数理解析研究所講究録
第2027巻 2017年 15-29

最小増加超距離木問題に対する局所探索アルゴリズム

石川累† 安藤和敏‡

† 静岡大学大学院総合科学技術研究科
‡ 静岡大学工学部

Rui Ishikawa† and Kazutoshi Ando‡

†Graduate School of Integrated Science and Technology, Shizuoka University

‡Faculty of Engineering, Shizuoka University

概要

超距離木とは根から葉までの距離が全て等しいような枝重み付き根付き木のことである. 超距離木 (T,l) に対して, $D_{(T,l)}[i,j]$ によって (T,l) の葉 i,j 間の距離を表す. L_p-最小増加超距離木問題とは, 相違行列 $M\colon S\times S\to\mathbb{R}$ が与えられたときに, 葉集合が S でありかつ $M[i,j]\leq D_{(T,l)}[i,j]$ $(i,j\in S)$ を満たすような超距離木 (T,l) の中から $\|D_{(T,l)}-M\|_p$ を最小化するものを見出す問題である. L_p-最小増加超距離木問題は, $p=\infty$ の場合は線形時間アルゴリズムが存在するが, p が有限の場合は NP 困難であることが知られている. 本研究では 2 分木の変形操作に基づいて, p が有限の場合の L_p-最小増加超距離木問題に対する局所探索アルゴリズムを開発した. 2 分木の変形操作として系統学においてよく知られている SPR 操作及び NNI 操作に加えて本研究で導入される SE 操作を用いる. さらに数値実験によってこのアルゴリズムの性能を検証した. その結果, NNI 操作を用いる局所探索アルゴリズムは非常に高速ではあるもののその解の品質は満足できるものではなかった. その一方で, SPR 操作あるいは SE 操作を用いる局所探索アルゴリズムの解の品質は NNI 操作を用いるそれよりも高いものの計算時間に関しては改良する必要があるということが明らかになった.

1 はじめに

系統樹とは生物の進化の歴史を表す木であり, 系統樹を推定することは分子系統学における重要な課題である. 系統樹の推定に使われる主な方法は距離法である. $S=\{1,\cdots,n\}$ を系統樹の構築における生物種の集合とする. 距離法では, まずこれらの生物種の DNA 配列などから生物種間の距離を表す行列 $M:S\times S\to\mathbb{R}_+$ を求める. 次に葉の集合が S であるような枝重み付き木 (T,l) の中で $\|D_{(T,l)}-M\|_p$ を最小化するものを求める. ここで, 各 $i,j\in S$ に対して, $D_{(T,l)}[i,j]$ は (T,l) における i,j 間の距離であり, $\|\cdot\|_p$ は L_p-ノルムを表す.

$$\|M\|_p=\begin{cases}(\sum_{i,j}|M[i,j]|^p)^{1/p} & \text{if } p<\infty,\\ \max_{i,j}|M[i,j]| & \text{if } p=\infty.\end{cases}\tag{1}$$

本研究では枝重み付き木 (T,l) として**超距離木**を考える. 超距離木とは根から葉までの距離がすべて等しいような枝重み付き根付き木のことである. 図 1 に超距離木の例を示した. 任意の超

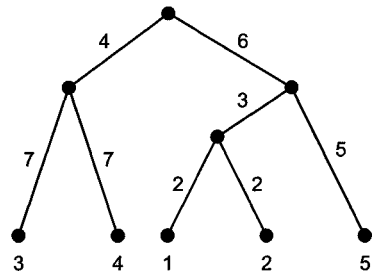

図 1: 超距離木 (T,l). T の葉集合は $S=\{1,2,3,4,5\}$ である.

距離木 (T,l) に対して, $D_{(T,l)}=D_{(T',l')}$ であり, かつ, 2 分木であるような超距離木 (T',l') が存在する. したがって 2 分木である超距離木だけを考えれば十分である. また, 多くの場合において DNA 配列から求めた生物種間の距離 $M[i,j]$ は真の距離の下限であると考えられる. その場合は, 我々が解くべき問題は $M[i,j]\leq D_{(T,l)}[i,j]$ $(i,j\in S)$ という条件を満たす超距離木の中から $\|D_{(T,l)}-M\|_p$ を最小化するものを見出す問題である. この問題は **L_p-最小増加超距離木問題**と呼ばれる. 言い換えると, L_p-最小増加超距離木問題とは, 与えられた相違行列 M に対して以下の問題の最適解 (T,l) を求める問題である.

$$\left|\begin{array}{ll}\min & \|D_{(T,l)}-M\|_p \\ \text{s.t.} & (T,l)\text{ は超距離木}, \\ & M\leq D_{(T,l)}.\end{array}\right. \tag{2}$$

ここで, $M\leq D_{(T,l)}$ はすべての $i,j\in S$ に対して $M[i,j]\leq D_{(T,l)}[i,j]$ が成り立つことを意味する.

L_p-最小増加超距離木問題は, $p=\infty$ の場合には線形時間アルゴリズム [3] が存在するが, $p=1,2$ の場合は NP 困難であることが証明されている [6]. $p<\infty$ のとき L_p-最小増加超距離木問題に対するアルゴリズムとして分枝限定法 [10] や, $O(kn^4)$ 時間で近似保証 $O(n^{\frac{1}{p}})$ を持つ解を求める近似アルゴリズム [8] 等がある. ここで, k は入力行列 M の相異なる成分の数である. 分枝限定法の時間計算量は非常に大きく, $n=20$ 程度の問題でさえ膨大な時間を要する. また, 近似アルゴリズムは時間計算量は小さいがその近似精度は低い.

局所探索法は数理的最適化の一般的手法の一つである. 系統樹推定問題の一つである最大節約問題 (maximum parsimony problem) に対しては SPR 操作 (Subtree Prune and Regraft operation) ([9],[2]) と呼ばれる 2 分木の変形操作に基づく局所探索アルゴリズム [7] が知られている. また, 本節の冒頭で紹介した系統樹推定問題で $p=2$ の場合には, NNI 操作 (Nearest Neighbor Interchange operation) に基づく局所探索アルゴリズム [5] が考察されている. しかしながら, 最小増加超距離木問題に対する局所探索法は筆者等の知る限り存在しない. 本研究では, 2 分木の変形操作に基づいて有限な p に対する L_p-最小増加超距離木問題に対する局所探索アルゴリズムを提案する. 2 分木の変形操作としては, 上述した SPR 操作と NNI 操作 の他に, 本研究で新しく導入する SE 操作 (Subtree Exchange operation) を用いる. さらに $p=1$ の場合にこの局所探索アルゴリズムを高速化する方法を示す. また, 提案するアルゴリズムの性能を評価するために $p=1$ の場合の L_p-最小増加超距離木問題に対してランダムに生成した相違行列を入力として数値実験を行う.

本論文は以下のように構成されている. 第 2 節では, 本論文で提案する L_p-最小増加超距離木問題に対する局所探索アルゴリズムと $p=1$ の場合におけるアルゴリズムの高速化について述べる. 第 3 節においては, $p=1$ の場合の L_p-最小増加超距離木問題に対して, 提案するアルゴリズムの

性能を数値実験によって検証する. 第4節ではまとめと結論を述べる.

2 局所探索アルゴリズム

Sを空でない有限集合とする. 行列$M : S \times S \to \mathbb{R}_+$は, 任意の$i, j \in S$に対して$M[i,j] = M[j,i]$かつ$M[i,i] = 0$を満たすとき, S上の**相違行列**と呼ばれる. 根付き2分木$T = (V, E)$に対して$L(T)$をTの葉集合とし, $v \in V$を根とするTの部分木をT_vと表す. 本論文では, 2分木は全ての枝が根から葉に向かって向き付けられた有向木であるとみなす.

2.1 局所探索アルゴリズム

任意のS上の相違行列Mと$L(T) = S$である根付き2分木$T = (V, E)$が与えられた時, $(T = (V, E), l)$がMに対する最小増加超距離木であるような枝重み関数$l : E \to \mathbb{R}_+$を見出す問題は**MUTT問題** (Minimum Ultrametric Tree with a given Topology problem) [10] と呼ばれる. 言い換えると, MUTT問題とは, 与えられたS上の相違行列MとSを葉集合として持つ2分木$T = (V, E)$に対して, (2)の最適解l: $E \to \mathbb{R}_+$を求める問題である.

以下のアルゴリズム1はWu他 [10] によって与えられたMUTT問題に対するアルゴリズムである.

入力: S上の相違行列M, $L(T) = S$であるような根付き2分木$T = (V, E)$.
出力: MとTに対するMUTT問題の最適解$l : E \to \mathbb{R}_+$.

```
1 for v ∈ V do h(v) ← 0;
2 for T の各内部点 v を後行順で do
3 |   u, w ← v の子;
4 |   h(v) = max{h(u), h(w), M[i,j]/2 | i ∈ L(T_u), j ∈ L(T_w)};
5 end
6 for e = (v, w) ∈ E do  l(e) ← h(v) − h(w) ;
```

アルゴリズム 1: MUTT問題に対するアルゴリズム.

アルゴリズム1によって得られる超距離木(T, l)の各点$v \in V$に対して, $\mathrm{height}_{(T,l)}(v)$を

$$\mathrm{height}_{(T,l)}(v) = \frac{1}{2} D_{(T,l)}[i,j] \tag{3}$$

と定義する. ここで, u, wをvの2つの子とするとき$(i,j) \in L(T_u) \times L(T_w)$である. すると, アルゴリズム1で用いられる変数hに対して$h = \mathrm{height}_{(T,l)}$である.

式(4)で与えられるMと図1の根付き2分木Tを入力としてアルゴリズム1を実行した結果を図2に示した.

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 4 & 7 \\ 2 & 0 & 8 & 3 & 6 \\ 5 & 8 & 0 & 1 & 10 \\ 4 & 3 & 1 & 0 & 9 \\ 7 & 6 & 10 & 9 & 0 \end{pmatrix} \end{array} \tag{4}$$

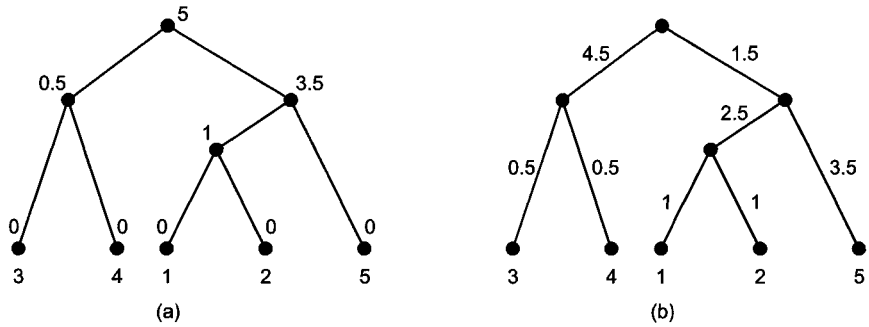

図 2: MUTT 問題に対するアルゴリズムの例. (a) $h(v)$ $(v \in V)$; (b) $l(e)$ $(e \in E)$.

定理 2.1 (Wu et al. [10]) アルゴリズム 1 は $O(n^2)$ 時間で MUTT 問題の解を出力する.

定理 2.1 より, 最小増加超距離木問題の最適解 (T^*, l^*) を見出す問題は, $\|D_{(T,l)} - M\|_p$ を最小にする根付き 2 分木 T を見出す問題になった. ここで, l は相違行列 M と根付き 2 分木 T を入力としたときの MUTT 問題の最適解である. したがって, 最適な 2 分木を求めるために以下の局所探索アルゴリズム (アルゴリズム 2) が導かれる.

入力: S 上の相違行列 M.
出力: $M \leq D_{(T,l)}$ なる超距離木 $(T = (V, E), l)$.
1 初期の根付き 2 分木 $T = (V, E)$ を構成する;
2 $l \leftarrow \mathrm{MUTT}(M, T)$;
3 **for** $T' \in \mathcal{N}(T)$ **do**
4 $\quad l' \leftarrow \mathrm{MUTT}(M, T')$;
5 $\quad$ **if** $\|D_{(T',l')} - M\|_p < \|D_{(T,l)} - M\|_p$ **then**
6 $\quad\quad (T, l) \leftarrow (T', l')$;
7 $\quad$ **end**
8 **end**

アルゴリズム 2: 局所探索アルゴリズム.

アルゴリズム 2 の中で, $\mathrm{MUTT}(M, T)$ は相違行列 M と 2 分木 T を入力とするしたときの, MUTT 問題の最適解である. また, $\mathcal{N}(T)$ は適切に定義された T の近傍からなる集合である. 以下では, T の近傍である SPR 近傍, NNI 近傍, 及び, SE 近傍について述べる.

2.2 SPR 近傍と NNI 近傍

$T = (V, E)$ を根 r を持つ 2 分木とし, $e = (v, w) \in E$ とする. $T \setminus T_w$ によって T から T_w を削除した木を表す. T に対する **SPR 操作** (Subtree Prune and Regraft operation) とは, T から別の 2 分木を得る以下の 3 つの操作のうちの一つである.

1. $v \neq r$ とする. $f = (v', w')$ を $v \neq v', w'$ なる $T \setminus T_w$ の枝とする. T から e を削除した後, f の中点に新しい点 u を挿入し, 枝 (u, w) を挿入する. 結果として生じる次数 2 の点とそれに接続する 2 本の枝を 1 本の枝で置き換える. 図 3(a),(b) を見よ.

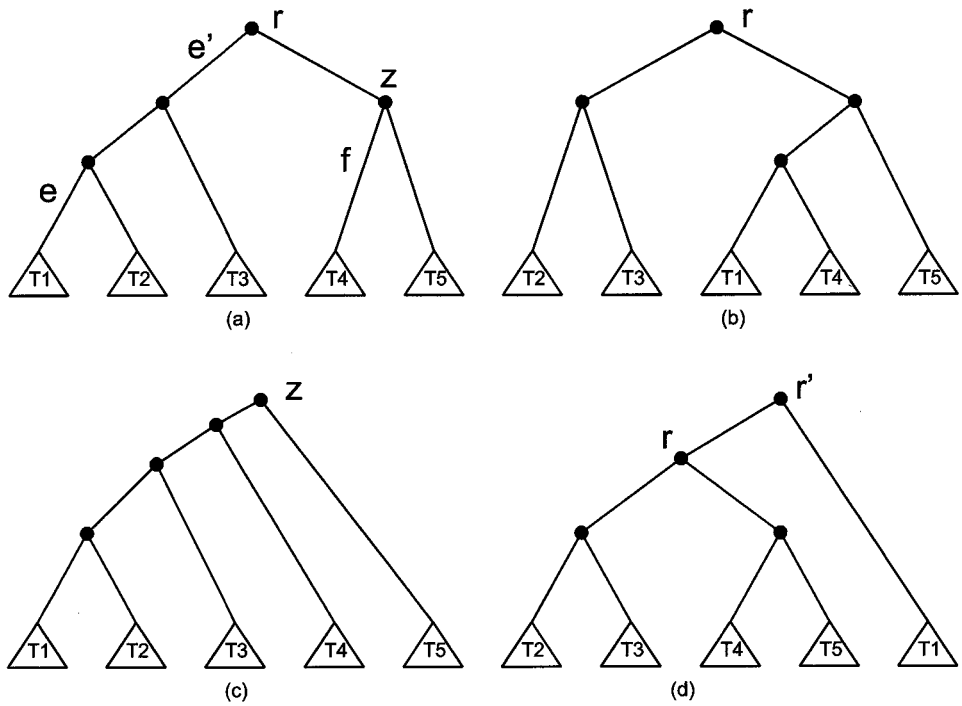

図 3: SPR 操作. T_i $(i = 1, \cdots, 5)$ は部分木を表す. (a) 操作の前の 2 分木 T; (b) SPR(e, f) 操作によって得られる 2 分木; (c) SPR(e', f) 操作によって得られる 2 分木; (d) SPR(e, r) 操作によって得られる 2 分木.

2. $v = r$ とする. $f = (v', w')$ を $v' \neq r$ なる $T \setminus T_w$ の枝とする. T から e を削除した後, f の中点に新しい点 u を挿入し, 枝 (u, w) を挿入する. T の根 r の w 以外の子を z とする. r を削除する. z が結果として生じる木の根となる. 図 3(a),(c) を見よ.

3. $v \neq r$ とする. T から e を削除した後, r' を新しい点として, 枝 $(r', r), (r', v)$ を挿入する. r' が新たに根となる. 結果として生じる次数 2 の点とそれに接続する 2 本の枝を 1 本の枝で置き換える. 図 3(a),(d) を見よ.

上記の SPR 操作 1 および 2 を, それに関連する 2 本の枝 e, f を明示するために **SPR(e, f) 操作**と呼ぶ. また, 上記の SPR 操作 3 を **SPR(e, r) 操作**と呼ぶ. ここで, r は T の根である. 2 分木 T から 1 回の SPR 操作によって得られる 2 分木は T の **SPR 近傍**と呼ばれる. SPR 操作及び SPR 近傍の概念は系統学において広く知られている. 例えば, 根無し木に対する SPR 近傍の概念については [1] を, 根付き木に対するそれについては [9] や [2] などを見よ.

SPR 操作の定義 1 または 2 において e と f の両方に隣接する枝が存在するとき, あるいは, 定義 3 において v が r (T の根) の子であるとき, T に対する SPR 操作は **NNI 操作** (Nearest Neighbor Interchange operation) と呼ばれ, 関連する枝などを明示したいときには **NNI(e, f)** 操作 (**NNI(e, r)** 操作) と呼ばれる. 図 3(a),(c) で示した SPR(e', f) 操作は NNI 操作である. また, T から 1 回の NNI 操作によって得られる 2 分木は T の **NNI 近傍**と呼ばれる.

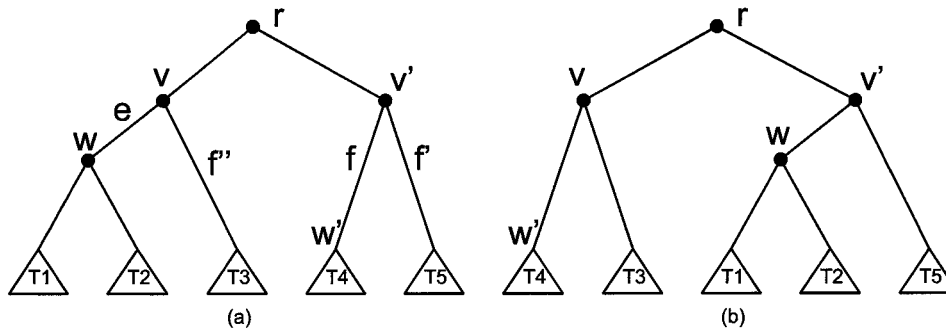

図 4: SE 操作. T_i $(i = 1, \cdots, 5)$ は部分木を表す. (a) T; (b) T に対する SE(e, f) 操作によって得られる二分木 T'.

命題 2.2 (Bordewich and Semple [2]) T と T' を任意の 2 分木とする. すると, T' は NNI 操作の列によって T から得ることができる. したがって, T' は SPR 操作の列によって T から得ることができる.

命題 2.3 T の SPR 近傍の数は $O(n^2)$ である. また, T の NNI 近傍の数は $O(n)$ である.

(証明) 葉の数が n であるような任意の 2 分木 T は $n-1$ 個の内部点を持つので T の点の数は $2n-1$ である. ゆえに, T の枝の数は $2n-2$ である. これより命題の主張がしたがう. □
SPR 近傍の数のより正確な見積もりは Song [9] を見よ.

命題 2.3 より, 局所探索アルゴリズム (アルゴリズム 2) において近傍集合として SPR 近傍を用いた場合は, アルゴリズムの 1 反復あたりの時間計算量は $O(n^4)$ である. 一方で, 近傍集合として NNI 近傍を用いた場合はアルゴリズムの 1 反復あたりの時間計算量は $O(n^3)$ である.

2.3 SE 近傍

$T = (V, E)$ を根 r を持つ 2 分木とする. $e = (v, w)$ は T の枝, $f = (v', w')$ は $v' \neq v$ なる $T \setminus T_w$ の枝で w' は w から r への道上にないものとする. T に対する **SE 操作** (Subtree Exchange operation) とは, T から枝 e, f を削除した後, 枝 $(v, w'), (v', w)$ を挿入する操作である. 図 4 を見よ. SE 操作を, それに関連する 2 本の枝 e, f を明示するために **SE**(e, f) **操作**と呼ぶこともある. T から 1 回の SE 操作を行うことによって得られる 2 分木を T の **SE 近傍**と呼ぶ. SE 操作及び SE 近傍の概念は本研究によって初めて導入される概念である.

命題 2.4 T と T' を任意の 2 分木とする. すると, T' は SE 操作の列によって T から得ることができる.

(証明) T と T' を任意の 2 分木とする. 命題 2.2 より, T' は NNI 操作の列によって T から得ることができる. 任意の 1 回の NNI 操作は 1 回の SE 操作であるため, T' は SE 操作の列によって T から得ることができる. □

命題 2.5 T を任意の 2 分木とする. T から 1 回の SE 操作によって得られる木は, T から 2 回の SPR 操作によって得ることができる.

(証明) 例えば，図 4(a) で示した T に対する SE 操作によって得られる図 4(b) の木 T' は，T に対して SPR(e, f') 操作を行った後，SPR(f, f'') 操作を行うことで得られる．一般の場合も同様である．□

命題 2.6 T を任意の 2 分木とする．T の SE 近傍の数は $O(n^2)$ である．

(証明) 命題 2.3 の証明と同様である．□

命題 2.6 より，局所探索アルゴリズム (アルゴリズム 2) において近傍集合として SE 近傍を用いた場合は，アルゴリズムの 1 反復あたりの時間計算量は $O(n^4)$ である．

2.4 $p = 1$ の場合の局所探索アルゴリズムの高速化

本節では $p = 1$ の場合の L_p-最小増加超距離木問題を考察し，この場合には第 2.1 節で導入した局所探索アルゴリズムが高速化可能であることを示す．

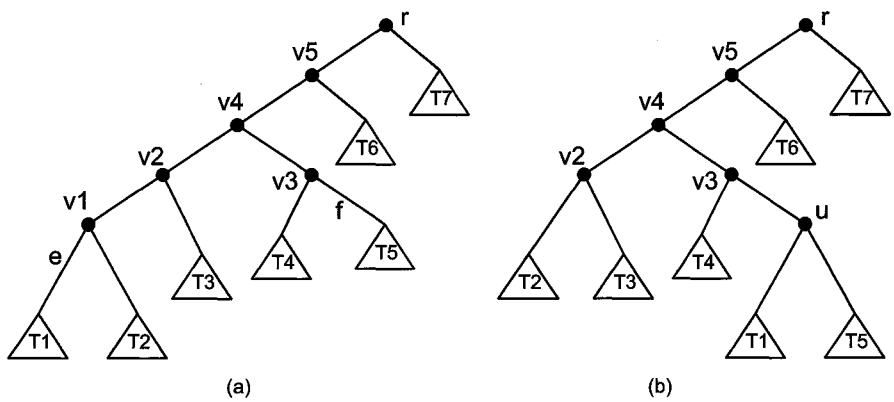

図 5: SPR 操作．T_i $(i = 1, \cdots, 7)$ は部分木を表す．(a) 操作前の 2 分木 T; (b) T に対する SPR(e, f) 操作によって得られる二分木 T'.

アルゴリズム 2 で近傍集合として SPR 近傍を用いる場合を考える．アルゴリズム 2 のある反復において現在得られている 2 分木 T が図 5(a) のようなものであるとする．T に対する最適な重み l は第 2 行あるいは第 6 行で，アルゴリズム 1 を用いて計算済みである．このときに，各点 v に対して $h(v) = \mathrm{height}_{(T,l)}(v)$ を保存しておくものとする．T の SPR 近傍として図 5(a) で示される SPR(e, f) 操作によって得られる木 T' は図 5(b) となる．次にアルゴリズム 2 の第 4 行で M と T' を入力する MUTT 問題をアルゴリズム 1 を用いて求めるのであるが，T' の点で $h(v)$ を計算し直す必要があるのは $v = v_2, v_3, u, v_4, v_5, r$ のみであり，したがって，l' を計算し直す必要がある T' の枝もこれらの点 v から左右の子 u, w に向かう枝のみである．また，これらの点 v に対して

$$2\mathrm{height}_{(T',l')}(v) = D_{(T',l')}[i, j] \quad (i \in L(T'_u), j \in L(T'_w))$$

であり, かつ,

$$\begin{aligned}\|D_{(T',l')}-M\|_1-\|D_{(T,l)}-M\|_1 &= \sum_{i,j\in S}(D_{(T',l')}[i,j]-M[i,j]-D_{(T,l)}[i,j]+M[i,j])\\ &= \sum_{i,j\in S}(D_{(T',l')}[i,j]-D_{(T,l)}[i,j])\\ &= \sum_{v}\sum_{i\in L(T'_u),j\in L(T'_w)}(2\mathrm{height}_{(T',l')}(v)-D_{(T,l)}[i,j])\end{aligned}$$

であるから, $\|D_{(T',l')}-M\|_1-\|D_{(T,l)}-M\|_1$ が非負であるかどうかの判定のための計算も効率化できる. ここで, 最後の式の総和は $\mathrm{height}_{(T',l')}(v)$ の再計算が必要な T' の点 v について取る. また, u,w は T' における v の子である. 以上の観察から, 入力の相違行列 M と 2 分木 T に関する MUTT 問題の最適解 l と $\mathrm{height}_{(T,l)}$ が所与であるときに, T に対する SPR(e,f) 操作によって得られる T' と M に関する MUTT 問題の最適解 l' と差分 $\Delta=\|D_{(T',l')}-M\|_1-\|D_{(T,l)}-M\|_1$ は以下のアルゴリズム 3 によって効率的に求められることが分かる.

入力: S 上の相違行列 M, 根付き 2 分木 T, $l=\mathrm{MUTT}(M,T)$, T に対する SPR(e,f) 操作で得られる 2 分木 T'.
出力: $l'=\mathrm{MUTT}(M,T')$, $\Delta=\|D_{(T',l')}-M\|_1-\|D_{(T,l)}-M\|_1$.

```
1 h = height_(T,l);
2 e = (v(e), w(e)), f = (v(f), w(f)) とし, f は SPR(e,f) 操作によって (v(f), u) と (u, w(f)) に分割されるとする;
3 Q を v(e) の親から T' の根までの道上にあるか, または, u から T' の根までの道上にあるすべての点とする;
4 for 各 v ∈ Q を後行順で do
5 |   u, w ← v の子;
6 |   h(v) ← max{h(u), h(w), M[i,j]/2 | i ∈ L(T'_u), j ∈ L(T'_w)};
7 |   l'(v,u) ← h(v) − h(u), l'(v,w) ← h(v) − h(w);
8 |   for (i,j) ∈ L(T'_u) × L(T'_w) do Δ ← Δ + (2h(v) − D_(T,l)[i,j]) ;
9 end
```

アルゴリズム 3: MUTT 問題に対するアルゴリズムの高速化.

アルゴリズム 3 の SPR(e,f) 操作を NNI(e,f) 操作あるいは SE(e,f) 操作に置き換えても同様のアルゴリズムが得られる.

以下のアルゴリズム 4 は, 第 2 節で導入した局所探索アルゴリズム (アルゴリズム 2) において, 近傍集合 $\mathcal{N}(T)$ として T の SPR 近傍を用い, かつ, MUTT 問題を解くアルゴリズムにアルゴリズム 3 を用いるように書き直したものである. 近傍集合 $\mathcal{N}(T)$ として SE 近傍を用いるものと NNI 近傍を用いるものも同様であるから省略する. 近傍探索を制限する可能性を含むために, SPR(e,f) 操作 (NNI(e,f) 操作, あるいは, SE(e,f) 操作) を T の枝部分集合 $\tilde{E}$ に制限していることに注意せよ. このように近傍に対する制限を行ったアルゴリズムを次節の数値実験で用いる.

3 数値実験

本節では, $p=1$ の場合の L_p-最小増加超距離木問題に対する局所探索アルゴリズム (例えば SPR 近傍を用いる場合は, アルゴリズム 4) を実装し, このアルゴリズムの性能を数値実験によって検

```
入力: S 上の相違行列 M, E の部分集合 Ẽ.
出力: M ≤ D_(T,l) なる超距離木 (T, l).
初期の根付き 2 分木 T = (V, E) を構成する;
l ← MUTT(M, T);
for e ∈ Ẽ do
    for SPR(e, f) 操作が適用可能なすべての f ∈ E ∪ {r} do
        T′ を T に対する SPR(e, f) 操作によって得られる 2 分木とする;
        (l′, Δ) を (M, T, l, T′) を入力として実行したアルゴリズム 3 の出力とする;
        if Δ < 0 then
            (T, l) ← (T′, l′);
        end
    end
end
```

アルゴリズム 4: SPR 近傍を用いる局所探索アルゴリズム.

証する.

3.1 枝集合 $\tilde{E}$ とその順序付け

数値実験では, アルゴリズム 4 において 2 分木の変形操作 (SPR(e,f) 操作, NNI(e,f) 操作, あるいは, SE(e,f) 操作) を行う枝 e の集合 $\tilde{E}$ とその順序付けについて複数のものを使用する.

2 分木 $T=(V,E)$ と自然数 k に対して, $E_k \subseteq E$ を

$$E_k = \{e=(v,w) \mid e \in E, |L(T_w)| = k\} \tag{5}$$

によって定義する. 特に, E_1 は T の茎 (葉に接続する枝) からなる集合である. $\tilde{E} \subseteq E$ とその選択の順序付けとして用いるものは以下の 3 つである.

1. $\tilde{E} = E_1 \cup E_2 \cup E_3$. E_1 の中の枝をそれらの枝が接続する葉の番号の昇順で選択した後, E_2 の枝を任意の順序で選択し, 最後に E_3 の枝を任意の順序で選択する.
2. $\tilde{E} = E$. $\tilde{E}$ の順序は根からの幅優先順である.
3. $\tilde{E} = E$. $\tilde{E}$ の順序は実装における枝番号の昇順である.

現在得られている 2 分木が図 6 の木であるとする. ここで, 枝のラベルは枝番号である. このとき,

$$E_1 = \{8,3,4,10,6,7\}, E_2 = \{1,9\}, E_3 = \{2\}$$

である. アルゴリズム 4 において e は, 方法 1 では $E_1 \cup E_2 \cup E_3$ の枝が $(8,3,4,10,6,7,1,9,2)$ という順序で, 方法 2 では E の枝が $(5,6,9,2,4,10,1,7,8,3)$ という順序で, 方法 3 では E の枝が枝番号の昇順で選択される.

$e=(v,w)$ に対する SPR(e,f) 操作及び SE(e,f) 操作における f の選択の順序は, $T \setminus T_w$ を v を始点とする幅優先探索によって訪問される順番である. 例えば, 現在の 2 分木 T が図 6 で与えられているとすると, $e=2$ に対して f は $(4,10,r,6)$ という順で選択される.

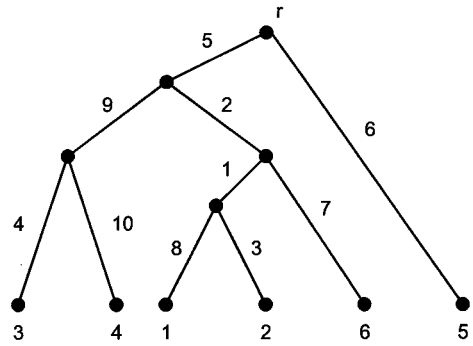

図 6: SPR 操作 (NNI 操作, あるいは, SE 操作) における枝 e の選択. 枝のラベルは枝番号を表す.

3.2 実験方法と環境

入力として与える相違行列として, $[0, 100]$ 上の一様実数乱数を要素とする相違行列とランダムに生成された距離行列を用意する. ランダムな距離行列は以下のように生成される S 上の相違行列 M である. 各 $i \in S$ に対して x_i と y_i を $\left[0, \frac{100}{\sqrt{2}}\right]$ 上の一様実数乱数とし,

$$M[i,j] = \|(x_i, y_i) - (x_j, y_j)\|_2 \quad (i,j \in S)$$

とする.

局所探索アルゴリズムの実装は C 言語で行い, 実験は以下の環境で行なった.

- コンパイラ: gcc バージョン 4.8.4 (Ubuntu 4.8.4-2ubuntu1 14.04.3)
- OS: ubuntu 14.04.4 LTS
- CPU: Intel® Core™ i7-3770 CPU @ 3.40GHz × 8
- メモリ: 7.7 GiB

SPR 近傍を用いる局所探索アルゴリズム (アルゴリズム 4) で, $\tilde{E}$ として第 3.1 節で示した方法 1 を用いるものを SPR(3), 方法 2 を用いるものを SPR(BFS), 方法 3 を用いるものを SPR(edge) によって表す. NII 近傍, 及び, SE 近傍を用いる局所探索アルゴリズム (アルゴリズム 4) についても同様に, NNI(3), NNI(BFS), NNI(edge), 及び, SE(3), SE(BFS), SE(edge) と表す. これら計 9 個の局所探索アルゴリズムを実装してその性能を比較する. また, 必要な初期の 2 分木は Complete-Linkage [4] で得られた解を使用する.

$n = 40, 80, 120, 160, 200$ とした場合についてそれぞれ 5 個の入力を与えて, それぞれの入力に対する各アルゴリズムの初期解から出力された解の L_1-ノルムの減少率と計算時間, 及び, 探索した近傍の総数の平均値を比較した. また, 減少率は

$$1 - \frac{||D_{(T,l)} - M||_1}{||D_{(T_0,l_0)} - M||_1}$$

で計算される. ここで, (T_0, l_0) は初期の超距離木であり, (T, l) はアルゴリズムの出力である.

3.3 実験結果

最初に, 一様実数乱数を要素とする相違行列を入力としたときの結果を示す. 初期解に対する減少率の平均値を表 1 に, 計算時間の平均値を表 2, 探索した近傍の総数の平均値を表 3 に示す.

まず, SPR 操作, NNI 操作, SE 操作の 3 つを比較する. 減少率については SPR 操作を用いるものが最も高く, それに次いで, SE 操作を用いるもの, NNI 操作を用いるものとなっており, これら 3 つの減少率の差は大きい. 計算時間に関しては, SPR 操作を用いるものが最も大きく, それに次いで, SE 操作を用いるもの, NNI 操作を用いるものとなっており, これら 3 つの計算時間の差は大きい.

続いて各 2 分木の変形操作 (SPR, NNI 及び SE 操作) に対して第 3.1 節で述べた $\tilde{E}$ の選択と順序付けに応じた 3 つのアルゴリズムについて比較した結果を述べる. SPR 操作を用いるアルゴリズムの中では, SPR(edge) が他の 2 つに比べてわずかに高い減少率を達成するが, 他の 2 つに比べて約 2 倍の計算時間がかかっている. NNI 操作を用いるアルゴリズムの中では, NNI(BFS) と NNI(edge) は NNI(3) に比べてやや高い減少率を持つが, この 2 つは NNI(3) の 2 倍以上の計算時間がかかっている. SE 操作を用いるアルゴリズムの中では SE(edge) が最も減少率が高いが, それとほぼ同じ減少率を達成する SE(3) の 2 倍以上の時間を要している. また, SE(BFS) は計算時間は最も小さいが, 他の 2 つに比べて減少率が著しく低い. これは探索している近傍の総数が非常に少ないためであると考えられる.

この入力に対する 9 個の局所探索アルゴリズム全てを比較すると, 全ての n に対して最も高い減少率を達成したのは SPR(edge) であるが, それと同程度の減少率であり計算時間が半分以下である SPR(3) が最も有用であると考えられる.

次にランダムな距離行列を入力としたときの結果を示す. 初期解からの減少率の平均値を表 4 に, 計算時間の平均値を表 5, 探索した近傍の総数の平均値を表 6 に示す.

SPR 操作, NNI 操作, SE 操作の 3 つを比較すると, 減少率については SPR 操作を用いるものが最も高く, それに次いで, SE 操作を用いるもの, NNI 操作を用いるものとなっているが, SE 操作を用いるアルゴリズムの減少率は SPR 操作を用いるアルゴリズムのそれとほぼ同じである. 計算時間に関しては, SPR 操作を用いるものが最も大きく, それに次いで, SE 操作を用いるもの, NNI 操作を用いるものとなっており, これら 3 つの計算時間の差は大きい.

続いて, 各 2 分木の変形操作 (SPR, NNI 及び SE 操作) に対して第 3.1 節で述べた $\tilde{E}$ の選択と順序付けに応じた 3 つのアルゴリズムについて比較した結果を述べる. SPR 操作を用いるアルゴリズムの中では SPR(edge) が最も高い減少率を達成しているが残りの 2 つと大きな差はない. 一方で計算時間は, SPR(BFS) が最も小さく, SPR(3) と SPR(edge) の約半分であることが観測された. NNI 操作を用いるアルゴリズムの中では, NNI(BFS) は他の 2 つと比較して高い減少率を達成している. NNI(3) は高速であるが減少率は最も低い. また, NNI(edge) は減少率があまり高くないにもかかわらず, 計算時間が NNI(3) の約 3 倍かかっていることが観測された. SE 操作を用いるアルゴリズムの中では, SE(edge) が最も高い減少率を達成するが計算時間も最も大きい. 一方で SE(3) は SE(edge) の 2 分の 1 以下の計算時間で SE(edge) とほぼ同じ減少率を達成している. SE 操作を用いるアルゴリズムにおいて, 一様乱数を要素とする相違行列を入力としたときと同様に, SE(BFS) の計算時間は最も小さいが残りの 2 つに比べて減少率は非常に低いことが観測された.

この入力に対する 9 個の局所探索アルゴリズム全てを比較すると, 全ての n に対して最も高い減少率を達成したのは SPR(edge) であるが, それより若干減少率は低くなるが計算時間は 10 分の 1 以下である SE(3) が最も有用であると考えられる.

一様実数乱数を要素とする相違行列を入力としたときには全体的に減少率は低いが, ランダム

な距離行列を入力としたときには全体的に減少率は高い．さらに，ランダムな距離行列を入力としたとき，各 n に対して減少率のばらつきが大きいことも観測された．これらの原因は，Complete-Linkage によって得られる超距離木は，一様実数乱数を要素とする相違行列を入力としたときには常に良い近似解を与えているが，ランダムな距離行列が入力であるときにはその近似精度が低いためであると推測される．

4 おわりに

L_p-最小増加超距離木問題は，有限集合 S 上の相違行列 M が与えられたとき，$M \leq D_{(T,l)}$ という条件の下で $||D_{(T,l)} - M||_p$ を最小化する超距離木 (T, l) を求める問題である．本研究では，SPR 操作，NNI 操作及び SE 操作と呼ばれる根付き 2 分木の変形操作に基づいて，p が有限の場合の L_p-最小増加超距離木問題に対する局所探索アルゴリズムを提案した．ここで，SE 操作は本研究において初めて導入された 2 分木の変形操作である．さらに，$p = 1$ の場合にこの局所探索アルゴリズムを高速化する方法を示した．

$p = 1$ の場合の L_p-最小増加超距離木問題に対して提案する局所探索アルゴリズムの性能を評価するために，ランダムに生成された相違行列を入力として数値実験を行なった．ここで，実験に用いた相違行列の次数 n は $n \leq 200$ である．その結果，近似精度と計算時間の両方を勘案すると，入力が一様実数乱数を要素とする相違行列の場合は SPR 操作を用いる局所探索アルゴリズム (SPR(3)) が，入力がランダムな距離行列の場合は SE 操作を用いる局所探索アルゴリズム (SE(3)) が最も有用であるという結論を得た．

より大きいサイズの入力に対しては，SPR 操作及び SE 操作を用いる局所探索アルゴリズムは膨大な時間を要するために，これらのアルゴリズムを用いて局所最適解を得ることは現実的に不可能である．こうした場合に唯一実行可能な選択肢は NNI 操作を用いるものであるが，本研究の実験結果からその近似精度は低いことが予想される．よって，より大きなサイズの入力に対して局所探索アルゴリズムを有用なものとするためには，NNI 操作を用いる局所探索アルゴリズムと同程度の計算時間を持ち，かつ，SPR 操作や SE 操作に近い減少率を達成するように近傍探索の方法を考える必要がある．例えば本研究の実験で用いられた SPR 操作を用いる局所探索アルゴリズムでは SPR(e, f) 操作はそれが適用可能な全ての f に対して実行されたが，SPR(e, f) 操作を実行する f を一部の枝に制限することによって，そのような局所探索アルゴリズムが得られると予想している．

謝辞

本研究は JSPS 科研費 15K00033 の助成を受けたものである．

参考文献

[1] B. L. Allen and M. Steel: Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics* **5** (2001) 1–15.

[2] M. Bordewich and C. Semple: On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics* **8** (2004) 409–423.

[3] W. H. E. Day: Computational complexity of inferring phylogenies from dissimilarity matrices. *Bulletin of Mathematical Biology* **49** (1987) 461–467.

[4] D. Defays: An efficient algorithm for a complete link method. *The Computer Journal* **20** (1977) 364–366.

[5] R. Desper and O. Gascuel: Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology* **9** (2002) 687–705.

[6] M. Farach, S. Kannan and T. Warnow: A robust model for finding optimal evolutionary trees. *Algorithmica* **13** (1995) 155–179.

[7] A. Goëffon, J.-M. Richer and J.-K. Hao: Progressive tree neighborhood applied to the maximum parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **5** (2008) 136–145.

[8] B. Harb, S. Kannan and A. McGregor: Approximating the best-fit tree under Lp norms. In: *Proceedings of 8th APPROX and 9th RANDOM* **99** (2005) 123–133.

[9] Y. S. Song: On the combinatorics of rooted binary phylogenetic trees. *Annals of Combinatorics* **7** (2003) 365–379.

[10] B. Y. Wu, K.-M. Chao and C. Y. Tang: Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. *Journal of Combinatorial Optimization* **3** (1999) 199–211.