Vol.2015-DPS-162 No.11
Vol.2015-CSEC-68 No.11
2015/3/5

業務指標の閾値算出手法を用いたプロセスディスカバリー業務分析

安部　麻里[1,a)]　工藤　道治[1,b)]

概要：従来，ログからプロセスを抽出するプロセスディスカバリー (Process Discovery) においては，ログの量が増えるに従い抽出されるプロセスは複雑なもの (いわゆるスパゲッティプロセス) になり，業務の本質となる構造が分析者にとって理解しづらく，そのために業務分析が困難になっていた．本論文では，業務分析に利用される業務指標 (KPI) に着目し，スパゲッティプロセスを紐解くための分析の切り口となる業務指標の閾値を，プロセスの編集距離に基づき自動的に算出する手法を提案する．閾値が自動的に算出されることで，従来業務分析の障害になっていた試行錯誤により閾値を探索する手間が軽減され，プロセスディスカバリーを用いた業務分析を支援することが可能となる．

キーワード：プロセスディスカバリー，業務分析，ビジネスプロセス管理

1. はじめに

プロセスマイニングは，様々なアプリケーション領域におけるビジネスプロセスを改善する手段として広く認知されつつある．プロセスマイニングの中でも，特にプロセス実行エンジンやアプリケーションの実装から出力されるログからプロセスを抽出するプロセスディスカバリー (Process Discovery) は，単にビジネスの現場における人の活動を可視化する手段にとどまらず，業務改善の施策内容を決定するための重要な分析手段として実践的に使われている [1], [2].

プロセスディスカバリーでは，ログの量が増えるに従い抽出されるプロセスは複雑なもの (いわゆるスパゲッティプロセス) になり，プロセスの分析が困難になることが問題点として指摘されてきた [3]．プロセスディスカバリーにおいては，複雑なプロセスを人が理解できる情報量にいかに絞り，有益な情報を得るかが重要であり，同時にプロセスディスカバリーが有用な業務分析手段として広く利用されるための課題でもある．

複雑なプロセスを人が見て理解しやすいものにする際の一アプローチとして，プロセスのログを特定の業務指標 (Key Performance Indicator: KPI) に基づきフィルターすることで単純化する方法が考えられる．例えば，オンラインによる商品購買の分析において，分析者は全てのユーザの購買行動を包括したプロセスを見るよりも，商品購買プロセスの開始から終了まで 10 分未満で終わるユーザと 10 分以上かかるユーザとでプロセスを比較することで，改善策を見出すことが考えられる．また，社内業務に遅延が発生した場合，業務の管理者は全ての業務を包括したプロセスを見るよりも，業務の曜日等に着目し遅延の原因を探る方法が考えられる．このような問題に対し，プロセスのログと，プロセスの所要時間など分析軸となる KPI をあわせた分析用のプラットフォームを構築することで，プロセスディスカバリーによる業務分析を支援する研究がなされている [4]

しかしながら，ある KPI に着目してログをフィルタする場合，どのように KPI の閾値を選ぶかは試行錯誤する必要があった．曜日など値の種類が少なく分類可能なデータ型の場合，値の選択は比較的単純であるが，何分以上/以下などといったスカラー値や値の種類数が非常に多い場合，閾値を選ぶ試行錯誤の手間が深刻であり，閾値をどこにおけばプロセスが変化するかわからず分析の障害となっていた．例えば，10 分未満/以上の 2 つのプロセスを比べても差がわからず，1 分刻みでログを分類しては逐一プロセスを抽出するような，試行錯誤的な作業が必要であった．

本論文は，ログから抽出されるプロセスを分析する際，分析の軸となる KPI の閾値をプロセスの編集距離に基づき自動的に算出する手法を提案する．提案手法を以下に要約する．

[1] 日本アイ・ビー・エム（株）　東京基礎研究所
5-6-2 Toyosu, Koto, Tokyo 135-8511, Japan
[a)] maria@jp.ibm.com
[b)] kudo@jp.ibm.com

行番号	イベント型	ユーザID	タイムスタンプ	リクエストURL	入出力パラメータ
1	request	user1	2013-08-20T08:30:33	calc_premium	
2	request	user2	2013-08-20T08:30:33	logon	
3	response	user1	2013-08-20T08:30:34		pageid="calc_premium_page"...
4	request	user3	2013-08-20T08:30:35	logon	
5	response	user2	2013-08-20T08:30:35		pageid="logon_page"...
6	request	user1	2013-08-20T08:30:48	submit_condition	sourcepageid="calc_premium_page", birthday="1965/01/01", gender="male"
7	response	user1	2013-08-20T08:30:49		pageid="insurance_option_page", msg="input options next"
8	response	user3	2013-08-20T08:30:50		pageid="logon_page"...
9	request	user1	2013-08-20T08:31:50	submit_product	sourcepageid="insurance_option_page, product="life insurance", premium="$50", period="10 years"
10	response	user1	2013-08-20T08:31:51		pageid="simulation_result_page", msg="need physical check-up"
11	request	user1	2013_12-20T08:32:00	save	sourcepageid="simulation_result_page"

図 1　アプリケーションログの例
Fig. 1　An example of application log

(1) KPI の計算式を外部定義として記述し，その外部定義を用いてログを処理し結果をプロセスの実行履歴 (プロセスインスタンス) として保存する．
(2) 業務処理時間，商品名や特定のページの利用回数などといった業務分析に必要な KPI もプロセスインスタンスにあわせて保存する．これにより，KPI とプロセスインスタンスが関連付けられた分析プラットフォームを生成することが可能となる．
(3) KPI に基づきプロセスインスタンスを分割し，分割されたインスタンス群からプロセスモデルを合成する．
(4) 合成されたプロセスモデル間の編集距離を求め，距離の近いものを集約することで，プロセスの変化が大きい KPI の閾値を算出する．

これにより，プロセスディスカバリーを用いた業務分析において，試行錯誤による閾値を探索する手間が軽減される．

以下，第 2 章でアプリケーションの実装から出力されるログを対象としたプロセスディスカバリーについて述べ，第 3 章で閾値を算出するアルゴリズムについて述べる．最後に第 4 章で本論文をまとめる．

2. アプリケーションの実装から出力されるログを対象としたプロセスディスカバリー

2.1 タスクとプロセスインスタンスの抽出

プロセスディスカバリーを用いた業務分析では，ログの分析が可能がどうか事前に精査をすることが必要である．ログは，フォーマットにより構造/非構造か，さらにログの出力元によりプロセス実行エンジンによるものか，アプリケーションの実装によるものかに分類される．場合によっては，社員番号や所属組織といったプロセス実行者の情報や，商品情報といったプロセスのメタ情報が別途与えられる場合もある．

フォーマットが非構造の場合，プロセスディスカバリーの前に自然言語処理などを用いたログの解析が必要となる．空白文字やカンマなどある程度ログが構造化されている場合は，ログの出力元がプロセス実行エンジンか，アプリケーションの実装かにより，分析アプローチと得られる分析結果が異なってくる．出力元がプロセス実行エンジンのログを対象としたプロセスディスカバリー手法と，アプリケーションの実装から出力されるログを対象としたプロセスディスカバリー手法との本質的な違いは，前者が 1 作業単位 (タスク) や一連のタスクの実行履歴からなる 1 プロセスインスタンス単位が識別子により明確に区分できることが多いのに対し，後者がログのセマンティクスに合わせて 1 タスクや 1 プロセスインスタンスを定義し切り分ける処理が必要となる点にある．また，後者はタスクやプロセスを切り出す処理が必要である一方で，実装レベルの情報であるため，ユーザインタフェース上での操作など抽象化されていない情報が含まれる場合が多いことも違いとして挙げられる．

例えば，プロセス実行エンジンのログ出力の設定によっては，プロセスの状態が変化したことで起こるタスクの遷移時間が記録されるものの，人が実際に作業を行った時間など時間に関する詳細な情報は記録されないことが多い．具体例として，タスクを割り当てられた人が作業を開始した後一旦入力データを保存し，昼休みを取る等一時的に作業を中断し，その後作業を再開する場合や，週末をまたい

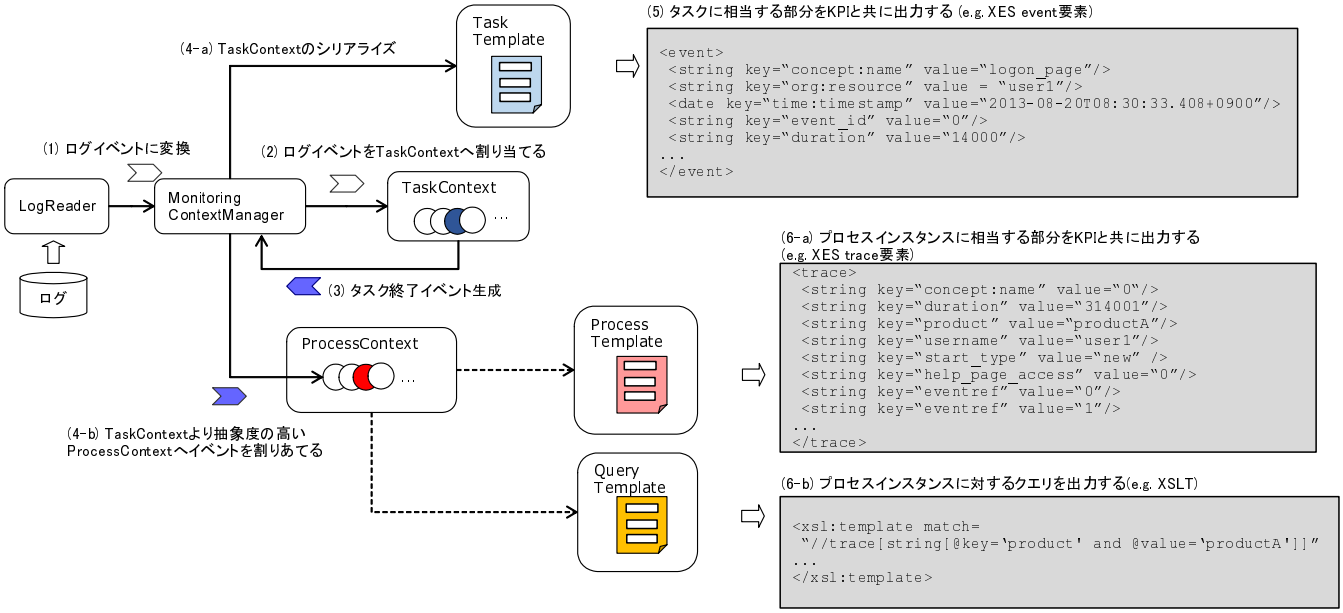


図 2 提案手法によるイベント処理と業務分析のためのプラットフォームを生成する流れ
Fig. 2 Flow of processing events and generating analytics platform

で作業を行った場合などは，実作業時間が同じでもタスクの開始から終了の期間が長く算出される．業務分析を行う上では，より詳細な情報が記録されているほうが，得られる情報量が多いという点で望ましい．本論文では，アプリケーションの実装 (例えば Web アプリケーションサーバ等) から出力されるログを対象とした，プロセスディスカバリーを提案する．

プロセスインスタンスを抽出する例を**図 1** を用いて説明する．実際のログはテキストファイルなどに空白文字やカンマで区切られているものが多いが，説明の便宜上表を用いる．ユーザが Web ページにアクセスすると，アプリケーションサーバはリクエストに応じて一行出力する．また，サーバがレスポンスを返す時も同様に一行出力する．リクエストかレスポンスかによりイベント型が決まり，同時にユーザ ID，タイムスタンプ，リクエスト URL，入出力パラメータなど，各項目が出力される．

プロセスインスタンスを切り出すためには，プロセスインスタンスの開始条件及び終了条件を決める必要がある．この例では，特定のユーザによるログオンページ (“logon_page”) へのアクセスが開始条件，結果ページ (“simulation_result_page”) へのアクセスが終了条件となる．プロセスインスタンスは，特定のユーザがアクセスするページのリストにより構成される．従って，タスクの識別子はユーザ ID とページ ID の組み合わせ，プロセスインスタンスの識別子はユーザ ID となる．ページ ID は入出力パラメータのパラメータ名から抽出する．

ユーザ ID が “user1” である行に着目した場合，行番号 1, 3, 6, 7, 9, 10, 11 が 1 プロセスインスタンスの基となる情報に相当する．イベント型が “response” の “pageid” の値と，イベント型が “request” の “soucepageid’ の値が一致することにより，特定のページの処理時間が確定する．例えば，行番号 3 と 6 の “calc_premium_page” という値により，“user1” が “calc_premium_page” を処理した時間 (14 秒) が計算出来る．

他のアプリケーション例として稟議書の承認プロセスが挙げられる．稟議書の承認プロセスでは，一つの稟議書に対して複数のユーザが承認していくため，タスクの識別子は稟議書 ID とユーザ ID の組み合わせ，プロセスインスタンスの識別子は稟議書 ID になる場合などがある．この様に，ログのセマンティクスにより識別子は様々である．

プロセスディスカバリーに必要な情報の取得方法はログのセマンティクスに依存する．迅速な業務分析を行うためには，ログの解析器にこれらのセマンティクスを外部定義として与え，解析器の実装を最小限に抑えることが求められる．

2.2 プロセスディスカバリーエンジン

ビジネスモニタリングは，ビジネスプロセス実行エンジンなどから出力されるビジネスイベントを入力とし，業務状態をリアルタイムに監視しダッシュボードなどに表示するフレームワークである．ビジネスモニタリングでは，ビジネスイベントから KPI を測定するために必要な情報を記

述するための，モニタリングコンテキストという外部定義が提案されている [5], [6], [7]．モニタリングコンテキストには，KPI の計算に必要なイベント型，イベントを処理すべきかどうか判断するためのキーとなる識別子，イベント中の変数値をコンテキストの状態変数として保持するマッピング関数，KPI の計算式などを記述することが可能である．ビジネスモニタリングの実行エンジンは，モニタリングコンテキストを入力として，様々なビジネスイベントから KPI を計算することが可能である．

モニタリングコンテキストはビジネスイベントから KPI を測定するために十分な表現力がある一方で，本来モニタリングを目的として設計されているため，その実行エンジンにはプロセスインスタンスを出力する機能が備わっていない．筆者らは，モニタリングコンテキストを入力とし，KPI を計算すると同時にプロセスインスタンスを出力するディスカバリーエンジンを提案した [8]．以下に提案プラットフォームを説明する．

図 2 にイベント処理と業務分析のためのプラットフォームを生成する流れを示す．まずはじめに，テキストファイルやデータベースに保存されているログを取得しイベントに変換する (図 2(1))．図 1 の例では，1 行が 1 イベントに相当し 1 つのタイムスタンプを持つ．MonitoringContextManager は，イベントを処理するクラスである MonitoringContext のインスタンスへイベントを割り当てる (図 2(2))．もし割り当てる MonitoringContext が存在しなかった場合，開始条件に合致する MonitoringContext をインスタンス化する．MonitoringContextManager は MonitoringContext のインスタンスのライフサイクルを管理する．

図 2 では，TaskContext 及び ProcessContext が MonitoringContext の実装クラスである．MonitoringContext はモニタリングの状態を保持するクラスであり，処理すべきイベントの識別子，イベントの型 (inbound event)，処理した結果に基づき生成するイベントの型 (outbound event)，KPI の計算方法，監視の開始・終了条件が定義されている．TaskContext はタスクが終了する際に，ログ 1 行に相当する最初のイベントより抽象度の高いタスクイベントを生成し，MonitoringContextManager に処理を依頼する (図 2(3))．

MonitoringContextManager は，TaskContext からイベントを受け取ると，イベントの生成元の TaskContext を終了させると同時に，内部状態をシリアライズする (図 2(4-a))．同時に，受け取ったイベントを ProcessContext に割り当てる (図 2(4-b))．

提案手法では，MonitoringContext の内部状態をシリアライズするために，業務分析のプラットフォームにあわせたテンプレートを用いる．テンプレートは，MonitoringContext の内部状態に相当する部分が変数として定義されており，MonitoringContext のシリアライズ処理により値が決まる．

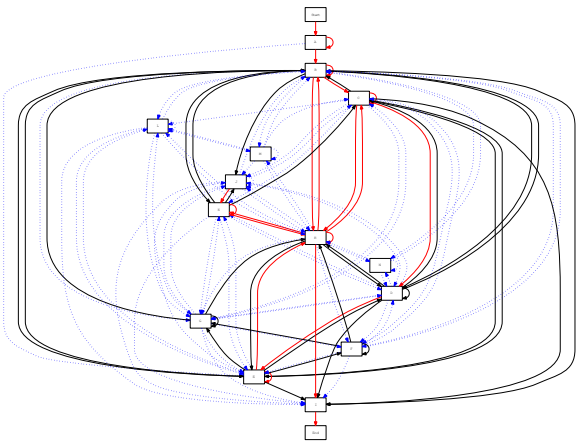


図 3 プロセスモデルの抽出
Fig. 3 Result of extracting process model

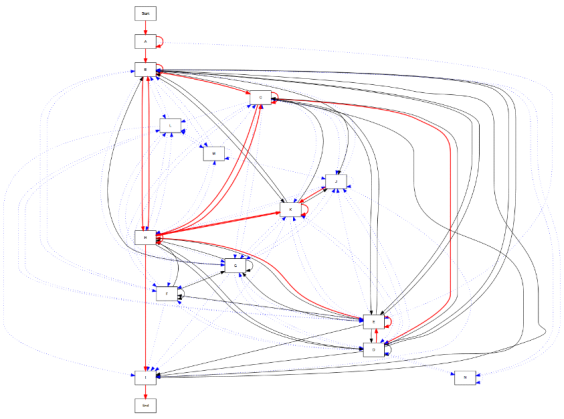
プロセスマイニングの分野で標準である，イベントログを記録するための XML 記述言語である XES[9] や，XML に対するクエリ言語 (XLST など) のテンプレートを介して，業務分析のためのプラットフォームが生成される．図 2(5) は XES の event 要素であり 1 タスクに相当する．event 要素には，ページ名 (“logon_page”)，ユーザ名 (“username”) のほかにも，タスク処理に要した時間 (“duration”) 等が KPI として記載されている．

イベント処理に伴い ProcessContext がプロセスの終了状態になった場合，TaskContext と同様に ProcessContext もテンプレートを用いてシリアライズされる．プロセスインスタンスの状態は XES の trace 要素として保存される (図 2(6-a))．この例では，プロセスインスタンスに相当する KPI として，プロセスの所要時間 (“duration”)，プロセスを実行するに当たりアプリケーションの利用方法が書いてあるページを参照した回数 (“help_page_access”)，プロセスが新しく開始されたものか中断から復帰されたものかを判断する属性 (“start_type”)，商品名 (“product”) 等が記載されている．“eventref” により，event 要素と関連付けることが可能である．trace 要素を生成すると同時に，プロセスインスタンスに対するクエリも出力することが可能である (図 2(6-b))．生成されたクエリは，KPI によるプロセスインスタンスの絞込みに利用する．

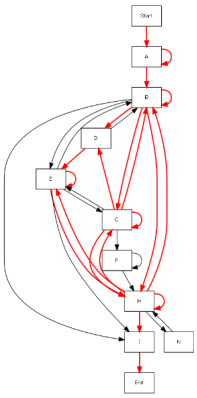
2.3 プロセスモデル合成

分析プラットフォームが生成された後，我々が開発した Process Discovery ツール [1] を用いてプロセスを合成した結果を**図 3** 及び**図 4** に示す．図 3 及び図 4 は，導入の初期段階にある実際のアプリケーションのログを対象とした実験結果である [8]．

Vol.2015-DPS-162 No.11
Vol.2015-CSEC-68 No.11
2015/3/5



(1) 3回未満



(2) 3回以上

図 4　KPI により区別されたプロセスモデル
Fig. 4　Result of extracting process model with KPI threshold

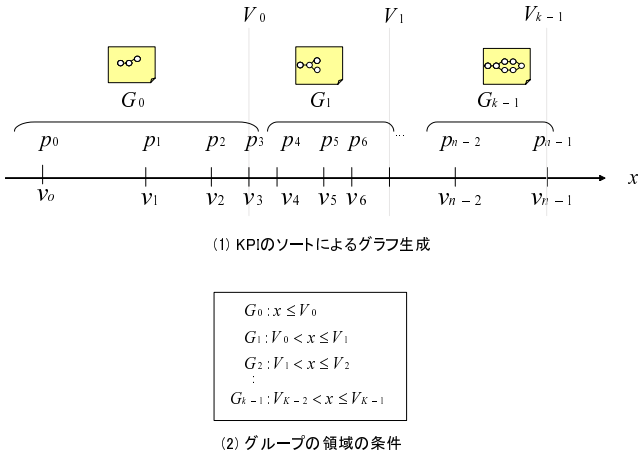
図 3 は，プロセスの開始条件/終了条件に合致するプロセスインスタンスから抽出されたプロセスモデルである．タスクは長方形で表示されており，一番上の長方形がプロセスの開始，一番下の長方形が終了となっている．長方形間を結ぶエッジはタスクの遷移を表す．遷移回数の少ないエッジは点線で示されている．ページ間の遷移に着目すると，回数の多少はあるもののほぼ全てのページ間の組み合わせに対して遷移があり，ページ間を右往左往している様子が分かる．

次に，生成されたクエリを図 3 のモデルに適用し，KPI によりプロセスモデルを区別した例を図 4 に示す．図 4(1) は，プロセスの開始から終了まで，アプリケーションの利用方法が書いてあるページを参照した回数が 3 回未満のプロセスインスタンスから合成されたプロセスモデル，図 4(2) は 3 回以上のプロセスインスタンスから合成されたプロセスモデルである．図 4(1) は図 3 と同様，ページ間を右往左往しているのに対し，図 4(2) は開始から終了まで，比較的試行錯誤せずプロセスを終えている様子が長方形間を結ぶリンクの数などから分かる．これらの結果から，アプリケーションの利用方法が書いてあるページを閲覧することは，アプリケーションの導入の初期段階において操作に不慣れなユーザがプロセスを実行するのに有効であり，3 回以上の利用がプロセスの分かれ目となることがわかった．

この様なプロセスを区別する閾値は，従来試行錯誤によりツールの利用者が特定する必要があった．第 3 章では，閾値を自動的に算出するアルゴリズムについて説明する．

Algorithm 1 プロセスインスタンスからのグラフ生成

```
1: プロセスインスタンスを p_0, ..., p_{n-1} ∈ P とする．
2: P に付与される KPI の種類を x_0, ..., x_{m-1} ∈ X とする
3: p_i に付与されている x_j の値を v_ij とする
4: if x_j が数値で順序に意味がある場合 then
5:     v_ij に基づきプロセスインスタンスをソートし k 分割する
6: else
7:     if x_j 数値でない or 数値であるが順序に意味がない場合 then
8:         k を値の種類数とし P を k 分割する
9:     end if
10: end if
11: 分割されたプロセスインスタンスを用いて G_0, ...G_{k-1} を生成
12: グラフごとに領域条件を保存
```



(1) KPIのソートによるグラフ生成

(2) グループの領域の条件

図 5　KPI の値によるプロセスインスタンスのソートとグラフの生成
Fig. 5　Graph genration based on sorted process instances

3. プロセスモデル間の編集距離に基づく KPI 閾値算出

本章では，KPI の付与されているプロセスインスタンスから，プロセスモデル間の編集距離に基づき KPI の閾値を求めるアルゴリズムについて説明する．プロセスインスタンスは図 2 (6-a) の trace 要素に相当し，KPI がそれぞれ付与されているものとする．プロセスインスタンスから KPI に基づき初期状態のプロセスモデルを生成するアルゴリズムを Algorithm 1 に示す．また，**図 5** に特定の KPI x

に着目し，プロセスインスタンス p_i をソートした例を示す．グラフ G は分割されたプロセスインスタンスから合成されるプロセスモデルに相当する．

まず，プロセスインスタンス p_i を KPI x に基づきソートする．x がプロセスの処理時間の様に数値であり順序に意味がある場合は，数値の低いほうから順に p_i を並べ，プロセスインスタンスを k 分割する．もし x が曜日など数値でない場合や，数値であるが順序に意味がない場合 (商品 A= 0, 商品 B= 1 等) は，値の種類によりプロセスインスタンスを分割する．最後に，k 分割されたプロセスインスタンスからそれぞれグラフ G を合成し，領域の条件を覚えておく (図 5(2))．

次に，生成された任意の 2 つのグラフに対し編集距離 [10] を求め類似するグラフを集約する (Algorithm 2, **図 6**)．まず，任意のグラフの組みあわせに対するテーブルを作成し，編集距離を求める．次に，編集距離の最短の 2 グラフを 1 つのグラフに集約し，新たなグラフとする．図 6 では，G_0 と G_2 の距離が最短であり，これらから 1 つのグループ $\{G_0, G_2\}$ を作成する．そして，G_0 及び G_2 のエントリを消し，新たに $\{G_0, G_2\}$ というエントリを作成し，編集距離を求める．以上のステップを，全ての距離が一定の距離以上になるまで繰り返す．最後に，KPI が連続値の場合，編集距離テーブルで最終的に同じグループになった領域をマージし，新たな領域の条件とする．

以上により，類似するグラフとその領域条件が求まり KPI の閾値となる．これにより，従来試行錯誤によりツールの利用者が閾値を特定する手間が軽減され，迅速な業務分析に役立てることが可能となる．

4. まとめ

本論文では，アプリケーションの実装から出力されるログを対象にしたプロセスディスカバリー手法を提案した．これにより，ログからプロセスを抽出すると同時に KPI を計算し，業務分析のためのプラットフォームを生成することが可能となった．さらに，KPI に基づきプロセスインスタンスを分割してプロセスモデルを合成し，プロセス間の編集距離を求め距離の近いグループを集約することで，プロセスモデルの変化が大きい KPI の閾値を算出する手法を提案した．今後の課題としては，閾値算出手法を実際のアプリケーションに適用し，有用性を検証していくことが挙げられる．複数の種類の KPI を組み合わせた時の分析手法に関しても，さらなる検討を行っていきたい．

Algorithm 2 グラフ編集距離を用いたグラフの集約

1: 任意のグラフの組み合わせに対して距離 d を算出しテーブルに入力
2: **while** 全ての距離が一定の距離以上になるまで **do**
3: 最短距離の 2 グラフを 1 つに集約しテーブルを更新
4: 新たなグループに対してグラフを生成
5: 編集距離を再計算しテーブルを更新
6: **end while**

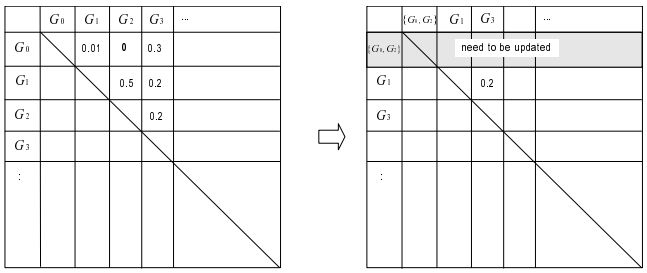


図 6 編集距離テーブルを用いたグラフの集約

Fig. 6 Merging of graphs using a table of graph edit distance entries

参考文献

[1] Kudo, M., Ishida, A. and Sato, N.: Businesss Process Discovery by using Process Skeletonization, *International Symposium on Data-Driven Process Discovery and Analysis* (2013).
[2] Kudo, M.: Operational Work Pattern Discovery Based On Human Behavior Analysis, *Service Research and Innovation Institute Global Conference* (2014).
[3] W. M. P. van der Aalst: *Process Mining*, Springer (2011).
[4] W.M.P. van der Aalst: Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining, *Asia Pacific Conference on Business Process Management (AP-BPM 2013)*, Lecture Notes in Business Information Processing, Vol. 159, Springer-Verlag, pp. 1–22 (2013).
[5] Chowdhary, P., Bhaskaran, K., Caswell, N., Chang, H., Chao, T., Chen, S., Dikun, M., Lei, H., Jeng, J., Kapoor, S., Lang, C., Mihaila, G., Stanoi, I. and Zeng, L.: Model Driven Development for Business Performance Management, *IBM Systems Journal*, Vol. 45, pp. 735–749 (2006).
[6] Lakshmanan, G. T., Keyser, P. T., Slominski, A. and Curbera, F.: A Business Centric Monitoring Approach for Heterogeneous Service Composites, *2011 IEEE International Conference on Services Computing*, pp. 671–678 (2011).
[7] Momm, C., Gebhart, M. and Abeck, S.: A Model-Driven Approach for Monitoring Business Performance in Web Service Compositions, *Fourth International Conference on Internet and Web Applications and Services*, pp. 343–350 (2009).
[8] Abe, M. and Kudo, M.: Business Monitoring Framework for Process Discovery with Real-Life Logs, *Proceedings of the International Conference on Business Process Management (BPM2014)*, pp. 417–424 (2014).
[9] Christian W. Günther and Eric Verbeek: XES Standard Definition Version 2.0, `http://www.xes-standard.org/xesstandarddefinition` (2014).
[10] Riesen, K., Emmenegger, S. and Bunke, H.: A Novel Software Toolkit for Graph Edit Distance Computation, *9th International Workshop on Graph Based Representations in Pattern Recognition*, pp. 142–151 (2013).