Pacific Symposium on Biocomputing 14:138-149 (2009)

AN EXACT SOLVER FOR THE DCJ MEDIAN PROBLEM

MENG ZHANG

College of Computer Science and Technology, Jilin University, China
Email: zhangmeng@jlu.edu.cn

WILLIAM ARNDT AND JIJUN TANG

Dept. of Computer Science and Engineering, Univ. of South Carolina, USA
Email: {arndtw, jtang}@cse.sc.edu

The "double-cut-and-join" (DCJ) model of genome rearrangement proposed by Yancopoulos et al. uses the single DCJ operation to account for all genome rearrangement events. Given three signed permutations, the DCJ median problem is to find a fourth permutation that minimizes the sum of the pairwise DCJ distances between it and the three others. In this paper, we present a branch-and-bound method that provides accurate solution to the multichromosomal DCJ median problems. We conduct extensive simulations and the results show that the DCJ median solver performs better than other median solvers for most of the test cases. These experiments also suggest that DCJ model is more suitable for real datasets where both reversals and transpositions occur.

1. Introduction

Once a genome has been annotated to the point where gene homologs can be identified, each gene family can be assigned a unique integer and each chromosome represented by an permutation of signed integers, where the sign indicates the strand. Rearrangement of genes under reversal (also called inversion), transposition, and other operations such as translocation, fusion and fission are known to be an important evolutionary mechanism [4] and have attracted great interests from phylogenists, evolutionary biologists and comparative genomicists.

Yancopoulos et al. [12] proposed a "universal" double-cut-and-join (DCJ) operation to account for all rearrangement events. Although there is no direct biological evidence for DCJ operations, these operations are very attractive because it provides a simpler and unifying model for genome rearrangement [2].

One important problem in genome rearrangement analysis is to find the

Pacific Symposium on Biocomputing 14:138-149 (2009)

median of three genomes, that is, finding a fourth genome that minimizes the sum of the pairwise genomic distances between it and the three given genomes [7]. This problem is important since it provides a maximum parsimony solution to the smallest binary tree, thus can be used as the basis for more complex methods. The median problem is NP-hard for when DCJ distance is used [9]. Since phylogenetic reconstruction based on reconstructing ancestral states may need to compute such medians repeatedly, fast approximations or heuristics are often needed, although exact methods have done well for small genomes [6,8]. With more and more whole genome information available, it becomes very important to develop accurate median solvers for these multichromosomal genomes.

In this paper, we present an exact solver for the DCJ median problem and show our experimental results on various combinations of probabilities of reversals and transpositions. Our tests suggest that this solver is efficient and accurate, and provides better median solutions for most of the cases.

2. Background and Notions

2.1. *Genome Rearrangements*

We assume a reference set of n genes $\{1, 2, \cdots, n\}$ through which a genome can be represented by an *ordering* of these genes. Gene i is assigned with an orientation that is either positive, written i, or negative, written $-i$. Specifically, we regard a multichromosomal genome as a set $A = A(1), \ldots, A(N_c)$ of N_c chromosomes partitioning genes $1, \ldots, n$; where $A(i) = \langle A(i)_1, \ldots, A(i)_{n_i} \rangle$ is the sequence of signed genes in the ith chromosome. In this paper, we also assume that each gene i occurs exactly once in the genome, and the chromosomes are undirected [10], i.e. the flip of chromosomes is regarded as equivalent.

Let G be the genome with signed ordering of $1, 2, \cdots, n$. An *reversal* between indices i and j ($i \leq j$), produces the genome with linear ordering $1, 2, \cdots, i-1, -j, -(j-1), \cdots, -i, j+1, \cdots, n$. A *transposition* on genome G acts on three indices i, j, k, with $i \leq j$ and $k \notin [i, j]$, picking up the interval $i, i+1, \cdots, j$ and inserting it immediately after k. Thus genome G is replaced by: $1, \cdots, i-1, j+1, \cdots, k, i, i+1, \cdots, j, k+1, \cdots, n$ (assume $k > j$). A *transversion* is a transposition followed by an inversion of the transposed subsequence.

There are additional events for multiple-chromosome genomes, such as *translocation* (the end of one chromosome is broken and attached to the end of another chromosome), *fission* (one chromosome splits and becomes

Pacific Symposium on Biocomputing 14:138-149 (2009)

two) and *fusion* (two chromosomes combine to become one).

Two genes i and j are said to be *adjacent* in genome A if i is immediately followed by j, or if $-j$ is immediately followed by $-i$. Given genomes A and B, a *breakpoint* is defined as an ordered pair of genes (i, j) such that i and j are adjacent in A but not in B.

2.2. *Genomic Distance and Median Problem*

We define the *edit distance* as the minimum number of events required to transform one genome into the other. Hannenhalli and Pevzner developed an elegant theory for unichromosomal genomes and provided a polynomial-time algorithm to compute the *reversal distance* (and the corresponding sequence of events). They also extends the algorithms to handle multichromosomes [5]. Yancopoulos et al. [12] proposed a universal double-cut-and-join (DCJ) operation that accounts for events such as reversals, translocations, fissions and fusions.

The median problem on three genomes is to find a single genome that minimizes the sum of pairwise distances between itself and each of the three given genomes. The median problems are generally named after the distance they use. For example, the *reversal median problem* uses reversal distances and the *DCJ median problem* uses DCJ distances. Many solvers have been proposed for the reversal median problem. Among them, the one proposed by Caprara [3] is the most accurate. As a branch-and-bound algorithm, Caprara's solver enumerates all possible solutions and tests them edge by edge. Very recently, Adam and Sankoff [1] presented a heuristic inspired by MGR for the DCJ median problem. Experiments on various biological data suggest it is at least as good as MGR. Xu and Sankoff [11] developed a decomposition theory based on multiple breakpoint graphs, enabling the possibility of very fast and exact algorithms for the DCJ median problem in case of circular genomes. Our new DCJ median solver presented in this paper is inspired by Caprara's solver, but adjusted to handle both DCJ edit distance and the complexities introduced by multiple linear and circular chromosomes.

3. Adjacency Graph for Undirected Genome

In this section we review the notations of the Adjacency Graph [2]. A gene is an oriented sequence of DNA that starts with a *head* and ends with a *tail*. We call these the *extremities* of the gene. The tail of a gene a is denoted by a^t, and its head is denoted by a^h. An adjacency of two consecutive genes

Pacific Symposium on Biocomputing 14:138-149 (2009)

a and b, depending on their respective orientation, can be of the following four different types: $\{a^h, b^t\}, \{a^h, b^h\}, \{a^t, b^t\}, \{a^t, b^h\}$. An extremity that is not adjacent to any other gene is called a *telomere*, represented by a singleton set $\{a^h\}$ or $\{a^t\}$.

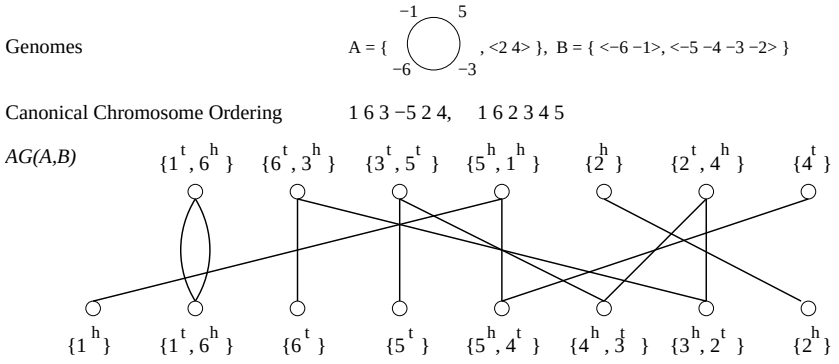

Figure 1. In $AG(A, B)$, $n = 6$, $c(A, B) = 1$,$|AB| = 2$, DCJ distance of A, B is 4.

We define the adjacency graph $AG(A, B)$ as a graph whose vertices are the adjacencies and telomeres of genomes A and B. For each vertices $u \in A$ and $v \in B$ there are $|u \cap v|$ edges between u and v. The adjacency graph is a union of *paths* and *cycles*. Paths of odd length (odd paths), connect telomeres of different genomes, and paths of even length (even paths), connect telomeres of the same genome. An example of adjacency graph is shown in Fig 1.

The double-cut-and-join (DCJ) operation cuts the chromosome in two places and joins the four ends of the cut in a new way. The DCJ distance can be computed using the following theorem:

Theorem 3.1. [2] *Let A and B be two genomes defined on the same set of n genes. Denote the DCJ distance for two genomes A and B by $d_{\mathrm{DCJ}}(A, B)$, then we have $d_{\mathrm{DCJ}}(A, B) = n - (C + I/2)$ where C is the number of cycles and I is the number of odd paths in the adjacency graph $AG(A, B)$.*

As a metric distance, the DCJ distance satisfies the triangle inequality:

Lemma 3.1. *Given three genomes A,B,C, $d_{\mathrm{DCJ}}(A, C)) + d_{\mathrm{DCJ}}(C, B) \geq d_{\mathrm{DCJ}}(A, B)$.*

4. Branch-and-bound algorithm

In this section, we describe a branch-and-bound algorithm for the multi-chromosomal DCJ median problem.

Pacific Symposium on Biocomputing 14:138-149 (2009)

4.1. *Basic Notions*

We formally define the DCJ Median Problem (DMP) here. For a genome G, let $Q := \{1, 2, 3\}$, $\gamma(G) = \sum_{k \in Q} d_{\mathrm{DCJ}}(G, G_k)$. We define DMP as the following: given undirected genomes G_1, G_2, G_3, find a genome G such that $\gamma(G)$ is minimized. The genome obtained after a DCJ operation may contain circular chromosome; thus we should allow circular chromosomes in the input genomes. In this paper, we also consider the solution genomes with circular chromosomes.

The following lower bound on DCJ median value is based on the fact that the DCJ distance satisfies the triangle inequality.

Lemma 4.1. *Given three undirected genomes G_1, G_2, G_3. Let γ^* be the optimal DCJ median value of these genomes,*

$$\gamma^* \geq \frac{d_{\mathrm{DCJ}}(G_1, G_2) + d_{\mathrm{DCJ}}(G_2, G_3) + d_{\mathrm{DCJ}}(G_3, G_1)}{2}. \tag{1}$$

We introduce the following operation in our algorithm. Given a genome A and a pair of extremities $\{p, q\}$. Let $\{p, x\}$, $\{q, y\}$ be adjacencies or telomeres of A. $DCJ(\{p, q\}, A)$ denotes the DCJ operation on A that cuts $\{p, x\}$ and $\{q, y\}$, then connects p with q and x with y. For input genomes G_1, G_2, G_3, we use $DCJ(\{p, q\})$ to denote the set $\{DCJ(\{p, q\}, G_1), DCJ(\{p, q\}, G_2), DCJ(\{p, q\}, G_3)\}$. By applying $DCJ(\{p, q\})$ to the input genomes, we get G'_1, G'_2, G'_3.

For a set of genomes and $DCJ(\{p, q\})$, the following lemma holds.

Lemma 4.2. *Given three undirected genomes G_1, G_2, G_3 and a pair of extremities $\{p, q\}$, an optimal DMP solution containing the adjacency $\{p, q\}$ is also an optimal solution of the set of genomes obtained by applying $DCJ(\{p, q\})$ to the input genomes.*

As a direct consequence of Lemma 4.1 and 4.2, a new lower bound on DCJ median can be computed from G'_1, G'_2, G'_3.

Lemma 4.3. *Given three undirected genomes G_1, G_2, G_3 and a pair of extremities $\{p, q\}$. Let γ^* be the optimal DCJ median value of these genomes, G'_1, G'_2, G'_3 are the resulting genomes by applying $DCJ(\{p, q\})$ to G_1, G_2, G_3. If $\{p, q\}$ is in an optimal DMP solution, then*

$$\begin{aligned} \gamma^* \geq\ & d_{\mathrm{DCJ}}(G_1, G'_1) + d_{\mathrm{DCJ}}(G_2, G'_2) + d_{\mathrm{DCJ}}(G_3, G'_3) \\ & + \frac{d_{\mathrm{DCJ}}(G'_1, G'_2) + d_{\mathrm{DCJ}}(G'_2, G'_3) + d_{\mathrm{DCJ}}(G'_3, G'_1)}{2}. \end{aligned} \tag{2}$$

Pacific Symposium on Biocomputing 14:138-149 (2009)

Proof. Let S be an optimal DMP solution that contains $\{p, q\}$. If $\{p, q\} \in G_i, i \in \{1, 2, 3\}$, then any cycle or path in $AG(S, G'_i)$ corresponds to a unique cycle or path in $AG(S, G_i)$. Otherwise $AG(S, G'_i)$ has the cycle from $\{p, q\}$ to $\{p, q\}$, and any cycle or path in $AG(S, G'_i)$ corresponds to a unique cycle or path in $AG(S, G_i)$. In the former case, $G_i = G'_i$, $d_{\rm DCJ}(G_i, S) = d_{\rm DCJ}(G'_i, S)$. In the latter case, $d_{\rm DCJ}(G_i, G'_i) = 1$ and $d_{\rm DCJ}(S, G_i) = d_{\rm DCJ}(S, G'_i) + 1$. Therefore, $\gamma^* = \sum_{k=1}^{3} d_{\rm DCJ}(S, G_k) = \sum_{k=1}^{3} d_{\rm DCJ}(G'_k, G_k) + \sum_{k=1}^{3} d_{\rm DCJ}(S, G'_k)$. By lemma 4.1, $\sum_{k=1}^{3} d_{\rm DCJ}(S, G'_k) \geq \frac{d_{\rm DCJ}(G'_1,G'_2)+d_{\rm DCJ}(G'_2,G'_3)+d_{\rm DCJ}(G'_3,G'_1)}{2}$. Thus the lemma is proved. □

4.2. *The Branch-and-Bound Algorithm*

By Lemma 4.2 and Lemma 4.3, we extend the branch-and-bound algorithm in [13] to deal with both circular and linear chromosomes. In each step, it either selects a gene (negative or positive) as an end or an inner gene of a chromosome of the solution (median) genome. In terms of the DCJ model, the former operation fixes a telomere in the solution genome and the latter one fixes an adjacency in the solution genome, these are called *telomere fixing* and *adjacency fixing* respectively. In the fixing of an adjacency, say α, $DCJ(\alpha)$ is applied to each input genome and a lower bound on the γ value of the solution is computed.

In the beginning, the algorithm enumerates all partial genomes by first fixing, in turn, all genes $\pm 1, \ldots, \pm n$ as the first gene, say f, in a chromosome of the solution. Since this chromosome is not completed yet, we call this chromosome an *opening chromosome*. Recursively, let the latest fixed gene be a, we proceed the enumeration by fixing in the solution gene b for all genes not fixed so far or fixing a as an end gene of the current opening chromosome. In the former case, the adjacency $\{a^t, b^h\}$ is fixed in the solution. In the latter case, the telomere $\{a^t\}$ or the adjacency $\{a^t, f^h\}$ is fixed and the current opening chromosome is closed as a linear or circular chromosome. A closing of a chromosome is followed by an opening of a new chromosome except that a complete genome is available.

In adjacency fixing, the operation $DCJ\big(\{p, q\}\big)$ is applied to the input genomes and the lower bound of all DMP solutions containing all adjacencies fixed so far up to $\{p, q\}$ in the median genomes can be derived according to Lemma 4.3. If it is greater than the lower bound $\{p, q\}$ is not an adja-

Pacific Symposium on Biocomputing 14:138-149 (2009)

cency in the best solution and can be pruned. Otherwise, $\{p, q\}$ is fixed in the current solution as an adjacency.

After adjacency fixing, the number of common adjacencies of the intermediate genomes increases by one. In other words, the three genomes move simultaneously by DCJ operations. In the case that $\{p, q\}$ is already an adjacency in one of the intermediate genomes, say G, the $DCJ\big(\{p, q\}, G\big)$ will not change G, therefore G will not move in this step.

The algorithm also generates three series of DCJ operations that transform G_1, G_2, G_3 to the median genome accordingly. Each of the three DCJ series is one of the shortest DCJ series from each of the input genomes to the median genome.

The lower bound can be computed very efficiently. For the current fixed adjacencies set $S_l = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$, let G_k^i, $k \in \{1, 2, 3\}$, be the genome after applying the operations $DCJ(\alpha_1, G_k), DCJ(\alpha_2, G_k), \ldots, DCJ(\alpha_i, G_k)$ to G_k, $G_k^0 = G_k$. We have that $\sum_{i=1}^{l} |\{k : \alpha_i \in G_k^{i-1}\}|$ equals the number of cycles in $AG(S_l, G_1), AG(S_l, G_2)$ and $AG(S_l, G_3)$. Since a DCJ operation alters at most two cycles or paths in the adjacency graph for two genomes, the new lower bound can be easily computed from the old one, avoiding the need to compute from scratch.

It's worth mentioning that the treatment of the telomere fixing plays an important role in reducing the complexity of the branch-and-bound algorithm. In adjacency fixing, we can find three unique DCJ operations such that Lemma 4.2 holds. However, in telomere fixing, it is difficult to find such DCJ operations with the property like Lemma 4.2. Other than treating the adjacency fixing and telomere fixing in a uniform way, we do not change any genome in telomere fixing and the number of paths in the adjacency graph of a solution and each input genome can be computed very efficiently as described above.The complexity is therefore avoided.

We also extend the efficient genome enumeration method in [13] to enumerate and test the genomes containing both linear and circular chromosomes at most once, which saves considerable computing time.

Let the linear chromosome $X = \langle X_1, \ldots, X_M \rangle$. The *canonical flipping* of X is $\langle -X_M, \ldots, -X_j, \ldots, -X_1 \rangle$, if $|X_M| < |X_1|$, otherwise, X itself. For a circular chromosome $Y = \langle Y_1, \ldots, Y_l, Y_m, Y_r, \ldots, Y_M \rangle$, let $|Y_m|$ be the minimal gene in Y. The *canonical flipping* of Y is $\langle Y_m, \ldots, Y_l \rangle$, if $Y_m > 0$; Otherwise, $\langle -Y_m, \ldots, -Y_r \rangle$. After canonical flipping of each chromosome, we order the chromosomes by their first genes in an increasing order. Obviously, each genome has a unique canonical ordering, and it can be uniquely represented by the canonical ordering along with the markers

Pacific Symposium on Biocomputing 14:138-149 (2009)

of open and closed genes and an indicator of linear or circular chromosome.

We proceed the branch-and-bound algorithm in a depth first order, which can be viewed as the depth-first traversal of the trie of all canonical chromosome ordering of genomes, as illustrated in Fig 2. With this scheme we can perform the lower bound test after each adjacency fixing as described in section 4.2.

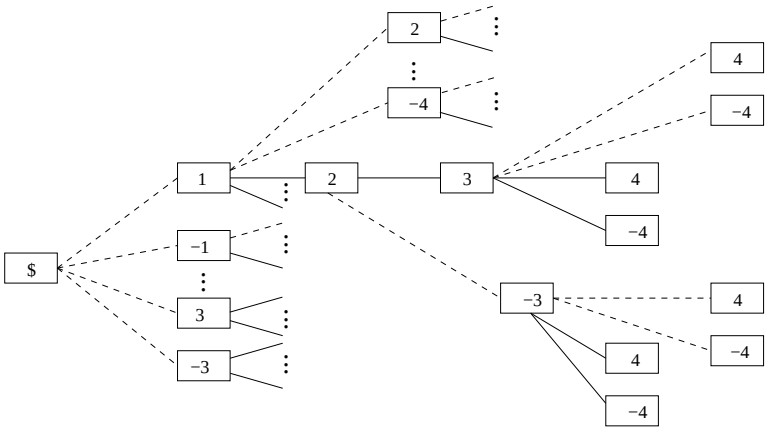

Figure 2. A sketch of the trie of canonical chromosome ordering of all genomes on gene 1..4. Solid edges correspond to adjacencies and each dotted edge corresponds to telomeres. The branch-and-bound algorithm can be viewed as the depth-first traversal of the trie. When lower bound test is failed, it backtracks and tries other genes. We assume that all genomes start at a virtual gene \$.

In our implementation, the initial lower bound LD is computed from the initial input genomes (Eq. 1) based on triangular inequalities. The branch-and-bound algorithm starts searching for a DMP solution of target value $L = LD$ and if a solution of value L is found, it is optimal and we stop, otherwise there is no solution of value LD. The algorithm then restarts with an increased target value $L = LD + 1$, and so on. The search stops as soon as we find a solution with the target value.

5. Experimental Results

We have implemented the algorithm and conducted simulations to assess its performance and compare its results against other solvers.

5.1. *Performance of the New Solver*

We test the performance of our new solver on genomes with $100 \sim 200$ genes and $2 \sim 8$ chromosomes. We create each dataset by first generating a tree

Pacific Symposium on Biocomputing 14:138-149 (2009)

topology with three leaves and assigning each edge with different length. We assign a genome G_0 to the root, then evolve the signed permutation down the tree, applying along each edge a number of operations that equals the assigned edge length. We test a large range of evolutionary rates: letting r denote the expected number of evolutionary events along an edge of the model tree, we used values of r in the range of 4 to 32. The actual number of events along each edge is sampled from a uniform distribution on the set $\{r/2, \ldots, 3r/2\}$. For each combination of parameters, we generate 10 datasets and average the results.

Table 1 shows the speed of the new solver on 200 genes (results of 100 genes are similar). From this table, we find that the number of chromosomes has very large impact on the speed and genomes with one or two chromosomes can be analyzed quickly, while genomes with 8 chromosomes are much more difficult to compute than unichromosomal genomes.

Table 1. Running time of 200 genes, $2 \sim 8$ chromosomes. For each combination of parameters, average running time is shown (in seconds) if all ten datasets finished in 1 hour; otherwise the percentage of finished datasets is shown.

	r=8	r=16	r=24	r=32	r=40	r=48
1 chromosome	< 1	< 1	< 1	< 1	1.1	70%
2 chromosomes	< 1	< 1	< 1	40.1	81.4	60%
4 chromosomes	< 1	26.1	142.2	421.4	40%	20%
8 chromosomes	62.0	2352.0	7691.6	20%	0%	0%

5.2. *Comparing with Other Median Solvers*

In this experimental study, each genome has 100 genes. We test each dataset with the following four methods: `MGR`, the most cited method for genome rearrangement analysis; Caprara's reversal median solver, the most accurate for the reversal median problem; breakpoint median solver; and the new DCJ median solver presented in this paper. We only use unichromosomal genomes in this study, because no multichromosomal breakpoint median solver is available. Comparing DCJ medians with breakpoint medians is interesting because the breakpoint model is independent of any particular rearrangement mechanism, hence can be viewed as a neutral model for genome rearrangements.

In this study, we used values of r in the range of 4 to 32. To test the robustness of the DCJ model, we use five combinations of events: 1) reversals are the only events; 2) 75% events are reversals, 25% are transpositions; 3) reversals and transpositions are equally likely; 4) 75% events are transpositions, 25% are reversals; 5) all three events (reversals, transpositions and

Pacific Symposium on Biocomputing 14:138-149 (2009)

transversions) are equally likely; 6) all events are transpositions.

Since the true ancestor is known in our simulations, we use the breakpoint distance between the inferred median and the true ancestor to assess the accuracy of a median solver. For each combination of parameters, we generate 10 datasets, and show the averaged results in Fig. 3 to Fig. 5.

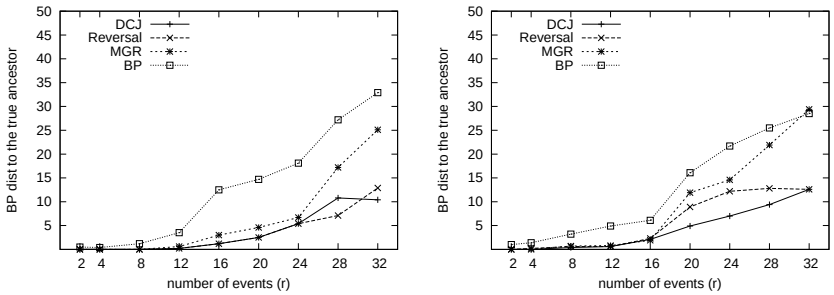

Figure 3. Breakpoint distance from the inferred median to the true ancestor when reversals are the only events (left), and when 75% of the events are reversals, 25% are transpositions (right).

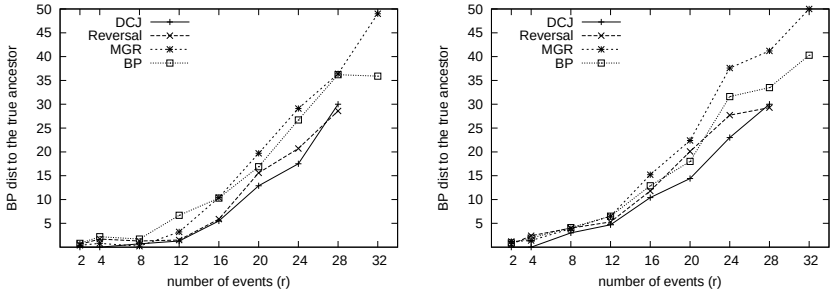

Figure 4. Breakpoint distance from the inferred median to the true ancestor when 50% of the events are reversals, 50% are transpositions (left), and when 25% of the events are reversals, 75% are transpositions (right). DCJ and reversal median solvers cannot finish all datasets for $r = 32$.

From these figures, we find that the DCJ median solver outperforms the other solvers for most of the test cases. Even for the case that favors the reversal median, i.e. when reversals are the only events, using the DCJ median solver has performance equals to the Caprara's solver for $r < 24$, and has better performance when $r = 32$. With the increasing probabilities

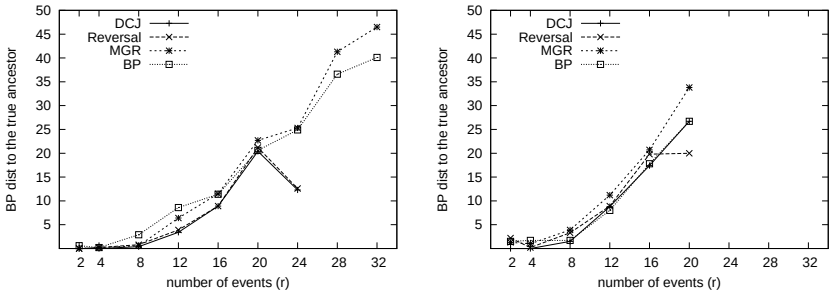

Figure 5. Breakpoint distance from the inferred median to the true ancestor when when all three events (reversals, transpositions and transversions) are equally likely (left), and when all events are transpositions (right). All median solvers cannot finish all datasets for $r > 24$.

of transpositions, the performance gap widens and the DCJ median solver has the best performance for almost all test cases. Although both DCJ and breakpoint models do not deal with actual biological events, the better performance of the DCJ median solver suggests that it is a better choice for real data where both reversals and transpositions occur.

6. Conclusions and Future Work

In this paper we present a new branch-and-bound method for the DCJ median problem. Our extensive experiments show that this method is more accurate than the existing methods. However, this solver is still primitive and further improvements of efficiency are needed. This solver (as almost all other solvers) requires equal gene content which limits the range of data we can analyze. To improve this solver, we need to develop new distance lower bounds so that complex events such as gene losses and duplications can be handled. Recent developments on breakpoint graph decomposition have shown potential and we plan to include some of these new results to further reduce the search space and improve the speed.

7. Acknowledgments

WA and JT were supported by US National Institutes of Health (NIH grant number R01 GM078991). All experiments were conducted on a 128-core shared memory computer supported by US National Science Foundation grant (NSF grant number CNS 0708391).

Pacific Symposium on Biocomputing 14:138-149 (2009)

References

1. Z. Adam and D. Sankoff. The abcs of mgr with dcj. *Evolutionary Bioinformatics*, 4:69–74, 2008.
2. A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In *Proc. 6th Int'l Workshop Algs. in Bioinformatics (WABI'06)*, number 4175 in Lecture Notes in Computer Science, pages 163–173, 2006.
3. A. Caprara. On the practical solution of the reversal median problem. In *Proc. 1st Int'l Workshop Algs. in Bioinformatics (WABI'01)*, volume 2149 of *Lecture Notes in Computer Science*, pages 238–251, 2001.
4. S.R. Downie and J.D. Palmer. Use of chloroplast DNA rearrangements in reconstructing plant phylogeny. In P. Soltis, D. Soltis, and J.J. Doyle, editors, *Plant Molecular Systematics*, pages 14–35. Chapman and Hall, 1992.
5. S. Hannenhalli and P.A. Pevzner. Transforming mice into men (polynomial algorithm for genomic distance problems). In *Proc. 36th Ann. IEEE Symp. Foundations of Comput. Sci. (FOCS'95)*, pages 581–592, 1995.
6. B.M.E. Moret, A.C. Siepel, J. Tang, and T. Liu. Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In *Proc. 2nd Int'l Workshop Algs. in Bioinformatics (WABI'02)*, volume 2452 of *Lecture Notes in Computer Science*, pages 521–536, 2002.
7. D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In *Proc. 3rd Int'l Conf. Computing and Combinatorics (COCOON'97)*, volume 1276 of *Lecture Notes in Computer Science*, pages 251–264. Springer Verlag, Berlin, 1997.
8. A.C. Siepel and B.M.E. Moret. Finding an optimal inversion median: Experimental results. In *Proc. 1st Int'l Workshop Algs. in Bioinformatics (WABI'01)*, volume 2149 of *Lecture Notes in Computer Science*, pages 189–203. Springer Verlag, Berlin, 2001.
9. E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal genome median and halving problems. In *Proc. 8th Int'l Workshop Algs. in Bioinformatics (WABI'08)*, volume 5251 of *Lecture Notes in Computer Science*, pages 1–13, 2008.
10. G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, 63(5):587–609, 2002.
11. W. Xu and D. Sankoff. Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In *Proc. 8th Int'l Workshop Algs. in Bioinformatics (WABI'08)*, volume 5251 of *Lecture Notes in Computer Science*, pages 25–37, 2008.
12. S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.
13. M. Zhang, W. Arndt, and J. Tang. A branch-and-bound method for the multichromosomal reversal median problem. In *Proc. 8th Int'l Workshop Algs. in Bioinformatics (WABI'08)*, volume 5251 of *Lecture Notes in Computer Science*, pages 14–24, 2008.