to enhance file search. *SIGOPS Oper. Syst. Rev.*, Vol. 39, No. 5, pp. 119–132, 2005.

[11] 吉川 正俊井ノ口 伸人. アクセス履歴を考慮したファイル間の関連度を用いたデスクトップ検索 (履歴応用, 夏のデータベースワークショップ dbws 2006). 情報処理学会研究報告. データベース・システム研究会報告, Vol. 2006, No. 77, pp. 141–146, 20060712.