多目的最短経路問題のための拡張ベルマンフォード法の提案

中野　壱帥

青山学院大学 宋研究室

January 30, 2019

目次

研究背景：最短経路問題

最短経路問題

重み付きグラフ $G = (V, E)$ の与えられた 2 つのノード s, t 間を結ぶ経路の中で，重みが最小の経路を求める最適化問題である．

- 2 頂点対最短経路問題：特定の 2 つのノード間の最短経路問題．
- 単一始点最短経路問題：特定の 1 つのノードから他の全ノードとの間の最短経路問題．
- 全点対最短経路問題：グラフ内の任意の 2 ノード間の最短経路問題．

単目的最短経路問題の主な解法

- ダイクストラ法：重みが最小のノードを更新対象として探索する．
- ベルマンフォード法：全辺を緩めることを単に $|V| - 1$ 回繰り返す．

単目的最短経路問題では負のサイクルが存在する場合，無限に更新が行われるため解を求めることができない．$\Rightarrow$ 負のサイクルの存在を報告する．

研究背景：多目的最適化問題

多目的最適化問題

特定の集合上で定義された複数の実数値関数または整数値関数に対して実行可能な最適解を求める問題である．
多目的最適化問題には，目的関数間にトレードオフの関係が存在するためパレート最適解を求める．

- トレードオフ：一方を追求すれば他方を犠牲にせざるを得ない関係であり，同時に最適にできない．

パレート最適解

取りうる値の範囲を全て考慮した上で支配されない解である．

- 支配：多目的最小化問題において，解 x, y が以下の条件を満たすとき，x は y を支配する．
  - $\forall i \in \{1, \ldots, k\}, f_i(x) \leq f_i(y)$
  - $\exists i \in \{1, \ldots, k\}, f_i(x) < f_i(y)$
  - k：最適化目的の数，f：目的関数

従来研究と研究目的

多目的最短経路問題

多目的最短経路問題は目的関数が複数である最短経路問題である.

- 道路ネットワーク，通信ネットワークなど.

- 本研究では多目的単一始点最短経路問題を扱う.

多目的単一始点最短経路問題の従来研究

拡張ダイクストラ法 [takahashi ら，2015]

- 従来では，非負の問題を扱っている.

- 本研究では，負の重みを考慮した問題を扱う．目的関数が複数あるため，負のサイクルが存在しない目的関数のみで解を求める.

研究目的

多目的単一始点最短経路問題における，負の重みを含むインスタンスへの拡張

多目的単一始点最短経路問題

多目的単一始点最短経路問題のインスタンスは頂点の全体集合 V と辺の全体集合 E で表されるグラフ $G(V, E)$，最適化目的の数 k，始点 s，各辺の重みを返す関数 $w : E \to \mathbb{R}^k$ で定義される．

- $V = \{1, 2, 3, \ldots, n\}$：頂点の全体集合
- $E = \{e_1, e_2, e_3, \ldots, e_m\}$：辺の全体集合
- $\vec{e} = \{e_{(1)}, e_{(2)}, \ldots, e_{(k)}\}$：辺に対する重み

多目的単一始点最短経路問題の解はパレート最適解の集合，すなわち s から各頂点への経路の集合 L である．

- 経路 l_v の評価 $\{l_{v(1)}, \ldots, l_{v(k)}\}$ は経路上の辺の重みの合計である．
  - 経路の経由する辺集合を E' とすると，$l_{v(1)} = \sum_{e' \in E'} e'_{(1)}$ となる．
- 多目的の場合，解の判定や削除に多くの計算時間がかかる．
  - 単目的 $\Rightarrow$ 1 頂点にたかだか 1 つの解
  - 多目的 $\Rightarrow$ 1 頂点に最大 $(n-2)!$ 個の解

ベルマンフォード法の拡張

入力： グラフ $G(V, E)$，最適化目的の数 k，始点 s，各辺の重みを返す関数 $w : E \to \mathbb{R}^k$

出力： パレート最適解である経路の集合 L

Step 1. $L \leftarrow l_s, \forall i \in \{1, \ldots, k\}, l_{s(i)} = 0$

Step 2. $\forall e \in E$ に対する以下の操作を $|V| - 1$ 回繰り返す.

> *Step 2-1.* $v, u \in e$ とするとき，$l'_u = l_v + \vec{e}, l_v \in L$ とする.
> *Step 2-2.* l'_u に対して解であるかの判定をする.

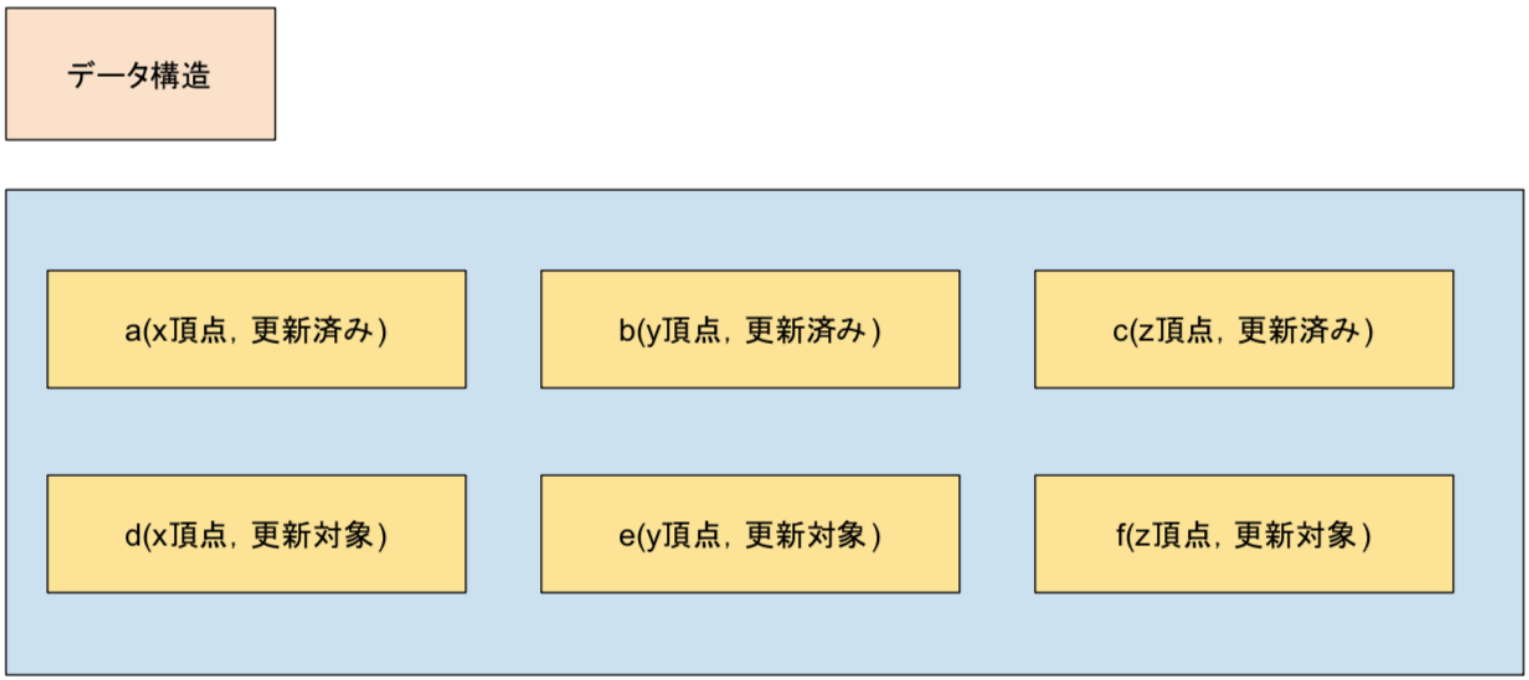
Step 3. 解を出力する.

非負の問題に対する解法のため辺を緩める操作を $|V| - 1$ 回行うのは無駄.
$\Rightarrow$ 更新できなくなったら探索終了.

従来のデータ構造

全ての経路が経路の集合である L に記憶されている.

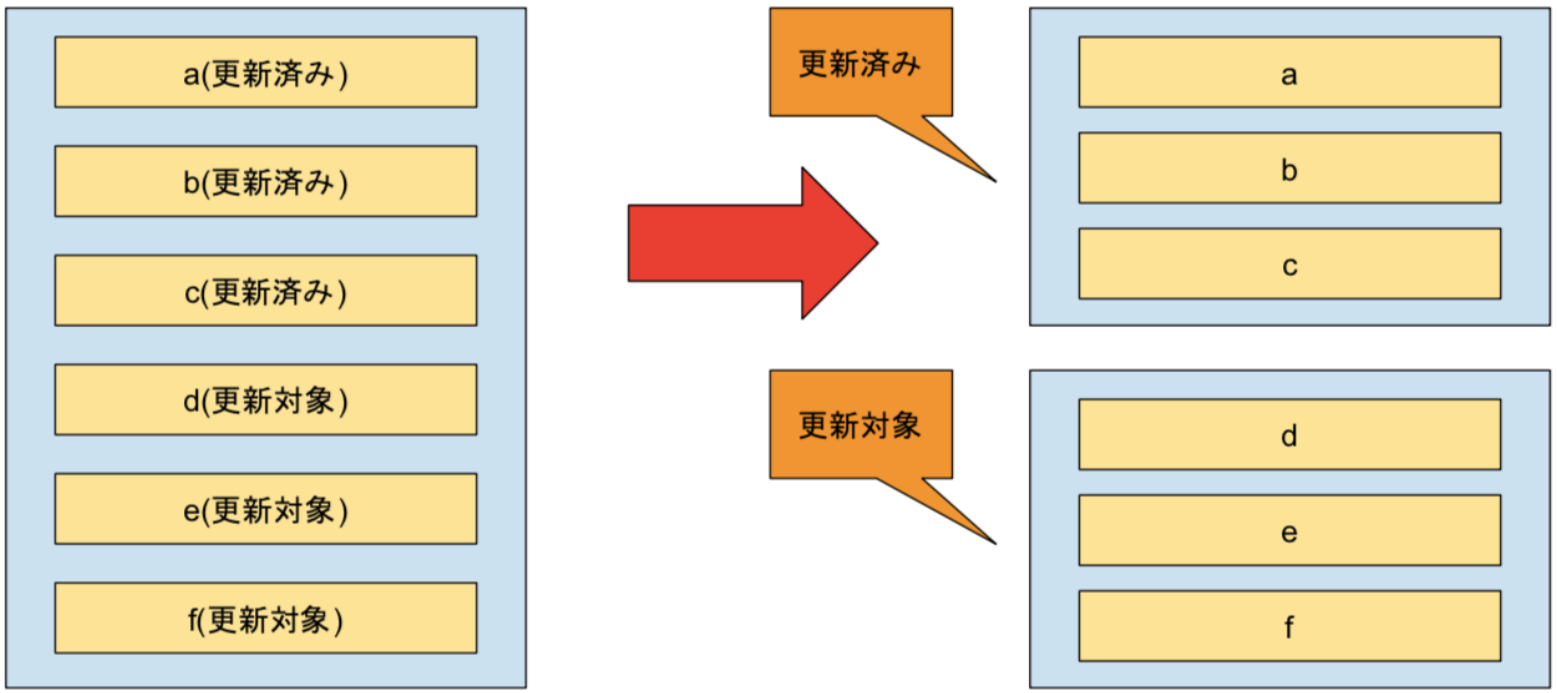
- 辺を緩める操作のとき，発見されている全ての経路を更新対象とするため，すでに更新した経路も更新対象となってしまう.
- l'_x という経路に対して解であるかの判定をする際，$l_x \in L_x$ となる経路を L から探索しなければならない.



データ構造の提案 1

更新対象とする経路と更新対象としない経路を分けて記憶する.

すでに更新した経路も更新対象となってしまう問題を解決する.

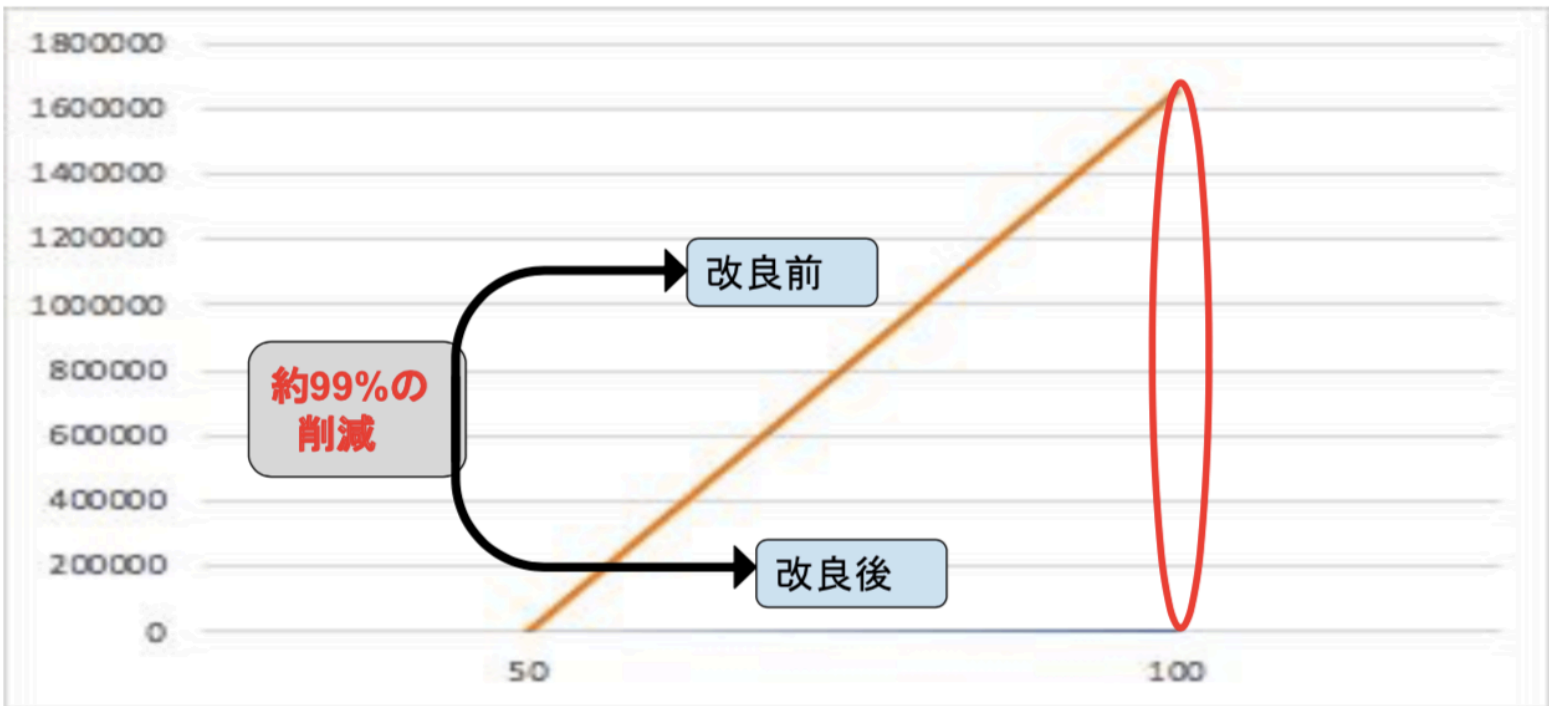


更新する経路が存在しない頂点に対して辺を緩める操作は無駄.
⇒ 経路に対する更新とする.

提案の評価

データ構造の提案 1 の評価

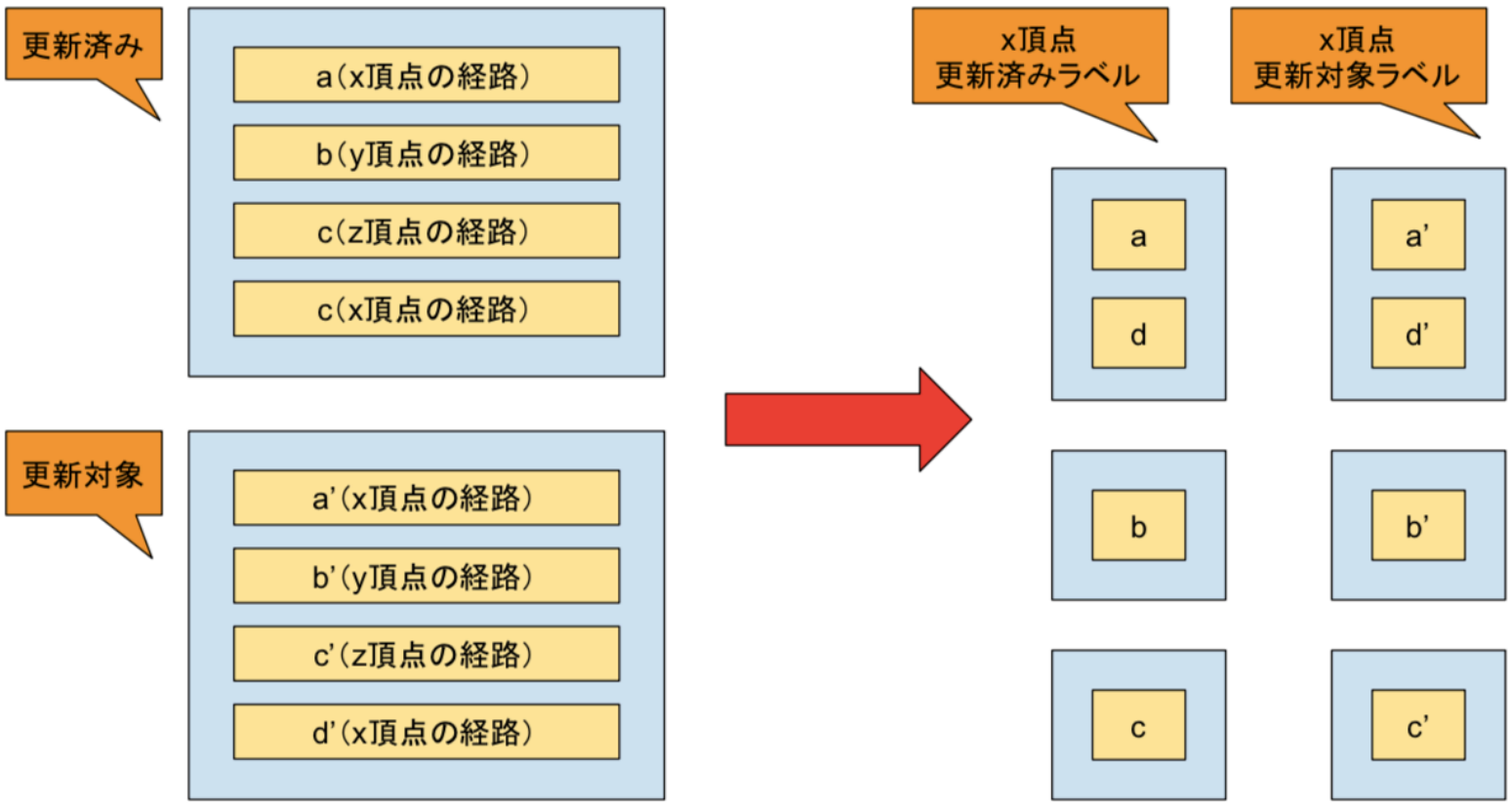
更新回数に対する計算量を $\mathcal{O}(n * S)$ から $\mathcal{O}(S)$ に削減した.
（生成された経路の数 S，頂点数 n）
全体の実装に対する計算時間を最大 99% 削減した.



データ構造の提案 2

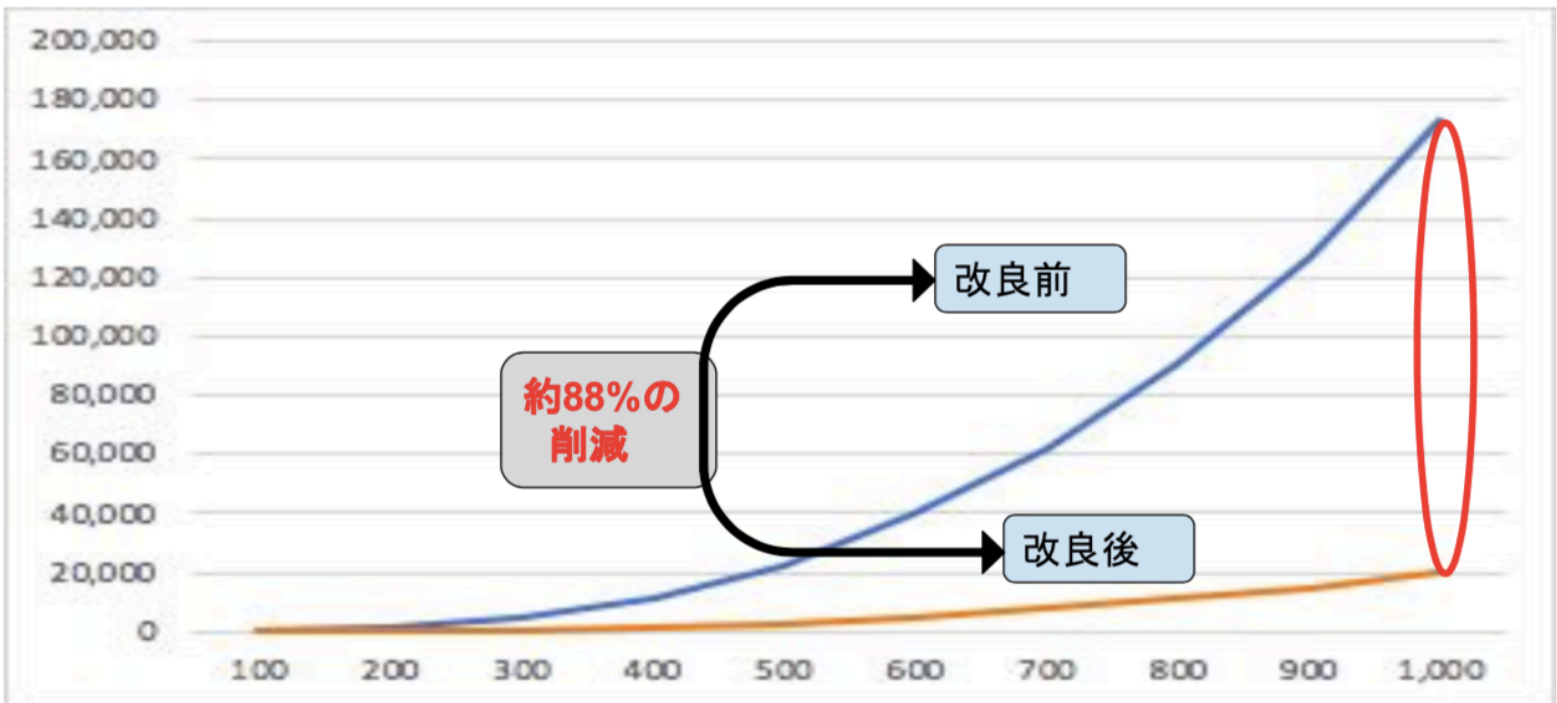
格納する経路を頂点ごと，n 個のラベルに分けて記憶する．

頂点 v への経路に対して解であるかの判定をする際，全体のラベル L から比較対象となる v のラベル L_v を探索する操作を削減する．



データ構造の提案２の評価

頂点 v への１つの経路に対するパレート解であるかの判定に要する計算量を $\mathcal{O}(k * L_v + L)$ から $\mathcal{O}(k * L_v)$ に削減した.
（最適化目的の数 k，更新時の全体のラベル L，更新時の v のラベル L_v）
全体の実装に対する計算時間を最大 88% 削減した.



生成される経路の数に対する考察

解であるかの判定

頂点 v への経路はすでに発見されている v への集合の要素全てと比較しなければならない.
つまり，すでに発見されている v への集合の大きさで計算時間が変わる.

すでに発見されている v への集合は解となる経路と後で削除される経路を含んでいる.
$\Rightarrow$ 後で削除される経路との比較は無駄.
後で削除される経路が 1 度解となったときに，更新対象とする．後で削除される経路によって生成された経路も後で削除されるため無駄となる.

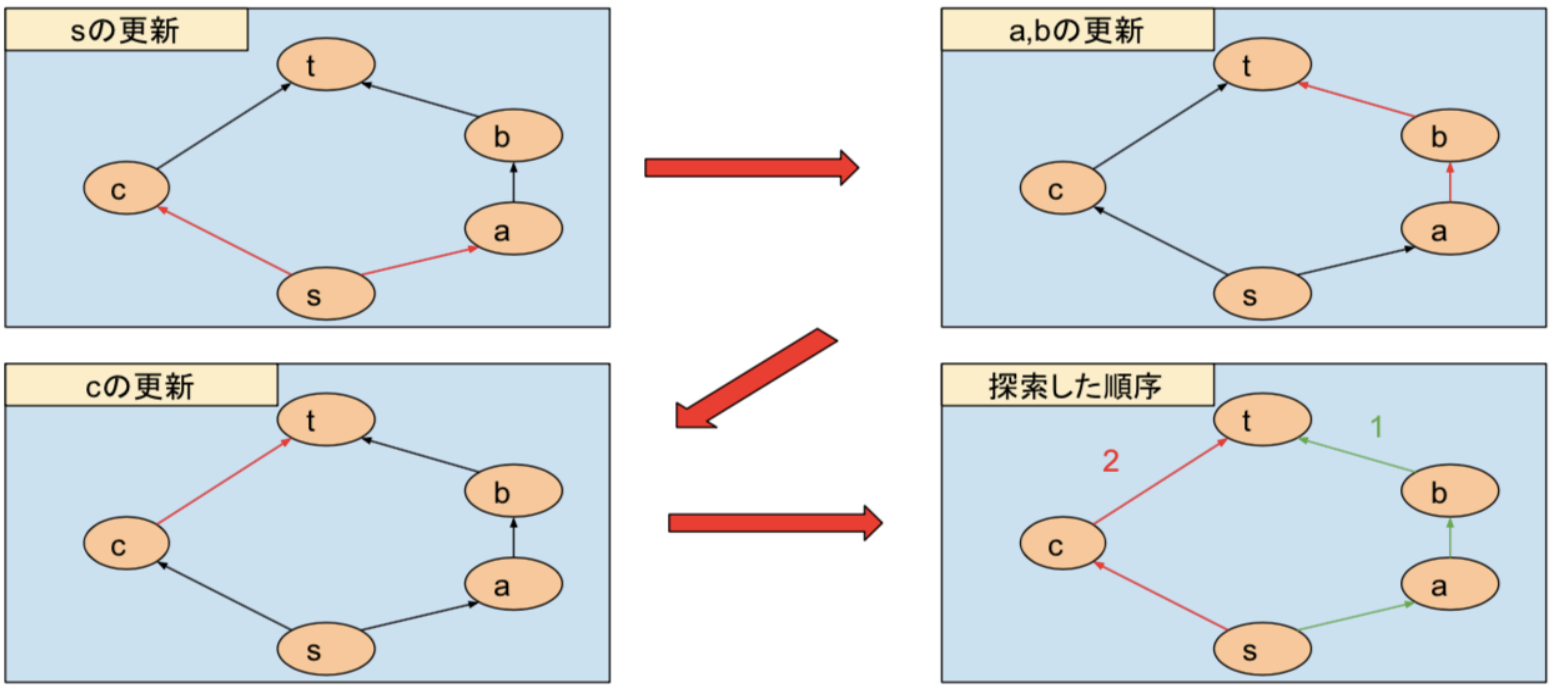
目的

後で削除される経路を削減する.

削除される経路が先に発見されなければ良い.

従来の探索順序

頂点の更新順序によって，経由する頂点数が多い経路が経由する頂点数が少ない経路より先に発見されてしまう．(更新順序：$s \to a \to b \to c$)

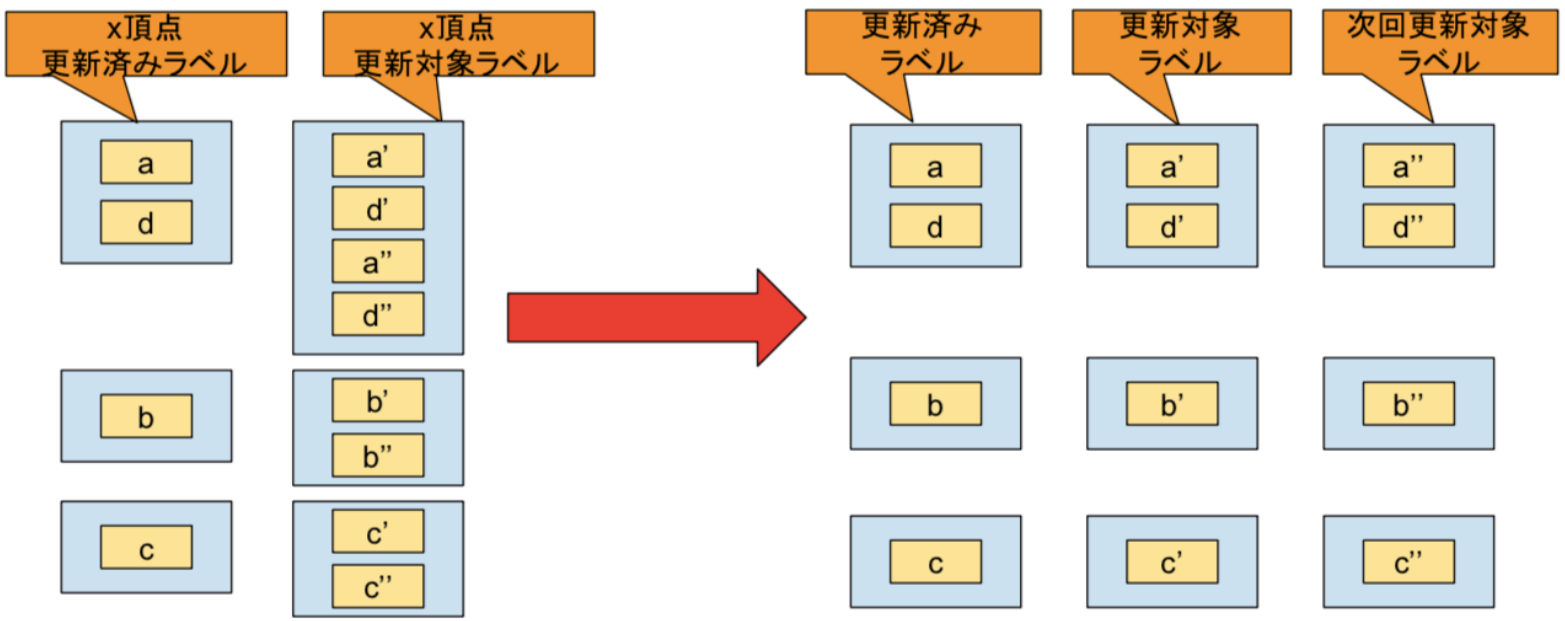


非負かつ重みの範囲が決まっている場合，経由する頂点が少ない経路の方が各目的関数の値が低くなる確率が高いため，経由する頂点が少ない経路を先に探索した方が良い．

データ構造の提案３

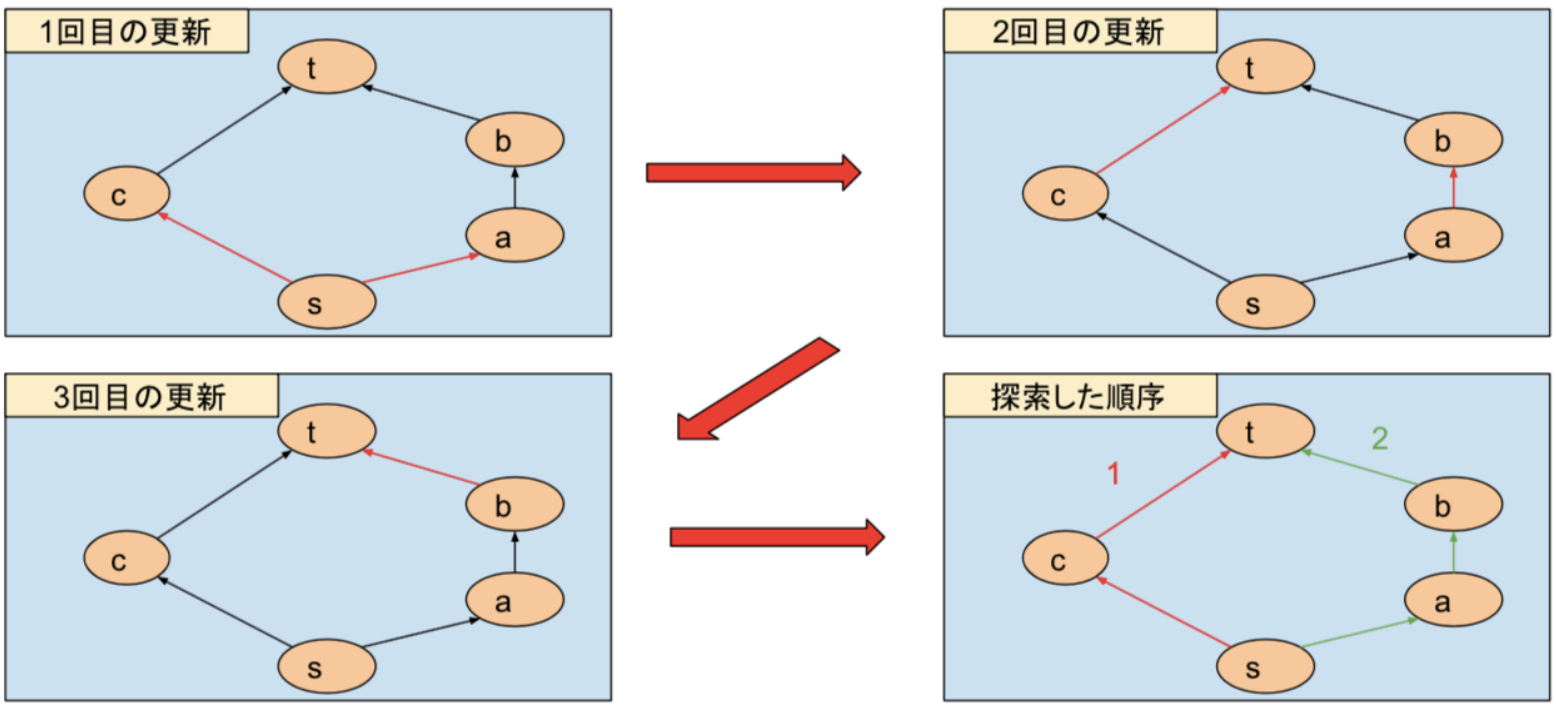
格納する経路を更新済み，更新対象，次回更新対象という３個のラベルに分けて記憶する．

更新内で探索された経路は次回の更新とすることで，経由する頂点数が少ない経路から発見されるようにする．



改良後の探索順序

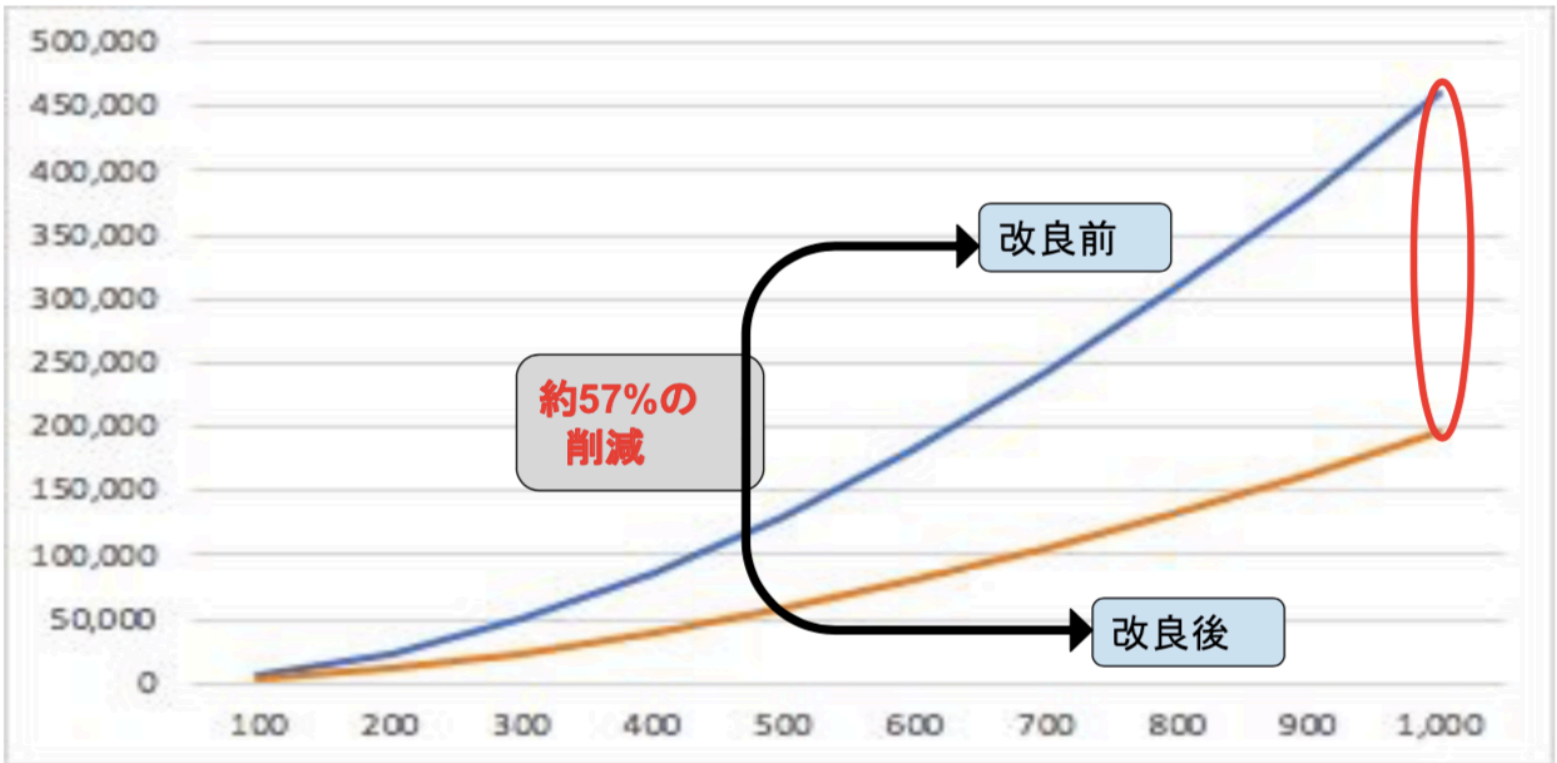
頂点の更新順序を指定することによって，経由する頂点数が少ない経路が経由する頂点数が多い経路より先に発見される．



提案の評価

削除される解の数の比較

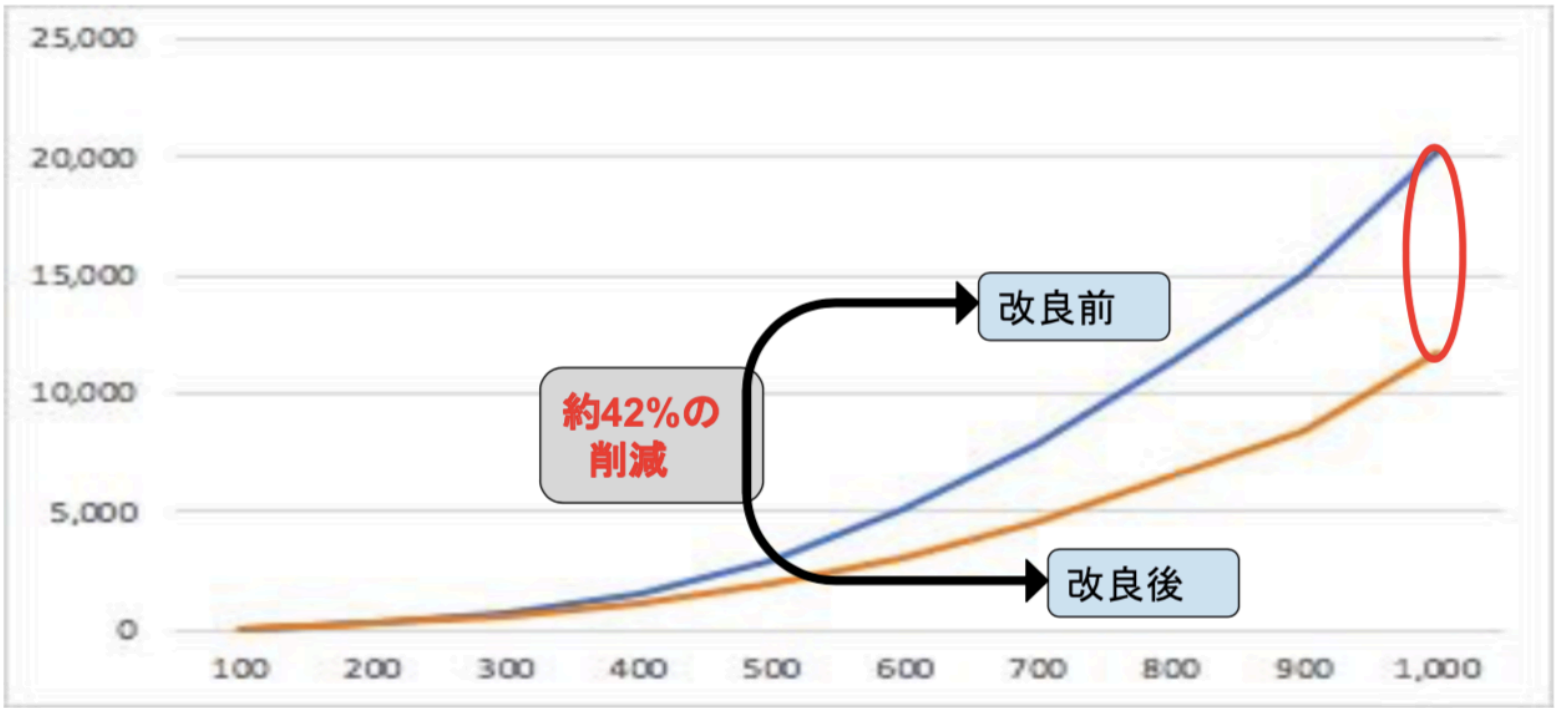
同じインスタンスに対して，削除される解の数を最大 57% 削減した.



提案の評価

探索順序変更による評価

全体の実装に対する計算時間を最大 42% 削減した.



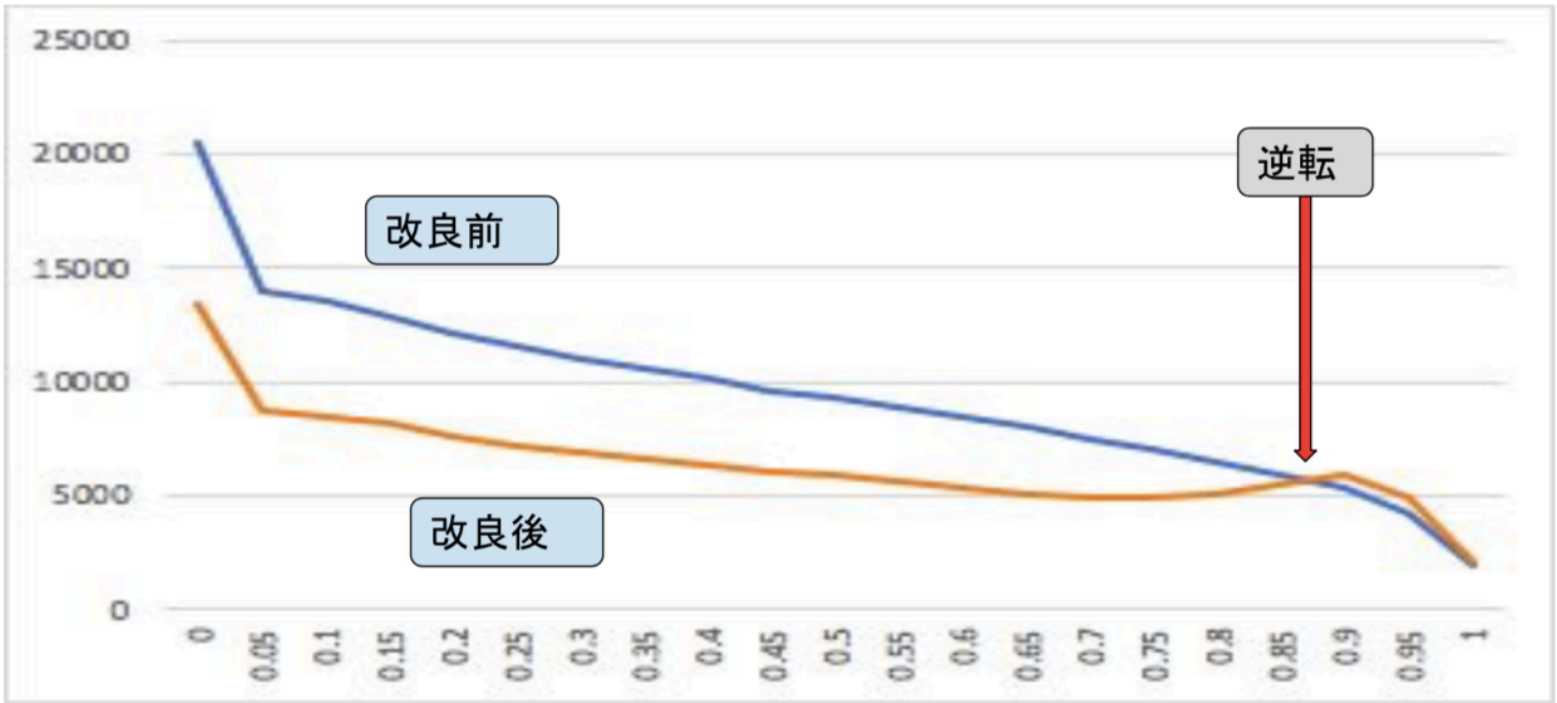
削除される解の数は 57% 削減したが，実装に対する計算時間は 42% の削減である.

$\Rightarrow$ ラベル間の移動操作に時間がかかっている.

提案の評価

条件ごとの評価

目的関数の相関係数が 0.87 以上のときは有効でない.



相関が 0.87 以下の場合のみ，無駄な探索をしないことによる時間の削減＞ラベル間の移動操作にかかる時間となる.

負のサイクルの検出

負のサイクルが存在する場合の解

ある目的関数に負のサイクルが存在するとき，その目的関数を無視することで負のサイクルが存在しない解を求める．
目的関数 j に負のサイクルが存在するとき，$\forall e \in E, e(j) = 0$ とする．

負のサイクルが存在する目的関数を求めたい．

従来の検出方法

辺を緩める操作を $|V| - 1$ 回行なった後，$|V|$ 回目の探索を行い重みが更新される頂点があれば負のサイクルが存在する．

多目的最短経路問題の場合，解となる経路が膨大に存在するため，負のサイクルが存在し $|V|$ 回の更新を行うと膨大な計算時間がかかってしまう．
$\Rightarrow |V|$ 回の更新を行わず負のサイクルが存在する目的関数を求めたい．

負のサイクルの検出

提案１：探索途中で負のサイクルを検出

同じ頂点を２度通った経路が解となっていないかを判定する.
解となっていた場合，１度目に通った経路と比較し負のサイクルが存在する目的関数を特定し削除する.（すでに発見されている経路に対しても適用し解であるかの判定をする）

提案２：探索開始前に負のサイクルを検出

各目的関数に対して単目的最短経路問題とみなしベルマンフォード法を適用する．$O(|E||V||k|)$ で検出が可能であり，非負の探索が適用できる.

提案３：探索開始前に負のサイクルを検出し，求めた解を活用

各目的関数ごとに負のサイクルを検出をした際，求めた解は確実に解となるため，求めた解を基に探索を開始する.

研究成果と今後の課題

多目的単一始点最短経路問題に対して，以下の成果が得られた．

成果

- 負の重みを含む問題のためにインスタンスの拡張を行った．
- データ構造の工夫によって計算量を削減した．
- 実験により，提案解法に対する優位性を示した．

今後の課題

全体の実装に対する計算量を求める．

本研究で提案した解法において，部分問題に対する計算量は求めたが，
全体の計算量は求められていない．