Available online at www.sciencedirect.com
SCIENCE DIRECT®

Journal of Discrete Algorithms 2 (2004) 229–255

JOURNAL OF DISCRETE ALGORITHMS

www.elsevier.com/locate/jda

A lower bound for the breakpoint phylogeny problem

David Bryant

McGill Centre for Bioinformatics, McGill University, 3775 University, Montreal H3A 2B4, QC, Canada

Abstract

Breakpoint phylogenies methods have been shown to be an effective tool for extracting phylogenetic information from gene order data. Currently, the only practical breakpoint phylogeny algorithms for the analysis of large genomes with varied gene content are heuristics with no optimality guarantee. Here we begin to address this lack by deriving lower bounds for the breakpoint median problem and for the more complicated breakpoint phylogeny problem. In both cases we employ Lagrange multipliers and sub-gradient optimization to tighten the bounds. The bounds have been implemented and are available as part of the GoTree package (http://www.math.mcgill.ca/bryant/gotree).
© 2003 Elsevier B.V. All rights reserved.

Keywords: Gene order data; Breakpoints; Phylogenetics; Lower bounds; Lagrange multiplies

1. Introduction

1.1. Gene order data

Evolutionary trees (phylogenies) have for a long time been inferred from genetic sequence data. The evolution of sequences has been modeled using a stochastic process involving local changes: insertion, deletion, and replacement of individual nucleotides or amino acids.

Recently, there has been a rapid increase in the number of completely sequenced genomes, giving access to a new source of phylogenetic data: the position of genes along the genome. As genomes evolve, the ordering of genes along the genome can change. Individual genes are inserted or deleted. Segments of the genome can be duplicated, or reversed

E-mail address: bryant@math.mcgill.ca (D. Bryant).

1570-8667/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1570-8667(03)00077-7

(an inversion), or removed and re-inserted in another position (a transposition). This leads to different genomes having equivalent or *homologous* genes arranged in different orders.

How can we use this gene order information to infer evolutionary relationships between the organisms? This question is especially pertinent for groups of organisms, such as the early branching eukaryotes, that have been difficult to analyse using conventional sequence based methods [30].

Gene order data has been used as a source of phylogenetic information for several years (e.g., [25]). Much of the early work in the field follows a distance-based approach. First, an evolutionary distance is estimated for each pair of genomes. Generally, this involved the calculation of the minimum number of mutations (e.g., reversals, transpositions) required to transform one genome into the other [10,25]. Next, these distances are used to construct trees using the variety of distance based methods available in phylogenetics.

A problem with distance based methods is that they give no indication as to the nature of genomes for ancestral species: the nodes in the interior of the tree. This shortcoming motivates the use of a parsimony type method. We try to find a tree and a collection of genomes for internal nodes of the tree such that the total length of the tree is minimum. The length of an edge in the tree is the distance between the gene orders at its endpoints, and the length of the entire tree is simply the sum of all of its edge lengths. Thus, we are looking for a Steiner tree on the space of genomes. Unfortunately, the extension of rearrangement distances to more than two genomes proved to be computationally difficult [5], though recently developed heuristic and Bayesian approaches are very promising [3,15,19].

1.2. Breakpoint methods

The *breakpoint phylogeny method* for the analysis of gene order data, introduced by [1], avoids many (but not all) of the computational difficulties inherent in rearrangement distance based methods. Breakpoints and breakpoint distances will be defined formally later (Section 2). For now we give just the intuitive idea.

Consider the effect of an inversion on a genome (Fig. 1). Two genes 2 and 3 that were adjacent in the original genome A are not adjacent in the resulting genome B; likewise for

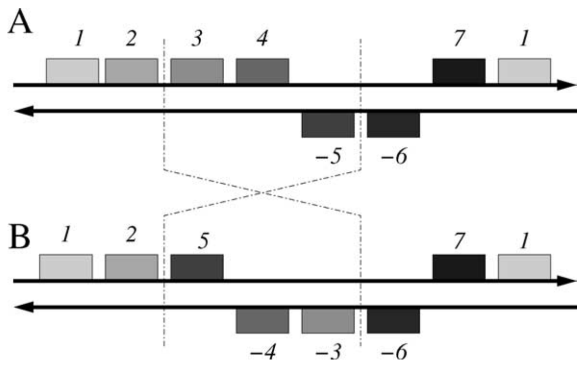

Fig. 1. The effect of an inversion on adjacencies. The inversion causes a segment to be reversed and swapped to the opposite strand.

the pair -5 and -6. These broken adjacencies are called *breakpoints*. Performing further inversions or mutations breaks the adjacencies between further pairs of genes, leading to more and more breakpoints. The breakpoint distance is based on the number of breakpoints between the two genomes, expressed as a proportion of the number of adjacencies. As the genomes become more scrambled there will be more breakpoints and the breakpoint distance increases. The number of breakpoints is a directly observable phenomenon so, in a sense, the breakpoint distance is "model-free".

The breakpoint distance can be extended to the case when there have been deletions and insertions into the genome and the two genomes have different collections of genes (that is, different *gene content*). In this case, the breakpoint distance between two genomes is computed by first removing any genes not appearing in both genomes, and then taking the distance between the two genomes that are left over [30,31].

The definitions of the breakpoint median and breakpoint phylogeny problems follow immediately from the definition of the breakpoint distance. The breakpoint median problem is to find a genome X that minimises the sum of the distances from X to each of a given collection of genomes. The breakpoint phylogeny problem is to find, for a tree with genomes at the leaves, an assignment of genomes for the internal nodes that minimizes the total length of the tree.

1.3. Existing work

The analysis of gene order data using breakpoints was introduced and developed in a series of papers by Blanchette, Bourque, Kunisawa and Sankoff [1,2,27–29]. They show how the breakpoint median problem can be solved by transforming it into an instance of the traveling salesman problem (TSP), provided that all genomes have the same gene content. Thus methods and software for the TSP can be applied directly to the breakpoint median problem.

Methods for the breakpoint median problem can in turn be extended to the breakpoint phylogeny problem using an iterative heuristic. First, the genomes at the internal nodes are initialised in some way (e.g., randomly). Next the program makes repeated passes through the tree, replacing the genome at each node with the median of its neighbouring genomes. In this way the length is decreased, and the procedure converges to a local optima. The method was implemented by Matthieu Blanchette and applied to a collection of animal mitochondrial gene orders. Blanchette's implementation has been recently re-optimized by Moret et al. [20], with impressive improvements in efficiency.

The breakpoint median problem was proved NP-hard by Pe'er and Shamir [21], who also developed an approximation algorithm for the problem [22]. Integral to the approximation algorithm is a lower bound method for the breakpoint median problem. We will discuss this bound in Section 3.6.

Several authors have proposed character encodings of gene orders [6,8]. Gene order data can be converted into character data for use with parsimony methods. These encodings give lower bounds for the breakpoint phylogeny problem. We show in Section 4.5 that these bounds are not as tight as other possible parsimony encodings and not as tight as the bounds developed in this paper.

In all of the work to date, the application of breakpoint methods to the analysis of gene order data has required the use of heuristic methods. To my knowledge, no one has been able to find a provably minimal solution of the breakpoint phylogeny problem for a non-trivial data set. We can only guess at how well our heuristics are working. Even simulation experiments such as those of Cosner et al. [6] do not help us here: the ancestral genomes generated using a random Markov process are not necessarily the most parsimonious.

Furthermore, almost all of the existing work only applies to the rather exceptional case when all of the input gene orders have the same gene content. If we abandon this constraint, the breakpoint problems become significantly more difficult. To date, there have been only a few papers on breakpoint analysis of gene order data for genomes with unequal gene content: Sankoff et al. [30] describe various heuristics and applied them to the analysis of mitochondrial gene order data from early branching eukaryotes; and a diverse collection of chloroplast data was analysed using breakpoint distances in Sankoff et al. [31]. In both cases, there was little guarantee provided that the breakpoint phylogeny criteria had been well optimised.

The determination of ancestral gene orders in a fixed tree is only one component of the problem: we still have to search for the phylogeny of minimum length. Moret et al. [19] show how lower bounds can be used to exclude large numbers of bad trees. They use a simple and fast circular ordering bound (see Section 4.5). The bounds we present here are provably tighter than the circular ordering bounds, even with the *swap-as-you-go* modification of [19]. For this reason, the claim of Moret et al. that circular ordering bounds were tighter than the bounds presented here is puzzling.

Moret et al. also claim that the bounds we present are "very slow"—too slow to justify the additional amount of work required. The connection with character parsimony introduced in Section 4.5 allows us to use PAUP* [32] to quickly evaluate the local optimum bound on huge numbers of trees. It took less than a minute (on a Mac G4) to evaluate all 34 million trees on 11 taxa when analyzing the animal mtDNA gene orders discussed in [2]. The local optimum bound eliminated all but 10,000 trees. In contrast, the circular ordering bound did not eliminate a single tree.

2. Terminology and notation

2.1. Genomes and successors

The genes will be denotated using lower case characters. We assume that we know the *orientation* of each gene, that is, the direction that the gene is read during transcription. The orientation is indicated by the sign of the gene, with $-a$ indicating the reverse orientation of a.

We will assume that the genomes are circular: linear genomes can be handled by inserting an extra symbolic gene representing the ends of the genome. A genome is a circular ordering of signed genes:

$$G = \langle g_1, g_2, \ldots, g_n, g_1 \rangle.$$

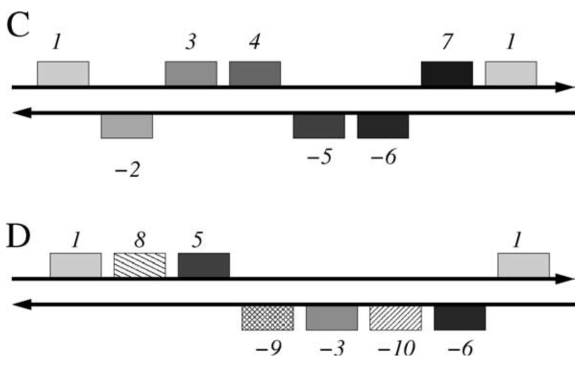

Fig. 2. Two circular genomes with different gene content.

Genome C in Fig. 2 would then be written $\langle 1, -2, 3, 4, -5, -6, 7, 1\rangle$. If we reverse both the order and the orientation of all the genes we get back to the same genome. So $-G = \langle -1, -7, 6, 5, -4, -3, 2, -1\rangle$ is just another way of writing down G. The set of genes of a genome G, with signs removed, is denoted $\mathcal{G}(G)$. For notational convenience we also define $\mathcal{G}^{\pm}(G)$ which contains one positive and one negative copy of each gene in $\mathcal{G}(G)$. Thus $\mathcal{G}(C) = \{1, 2, 3, 4, 5, 6, 7\}$ and

$$\mathcal{G}^{\pm}(C) = \{-7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, 7\}.$$

Given a vector of genomes $\mathbf{A} = [A_1, A_2, \dots, A_N]$ we define $\mathcal{G}(\mathbf{A}) = \bigcup_{i=1}^{N} \mathcal{G}(A_i)$ and $\mathcal{G}^{\pm}(\mathbf{A}) = \bigcup_{i=1}^{N} \mathcal{G}^{\pm}(A_i)$.

The *successor* of a gene g in a genome G is the genome immediately following g, and is denoted $\operatorname{succ}(g, G)$. Thus in the figure, $\operatorname{succ}(1, C) = -2$, $\operatorname{succ}(-2, C) = 3$ and so on. Since C and $-C$ represent the same genome, we also have $\operatorname{succ}(-1, C) = -7$, $\operatorname{succ}(2, C) = -1$ and so on. In general, $\operatorname{succ}(g, G) = h$ if and only if $\operatorname{succ}(-h, G) = -g$. If g is not a gene in G then we define $\operatorname{succ}(g, G) = \emptyset$.

If G is a genome and X is a set of genes, then the induced genome $G|_X$ is obtained by removing all the genes of G that are not in X (either positive or negative), but leaving the rest of the genes in the same order. For example, if $X = \{1, 3, 5, 6\}$ and C is the genome in Fig. 2 then $C|_X = \langle 1, 3, -5, -6, 1\rangle$.

2.2. The breakpoint distance

Let A and B be two genomes. Let $X = \mathcal{G}^{\pm}(A) \cap \mathcal{G}^{\pm}(B)$ be the signed genes they have in common. The (*normalised*) *breakpoint distance* between two genomes A and B is defined

$$d(A, B) = \frac{1}{|X|}\bigl|\bigl\{g \in X\colon \operatorname{succ}(g, A|_X) \neq \operatorname{succ}(g, B|_X)\bigr\}\bigr|. \tag{1}$$

For example, if C and D are the two genomes in Fig. 2 then $X = \{1, -1, 2, -2, 5, -5, 6, -6\}$ and $d(C, D) = \frac{1}{8}4 = \frac{1}{2}$.

Note that the distance $d(A, B)$ is unaffected by genes that only appear in one of the genomes or that do not appear in any of the genomes. Also note that this breakpoint distance differs from that used in [2] by the introduction of the scaling factor $1/|X|$. This factor was introduced by [30] to overcome a problem of the unnormalised breakpoint distances: when there is a great deal of variation in gene content, unnormalised breakpoint distances will tend to produce trees with large genomes forced apart.

2.3. The breakpoint median problem

Let $\mathbf{A} = [A_1, \dots, A_N]$ be a vector of genomes. For any other genome G we define

$$\delta(G, \mathbf{A}) = \sum_{i=1}^{N} d(G, A_i).$$

The breakpoint median problem for $\mathbf{A}$ is to find a genome G with gene set $\mathcal{G}(G) = \mathcal{G}(\mathbf{A})$ such that $\delta(G, \mathbf{A})$ is minimized.

2.4. The breakpoint phylogeny problem

Let $\mathbf{A} = [A_1, \dots, A_N]$ be a vector of genomes and let T be a rooted tree with leaves labelled bijectively by numbers 1 to N. We will assume that T is binary—every internal node has exactly two children. We can make this assumption because every minimal breakpoint phylogeny, as with every Steiner tree, can be extended to a binary tree through the addition of zero length edges. The set $E(T)$ of edges of T is the set of ordered pairs (u, v) such that u is a child of v. We use $\mathrm{par}(v)$ to denote the parent of node v and $\mathrm{root}(T)$ to denote the root of T.

An *assignment* of genomes to T is defined formally as a mapping ϕ from nodes of T to genomes that satisfies

1. If v is a leaf and i is the label of this leaf then $\phi(v) = A_i$.
2. If v is an internal node and v has children u_1 and u_2 then the gene set of $\phi(v)$ is equal to the union of the gene sets of $\phi(u_1)$ and $\phi(u_2)$.

The mapping ϕ describes a history of the evolution of the gene orders that correspond to the leaves. Property (2) models the situation where the differences in gene content are completely explained by deletions. For this to be possible, the gene set of an ancestral gene order must contain all the genes present in the gene orders of its descendents. Any ancestral genes that do not appear in any of the descendent gene orders can be removed from the analysis without affecting the final result. We will always use the Greek letters ϕ or ψ to denote assignments.

To simplify presentation later on we recursively define a gene set $\mathcal{G}^{\pm}(v)$ for each vertex v in T:

1. If v is a leaf with label i then $\mathcal{G}^{\pm}(v) = \mathcal{G}^{\pm}(A_i)$.
2. If v is an internal node with children u_1 and u_2 then $\mathcal{G}^{\pm}(v) = \mathcal{G}^{\pm}(u_1) \cup \mathcal{G}^{\pm}(u_2)$.

Thus for all assignments ϕ and all vertices v in T we have $\mathcal{G}^{\pm}(\phi(v)) = \mathcal{G}^{\pm}(v)$.

Given genomes **A** and tree T, the *length* of an assignment ϕ is defined

$$l(\phi, \mathbf{A}, T) = \sum_{(u,v)\in E(T)} d\big(\phi(u), \phi(v)\big).$$

The *breakpoint phylogeny problem* for **A** and T is to find an assignment ϕ of minimum length.

3. A lower bound for the breakpoint median problem

In this section we describe a new lower bound for the breakpoint median problem. It is inspired by a lower bound used for the TSP, though the conversion of the TSP bound is complicated substantially by the problem of unequal gene content. After describing an initial basic bound, we show how to improve the bound using Lagrange multipliers. This is also a technique borrowed from work on the TSP, and it is also complicated by the unequal gene content. Later on in the paper (Section 4) we show how the breakpoint median bound can be extended to give a lower bound for the breakpoint phylogeny problem. The intuitive idea behind both bounds is the same: understanding the median bound is important for understanding the phylogeny bound.

3.1. *The closest neighbour bound for the TSP*

Suppose that we have an instance of the TSP: a distance matrix D defined on a finite collection of cities $1, 2, \dots, n$. The length of any tour $\tau = \langle x_1, x_2, \dots, x_n, x_1\rangle$ is defined

$$\text{length}(\tau) = \sum_{i=1}^{n} D[x_i, x_{i+1}]$$

where x_{n+1} is identified with x_1. For each $i = 1, \dots, n$ let y_i be the closest city to x_i. Thus $D[x_i, x_{i+1}] \geqslant D[x_i, y_i]$ and

$$\text{length}(\tau) \geqslant \sum_{i=1}^{n} D[x_i, y_i].$$

The right hand side is independent of τ and is therefore a lower bound for any tour length. This TSP lower bound can be viewed as the sum of local optimizations. For each city, we optimize the length of the outgoing step. Summing these local optima gives a lower bound for the global optimum.

Our lower bound for the breakpoint median (and later breakpoint phylogeny) problem works by the same principle. For each signed gene, we solve a ‘localised median problem’ that focuses only on the successors of that gene. The global bound is then found by summing up over all of the signed genes.

3.2. Local breakpoints and local medians

The definition of breakpoint distance (Eq. (1)) incorporates the cardinality of the set:

$$\{g \in X\colon \operatorname{succ}(g, A) \neq \operatorname{succ}(g, B)\}.$$

A standard way to define cardinality is to define a characteristic function for elements of the set and sum over these. We modify this idea to re-express the breakpoint distance.

Let A and B be two genomes and let $X = \mathcal{G}^{\pm}(A) \cap \mathcal{G}^{\pm}(B)$. For each signed gene g define

$$d_g(A, B) = \begin{cases} 1/|X|, & \text{if } g \in X \text{ and } \operatorname{succ}(g, A|_X) \neq \operatorname{succ}(g, B|_X); \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

We can rewrite the breakpoint distance by summing up over all signed genes:

$$d(A, B) = \sum_g d_g(A, B). \tag{3}$$

We decompose the median function δ in a similar way. Let $\mathbf{A}$ be a vector of genomes $[A_1, A_2, \ldots, A_N]$ and define

$$\delta_g(G, \mathbf{A}) = \sum_{i=1}^{N} d_g(G, A_i). \tag{4}$$

Then

$$\delta(G, \mathbf{A}) = \sum_{i=1}^{N} d(G, A_i) \tag{5}$$
$$= \sum_{i=1}^{N} \sum_g d_g(G, A_i) \tag{6}$$
$$= \sum_g \sum_{i=1}^{N} d_g(G, A_i) \tag{7}$$
$$= \sum_g \delta_g(G, \mathbf{A}). \tag{8}$$

3.3. The local optimum bound

Having decomposed the function δ into a sum of functions δ_g, we obtain a bound by optimizing each δ_g separately. The *local optimal bound* for $\mathbf{A}$ is defined

$$L(\mathbf{A}) = \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \min_G \{\delta_g(G, \mathbf{A})\colon \mathcal{G}(G) = \mathcal{G}(\mathbf{A})\}. \tag{9}$$

The next step is to prove that the local optimal bound is indeed a lower bound for the breakpoint median problem.

Lemma 1. *Let* $\mathbf{A}$ *be a vector of genomes. For all genomes* H *such that* $\mathcal{G}(H) = \mathcal{G}(\mathbf{A})$ *we have*

$$\delta(H, \mathbf{A}) \geqslant L(\mathbf{A}). \tag{10}$$

Proof.

$$\begin{aligned}\delta(H, \mathbf{A}) &= \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \delta_g(H, \mathbf{A}) & (11)\\ &\geqslant \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \min_G \bigl\{\delta_g(G, \mathbf{A})\colon \mathcal{G}(G) = \mathcal{G}(\mathbf{A})\bigr\} & (12)\\ &= L(\mathbf{A}). \quad \square & (13)\end{aligned}$$

We now turn to the problem of efficiently computing this bound. In order to minimize the localised median score $\delta_g(G, \mathbf{A})$ we would hope that we only have to consider the genes close to g in the input genomes. This is what we establish in the next lemma.

Lemma 2. *Let* $\mathbf{A} = [A_1, A_2, \ldots, A_N]$ *be a vector of genomes and* g *a signed gene in* $\mathcal{G}^{\pm}(\mathbf{A})$. *There is a genome* G *with* $\mathcal{G}(G) = \mathcal{G}(\mathbf{A})$ *minimizing* $\delta_g(G, \mathbf{A})$ *such that for all* $i = 1, \ldots, N$ *either*

$$\operatorname{succ}(g, G|_{\mathcal{G}(A_i)}) = \emptyset$$

(*that is,* $g \notin \mathcal{G}^{\pm}(A_i)$) *or*

$$\operatorname{succ}(g, G|_{\mathcal{G}(A_i)}) = \operatorname{succ}(g, A_j)$$

for some $j \in \{1, 2, \ldots, N\}$.

Proof. Let H be a genome minimizing $\delta_g(H, \mathbf{A})$ such that $\mathcal{G}(H) = \mathcal{G}(\mathbf{A})$. Starting with $\operatorname{succ}(g, H)$ we proceed along the genome H, considering each gene h in turn. If $h \neq \operatorname{succ}(g, A_j)$ for all $j = 1, \ldots, N$ then we remove h from the genome H and re-insert it directly before the gene g. We then pass to the gene that was originally the successor of h in H. We continue this way until we have considered all of the genes in $\mathcal{G}(H) - \{g, -g\}$. Let G be the resulting genome.

If $g \notin \mathcal{G}^{\pm}(A_i)$ or $\operatorname{succ}(g, H|_{\mathcal{G}(A_i)}) = \operatorname{succ}(g, A_j)$ for some $j = 1, \ldots, N$ then

$$\operatorname{succ}(g, G|_{\mathcal{G}(A_i)}) = \operatorname{succ}(g, H|_{\mathcal{G}(A_i)})$$

and so $d_g(G, A_i) = d_g(H, A_i)$. On the other hand, if $g \in \mathcal{G}^{\pm}(A_i)$ but $\operatorname{succ}(g, H|_{\mathcal{G}(A_i)}) \neq \operatorname{succ}(g, A_j)$ for all $j = 1, \ldots, N$ then

$$\operatorname{succ}(g, H|_{\mathcal{G}(A_i)}) \neq \operatorname{succ}(g, A_i)$$

and so

$$d_g(H, A_i) = \frac{1}{|\mathcal{G}^{\pm}(A_i) \cap \mathcal{G}^{\pm}(G)|} \geqslant d_g(G, A_i).$$

G	g	h_1	h_2	h_3	h_4	$\dots$
$G\vert_{\mathcal{G}(A_1)}$	g	$\mathbf{h_1}$	h_2		h_4	$\dots$
$G\vert_{\mathcal{G}(A_2)}$	g	$\mathbf{h_1}$	h_2	h_3	h_4	$\dots$
$G\vert_{\mathcal{G}(A_3)}$	g			$\mathbf{h_3}$	h_4	$\dots$
$G\vert_{\mathcal{G}(A_4)}$	g		$\mathbf{h_2}$		h_4	$\dots$
$G\vert_{\mathcal{G}(A_5)}$		h_1		h_3	h_4	$\dots$

Fig. 3. A genome G and five induced genomes $G|_{\mathcal{G}(A_1)}, G|_{\mathcal{G}(A_2)}, \dots, G|_{\mathcal{G}(A_5)}$, providing an example of the choice of induced successors for a gene. The choices for $\operatorname{succ}(g, G|_{A_i})$ are in boldface.

It follows that $\delta_g(G, \mathbf{A}) \leqslant \delta_g(H, \mathbf{A})$. Since H was already minimal, so is G and it is a genome satisfying the conditions of the lemma. $\square$

Lemma 2 tells us that to find the minimum value for $\delta_g(G, \mathbf{A})$ we need only consider all the possible choices of genes for each $\operatorname{succ}(g, G|_{\mathcal{G}(A_i)})$, $i = 1, \dots, N$. Furthermore, these genes all come from the set of successors $\{\operatorname{succ}(g, A_j)\colon\ j = 1, \dots, N\}$. These two observations are almost enough for a polynomial time algorithm to minimize $\delta_g(G, \mathbf{A})$ (for bounded N). We merely have to check all choices of genes for $\operatorname{succ}(g, G|_{\mathcal{G}(A_i)})$, $i = 1, \dots, N$, that can be realised by some genome G. This we can do using a search tree. We proceed by way of an example.

Fig. 3 represents a candidate genome G and the five induced genomes

$$G|_{\mathcal{G}(A_1)},\ G|_{\mathcal{G}(A_2)},\ \dots,\ G|_{\mathcal{G}(A_5)}.$$

The gene g is present in only four of these. The first successor of g, h_1, appears in two genomes. The next successor h_2 appears in three, two of which (A_1 and A_2) contain h_1. The third successor appears in two genomes, one of which does not contain h_1 or h_2. At this point, all of the successors $\operatorname{succ}(g, G|_{\mathcal{G}(A_i)})$, $i = 1, \dots, 5$, have been chosen.

To generalise this example: we are looking for a sequence of signed genes $h_1, h_2, \dots, h_k$ such that

(P1) Each $h_i = \operatorname{succ}(g, A_j)$ for some $j \in \{1, 2, \dots, N\}$.
(P2) For each h_i there is a genome A_j such that $\{g, h_i\} \subseteq \mathcal{G}^{\pm}(A_j)$ but $h_k \notin \mathcal{G}^{\pm}(A_j)$ for all $1 \leqslant k < i$.

Now we have enough to sufficiently limit our search space:

Theorem 3. *The lower bound $L(\mathbf{A})$ can be computed in polynomial time when either the number of genomes is bounded or all genomes have equal gene content.*

Proof. For each signed gene $g \in \mathcal{G}^{\pm}(\mathbf{A})$ we need to search through all sequences satisfying properties (P1) and (P2) above. The maximum length of such a sequence is at most N, and for each h_i there are at most $(N - i)$ possibilities, as (P2) rules out the possibility of repetitions. Hence the number of sequences to examine is at most $O(N!)$. In the special case that all genes have the same gene content, (P2) forces the maximum sequence length to be one, so the number of sequences is $O(N)$. $\square$

Theorem 3 leads to a polynomial time algorithm when the genomes have equal gene content, or there is only a bounded number of genomes. In many practical applications, such as the iterate median heuristic of [30], the number of genomes in the median problem is bounded by two or three. However, it is important to ask whether a fast algorithm exists for the general case. The answer to this question appears to be "probably no":

Theorem 4. *It is an NP-hard problem to compute $L(\mathbf{A})$ for a vector of genomes $[A_1, \dots, A_N]$ with unequal gene sets. Hence it is also NP-hard to minimize $\delta_g(G, \mathbf{A})$ for a signed gene g.*

Proof. We provide a reduction from FEEDBACK ARC SET [13], which is NP-complete even when there is cycle of length two. Let the directed graph $G = (V, E)$ and number $K \leqslant |E|$ make up an arbitrary instance of FEEDBACK ARC SET satisfying this condition. Put $\mathcal{G} = V \cup \{x\}$ where x is a new gene. Label the arcs in E as $(a_1, a'_1), \dots, (a_m, a'_m)$. Construct a vector of genomes

$$\mathbf{A} = \big[\langle x, a_1, a'_1, x\rangle,\ \langle x, a_2, a'_2, x\rangle, \dots,\ \langle x, a_m, a'_m, x\rangle\big] \tag{14}$$

which all have genes in $\mathcal{G}$. We claim that $G = (V, E)$ has a feedback arc set $E' \subseteq E$ of size K if and only if $L(\mathbf{A}) \leqslant K$.

Fix $a \in V$, and let $B = \{b\colon (a, b) \in E\}$, let $b_1, \dots, b_k$ be an arbitrary ordering of B, and consider the subsequence $a, b_1, \dots, b_k, x$. By inserting all other genes directly before x we obtain a genome H_a for which $\delta_a(H_a, \mathbf{A}) = 0$. The same trick (in reverse) gives a genome H_{-a} such that $\delta_{-a}(H_{-a}, \mathbf{A}) = 0$. Hence

$$L(\mathbf{A}) = \frac{1}{2}\min\big\{\delta_x(H, \mathbf{A})\colon \mathcal{G}^{\pm}(H) = \mathcal{G}^{\pm}(\mathbf{A})\big\} + \frac{1}{2}\min\big\{\delta_{-x}(H, \mathbf{A})\colon \mathcal{G}^{\pm}(H) = \mathcal{G}^{\pm}(\mathbf{A})\big\}.$$

Suppose that E' is a feedback arc set of size K. Let $a_1, \dots, a_n$ be an ordering of V such that for each arc $(a_i, a_j) \in E - E'$ we have $i < j$. Consider $H = \langle x, a_1, a_2, \dots, a_n, x\rangle$. Then $\delta_x(H, \mathbf{A})$ is the number of arcs $(a_j, a_i) \in E$ such that $j > i$, and $\delta_{-x}(H, \mathbf{A}) = \delta_x(H, \mathbf{A})$. Hence $L(\mathbf{A}) \leqslant |E'| \leqslant K$.

Conversely, suppose that $L(\mathbf{A}) \leqslant K$. Without loss of generality there is a genome H such that $\delta_x(H, \mathbf{A}) \leqslant K$. Write H as $H = \langle x, a_1, a_2, \dots, a_n, x\rangle$. There are at most K genomes $\langle x, a_j, a_i, x\rangle$ in $\mathbf{A}$ such that $j > i$, and the corresponding edges form a feedback arc set for G. □

We stress that in most current applications the number of genomes N in the breakpoint median problem is bounded (typically $N \leqslant 3$). In these cases, the above NP-hardness result is irrelevant.

3.4. Improving the bound with Lagrange multipliers

Lagrange multipliers provide us with a ratchet technique for cranking up the lower bound by applying a system of weights. For each set of weights, we obtain a new bound—the goal is to find a set of weights that produces as large a lower bound as we can.

To illustrate the concept, return once again to the standard TSP and the "closest cities" bound that we derived before. Suppose that in addition to distances between cities we also assign a cost for visiting each city; the cost of a hotel if you like. Let $c(x_i)$ denote the cost of each visit to city x_i. Let y_i denote the cheapest and closest city to x_i, that is, the city for which $D[x_i, y_i] + c(y_i)$ is minimum. For any tour τ we have

$$\text{length}(\tau) = \sum_{i=1}^{n}\bigl(D[x_i, x_{i+1}] + c(x_{i+1})\bigr) - \sum_{i=1}^{n} c(x_i) \tag{15}$$

$$\geqslant \sum_{i=1}^{n}\bigl(D[x_i, y_i] + c(y_i)\bigr) - \sum_{i=1}^{n} c(x_i). \tag{16}$$

The right hand side is independent of τ so is a lower bound for any tour length, no matter which values we choose for the costs $c(x_i)$. Clearly, different choices of costs are going to give different bounds. We are able to choose a set of costs that gives the largest bound, keeping in mind that any choice of costs still gives a guaranteed lower bound.

We return to the breakpoint median problem. Let $\mathbf{A}$ be the set of input genomes. The analogue of a hotel cost is a *Lagrange multiplier*. We define one Lagrange multiplier for each genome index $i = 1, \ldots, N$ and for each signed gene $g \in \mathcal{G}^{\pm}(A_i)$. We denote this Lagrange multiplier by $\lambda[i, g]$ and the array of Lagrange multiplies by $\boldsymbol{\lambda}$. For convenience of notation we define $\lambda[i, \emptyset] = 0$ for all i and $\lambda[i, g] = 0$ for all $g \notin \mathcal{G}^{\pm}(A_i)$. This simplifies several formulae further on.

Now to the analogue of the "closest cheapest" city. For each signed gene g define

$$\delta_g(G, \mathbf{A}, \boldsymbol{\lambda}) = \delta_g(G, \mathbf{A}) + \sum_{i=1}^{N} \lambda\bigl[i, \operatorname{succ}(g, G|_{\mathcal{G}(A_i)})\bigr] \tag{17}$$

which is equivalent to

$$\delta_g(G, \mathbf{A}, \boldsymbol{\lambda}) = \sum_{i=1}^{N}\bigl(\delta_g(G, A_i) + \lambda\bigl[i, \operatorname{succ}(g, G|_{\mathcal{G}(A_i)})\bigr]\bigr). \tag{18}$$

The *weighted local optimum bound* is then defined

$$L(\mathbf{A}, \boldsymbol{\lambda}) = \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \min_{G} \bigl\{\delta_g(G, \mathbf{A}, \boldsymbol{\lambda}) \colon \mathcal{G}(G) = \mathcal{G}(\mathbf{A})\bigr\} - \frac{1}{2}\sum_{i=1}^{N} \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \lambda[i, g]. \tag{19}$$

We prove that, for all choices of $\boldsymbol{\lambda}$, the weighted local optimum bound $L(\mathbf{A}, \boldsymbol{\lambda})$ is a lower bound.

Lemma 5. *Let* $\mathbf{A}$ *be a vector of genomes and let* $\boldsymbol{\lambda}$ *be a vector of Lagrange multipliers for signed genes in* $\mathbf{A}$. *For all genomes* H *such that* $\mathcal{G}(H) = \mathcal{G}(\mathbf{A})$ *we have*

$$\delta(H, \mathbf{A}) \geqslant L(\mathbf{A}, \lambda). \tag{20}$$

Proof.

$$\begin{aligned}
\delta(H,\mathbf{A}) &= \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \delta_g(H,\mathbf{A}) \\
&= \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \left(\delta_g(H,\mathbf{A}) + \sum_{i=1}^{N} \lambda[i,g]\right) - \sum_{i=1}^{N} \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \lambda[i,g] \\
&= \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \delta_g(H,\mathbf{A},\boldsymbol{\lambda}) - \sum_{i=1}^{N} \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \lambda[i,g] \\
&\geqslant \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \min_G \bigl\{\delta_g(G,\mathbf{A},\boldsymbol{\lambda})\colon \mathcal{G}(G) = \mathcal{G}(\mathbf{A})\bigr\} - \sum_{i=1}^{N} \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \lambda[i,g] \\
&= L(\mathbf{A},\boldsymbol{\lambda}). \quad \square
\end{aligned}$$

Once again, we turn to computational issues. The introduction of weights makes the problem of minimizing $\delta_g(G,\mathbf{A},\boldsymbol{\lambda})$ harder than the problem of minimizing $\delta_g(G,\mathbf{A})$. For one thing, Lemma 2 no longer holds: the addition of weights means that there may be no genome minimizing $\delta_g(G,\mathbf{A},\boldsymbol{\lambda})$ for which all of the successors genes $\operatorname{succ}(g, G_{\mathcal{G}(A_i)})$ come from $\{\operatorname{succ}(g,A_j)\colon j=1,\dots,N\}$. Our first step is to increase the set which is guaranteed to contain the optimal successor genes.

Let I_{in} and I_{out} be two disjoint subsets of $\{1,2,\dots,N\}$. Let $X[I_{\text{in}},I_{\text{out}}]$ be the set of signed genes defined as follows

$$X[I_{\text{in}},I_{\text{out}}] = \bigl\{h\in\mathcal{G}^{\pm}(\mathbf{A})\colon\ h\in\mathcal{G}^{\pm}(A_i) \text{ for all } i\in I_{\text{in}}, \text{ and } h\notin\mathcal{G}^{\pm}(A_i) \text{ for all } i\in I_{\text{out}}\bigr\}.$$

In pseudo-English: $X[I_{\text{in}},I_{\text{out}}]$ contains all of the genes that appear in every one of the genomes with indices in I_{in} but none of the genomes with indices in I_{out}. For example, in Fig. 3, $X[\{1,2\},\{3\}] = \{h_1,h_2\}$, $X[\{1,2\},\{4\}] = \{h_1\}$, $X[\{2\},\{1,4\}] = \{h_3\}$ and $X[\{2\},\{4,5\}] = \emptyset$.

For each choice of I_{in} and I_{out} such that $X[I_{\text{in}},I_{\text{out}}]$ is non-empty we choose a signed gene $x\in X[I_{\text{in}},I_{\text{out}}]$ for which $\sum_{i\in I_{\text{in}}}\lambda[i,x]$ is minimal and a second gene $y\in X[I_{\text{in}},I_{\text{out}}] - \{x,-x\}$ for which $\sum_{i\in I_{\text{in}}}\lambda[i,y]$ is minimal. We use $M[\mathbf{A},\boldsymbol{\lambda}]$ to denote the set of all the x's and y's that are chosen for at least one I_{in} and I_{out}.

Lemma 6. *Let $\mathbf{A} = [A_1,A_2,\dots,A_N]$ be a vector of genomes, $\boldsymbol{\lambda}$ an array of Lagrange multipliers associated to $\mathbf{A}$, and g a signed gene in $\mathcal{G}^{\pm}(\mathbf{A})$. There is a genome G with $\mathcal{G}(G)=\mathcal{G}(\mathbf{A})$ minimizing $\delta_g(G,\mathbf{A})$ such that for all $i=1,\dots,N$ either*

$$\operatorname{succ}(g, G|_{\mathcal{G}(A_i)}) = \emptyset$$

or

$$\operatorname{succ}(g, G|_{\mathcal{G}(A_i)}) = \operatorname{succ}(g, A_j)$$

for some $j \in \{1, 2, \dots, N\}$ *or*

$$\operatorname{succ}(g, G|_{\mathcal{G}(A_i)}) \in M[\mathbf{A}, \boldsymbol{\lambda}].$$

Proof. Let H be a genome minimizing $\delta_g(H, \mathbf{A})$ such that $\mathcal{G}(H) = \mathcal{G}(\mathbf{A})$ that does not satisfy the conditions of the lemma. Let $h_1, h_2, \dots, h_m$ be the signed genes

$$\big\{\operatorname{succ}(g, H|_{\mathcal{G}(A_i)}):\ i = 1, \dots, N\big\}$$

in the same order that they appear after g in the genome H. Let h_k be the first gene in this sequence that is not a member of $M[\mathbf{A}, \boldsymbol{\lambda}]$ and does not equal $\operatorname{succ}(g, A_j)$ for some j.

Set

$$I_{\mathrm{in}} = \big\{i:\ \operatorname{succ}(g, H|_{\mathcal{G}(A_i)}) = h_k\big\}$$

and

$$I_{\mathrm{out}} = \big\{i:\ \operatorname{succ}(g, H|_{\mathcal{G}(A_i)}) \in \{h_{k+1}, h_{k+2}, \dots, h_m\}\big\}.$$

Then I_{in} contains those genomes of which h_k is a member but no gene $h_1, h_2, \dots, h_{k-1}$ is a member while I_{out} contains those genomes of which none of $h_1, h_2, \dots, h_k$ are members. Furthermore, $h_k \in X[I_{\mathrm{in}}, I_{\mathrm{out}}]$.

Let x be a gene in $M[\mathbf{A}, \boldsymbol{\lambda}]$ that minimizes $\sum_{i \in I_{\mathrm{in}}} \lambda[i, x]$ over all $x \in X[I_{\mathrm{in}}, I_{\mathrm{out}}] - \{g, -g\}$. Since the genes $h_1, \dots, h_{k-1}$ are not genes in genome A_i for any $i \in I_{\mathrm{in}}$ the gene x cannot equal any of genes $h_1, \dots, h_{k-1}$. Similarly, since x is not in any of the genomes A_i for $i \in I_{\mathrm{out}}$, the gene x does not equal any of $h_{k+1}, \dots, h_m$.

We modify the genome H by swapping h_k with x. Let H' be the genome we obtain. Then

$$\begin{aligned}
\delta_g(H', \mathbf{A}, \boldsymbol{\lambda}) &= \sum_{i \in I_{\mathrm{in}}} \big(d_g(H', A_i) + \lambda\big[i, \operatorname{succ}(H'|_{\mathcal{G}(A_i)})\big]\big) \\
&\quad + \sum_{i \notin I_{\mathrm{in}}} \big(d_g(H', A_i) + \lambda\big[i, \operatorname{succ}(H'|_{\mathcal{G}(A_i)})\big]\big) \\
&= \sum_{i \in I_{\mathrm{in}}} \big(d_g(H', A_i) + \lambda[i, x]\big) + \sum_{i \notin I_{\mathrm{in}}} \big(d_g(H, A_i) + \lambda\big[i, \operatorname{succ}(H|_{\mathcal{G}(A_i)})\big]\big) \\
&\leqslant \sum_{i \in I_{\mathrm{in}}} \big(d_g(H, A_i) + \lambda[i, h_k]\big) + \sum_{i \notin I_{\mathrm{in}}} \big(d_g(H, A_i) + \lambda\big[i, \operatorname{succ}(H|_{\mathcal{G}(A_i)})\big]\big) \\
&= \delta_g(H, \mathbf{A}, \boldsymbol{\lambda}).
\end{aligned}$$

Since H is already minimal, so must be H'. We can repeat the process, increasing k until we obtain a minimal genome satisfying the conditions of the lemma. □

The problem of searching for genomes to minimize $\delta_g(G, \mathbf{A}, \boldsymbol{\lambda})$ is therefore almost the same as the unweighted problem, except, of course, that we have to consider a larger range of genes at each node of the search tree. Specifically, we are searching for sequences of signed genes $h_1, h_2, \dots, h_k$ that satisfy

(P1′) Each h_i is either in $M[\mathbf{A}, \boldsymbol{\lambda}]$ or equals $\mathrm{succ}(g, A_j)$ for some $j \in \{1, 2, \ldots, N\}$.
(P2) For each h_i there is a genome A_j such that $\{g, h_i\} \subseteq \mathcal{G}^{\pm}(A_j)$ but $h_k \notin \mathcal{G}^{\pm}(A_j)$ for all $1 \leqslant k < i$.

We now have

Theorem 7. *The lower bound $L(\mathbf{A}, \boldsymbol{\lambda})$ can be computed in polynomial time when either the number of genomes is bounded or all genomes have equal gene content.*

Of course, when N is unbounded and the genomes have unequal gene content, the NP-hardness result of Theorem 4 extends to the weighted case.

3.5. Choosing the best Lagrange multipliers: sub-gradient optimization

We now have a way of computing a lower bound $L(\mathbf{A}, \boldsymbol{\lambda})$ for each choice of $\boldsymbol{\lambda}$. To take advantage of this feature, we want $\boldsymbol{\lambda}$ that gives the largest, and therefore tightest, bound possible. To this end, we need to study the function taking $\boldsymbol{\lambda}$ to $L(\mathbf{A}, \boldsymbol{\lambda})$. This function shares many of the properties of the lower bound function for the Held–Karp bound (cf. [11,12,23]), including continuity, piecewise linearity, and concavity. Hence we are able to use a technique called sub-gradient optimization to find an optimal, or hopefully close to optimal, choice for $\boldsymbol{\lambda}$.

First we need an ascent direction.

Suppose that we have calculated $L(\mathbf{A}, \boldsymbol{\lambda})$ for some choice of $\boldsymbol{\lambda}$. To do this, we took each signed gene $g \in \mathcal{G}^{\pm}(\mathbf{A})$ in turn and determined a genome H_g optimizing $\delta_g(H_g, \mathbf{A}, \boldsymbol{\lambda})$. In actual fact, we were really only interested in the successor genes $\mathrm{succ}(g, H_g|_{\mathcal{G}^{\pm}(A_i)})$, $i = 1, \ldots, N$ as the rest of H_g does not affect the score.

Fix a signed gene h and index i such that $h \in \mathcal{G}^{\pm}(A_i)$. Let $f[i, h]$ denote the number of genes g for which the optimal genome H_g that we found satisfies $\mathrm{succ}(g, H_g|_{\mathcal{G}^{\pm}(A_i)}) = h$. That is, $f[i, h]$ is the number of time that h gets chosen as a successor of g in some $H_g|_{\mathcal{G}^{\pm}(A_i)}$.

Define the array $\boldsymbol{\Delta} = (\Delta[i, h])$ by

$$\Delta[i, h] = f[i, h] - 1 \tag{21}$$

for all $i = 1, \ldots, N$ and $h \in \mathcal{G}^{\pm}(A_i)$. When $h \notin \mathcal{G}^{\pm}(A_i)$ we let $\Delta[i, h] = 0$. The array $\boldsymbol{\Delta}$ has the same dimensions as $\boldsymbol{\lambda}$ and provides our sub-gradient vector.

Theorem 8. *If $\boldsymbol{\Delta}$ is non-zero then there is $\varepsilon > 0$ such that $L(\mathbf{A}, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) > L(\mathbf{A}, \boldsymbol{\lambda})$.*

Proof. The function $L(\mathbf{A}, \boldsymbol{\lambda})$ is piecewise linear: the space of all $\boldsymbol{\lambda}$ can thus be divided up into closed regions on which every $f[i, h]$ is constant and $L(\mathbf{A}, \boldsymbol{\lambda})$ is linear. There is $\varepsilon > 0$ such that $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}$ belong to the same region. This is true even if $\boldsymbol{\lambda}$ lies on the boundary between two regions.

For each $g \in \mathcal{G}^{\pm}(\mathbf{A})$ there is a genome H_g that minimizes both $\delta_g(H, \mathbf{A}, \boldsymbol{\lambda})$ and $\delta_g(H, \mathbf{A}, \boldsymbol{\lambda} + \varepsilon\Delta)$. Expanding $\delta_g(H, \mathbf{A}, \boldsymbol{\lambda})$ we obtain

$$\delta_g(H_g, \mathbf{A}, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) - \delta_g(H_g, \mathbf{A}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} \varepsilon\Delta\big[i, \operatorname{succ}(g, H_g|_{\mathcal{G}(A_i)})\big]. \tag{22}$$

Thus

$$\begin{aligned}
&L(\mathbf{A}, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) - L(\mathbf{A}, \boldsymbol{\lambda}) \\
&\quad = \left(\sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \big(\delta_g(H_g, \mathbf{A}, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) - \delta_g(H_g, \mathbf{A}, \boldsymbol{\lambda})\big)\right) - \left(\sum_{i=1}^{N} \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \varepsilon\Delta[i, g]\right) \\
&\quad = \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{i=1}^{N} \varepsilon\Delta\big[i, \operatorname{succ}(g, H_g|_{\mathcal{G}^{\pm}(A_i)})\big] - \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{i=1}^{N} \varepsilon\Delta[i, g] \\
&\quad = \varepsilon \sum_{h\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{i=1}^{N} \Delta[i, h]\big(f[i, h] - 1\big) \\
&\quad = \varepsilon \sum_{h\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{i=1}^{N} \big(\Delta[i, h]\big)^2 \\
&\quad > 0,
\end{aligned} \tag{23}$$

since $\boldsymbol{\Delta} \neq 0$. □

We therefore have an ascent direction. The next question is how far to go in this direction. We have implemented three strategies:

Simple line search. The first strategy was to use the simple line search algorithm already being used for the Held–Karp bound (cf. [23], p. 176). The algorithm is given an initial step length and correction factor. At each iteration we proceed along the sub-gradient vector by the given step length, and then shorten the step length using the correction factor. Eventually the algorithm will either reach a local (and therefore global) optimum, or grind to a halt as the step length becomes too small. One obvious shortcoming of this approach is that we have to come up with some value for the initial step length.

Exact line search. The fact that the function is concave allows us to perform exact line searches. Consider the function $L(\mathbf{A}, \boldsymbol{\lambda} + t\boldsymbol{\Delta})$ for $t \geqslant 0$. Since $\boldsymbol{\Delta}$ is a sub-gradient, the function is initially increasing. The function is concave, so it will ascend monotonically, reach a maximum, then descend. We search for the first value $t_1 > 0$ such that $L(\mathbf{A}, \boldsymbol{\lambda} + t_1\boldsymbol{\Delta}) \leqslant L(\mathbf{A}, \boldsymbol{\lambda})$. The maximum must then equal $L(\mathbf{A}, \boldsymbol{\lambda} + t\boldsymbol{\Delta})$ for some $t \in [0, t_1]$. We subdivide the interval $[0, t_1]$ into three equal segments: $[0, t_2]$, $[t_2, t_3]$, $[t_3, t_1]$. If $L(\mathbf{A}, \boldsymbol{\lambda} + t_2\boldsymbol{\Delta}) < L(\mathbf{A}, \boldsymbol{\lambda} + t_3\boldsymbol{\Delta})$ then the maximum occurs when $t \in [t_2, t_1]$, otherwise when $t \in [0, t_3]$. Recursing, we obtain a small interval that is guaranteed to contain the maximum.

Almost exact line search. Given that exact line searching takes a lot of time, and is not necessarily the best strategy, we also tried an approximate line search. After locating the smallest $t_1 > 0$ such that $L(\mathbf{A}, \boldsymbol{\lambda} + t_1\boldsymbol{\Delta}) \leqslant L(\mathbf{A}, \boldsymbol{\lambda})$ we use the mid-point $\boldsymbol{\lambda} + t_1/2\boldsymbol{\Delta}$ as the return value of the line search.

3.6. Comparison with other breakpoint median bounds

The lower bounds $L(\mathbf{A})$ and $\max_{\boldsymbol{\lambda}} L(\mathbf{A}, \boldsymbol{\lambda})$ are currently the only two bounds that we can use for the general breakpoint median problem. If we forget this for the moment and concentrate on the equal gene set case we find several bounds that we can make comparisons with.

First of all, when all genomes have equal gene content the breakpoint distance becomes a metric. We can therefore apply the standard three point Steiner bound:

$$\delta\big(G, [A_1, A_2, A_3]\big) \geqslant \frac{1}{2}\big(d(A_1, A_2) + d(A_1, A_3) + d(A_2, A_3)\big).$$

As usual, this bound is quite weak. Even the simple local optimum bound improves on the Steiner bound. We can show that for all vectors $\mathbf{A} = [A_1, A_2, A_3]$ of three genomes with equal gene content,

$$\frac{1}{2}\big(d(A_1, A_2) + d(A_1, A_3) + d(A_2, A_3)\big) \leqslant L(\mathbf{A}) \leqslant \max_{\boldsymbol{\lambda}} L(\mathbf{A}, \boldsymbol{\lambda}).$$

Another breakpoint median bound appears as part of the approximation algorithm developed by Pe'er and Shamir [22]. One can show that the bound they use is at least as tight as $L(\mathbf{A})$. The bound has not been implemented, so at the moment we don't know whether the bound also improves on $\max_{\boldsymbol{\lambda}} L(\mathbf{A}, \boldsymbol{\lambda})$. In any case, the Pe'er and Shamir bound is restricted to only three genomes with equal gene content. There appears to be no direct extension of the bound for a larger number of genomes nor for genomes with unequal gene content.

The transformation from the breakpoint median problem to the TSP provides even further scope for new and tighter lower and upper bounds. However, like the Pe'er and Shamir bound, this transformation breaks down completely when the genomes have different gene content and does not help us with the general breakpoint median problem.

4. A lower bound for the breakpoint phylogeny problem

Everything we did for medians we now do for trees. We will decompose the length of a tree into local scores, define local optima, and use these to derive a lower bound. We then show how Lagrange multipliers can be applied to this problem, and how dynamic programming can be used to compute the value of the bound for each choice of multipliers. Apart from the complications added by dealing with trees and larger numbers of genomes, the intuition behind the breakpoint median and the breakpoint phylogeny bounds is the same.

4.1. Local breakpoints and local breakpoint phylogenies

In Section 3.2 we saw how the breakpoint distance $d(A, B)$ between two genomes can be decomposed into the sum of local breakpoint distances $d_g(A, B)$. Likewise, the median score $\delta_g(G, \mathbf{A})$ can be expressed as the sum of local median scores $\delta_g(G, \mathbf{A})$. Here we do the same for tree length.

The length of an assignment ϕ was defined in Section 2.4 as

$$l(\phi, \mathbf{A}, T) = \sum_{(u,v)\in E(T)} d\big(\phi(u), \phi(v)\big).$$

For each signed gene $g \in \mathcal{G}^{\pm}(\mathbf{A})$ we define

$$l_g(\phi, \mathbf{A}, T) = \sum_{(u,v)\in E(T)} d_g\big(\phi(u), \phi(v)\big).$$

Then

$$\begin{aligned} l(\phi, \mathbf{A}, T) &= \sum_{(u,v)\in E(T)} \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} d_g\big(\phi(u), \phi(v)\big) && (24)\\ &= \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{(u,v)\in E(T)} d_g\big(\phi(u), \phi(v)\big) && (25)\\ &= \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} l_g(\phi, \mathbf{A}, T). && (26) \end{aligned}$$

4.2. The local optimum bound for phylogenies

We have decomposed the length $l(\phi, \mathbf{A}, T)$ of an assignment ϕ into the sum of local lengths $l_g(\phi, \mathbf{A}, T)$. To derive a lower bound on $l(\phi, \mathbf{A}, T)$ we optimize each of these local lengths separately and then sum. The *local optimum bound* for $\mathbf{A}$ and T is defined

$$L(\mathbf{A}, T) = \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \min_{\psi} \big\{ l_g(\psi, \mathbf{A}, T)\colon\ \psi \text{ is an assignment for } \mathbf{A} \text{ and } T \big\}. \qquad (27)$$

Lemma 9. *Let* $\mathbf{A} = [A_1, A_2, \dots, A_N]$ *be a vector of genomes and let* T *be a tree with leaves labelled* $1, 2, \dots, N$. *For all assignments* ϕ *for* $\mathbf{A}$ *and* T *we have* $l(\phi, \mathbf{A}, T) \geqslant L(\mathbf{A}, T)$.

Proof.

$$\begin{aligned} L(\mathbf{A}, T) &= \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \min_{\psi} \big\{ l_g(\psi, \mathbf{A}, T)\colon\ \psi \text{ is an assignment for } \mathbf{A} \text{ and } T \big\} && (28)\\ &\leqslant \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} l_g(\phi, \mathbf{A}, T) && (29)\\ &= l(\phi, \mathbf{A}, T). \quad \square && (30) \end{aligned}$$

We can compute $l(\phi, \mathbf{A}, T)$ in $O(n^2 N)$ time using dynamic programming, though we will omit details and correctness proof because the problem can be solved using the algorithm we present in the following section.

4.3. Improving the bound with Lagrange multipliers

We have seen how Lagrange multipliers can be applied to the breakpoint median problem. Here we show how to apply them to the breakpoint phylogeny problem. We use one Lagrange multiplier for each node v in the tree and each signed gene $g \in \mathcal{G}^{\pm}(v)$. (The set $\mathcal{G}^{\pm}(v)$ was defined in Section 2.4.) Denote this Lagrange multiplier by $\lambda[v, g]$ and the array of Lagrange multipliers by $\boldsymbol{\lambda}$. As before, we define $\lambda[v, \emptyset] = 0$ for all v and $\lambda[v, g] = 0$ for all $g \notin \mathcal{G}^{\pm}(v)$.

Let $par(v)$ denote the parent of v in T and root(T) the root of T. Let ϕ be an assignment for T. For each signed gene g define

$$l_g(\phi, \mathbf{A}, T, \boldsymbol{\lambda}) = l_g(\phi, \mathbf{A}, T) + \sum_{u \neq \mathrm{root}(T)} \lambda\bigl[u, \mathrm{succ}\bigl(g, \phi\bigl(\mathrm{par}(u)\bigr)\big|_{\mathcal{G}^{\pm}(u)}\bigr)\bigr] \tag{31}$$

which is equivalent to

$$l_g(\phi, \mathbf{A}, T, \boldsymbol{\lambda}) = \sum_{(u,v) \in E(T)} d_g\bigl(\phi(u), \phi(v)\bigr) + \lambda\bigl[u, \mathrm{succ}\bigl(g, \phi(v)|_{\mathcal{G}^{\pm}(u)}\bigr)\bigr]. \tag{32}$$

The *weighted local optimum bound* is then defined

$$L(\mathbf{A}, T, \boldsymbol{\lambda}) = \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \min_{\psi} \bigl\{ l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda})\colon \psi \text{ is an assignment for } T \text{ and } \mathbf{A} \bigr\} \tag{33}$$

$$- \sum_{v \neq \mathrm{root}(T)} \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \lambda[v, g]. \tag{34}$$

For all choices of $\boldsymbol{\lambda}$, the bound $L(\mathbf{A}, T, \boldsymbol{\lambda})$ is a lower bound for $l(\phi, \mathbf{A}, T)$:

Lemma 10. *Let $\mathbf{A} = [A_1, A_2, \ldots, A_N]$ be a vector of genomes and T a tree with leaves labelled $1, 2, \ldots, N$. For all assignments ϕ for T and $\mathbf{A}$ and all choices of $\boldsymbol{\lambda}$ we have $l(\phi, \mathbf{A}, T) \geqslant L(\mathbf{A}, T, \boldsymbol{\lambda})$.*

Proof.

$$\begin{aligned} l(\phi, \mathbf{A}, T) &= \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} l_g(\phi, \mathbf{A}, T) \\ &= \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} l_g(\phi, \mathbf{A}, T, \boldsymbol{\lambda}) - \sum_{v \neq \mathrm{root}(T)} \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \lambda[v, g] \\ &\geqslant \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \min_{\psi} \bigl\{ l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda})\colon \psi \text{ is an assignment for } T \text{ and } \mathbf{A} \bigr\} \\ &\quad - \sum_{v \neq \mathrm{root}(T)} \sum_{g \in \mathcal{G}^{\pm}(\mathbf{A})} \lambda[v, g] \\ &= L(\mathbf{A}, T, \boldsymbol{\lambda}). \quad \square \end{aligned} \tag{35}$$

We return to computational issues. We show that $L(\mathbf{A}, T, \boldsymbol{\lambda})$ can also be computed in $\mathrm{O}(n^2 N)$ time, where n is the number of genes and N the number of genomes. Let g be

```
foreach node $v$ in a post-order traversal of $T$ do
    if $v$ is a leaf then
        Let $i$ be the label of $v$.
        $m[v,h] = \begin{cases} 0 & \text{if } h = \text{succ}(g, A_i) \\ \infty & \text{otherwise} \end{cases}$
    else
        Set $m[v,h] \leftarrow 0$ for all $h \in \mathcal{G}^{\pm}(v)$. Let $u_1$ and $u_2$ be the children of $v$.
        foreach child $u_i$ such that $g \in \mathcal{G}^{\pm}(u_i)$ do
            $X_i \leftarrow \min_x\{m[u_i, x] + \lambda[u_i, x]\colon\ x \in \mathcal{G}^{\pm}(u_i) - \{g, -g\}\}$.
            $Y_i \leftarrow \min_y\{m[u_i, y]\colon\ y \in \mathcal{G}^{\pm}(u_i) - \{g, -g\}\}$.
            $Z_i \leftarrow \min_z\{m[u_i, z]\colon\ z \in \mathcal{G}^{\pm}(u_i) - \{g, -g\}\}$.
            $\epsilon \leftarrow \frac{1}{|\mathcal{G}^{\pm}(u_i)|}$.
            foreach signed gene $h \in \mathcal{G}^{\pm}(v) - \{g, -g\}$ do
                if $h \in \mathcal{G}^{\pm}(u_i)$ then
                    $m[v,h] \leftarrow m[v,h] + \lambda[v,h] + \min\{m[u_i,h], Y_i + \epsilon\}$.
                else
                    $m[v,h] \leftarrow m[v,h] + \min\{X_i, Y_i + Z_i + \epsilon\}$.
Let $v$ be the root of $T$.
Return $\min_h\{m[v,h]\colon\ h \notin \{g, -g\}\}$. end.
```

Algorithm 1. Dynamic programming algorithm that computes $\min_{\psi} l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda})$.

a signed gene in $\mathcal{G}^{\pm}(\mathbf{A})$. We construct a array $m[v,h]$ indexed by nodes of T and signed genes in $\mathcal{G}^{\pm}(\mathbf{A})$. Algorithm 1 fills in the array and returns the minimum of $l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda})$ over all valid assignments ψ.

Theorem 11. *If v_0 is the root of T and m is the array constructed using Algorithm* 1 *then*

$$\min\bigl\{m[v_0,h]\colon\ h \notin \{g,-g\}\bigr\} = \min_{\psi}\bigl\{l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda})\colon\ \psi \text{ is an assignment for } \mathbf{A} \text{ and } T\bigr\}.$$

Proof. For each node v of T let T_v denote the subtree of T rooted at v, let $E(T_v)$ denote the edges in this subtree and define

$$l_g(\phi, \mathbf{A}, T_v) = \sum_{(x,y)\in E(T_v)} \Bigl(d_g\bigl(\phi(x), \phi(y)\bigr) + \lambda\bigl[x, \text{succ}\bigl(g, \phi(y)|_{\mathcal{G}^{\pm}(x)}\bigr)\bigr]\Bigr). \tag{36}$$

Thus if v_0 is the root of T then $T = T_{v_0}$ and $l_g(\phi, \mathbf{A}, T_{v_0}) = l_g(\phi, \mathbf{A}, T)$.

We claim that after the algorithm has finished, the value of $m[v,h]$ equals the minimum of $l_g(\psi, \mathbf{A}, T_v, \boldsymbol{\lambda})$ over all assignments ψ such that $\text{succ}(g, \psi(v)) = h$. The proof is by induction on the height of v.

Induction base. If v is a leaf which has label i, and $g \in \mathcal{G}(v)$, then $l_g(\psi, \mathbf{A}, T_v, \boldsymbol{\lambda}) = 0$ for all assignments ψ. By the definition of an assignment we must have $\text{succ}(g, \psi(v)) = \text{succ}(g, A_i)$ for all assignments ψ. Thus we set $m[v,h] = 0$ for $h = \text{succ}(g, A_i)$ and $m[v,h] = \infty$ for $h \neq \text{succ}(g, A_i)$.

Induction step. Next suppose that v is not a leaf and that the claim holds for all descendents of v. We will construct an assignment ϕ for which

- $\operatorname{succ}(g, \phi(v)) = h$,
- $l_g(\phi, \mathbf{A}, T_v, \boldsymbol{\lambda}) = m[v, h]$, and
- $l_g(\phi, \mathbf{A}, T_v, \boldsymbol{\lambda}) \leqslant l_g(\psi, \mathbf{A}, T_v, \boldsymbol{\lambda})$ for all assignments ψ with $\operatorname{succ}(g, \psi(v))$ equal to h.

We initialise ϕ as any assignment for which $\operatorname{succ}(g, \phi(v)) = h$.

Let u_1 and u_2 be the children of v. Fix $i \in \{1, 2\}$ and consider three cases (i), (ii) and (iii).

Case (i). $g \in \mathcal{G}^{\pm}(u_i)$ and $h \in \mathcal{G}^{\pm}(u_i)$.
Choose y that minimizes $m[u_i, y]$. If

$$m[u_i, h] \leqslant m[u_i, y] + \frac{1}{|\mathcal{G}^{\pm}(u_i)|}$$

then set $\hat{h} = h$, otherwise set $\hat{h} = y$. By the induction hypothesis we can set $\phi(x)$ for all nodes x in T_{u_i} so that $\operatorname{succ}(g, \phi(u_i)) = \hat{h}$ and $l_g(\phi, T_{u_i}, \mathbf{A}, \boldsymbol{\lambda}) = m[u_i, \hat{h}]$. For all assignments ψ such that $\operatorname{succ}(g, \psi(v)) = h$ we have

$$\begin{aligned}
&l_g(\psi, T_{u_i}, \mathbf{A}, \boldsymbol{\lambda}) + d_g\big(\psi(u_i), \psi(v)\big) + \lambda[u_i, h] \\
&\quad \geqslant \min\left\{m[u_i, h], m[u_i, y] + \frac{1}{|\mathcal{G}^{\pm}(u_i)|}\right\} + \lambda[u_i, h] \\
&\quad = l_g(\phi, T_{u_i}, \mathbf{A}, \boldsymbol{\lambda}) + d_g\big(\phi(u_i), \phi(v)\big) + \lambda[u_i, h],
\end{aligned}$$

noting that

$$\lambda\big[u_i, \phi(v)|_{\mathcal{G}^{\pm}(u_i)}\big] = \lambda\big[u_i, \psi(v)|_{\mathcal{G}^{\pm}(u_i)}\big] = \lambda[u_i, h].$$

Case (ii). $g \notin \mathcal{G}^{\pm}(u_i)$.

For any assignment ψ we have

$$l_g(\psi, \mathbf{A}, T_{u_i}, \boldsymbol{\lambda}) + d_g\big(\psi(u_i), \psi(v)\big) = 0.$$

For each node x in T_{u_i} we set $\phi(x)$ to be an arbitrary genome with genes $\mathcal{G}^{\pm}(v)$. For all assignments ψ such that $\operatorname{succ}(g, \psi(v)) = h$ we then have

$$\begin{aligned}
0 &= l_g(\psi, \mathbf{A}, T_{u_i}, \boldsymbol{\lambda}) + d_g\big(\psi(u_i), \psi(v)\big) + \lambda\big[u_i, \psi(v)|_{\mathcal{G}^{\pm}(u_i)}\big] && (37)\\
&= l_g(\phi, \mathbf{A}, T_{u_i}, \boldsymbol{\lambda}) + d_g\big(\phi(u_i), \psi(v)\big) + \lambda\big[u_i, \phi(v)|_{\mathcal{G}^{\pm}(u_i)}\big], && (38)
\end{aligned}$$

noting that

$$\lambda\big[u_i, \phi(v)|_{\mathcal{G}^{\pm}(u_i)}\big] = \lambda\big[u_i, \psi(v)|_{\mathcal{G}^{\pm}(u_i)}\big] = \lambda[u_i, \emptyset] = 0.$$

Case (iii). $g \in \mathcal{G}^{\pm}(u_i)$ but $h \notin \mathcal{G}^{\pm}(u_i)$.

Choose $x, y, z \in \mathcal{G}^{\pm}(u_i) - \{g, -g\}$ that minimize $m[u_i, x] + \lambda[u_i, x]$, $m[u_i, y]$, and $\lambda[u_i, z]$ respectively. If

$$m[u_i, y] + \lambda[u_i, z] + \frac{1}{|\mathcal{G}^{\pm}(u_i)|} < m[u_i, x] + \lambda[u_i, x], \tag{39}$$

then set $\hat{h} = y$, remove z from $\phi(v)$ and insert it directly after h. Thus $\lambda[u_i, \phi(v)|_{\mathcal{G}^\pm(u_i)}] = z$. If (39) does not hold then we set $\hat{h} = x$, remove x from $\phi(v)$ and insert it directly after h, giving $\lambda[u_i, \phi(v)|_{\mathcal{G}^\pm(u_i)}] = x$. Case (iii) applies to at most one of the children u_1 and u_2 so there is conflict between the two children over the successor of h in $\phi(v)$.

By the induction hypothesis we can set $\phi(x)$ for all nodes x in T_{u_i} so that $\operatorname{succ}(g, \phi(u_i)) = \hat{h}$ and $l_g(\phi, \mathbf{A}, T_{u_i}, \boldsymbol{\lambda}) = m[u_i, \hat{h}]$. For all assignments ψ such that $\operatorname{succ}(g, \psi(v)) = h$ we have

$$\begin{aligned}
&l_g(\psi, T_{u_i}, \mathbf{A}, \boldsymbol{\lambda}) + d_g\big(\psi(u_i), \psi(v)\big) + \lambda\big[u_i, \psi(v)|_{\mathcal{G}^\pm(u_i)}\big] \\
&\quad \geqslant \min\left\{m[u_i, x] + \lambda[u_i, x],\ m[u_i, y] + \frac{1}{|\mathcal{G}^\pm(u_i)|} + \lambda[u_i, z]\right\} \\
&\quad = l_g(\phi, \mathbf{A}, T_{u_i}, \boldsymbol{\lambda}) + d_g\big(\phi(u_i), \phi(v)\big) + \lambda\big[u_i, \phi(v)|_{\mathcal{G}^\pm(u_i)}\big].
\end{aligned}$$

Bringing things together, we have that for all assignments ψ such that $\operatorname{succ}(g, \psi(v)) = h$,

$$\begin{aligned}
l_g(\psi, \mathbf{A}, T_v, \boldsymbol{\lambda}) &= \sum_{i=1}^{2} l_g(\psi, T_{u_i}, \mathbf{A}, \boldsymbol{\lambda}) + d_g\big(\psi(u_i), \psi(v)\big) + \lambda\big[u_i, \psi(v)|_{\mathcal{G}^\pm(u_i)}\big] \\
&\geqslant \sum_{i=1}^{2} l_g(\phi, T_{u_i}, \mathbf{A}, \boldsymbol{\lambda}) + d_g\big(\phi(u_i), \phi(v)\big) + \lambda\big[u_i, \phi(v)|_{\mathcal{G}^\pm(u_i)}\big] \\
&= l_g(\phi, \mathbf{A}, T_v, \boldsymbol{\lambda}).
\end{aligned}$$

Furthermore, $l_g(\phi, \mathbf{A}, T_v, \boldsymbol{\lambda})$ equals the value $m[v, h]$ computed by Algorithm 1.

This proves the induction hypothesis. To prove the theorem we observe that if ϕ minimizes $l_g(\phi, \mathbf{A}, T, \boldsymbol{\lambda})$ and v_0 is the root of T then

$$l_g(\phi, \mathbf{A}, T, \boldsymbol{\lambda}) = m\big[v_0, \operatorname{succ}\big(g, \phi(v)\big)\big] = \min\big\{m[v_0, h]\colon\ h \in \mathcal{G}^\pm v_0\big\}. \quad \square$$

4.4. *Choosing the best Lagrange multipliers: sub-gradient optimization*

As with the breakpoint median problem, we want to find values for $\boldsymbol{\lambda}$ such that $L(\mathbf{A}, T, \boldsymbol{\lambda})$ is maximized. It is, of course, not necessary to find a global optimum: every choice of $\boldsymbol{\lambda}$ gives a lower bound. Nevertheless we do want to find the best bound that we can.

We are faced, then, with the problem of maximizing the high dimensional, concave, continuous and piecewise linear function $L(\mathbf{A}, T, \boldsymbol{\lambda})$. Once again, we employ sub-gradient optimization.

For each signed gene $g \in \mathcal{G}^\pm(\mathbf{A})$ let ψ_g be an assignment that minimizes $l_g(\psi_g, \mathbf{A}, T, \boldsymbol{\lambda})$. For each node $u \neq \operatorname{root}(T)$ and each signed gene $h \in \mathcal{G}^\pm(u)$ define

$$f[u, h] = \big|\big\{g\colon\ \psi_g\big(\operatorname{par}(u)\big)\big|_{\mathcal{G}^\pm(u)} = h\big\}\big|. \tag{40}$$

In pseudo-English, $f[u, h]$ is the number of genes g for which h equals the successor of g in the genome $\psi_g(v)$ restricted to $\mathcal{G}^\pm(u)$, where v is parent of u.

Define the array $\boldsymbol{\Delta} = (\Delta[v, h])$ by

$$\Delta[v, h] = f[v, h] - 1 \tag{41}$$

for all nodes $v \neq \mathrm{root}(T)$ and $h \in \mathcal{G}^{\pm}(v)$. When $h \notin \mathcal{G}^{\pm}(v)$ we let $\Delta[v,h] = 0$. The array $\boldsymbol{\Delta}$ provides the sub-gradient. The following is the phylogeny analogue of Theorem 8.

Theorem 12. *If $\boldsymbol{\Delta}$ is non-zero then there is $\varepsilon > 0$ such that $L(\mathbf{A}, T, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) > L(\mathbf{A}, T, \boldsymbol{\lambda})$.*

Proof. The function $L(\mathbf{A}, T, \boldsymbol{\lambda})$ is piecewise linear: the space of all $\boldsymbol{\lambda}$ can be divided up into closed regions on which every $f[v,h]$ is constant and $L(\mathbf{A}, T, \boldsymbol{\lambda})$ is linear. There is $\varepsilon > 0$ such that $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}$ belong to the same region. This is true even if $\boldsymbol{\lambda}$ lies on the boundary between two regions.

For each $g \in \mathcal{G}^{\pm}(\mathbf{A})$ there is an assignment ψ_g that minimizes both $l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda})$ and $l_g(\psi, \mathbf{A}, T, \boldsymbol{\lambda} + \varepsilon\Delta)$. Expanding $l_g(\psi_g, \mathbf{A}, T, \boldsymbol{\lambda})$ we obtain

$$\begin{aligned} &l_g(\psi_g, \mathbf{A}, T, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) - l_g(\psi_g, \mathbf{A}, T, \boldsymbol{\lambda}) \\ &\quad = \sum_{(u,v)\in E(T)} \varepsilon\Delta\big[u, \mathrm{succ}\big(g, \psi_g(v)|_{\mathcal{G}(u)}\big)\big]. \end{aligned} \tag{42}$$

Thus

$$\begin{aligned} &L(\mathbf{A}, T, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) - L(\mathbf{A}, T, \boldsymbol{\lambda}) \\ &\quad = \left(\sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \big(l_g(\psi_g, \mathbf{A}, T, \boldsymbol{\lambda} + \varepsilon\boldsymbol{\Delta}) - \delta_g(\psi_g, \mathbf{A}, T, \boldsymbol{\lambda})\big)\right) \\ &\qquad - \left(\sum_{(u,v)\in E(T)} \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \varepsilon\Delta[u,g]\right) \\ &\quad = \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{(u,v)\in E(T)} \varepsilon\Delta\big[u, \mathrm{succ}\big(g, \psi_g(v)|_{\mathcal{G}^{\pm}(u)}\big)\big] - \sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{i=1}^{N} \varepsilon\Delta[i,g] \\ &\quad = \sum_{h\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{u\neq\mathrm{root}(T)} \varepsilon\Delta[u,h]\big(f[u,h]-1\big) \\ &\quad = \sum_{h\in\mathcal{G}^{\pm}(\mathbf{A})} \sum_{u\neq\mathrm{root}(T)} \varepsilon\big(f[u,h]-1\big)\big(f[u,h]-1\big) \\ &\quad > 0. \quad \square \end{aligned}$$

Thus $\boldsymbol{\Delta}$ is an ascent direction. All three line-search strategies are available in the implementation (see Section 5).

4.5. Comparison with other breakpoint phylogeny lower bounds

There are currently no other lower bounds for the breakpoint phylogeny problem with genomes that have unequal gene sets. Hence we consider the case when all genomes in $\mathbf{A}$ have equal gene sets, and compare the existing bounds with the breakpoint phylogeny bounds we have derived.

The normalised breakpoint distance becomes a metric when we restrict our attention to genomes on the same gene set. We can therefore apply the standard “once around the tree

Steiner bound" as suggested by [19]. Let $a_1, a_2, \ldots, a_n$ be an ordering of the leaf labels $1, 2, \ldots, N$ given by a pre-order traversal of T. If we define

$$ST(\mathbf{A}, T) = \frac{1}{2}\big(d(A_{a_1}, A_{a_2}) + d(A_{a_2}, A_{a_3}) + \cdots + d(A_{a_{N-1}}, A_{a_N}) + d(A_{a_N}, A_{a_1})\big)$$

and then for all assignments ϕ,

$$ST(\mathbf{A}, T) \leqslant l(\phi, \mathbf{A}, T).$$

This bound applies to all metric spaces irrespective of their structure, so one would expect it be quite weak. It is, however, very quick to compute and [19] claim that it eliminated a large proportion of bad trees when applied to their one real and multiple simulated datasets.

The two additional bounds that we consider stem from two different character encodings of gene order data. In the first, Maximum parsimony on binary encodings of genomes (MPBE) [6], we use binary characters to denote whether or not a particular adjacency is present in a genome. There is one site for each ordered pair of signed genes (a, b). The sequence representing a genome A_i has a 1 at the site corresponding to (a, b) if $\mathrm{succ}(a, A_i) = b$, and 0 otherwise. A second encoding, with we shall call the SB-encoding, was introduced by Sankoff and Blanchette in quite a different context [29]. In this encoding, the sequences have one site for each signed gene and the state set equals the set of signed genes. The sequence representing a genome A_i has $\mathrm{succ}(g, A_i)$ at the site corresponding to signed gene g.

With both encodings, we use the Hamming distance to measure the difference between sequences. Let $MPBE(\mathbf{A}, T)$ denote the minimum length of T under the MPBE encoding and let $SB(\mathbf{A}, T)$ denote the minimum length of T under the SB-encoding.

Theorem 13. *Let $\mathbf{A} = [A_1, A_2, \ldots, A_N]$ be a vector of genomes, all with the same set of genes. Let T be a tree with leaves labelled by $1, 2, \ldots, N$. Let ϕ be an arbitrary assignment for T and $\mathbf{A}$. Then*

$$\begin{aligned} ST(\mathbf{A}, T) &\leqslant \frac{1}{|\mathcal{G}^{\pm}(\mathbf{A})|} MPBE(\mathbf{A}, T) \leqslant \frac{1}{|\mathcal{G}^{\pm}(\mathbf{A})|} SB(\mathbf{A}, T) = L(\mathbf{A}, T) \\ &\leqslant \max_{\boldsymbol{\lambda}} \big\{L(\mathbf{A}, T, \boldsymbol{\lambda})\big\} \leqslant l(\phi, \mathbf{A}, T). \end{aligned}$$

Proof. The space of binary encodings with metric $1/|\mathcal{G}^{\pm}(\mathbf{A})|$ times the Hamming distance is a metric space for which the distance between encodings of genomes equal the respective normalised breakpoint distances. Since the Steiner bound holds for all metric spaces, $ST(\mathbf{A}, T) \leqslant \frac{1}{|\mathcal{G}^{\pm}(\mathbf{A})|} MPBE(\mathbf{A}, T)$.

Every SB sequence S can be converted into an MPBE encoding: the MPBE sequence has a 1 at position (a, b) if the SB sequence has a b at position a; otherwise it has 0. The Hamming distance between two converted SB sequences equals their original distance. We therefore have an isometric mapping from SB sequences to MPBE sequences. Not every MPBE sequence can be converted into a corresponding SB sequence. For example, the binary character encoding may have a 1 at positions (a, b) and (a, c) for $b \neq c$. Therefore the isometric mapping takes SB sequences into a strict subset of MPBE sequences. Finding a minimum length assignment for SB encodings on a tree T is therefore equivalent to finding a minimum length assignment on a restricted subset of MPBE sequences. This restriction can only increase the minumum possible length.

To prove that $SB(\mathbf{A},T) = \frac{1}{|\mathcal{G}^{\pm}(\mathbf{A})|}L(\mathbf{A},T)$ we observe that

$$\min_{\psi}\left\{l_g(\psi,\mathbf{A},T)\colon\ \psi \text{ is an assignment for } T \text{ and } \mathbf{A}\right\}$$

equals $1/|\mathcal{G}(\mathbf{A})|$ times the minimum number of changes along the tree for the site corresponding to g in the SB encoding of the genomes. Hence

$$\begin{aligned}
&\frac{1}{|\mathcal{G}^{\pm}(\mathbf{A})|}SB(\mathbf{A},T)\\
&\quad = \frac{1}{2}\sum_{g\in\mathcal{G}^{\pm}(\mathbf{A})}\min_{\psi}\left\{l_g(\psi,\mathbf{A},T)\colon\ \psi \text{ is an assignment for } \mathbf{A} \text{ and } T\right\} && (43)\\
&\quad = L(\mathbf{A},T). && (44)
\end{aligned}$$

The remaining inequalities follow from Lemma 10 and the observation that $L(\mathbf{A},T,\boldsymbol{\lambda}) = L(\mathbf{A},T)$ when $\boldsymbol{\lambda} = 0$. □

Theorem 13 has considerable practical implications for tree searching. Using the SB encoding, we can compute local optimum bounds using efficient phylogenetic estimation software such as PAUP* [32].

5. Discussion and future work

The lower bounds described in this paper have been implemented as part of the package GoTree, available from http://www.mcb.mcgill.ca/~bryant. The package is based on the nexus class library of Paul Lewis [16].

I have applied the bound to the gene order data sets analysed in [2,30], each time comparing the lower bounds to heuristic upper bound on a number of different trees. The results were mixed. In general:

1. The lower bound for the breakpoint median problem fell within 1% (on average) of the upper bound when genes had equal gene content.
2. The lower bound for the breakpoint phylogeny fell within 8% (on average) of the upper bound when genomes had equal gene content.
3. For both problems, the bound worsened considerably when genomes had unequal gene content: the average median bound fell to 12% of the upper bound while the phylogeny bound dropped to 50% of the upper bound.
4. In all cases, Lagrangian optimization led to a significant improvement in the bound.
5. Implementation accuracy plays a significant role in any simulation experiment.

These preliminary observations can be tested by applying the methods to a large and varying range of gene order data. One could also attempt simulation experiments with synthetic data to detect potential biases and areas for improvement. Such experiments must be conducted with care, though. The underlying processes of the evolution of gene order are still not well understood, and we can only draw limited conclusions using simplistic probabilistic models. Secondly, the performance of the bounds can only be measured compared to heuristic upper bounds, as we have as yet no means for generating random synthetic data for which the most parsimonious assignment is known.

However the real test of these bounds will be in application, particularly in algorithm development and tree searching. We are currently exploring their incorporation into a branch and bound technique. There are many computational obstacles to overcome.

A second direction for future work is the extension of these results to other measures of gene order divergence. Upper bound heuristics for parsimony based on edit distances have been developed by [3,19], though, as for breakpoints, there is little guarantee that the heuristics are obtaining global optima. Moret et al. [18,19] use randomly generated data to argue that edit based heuristics outperform breakpoint heuristics when estimating phylogenies and ancestral gene orders. Whether or not the same applies for real genomes, in all contexts, depends on the accuracy and breadth of the model used to generate the synthetic data. The development and validation of models for gene order evolution is an area of active and ongoing research.

6. For further reading

The following references could also be of interest to the reader: [4,7,9,14,17,24,26].

References

[1] M. Blanchette, G. Bourque, D. Sankoff, Breakpoint phylogenies, in: S. Miyano, T. Takagi (Eds.), Genome Informatics, Universal Academy Press, 1997, pp. 5–34.

[2] M. Blanchette, T. Kunisawa, D. Sankoff, Gene order breakpoint evidence in animal mitochondrial phylogeny, J. Molecular Evolution 49 (1999) 193–203.

[3] G. Bourque, P. Pevzner, Genome-scale evolution: reconstructing gene orders in the ancestral species, Genome Res. 12 (2002) 26–36.

[4] D. Bryant, M. Deneault, D. Sankoff, The breakpoint median problem, Centre de recherches mathématiques, Université de Montréal. ms, 1999.

[5] A. Caprara, Formulations and hardness of multiple sorting by reversals, in: S. Istrail, P.A. Pevzner, M. Waterman (Eds.), Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB 99), ACM, New York, 1999, pp. 84–93.

[6] M.E. Cosner, R.K. Jansen, B.M.E. Moret, L.A. Raubeson, L. Wang, T. Warnow, S. Wyman, An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae, in: D. Sankoff, J.H. Nadeau (Eds.), Comparative Genomics, Kluwer, Dordrecht, 2000, pp. 99–121.

[7] W.M. Fitch, Toward defining the course of evolution: minimal change for a specific tree topology, Syst. Zool. 20 (1971) 406–416.

[8] C. Gallut, V. Barriel, R. Vignes, Gene order and phylogenetic information, in: D. Sankoff, J.H. Nadeau (Eds.), Comparative Genomics, Kluwer, Dordrecht, 2000, pp. 123–132.

[9] S. Hannenhalli, C. Chappey, E.V. Koonin, P.A. Pevzner, Genome sequence comparison and scenarios for gene rearrangements: a test case, Genomics 30 (1995) 299–311.

[10] S. Hannenhalli, P.A. Pevzner, Transforming cabbage into turnip. (polynomial algorithm for sorting signed permutations by reversals), J. ACM 46 (1999) 1–27.

[11] M. Held, R. Karp, The traveling salesman problem and minimum spanning trees: part II, Math. Programming 1 (1970) 16–25.

[12] M. Held, R. Karp, The traveling salesman problem and minimum spanning trees, Oper. Res. 18 (1970) 1138–1162.

[13] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, pp. 85–103.

[14] M. Korab-Laskowska, P. Rioux, N. Brossard, T.G. Littlejohn, M.W. Gray, B.F. Lang, G. Burger, The Organelle Genome Database Project (GOBASE), Nucleic Acids Research 26 (1998) 139–146, http://megasun.bch.umontreal.ca/gobase/gobase.html.

[15] Larget, D. Simon, Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees, Mol. Biol. Evol. 16 (1999) 750–759.

[16] P. Lewis, Nexus Class Library (NCL), 1999, http://lewis.eeb.uconn.edu/lewishome.

[17] C. Lemieux, C. Otis, M. Turmel, Ancestral chloroplast genome in *Mesostigma viride* reveals an early branch of green plant evolution, Nature 403 (2000) 649–652.

[18] B.M.E. Moret, A.C. Siepel, J. Tang, T. Liu, Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data, in: Proc. 2nd Internat. Workshop on Algorithms in Bioinformatics (WABI'02), in: Lecture Notes in Computer Science, vol. 2452, 2002, submitted for publication.

[19] B.M.E. Moret, J. Tang, L. Wang, T. Warnow, Steps towards accurate reconstructions of phylogenies from gene-order data, J. Comput. Syst. Sci. (2002), submitted for publication.

[20] B.M.E. Moret, S. Wyman, D.A. Bader, T. Warnow, M. Yan, A new implementation and detailed study of breakpoint analysis, in: Proc. 6th Pacific Symp. on Biocomputing (PSB 2001), World Scientific, Hawaii, 2001, pp. 583–594.

[21] I. Pe'er, R. Shamir, The median problems for breakpoints are NP-complete, Electronic Colloquium on Computational Complexity Technical Report 98-071, http://www.eccc.uni-trier.de/eccc, 1998.

[22] I. Pe'er, R. Shamir, Approximation algorithms for the median problem in the breakpoint model, in: D. Sankoff, J.H. Nadeau (Eds.), Comparative Genomics, Kluwer, Dordrecht, 2000, pp. 225–241.

[23] G. Reinelt, The Traveling Salesman—Computational Solutions for TSP Applications, Springer-Verlag, Berlin, 1991.

[24] D. Sankoff, P. Rousseau, Locating the vertices of a Steiner tree in an arbitrary metric space, Math. Programming 9 (2) (1975) 240–246.

[25] D. Sankoff, Edit distance for genome comparison based on non-local operations, in: A. Apostolico, M. Crochemore, Z. Galil, U. Manber (Eds.), Combinatorial Pattern Matching. 3rd Annual Symposium, in: Lecture Notes in Computer Science, vol. 644, Springer-Verlag, New York, 1992, pp. 121–135.

[26] D. Sankoff, G. Sundaram, J. Kececioglu, Steiner points in the space of genome rearrangements, Internat. J. Found. Comput. Sci. 7 (1996) 1–9.

[27] D. Sankoff, M. Blanchette, The median problem for breakpoints in comparative genomics, in: T. Jiang, D.T. Lee (Eds.), Computing and Combinatorics, Proceedings of COCOON'97, New York, in: Lecture Notes in Comput. Sci., pp. 251–263.

[28] D. Sankoff, M. Blanchette, Multiple genome rearrangement and breakpoint phylogeny, J. Comput. Biol. 5 (1998) 555–570.

[29] D. Sankoff, M. Blanchette, Probability models for genome rearrangement and linear invariants for phylogenetic inference, in: Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB 99), ACM, New York, 1999, pp. 302–309.

[30] D. Sankoff, D. Bryant, M. Denault, B.F. Lang, G. Burger, Early eukaryote evolution based on mitochondrial gene order breakpoints, J. Comput. Biol., 2000, submitted for publication.

[31] D. Sankoff, M. Deneault, D. Bryant, C. Lemieuex, M. Turmel, Chloroplast gene order and the divergence of plants and algae, from the normalised number of induced breakpoints, in: D. Sankoff, J.H. Nadeau (Eds.), Comparative Genomics, Kluwer, pp. 89–98.

[32] D. Swofford, PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods), Sinauer Associates, Sunderland, MA, 1998.