小口研究室　研究紹介　(2024年度)
(お茶の水女子大学理学部情報科学科)

ブロックチェーンベースのカーボンフットプリントデータ管理基盤の構築と自動車部品製造業への応用における評価
（研究担当:堀　遥）

研究背景

- SDGsなどの観点から，サプライチェーン全体でのカーボンフットプリントデータ管理の需要増
- 横断的データ連携基盤の実現を目指す
- 実証実験では自動車部品製造業がテーマ
- システム実現における3つの課題
  - ① 企業同士の連携の信頼性の担保
  - ② 社内外で発生する改ざんの検知
  - ③ カーボンフットプリントは算出は再帰計算

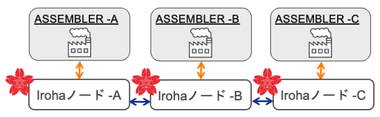
➡ **効率的なデータ検証機能を備えるブロックチェーンベースのシステムを提案**

提案手法

- 定義
  - ✓ **部品木**:構成する子部品らの関係を階層的に表現
  - ✓ CFP(P) : 部品Pの製造企業で発生したCO_2排出量
  - ✓ TotalCFP(P) : 部品Pのカーボンフットプリント

(例)部品Aが部品B,C,D,さらに部品Bが部品E,Fで構成されるとき…
TotalCFP(A) = CFP(A) + TotalCFP(B) + TotalCFP(C) + TotalCFP(D)
TotalCFP(B) = CFP(B) + TotalCFP(E) + TotalCFP(F) …

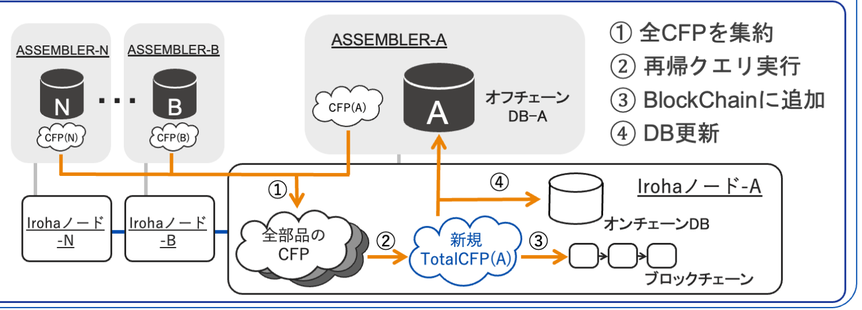
- 全体構成
  - ✓ ブロックチェーンプラットフォーム Hyperledger Irohaで各社を接続
  - ✓ データ検証機能を実装



メインプロセス1 TotalCFP算出プロセス

TotalCFPデータを検証可能な形で保存するプロセス

Smart Contract技術の組み込みコマンドで実装



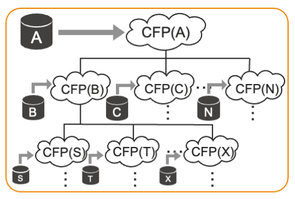
メインプロセス2 TotalCFP検証プロセス

TotalCFPデータの健全性を検証するプロセス

- 検証粒度の異なる2つの検証プロセスを提案

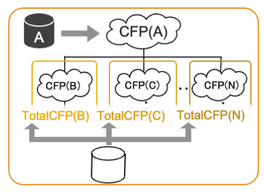
✓ **Fine-grained手法**

②では検証対象の部品を構成する全ての部品で再算出
→ TotalCFP算出プロセスで実装



✓ **Coarse-grained手法**

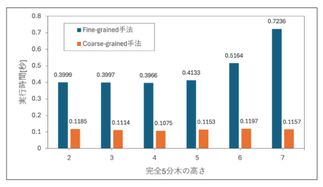
②では部品木で検証対象部品の直下に位置する子部品のみで算出
→ オンチェーンDB上の子部品のTotalCFP値を利用



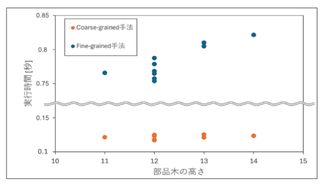
評価

- 制約のもとで多様な部品木を生成
- 部品木に対して2つのデータ検証手法を実行，時間を測定

■ 部品木が完全5分木の場合



■ 総節点数を固定で部品木をランダム生成



✓ Fine-grained:部品木の大きさに対し実行時間が単調増加
✓ Coarse-grained:部品木に関係なく安定に高速

- 結論
  2つのデータ検証手法は
  改ざん検知範囲と速度がトレードオフ

検証手法	Fine-grained	Coarse-grained
改ざん検知範囲	部品木全体	直下の下位部品まで
検証速度の比（最大）	1	1／6

クローズドパターン抽出を用いたインタラクティブなシーケンシャルパターンマイニングの高速化の提案と評価
（研究担当:青柳　結衣）

1.研究背景

- 様々なビッグデータの蓄積に伴い、シーケンシャルパターンマイニング(SPM)が注目
- インタラクティブなSPMが不可欠
  - 最適な閾値はデータセットや分析要求等に依存し、発見までにSPMを複数回実行する必要あり
- 既存研究：生成される頻出シーケンシャルパターンは冗長なものが多数存在し、実行時間が長い

➡ 実行中に生成されるシーケンス数を減少させ、アルゴリズムの高速化を目指す

2.関連研究

KISP[1]　1.指定されたminsupとKB.baseの比較 → 2.新しい候補シーケンス生成 → 3.候補シーケンスのサポート値算出

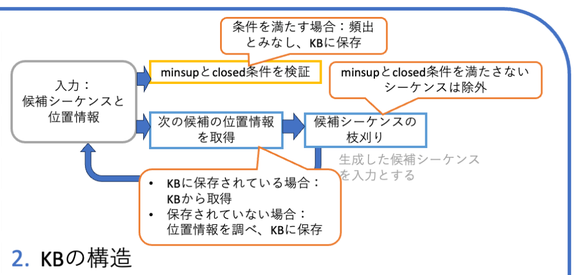
1. 指定されたminsupとKB.base(今までの最小minsup)の比較
   - minsup≥KB.base：データセットにアクセスせず取得可能
2. 新しい候補シーケンスの生成
3. 候補シーケンスのサポート値算出
   - 算出後、KB(知識ベース)に保存

[1]Ming-Yen Lin and Suh-Yin Lee ,"Improving the Efficiency of Interactive Sequential Pattern Mining by Incremental Pattern Discovery" , Proceedings of the 36th Hawaii International Conference on System Sciences - 2003

3.提案手法

指定されたminsupとKB.baseの比較
minsup<KB.base：探索
minsup≥KB.base：KBから取得
→ 新しいクローズド候補シーケンス生成（以下の情報をKBに蓄積：SPの位置情報(ハッシュ構造)）
→ 候補シーケンスのサポート値算出（以下の情報をKBに蓄積：KB.base、FCSP＋サポート値(ツリー構造)）

1. クローズドシーケンスのみを候補シーケンスとして生成
   - クローズドシーケンス：スーパーシーケンスで同じサポート値を持つものが存在しない
   - ◆ (A→B)サポート値4、(A→B→C)サポート値4の場合、クローズドシーケンスである(A→B→C)のみを採用
   - パターン数が減少し、アルゴリズムの高速化に繋がる



2. KBの構造
   - 頻出クローズドシーケンシャルパターン(FCSP: Frequent Closed Sequential Pattern)とサポート値をツリー構造で管理
   - 候補シーケンスの位置情報をハッシュ構造で管理

4.実験結果

増加幅	PrefixSpan(s)	提案手法(s)	提案手法/PrefixSpan	減少幅	PrefixSpan(s)	提案手法(s)	提案手法/PrefixSpan
0.0005	14.86	15.16	102%	0.0005	14.67	27.35	186%
0.0004	18.43	15.29	83%	0.0004	18.44	29.71	161%
0.0003	24.75	16.02	65%	0.0003	24.75	34.41	139%
				0.0002	36.74	40.91	111%
				0.0001	73.98	62.67	85%

1. KBによる実行時間の検証
   - minsupを増加する時と減少する時を検証
     - 増減幅を小さくすると実行回数が増加
   - ✓ 実行回数が増えると提案手法の方が高速
   - ✓ 増加する場合は減少する場合より実行回数が少なくても効果がみられる

2. クローズド考慮による実行時間の検証
   - データセットを変更し提案手法を適用
     - 実行時間と候補シーケンス数を測定
   - ✓ どちらのデータセットもクローズド考慮すると高速
     - ✓ 候補シーケンス数が減少したため
   - ✓ クローズド考慮することによる実行時間への影響はデータセットに依存

実行時間	クローズド考慮あり(s)	クローズド考慮なし(s)	考慮あり/考慮なし
BMSWebView1	532.73	1,154.66	46%
BMSWebView2	423.13	456.90	92%

候補シーケンス数	クローズド考慮あり	クローズド考慮なし	考慮あり/考慮なし
BMSWebView1	85,186,590	195,240,402	44%
BMSWebView2	53,330,915	67,693,468	79%

minsup(値)	60	59	58	57	56	55
パターン1 (s)	43.42	487.45	539.31	600.99	674.84	753.35
パターン2 (s)	7.41	7.76	8.14	8.53	9.00	9.56
メモリサイズ(GB)	1.29	1.36	1.43	1.51	1.61	1.71

3. 候補シーケンスの位置情報の保持
   - パターン1：位置情報の保存あり
     パターン2：位置情報の保存なし
   - ✓ 位置情報を保存しない方が高速

5.まとめと今後の課題

まとめ

- ✓ KBを利用することの有効性を示した
- ✓ クローズドシーケンシャルパターンのみを抽出することで候補シーケンス数が減少し、アルゴリズムの高速化を検証できた

今後の課題

- ✓ 位置情報を保存する構造の検討
- ✓ 他のデータセット(医療電子カルテ等)へ適用し、評価
- ✓ 従来手法との比較

Unixドメインソケットとseccompを用いた強制アクセス制御実現の検討
（研究担当:喜多　陽花）

研究背景

- 不正アクセスは深刻な脅威として注目されている
- 不正アクセスを防ぐためのアクセス制御技術はカーネル空間(Linux上)・ユーザ空間で実装可能
- ユーザ空間における実装ではアプリケーション単位できめ細かくアクセスポリシを設定できる可能性
  - OAuthプロトコル:ユーザが関与する前提→アプリケーション同士の認証や認可には対応していない
  - DDSSecurity:単一計算機上の通信でも暗号化によるオーバーヘッドが発生

➡ 単一計算機上で強制アクセス制御を実現するユーザ空間の認証・認可デーモンを提案

設計

アプリケーションの署名

- 管理者の操作
  - アクセス制御リスト(ACL)を作成し，アプリケーションのアクセス権限を設定
  - アプリケーションの証明書を作成し，拡張ファイル属性に設定

アプリケーションプロセスの起動

- seccompを利用
  - サーバプロセス:socket()，bind()の直接呼出を禁止
  - クライアントプロセス:socket()，connect()の直接呼出を禁止
- LD_PRELOAD環境変数を利用し，seclibライブラリを適用
  - socket()→sec_socket()を実行し，デーモンとUnixドメインソケットで接続
  - bind()→sec_bind()を実行し，使用したいIPアドレスとポート番号をデーモンに送信，認証・認可後にソケットのファイルディスクリプタを取得
  - connect()→sec_connect()を実行し，使用したいIPアドレスとポート番号をデーモンに送信，認証・認可後にソケットのファイルディスクリプタを取得

デーモン接続

- サーバプロセス・クライアントプロセスはsocket()を呼び出すとsec_socket()が実行され，デーモンとUnixドメインソケットで接続

サーバプロセスの認証・認可要求

- bind()を呼び出すとsec_bind()が実行され，使用したいIPアドレスとポート番号をsendmsg()でデーモンに送信
- プロセスIDの取得
- 署名の検証
  - デーモンは拡張ファイル属性から証明書を取得
- アクセス制御リストを確認

サーバプロセスへのファイルディスクリプタ送信

- デーモンがサーバプロセス用のソケットを作成
- bind()で受け取ったIPアドレスとポート番号をソケットに割り当て
- 作成したサーバ用ソケットのファイルディスクリプタをsendmsg()でサーバプロセスに送信
- サーバプロセスはlisten()を呼び出し，接続を待機

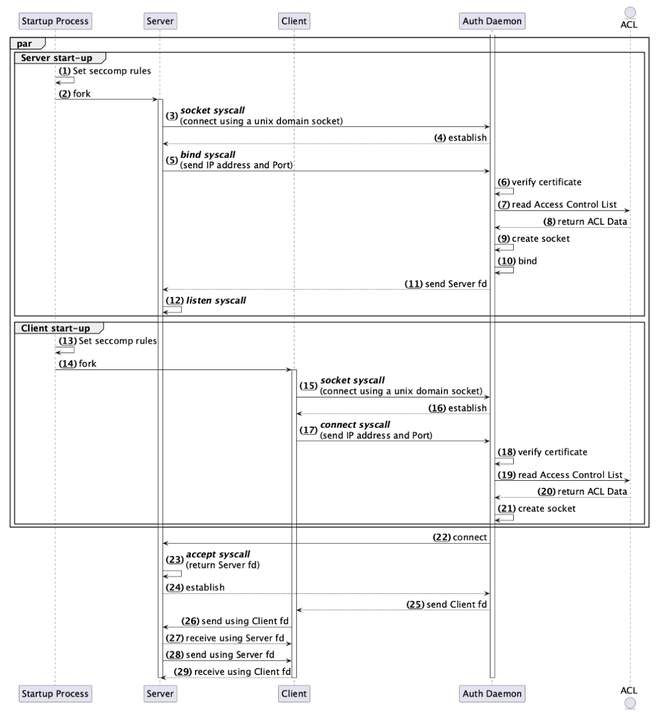
クライアントプロセスの認証・認可要求

- connect()を呼び出すとsec_connect()が実行され，使用したいIPアドレスとポート番号をsendmsg()でデーモンに送信
- サーバプロセスと同じように認証・認可

クライアントプロセスへのファイルディスクリプタ送信

- クライアントプロセスの認証・認可後，デーモンがクライアントプロセス用のソケットを作成
- クライアントのIPアドレス・ポート番号を使用し，デーモンがサーバへconnect()
- サーバプロセスがaccept()を呼び出し，ファイルディスクリプタを取得
- サーバプロセスとデーモンが接続
- デーモンがサーバと接続したクライアント用ソケットのファイルディスクリプタをsendmsg()でクライアントに送信

サーバプロセス・クライアントプロセスはデーモンから受け取ったファイルディスクリプタを用いて通信



https://www.is.ocha.ac.jp/~oguchi_lab/　　oguchi@is.ocha.ac.jp