データ解析型プロジェクトにおけるバージョン管理手法の検討

九州工業大学

〇李林

服部祐一

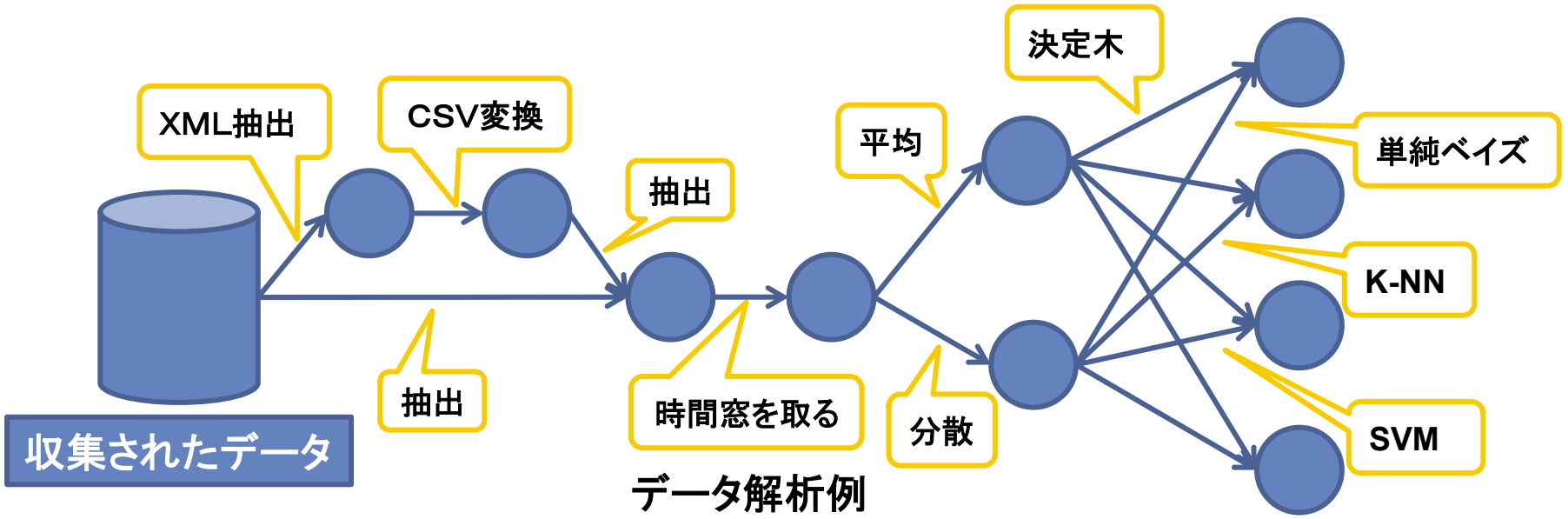
井上創造

目次

- 背景
- 関連研究
- バージョン管理手法の提案
- まとめ
- 今後の研究

背景

- 近年、Web技術とセンサ技術の発展により、大量のデータが収集され、解析されるようになった
- データ解析には複数の段階があり、毎段階の中間結果は生成する、生成した結果は引き続く解析したり、前の中間結果を戻して別の方法で再解析したりする



これらのデータをバージョン管理することできれば、
何度も同じ計算をする必要がなくなる

問題点

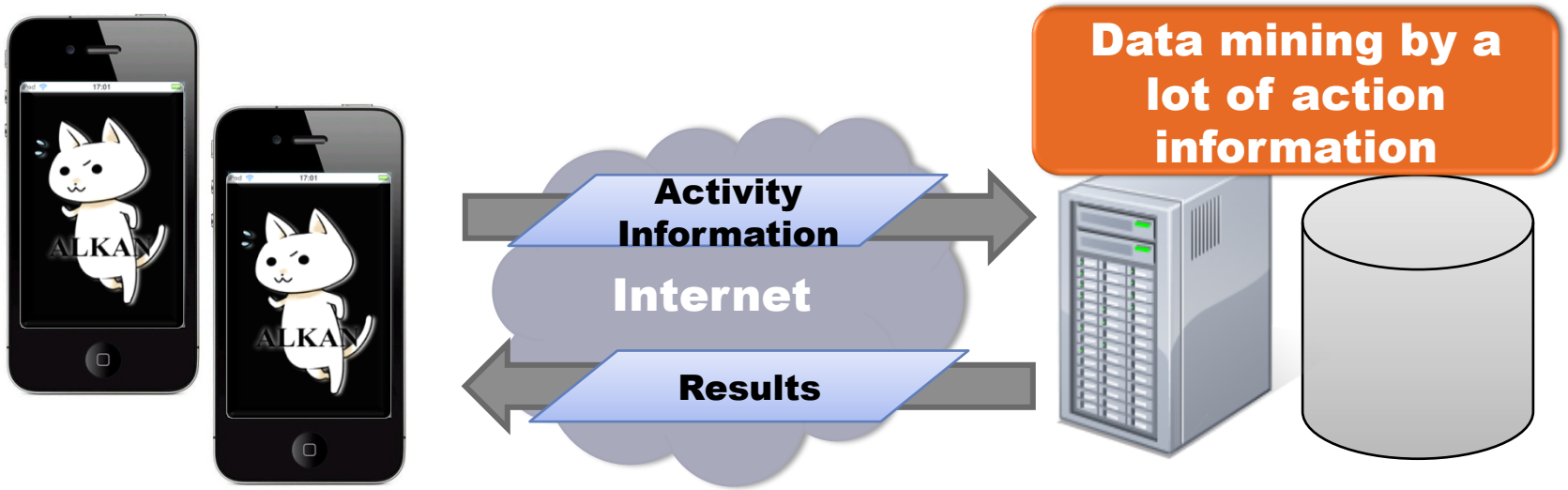
- しかし、データ解析型プロジェクトでは、データとデータ、プログラムとデータなどの間に依存関係がある

- 一般的なバージョン管理方法によるデータとプログラムの依存関係を把握することが困難である

本研究では、上記の問題に対して、データ解析プロジェクトにおけるプログラムのソースコードだけではなく、解析に用いるデータを合わせて、バージョン管理の手法を検討し、データフローの視点から、自動的にデータを更新判断するアルゴリズムを提案する

関連研究 1　大規模行動情報システム「ALKAN」

- 3軸加速度センサを搭載した携帯情報端末を用いた
  大規模行動情報収集システム
- 送信された行動情報をサーバに格納
- 現在の行動情報収集件数：　約40,000件

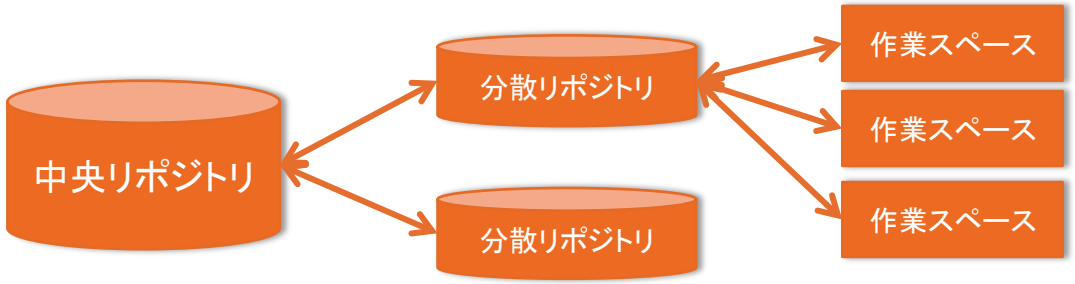


Yuichi Hattori, Sozo Inoue, Go Hirakawa,"A Large Scale Gathering System for Activity Data with Mobile Sensors", ISWC2011, pp. 97-100, June 12, 2011, San Francisco, CA, USA.

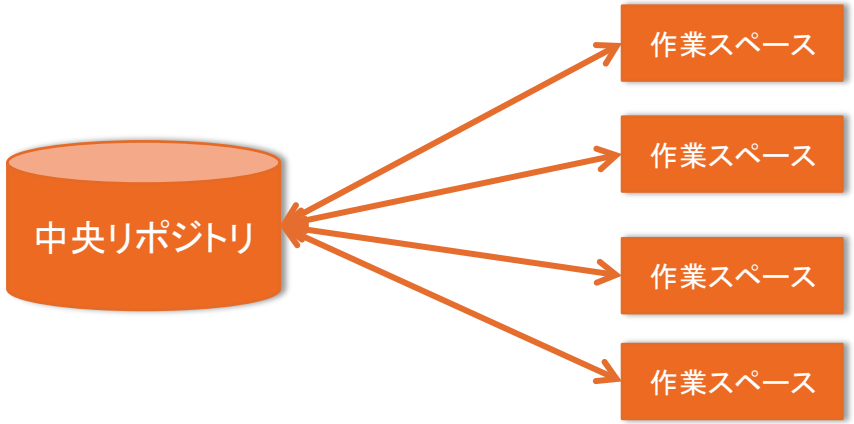
関連研究 2　バージョン管理システム

現在の主流のバージョン管理システム

① 分散型バージョン管理システム（Git、Mercurial）

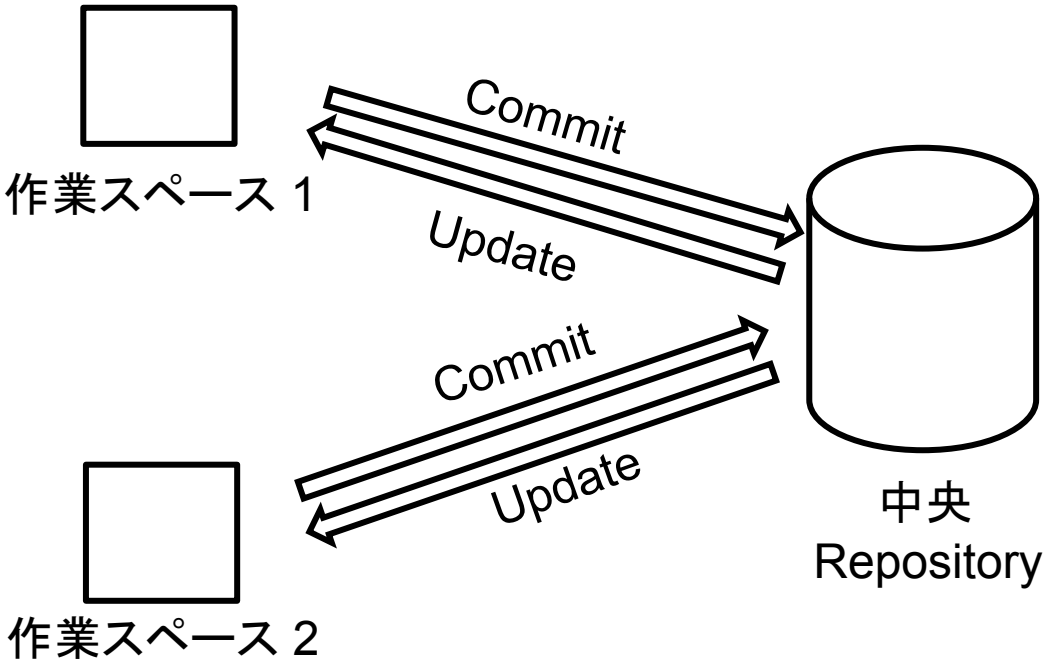


② 集中型バージョン管理システム(CVS,Subversion)



関連研究 3　SUBVERSION(SVN)

- 時間と伴に変化するファイルやディレクトリを管理することができる
- 古いバージョンのデータに戻したり、変更履歴を確認したりすることもできる



バージョン管理の提案　定式化

我々は有限的な有向グラフ

$$G = (V, E)$$

を定義する.

V : ノードの集合（データ）

E : 枝の集合（プログラム）

とする

$$V = \{v_1, v_2, v_3, \cdots, v_n\}$$

$$E = \{(v_1, v_2), \cdots, (v_m, v_n)\}$$

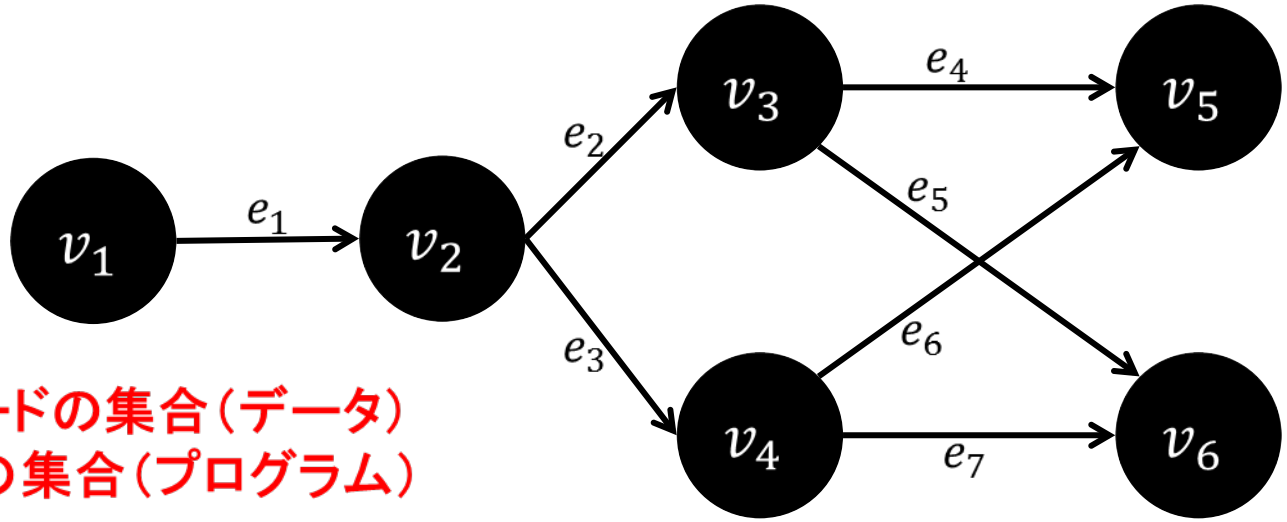
バージョン管理の提案　実例

- ある数式Gを下記のように定義している.

$$G = (V, E)$$

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \left\{ \begin{matrix} (v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_5), \\ (v_3, v_6), (v_4, v_5), (v_4, v_6) \end{matrix} \right\}$$



V：ノードの集合（データ）
E：枝の集合（プログラム）

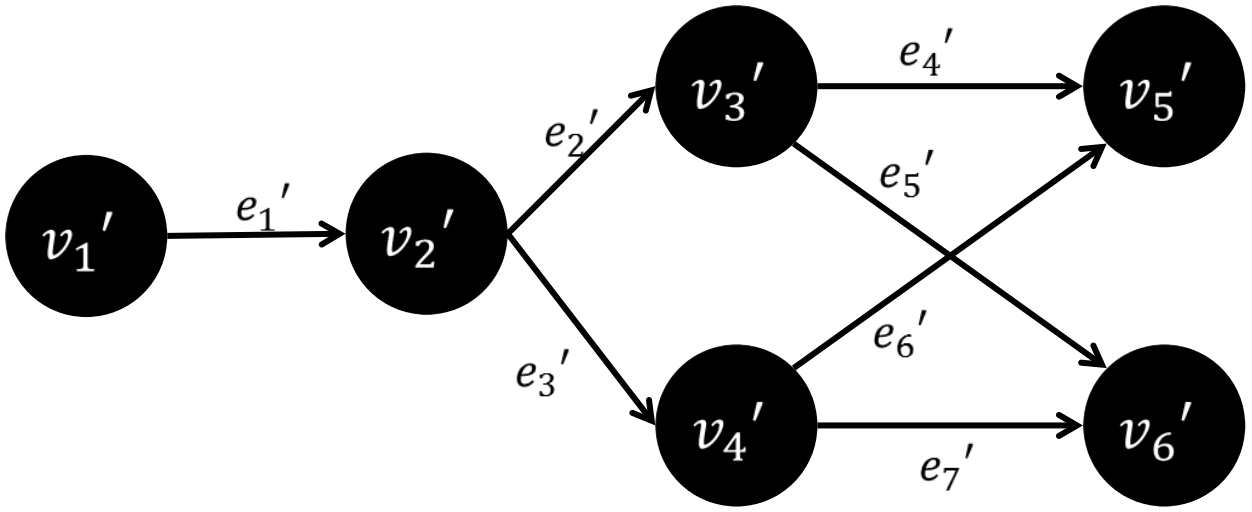
バージョン管理の提案　実例

- 次に，比較のためにG'を定義する．G'はGに対応していくつかファイルが更新された状態である．以下のように表す．データの中身が異なっている場合は$v \neq v'$,同じなる場合$v = v'$と書く

$$G' = (V', E')$$

$$V' = \{v_1', v_2', v_3', v_4', v_5', v_6'\}$$

$$E' = \left\{ \begin{matrix} (v_1', v_2'), (v_2', v_3'), (v_2', v_4'), (v_3', v_5'), \\ (v_3', v_6'), (v_4', v_5'), (v_4', v_6') \end{matrix} \right\}$$



アルゴリズム（あるノードvが更新される条件に対して判断する）

Algorithm 1 need_update

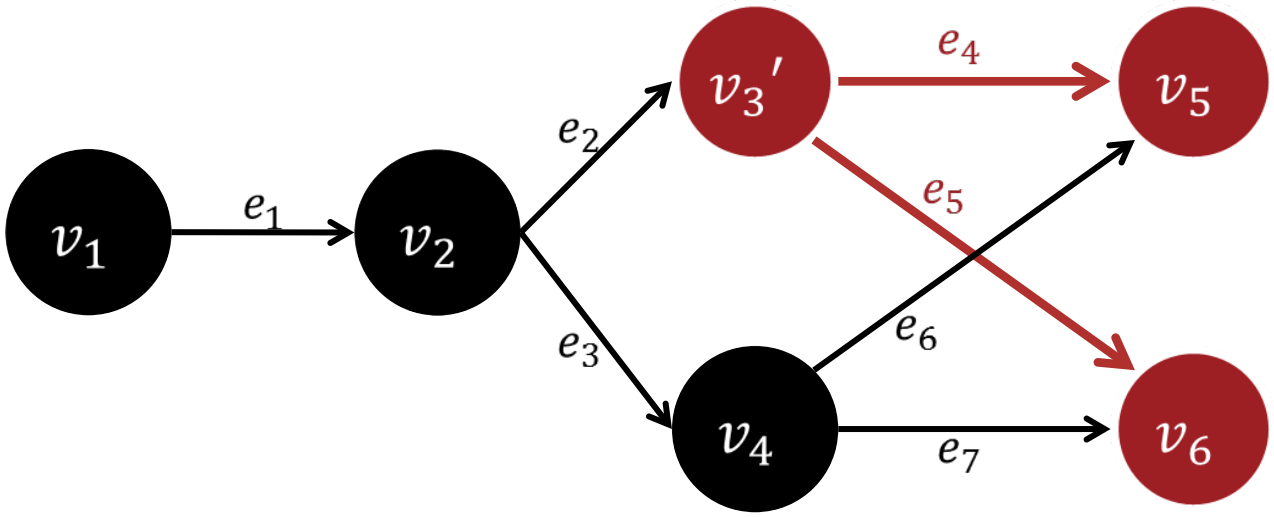
Input G, G', v, v'
Output $TRUE, FALSE$

```
1.  if v ≠ v' then return TRUE
2.    if parents(v) ≠ ∅
3.      for each w ∈ parents(v) do
4.        if(need_update(G, G', w, w'))
5.          then return TRUE
6.      end for
7.      for each e ∈ inedge(v)
8.        if e ≠ e' then return TRUE
9.       end for
10. return FALSE
```

注：1.有向グラフでは、頂点v_xを指向している辺の集合を$inedge(v_x)$と書く
2.v_xの親ノード集合を$parent(v_x)$と書き，$(v_y, v_x) \in E$となるようなv_yの集合である

バージョン管理手法　再計算の例

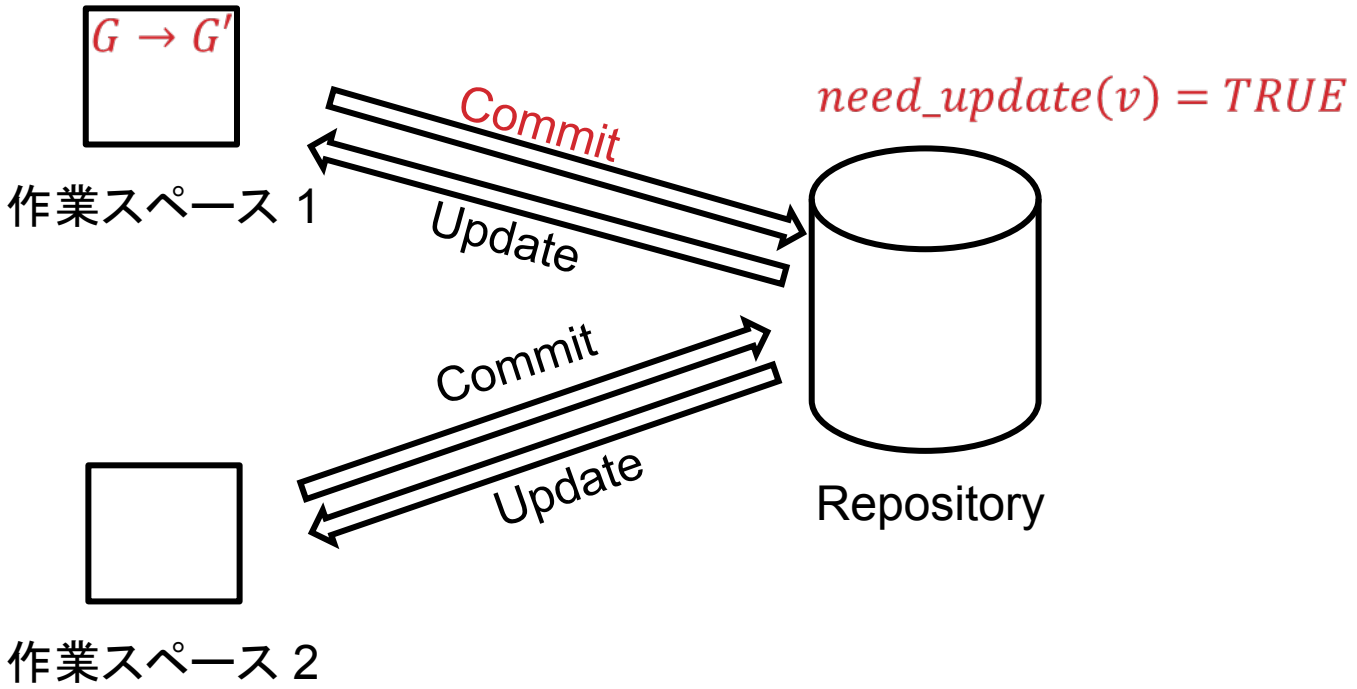
上記のアルゴリズムに対して、例えば$v_3 = v_3'$の時$need_update(v_5) = TRUE$なのでe_4とe_5を再計算する必要がある



※赤い部分を再計算

バージョン管理手法　SUBVERSIONと連携

例えば、作業スペース1であるGがG'になった、クライアントからサーバにコミットすると、リポジトリではすべての$v \in V'$対して自動的にこのアルゴリズムを実行する.
もしあるv に対し$need_update(v)$　がTRUEならば,
データを再計算しなければならない.



まとめ

- 本研究はデータ解析型プロジェクトの特徴に対して数式でモデル化、管理手法を提案した
- あるデータの更新条件を自動的に判断できるアルゴリズムを作った
- Subversionと連携の検討を行った

今後の研究

- グラフ表現にはプログラムの入力と出力が複数の場合を考える
- 再計算の順序の特定アルゴリズムを開発する
- Subversionと連携して実装する
- ALKANプロジェクトの管理を実験し、効率について評価を行う
- 並列分散処理に対する適用をする

ご清聴ありがとうございました