J Comb Optim (2016) 32:1165–1181
DOI 10.1007/s10878-015-9940-4

Exemplar or matching: modeling DCJ problems with unequal content genome data

Zhaoming Yin[2,5] · **Jijun Tang**[1,3] · **Stephen W. Schaeffer**[4] · **David A. Bader**[2]

Published online: 12 August 2015

Abstract The edit distance under the *DCJ* model can be computed in linear time for genomes with equal content or with *Indels*. But it becomes **NP**-Hard in the presence of duplications, a problem largely unsolved especially when *Indels* (i.e., insertions and deletions) are considered. In this paper, we compare two mainstream methods to deal with duplications and associate them with *Indels*: one by deletion, namely *DCJ-Indel-Exemplar* distance; versus the other by gene matching, namely *DCJ-Indel-Matching* distance. We design branch-and-bound algorithms with set of optimization methods to compute exact distances for both. Furthermore, median problems are discussed in alignment with both of these distance methods, which are to find a median genome that minimizes distances between itself and three given genomes. Lin–Kernighan heuristic is leveraged and powered up by sub-graph decomposition and search space reduction technologies to handle median computation. A wide range of experiments are conducted on synthetic data sets and real data sets to exhibit pros and cons of

✉ Jijun Tang
jtang@cse.sc.edu

✉ David A. Bader
http://www.cc.gatech.edu/~bader

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China

[2] School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, USA

[3] Department of Computer Science and Engineering, University of South Carolina, Columbia, USA

[4] Department of Biology, The Pennsylvania State University, State College, USA

[5] Oracle Corporation, 400 Oracle Parkway, Redwood City, CA 94065, USA

these two distance metrics per se, as well as putting them in the median computation scenario.

Keywords Genome rearrangement · Double-cut and join (*DCJ*) · Lin–Kernighan heuristic

1 Introduction

Over the last years, many distance metrics have been introduced to calculate the dissimilarity between two genomes by genome rearrangement (Blin et al. 2004; Bader et al. 2001; Bafna and Pevzner 1998; Yancopoulos et al. 2005). Among them, *DCJ* distance is largely studied in recent years due to its capability to model various forms of rearrangement events, with a cheap cost of linear time computation. However, when considering duplications, the distance computation becomes **NP**-hard (Chauve et al. 2006) and **APX**-hard (Angibaud et al. 2009; Chen et al. 2012) for various distance models. There are two approaches to treat duplications, both are targeted at removing duplicated genes, so that existing linear algorithms can be utilized subsequently.

The first approach identifies the so called exemplar genes (Sankoff 1999) in order to retain one copy gene in each duplicated gene family, while the other assigns one-to-one matching to every duplicated genes in each gene family (Shao and Lin 2012; Shao et al. 2014). Situated in the context of duplications, gene insertion and deletion (*Indels*), are also important rearrangement events that results in unequal contents (Brewer et al. 1999). Strictly speaking, an *Indel* refers to both insertions and deletions either when what event took place is unsure or all other sequence length variation events in the genome. In this paper, we interpret *Indel* as one genome has single or multiple copy(ies) of a given gene, but another genome has none. Pioneer works were conducted to study the sorting and distance computation by reversals with *Indels* (Mabrouk 2001). Later on, the *DCJ-Indel* distance metric was introduced to take advantages of the *DCJ* model. Braga et al. (2010) proposed the first framework to compute the *DCJ-Indel* distance; Compeau later simplified the problem with a much more elegant distance formula (Compeau 2012). In this paper, we adapt the previous research results to design algorithms that procure the ability to handle both duplications and *Indels* when computing *DCJ* distance. To be more specific, in Sankoff (1999), a combinatorial problem for computing exemplar distance was discussed, but a tool for analytics such as breakpoint graphs (BPG) were not provided. With respect to paper (Shao et al. 2014), a LP method was formulated, but *Indels* were not considered in their solution. In general, our method has no constraints as opposed to these two methods, and can deal with data that is close to real world scenario.

As evolutionary analysis generally involves more than two species, it is necessary to extend the above distances to deal with multiple genomes. Because three species form the smallest evolutionary tree, it is critical to study the median problem, which is to construct a genome that minimizes the sum of distances from itself to the three input genomes (Moret et al. 2002; Bourque and Pevzner 2002). The median problem is **NP**-hard under most distance metrics (Pe’er and Shamir 1998; Caprara 2003; Xu

2009b; Bergeron et al. 2005). Several exact algorithms have been implemented to solve the *DCJ* median problems on both circular (Xu and Sankoff 2008; Xu 2009b) and linear chromosomes (Xu 2009a; Xu and Moret 2011). Some heuristics are brought forth to improve the speed of median computation, such as linear programming (*LP*) (Caprara 2003), local search (Lenne et al. 2008), evolutionary programming (Gao et al. 2013), or simply searching on one promising direction (Rajan et al. 2010). All these algorithms are intended for solving median problems with equal content genomes, which are highly unrealistic in nature. In this paper, we implement a Lin–Kernighan heuristic leveraging the aforementioned distance metric to compute *DCJ* median when duplications and *Indels* are considered.

2 Background

2.1 Genome rearrangement events and their graph representations

2.1.1 Genome rearrangement events

The ordering of a genome can be changed through rearrangement events such as reversals and transpositions. Figure 1 shows examples of different events of a single chromosome (1 −2 3 4 −5 6 7). In the examples, we use signed numbers to represent different genes and their orientations. Genome rearrangement events involve multiple combinatorial optimization problems and graph representation is common to these problems. In this part, we will address the foundations of using the BPG to genome rearrangement events.

2.1.2 Breakpoint graph

Given an alphabet $\mathcal{A}$, two genomes Γ and Π are represented by two strings of signed ($+$ or $-$) numbers (representing genes) from $\mathcal{A}$. Each gene $a \in \mathcal{A}$ is represented by a pair of vertices head a_h and tail a_t; If a is positive a_h is putted in front of a_t, otherwise a_t is putted in front of a_h. For $a, b \in \mathcal{A}$, if $a, b \in \Gamma$ and are adjacent to each other, their adjacent vertices will be connected by an edge. For a telomeric gene, if it exists in a circular chromosome, two end vertices will be connected by an edge; if it exists in a linear chromosome, two end vertices will be connected to a

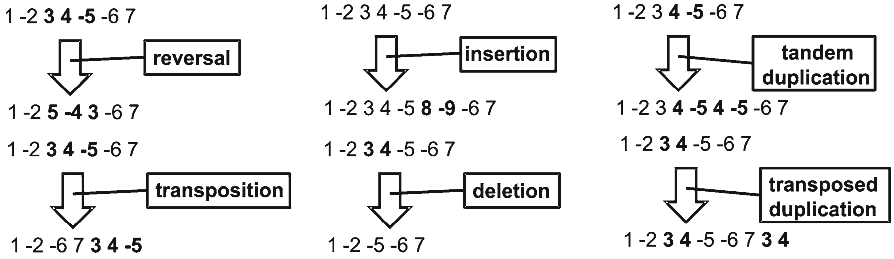

Fig. 1 Example of different rearrangement events

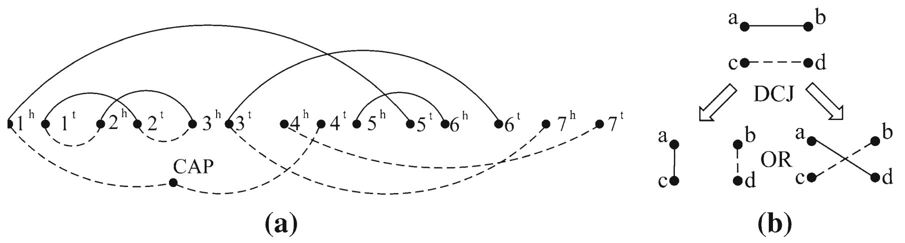

Fig. 2 Examples of *BPG*; and *DCJ* operations. **a** Example of *BPG*. **b** Example of *DCJ*

special vertex called *CAP* vertex. If we use one type of edge to represent adjacencies of gene order Γ and another type of edges to represent adjacencies of gene order Π, the resulting graph with two types of edges is called a *BPG*. Figure 2a shows the *BPG* for gene order Γ (1,−2,3,−6,5) (edge type: solid edges) which has one circular chromosome and Π (1,2,3,7,4) (edge type: dashed edges) which has one linear chromosome.

2.1.3 DCJ operation

Double-cut and join (*DCJ*) operations are able to simulate all rearrangement events. In a *BPG*, these operations cut two edges (within one genome) and rejoin them using two possible combinations of end vertices (shown in Fig. 2b).

2.2 Distance computation

2.2.1 DCJ distance

DCJ distance of genomes with the same content can be easily calculated by enumerating the number of cycles/paths in the *BPG* (Yancopoulos et al. 2005), which is of linear complexity.

2.2.2 DCJ-Indel distance

When *Indels* are introduced in *BPG*, with two genomes Γ and Π, the vertices and edges of a closed walk form a cycle. In Fig. 2a, the walk $[1^t$, $(1^t; 2^h)$, 2^h, $(2^h; 3^h)$, 3^h, $(3^h; 2^t)$, 2^t, $(2^t; 1^t)$, $1^t]$ is a cycle. A vertex v is π*-open* (γ*-open*) if $v \notin \Gamma$ ($v \notin \Pi$). An unclosed walk in *BPG* is a path. Based on different kinds of ends points of paths, we can classify paths into different types. If the two ends of a path are *CAP* vertices, we simply denote this path as p^0. If a path is ended by one open vertex and one *CAP*, we denote it as p^{π} (p^{γ}). If a path is ended by two open vertices, we denote it by the types of its two open vertices: for instance, $p^{\pi,\gamma}$ represents a path that ends with a π*-open* vertex and a γ*-open* vertex. In Fig. 2a, the walk $[5^t$, $(5^t; 1^h)$, 1^h, $(1^h; CAP)$, $CAP]$ is a p^{γ} path and the walk $[6^t$, $(6^t; 3^t)$, 3^t, $(3^t; 7^h)$, $7^h]$ is a $p^{\gamma,\pi}$ path. A path is even (odd), if it contains even (odd) number of edges. In Compeau (2012), if $|\mathcal{A}| = N$ the *DCJ* distance between two genomes with *Indels* but without duplications is calculated by Eq. (1). We call this distance *DCJ-Indel* distance. From this equation, we can easily get the *DCJ-Indel* distance between Γ and Π in Fig. 2a as 4.

$$d_{indel}(\Gamma, \Pi) = N - \left[\left|c\right| + \left|p^{\pi,\pi}\right| + \left|p^{\gamma,\gamma}\right| + \lfloor p^{\pi,\gamma} \rfloor \right. \\ \left. + \frac{1}{2}\left(\left|p^{0}_{even}\right| + min\left(\left|p^{\pi}_{odd}\right|, \left|p^{\pi}_{even}\right|\right) + min\left(\left|p^{\gamma}_{odd}\right|, \left|p^{\gamma}_{even}\right|\right) + \delta \right) \right] \tag{1}$$

where $\delta = 1$ only if $p^{\pi,\gamma}$ is odd and either $|p^{\pi}_{odd}| > |p^{\gamma}_{even}|, |p^{\gamma}_{odd}| > |p^{\gamma}_{even}|$ or $|p^{\pi}_{odd}| < |p^{\gamma}_{even}|, |p^{\gamma}_{odd}| < |p^{\gamma}_{even}|$; Otherwise, $\delta = 0$.

2.2.3 DCJ-exemplar (matching) distance

In general, there are two approaches to cope with duplicated genes. One is by removing all but one copy in a gene family to generate an exemplar pair (Sankoff 1999) and the other is by relabeling duplicated genes to ensure that every duplicated gene has unique number (Shao et al. 2014; Shao and Lin 2012). Both of these two distances can be computed with *BPG* using branch-and-bound methods. For both of the distance metrics, the upper bound can be easily derived by assigning an arbitrary mapping to two genomes then computing their mutual distance. In Sankoff's paper (1999) regarding exemplar distance, it is proved that by removing all occurrences of unfixed duplicated gene families, the resulting distance is monotony decreasing, hence the resulting distance serves as a lower bound. In Chen et al.'s paper (2005) regarding matching distance, the authors proposed a way for computing lower bounds by measuring the number of breakpoints between two genomes, which might not directly imply the lower bound between genomes with *Indels*. However, it is still possible to use this method to find a 'relaxed' lower bound.

2.2.4 Distance estimation

Note that the mathematically optimized distance might not reflect the true number of biological events, thus several estimation methods such as *EDE* or *IEBP* are used to rescale these computed distances (Moret et al. 2001) to better fit true evolutionary history.

2.3 Median computation

If there are three given genomes, the graph constructed by pre-defined *BPG* rule is called a multiple breakpoint graph (*MBG*). Figure 3a shows an example of *MBG* with three input genomes. When genomes have equal gene content, the *DCJ* median problem can be briefly described by finding a maximum matching (which is called 0-*matching*) in *MBG*. Figure 3b shows an example of 0-*matching* which is represented by gray edges. In Xu and Sankoff's paper (2008), it is proven that a type of sub-graph called adequate sub-graph (*AS*) could be used to decompose the graph with edge shrinking operations, which are shown in Fig. 3c. Unfortunately, there is no branch-

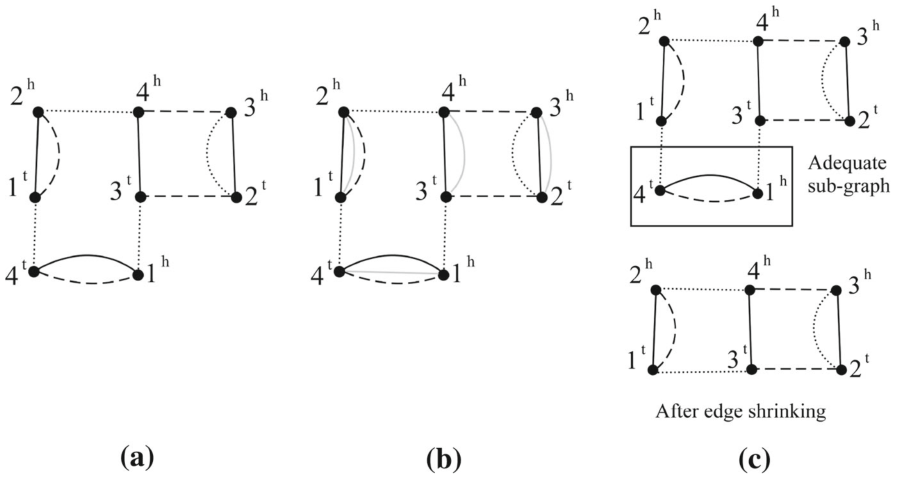

Fig. 3 (*Top*) Examples of *MBG* with three input genomes: (1,2,3,4) (*solid* edges); (1,2,−3,4) (*dashed* edges) and (2,3,1,−4) (*dotted* edges).; (*middle*) 0-matching operation; (*bottom*) edge shrinking operations. **a** *MBG*. **b** *0-matching*. **c** Adequate subgraph and edge shrinking

and-bound based median algorithm that deals with unequal content genomes. In the following section, we will show that it is actually difficult to design such algorithm.

3 Approaches

3.1 Proposed distance metrics

We have discussed *DCJ*, *DCJ-Indel* and *DCJ-Exemplar(Matching)* distances, here we formally define the *DCJ-Indel-Exemplar(Matching)* distances as follows:

Definition 1 An *exemplar* string is constructed by deleting all but one occurrence of each gene family. Among all possible exemplar strings, the minimum distance that one exemplar string returns is the *DCJ-Indel-Exemplar* distance.

Definition 2 A *matching* string is constructed by assigning a one-to-one mapping to each occurrence of genes in a gene family and relabel them to distinct markers. Among all possible matching strings, the minimum distance that one matching string returns is the *DCJ-Indel-Matching* distance.

Figure 4 shows examples of *BPG* representation of exemplar mapping from genome Γ (1, −2, 3, 2, −6, 5) and genome Π (1, 2, 3, 7, 2, 4) to Γ (1, 3, 2, −6, 5) and genome Π (1, 3, 7, 2, 4), and a matching that mapping from genome Γ (1, −2, 3, 2, −6, 5) and genome Π (1, 2, 3, 7, 2, 4) to Γ (1, −2, 3, 2′, −6, 5) and genome Π (1, 2′, 3, 7, 2, 4).

We can use branch-and-bound methods which are applied in *DCJ-Exemplar (Matching)* distances to solve these two distances.

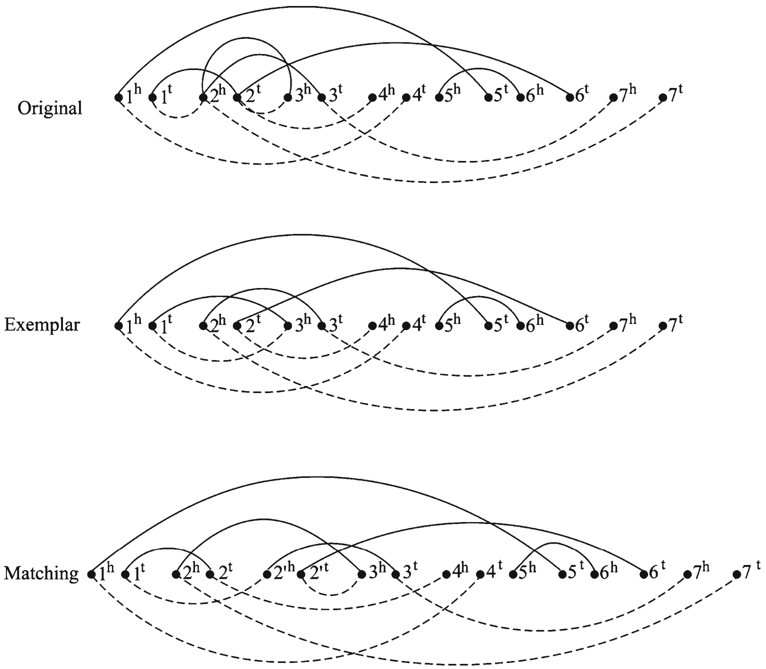
Original
1h 1t 2h 2t 3h 3t 4h 4t 5h 5t 6h 6t 7h 7t
Exemplar
1h 1t 2h 2t 3h 3t 4h 4t 5h 5t 6h 6t 7h 7t
Matching
1h 1t 2h 2t 2'h 2't 3h 3t 4h 4t 5h 5t 6h 6t 7h 7t

Fig. 4 Examples of exemplar and matching distance in the form of *BPG* representation

3.2 Optimization methods

3.2.1 Optimal assignments

Although branch-and-bound algorithms are based on enumerating the number of cycles/path in *BPG*, it is not necessary to enumerate every component in the graph, as both Shao et al. (2014), Chen et al. (2005) indicated that there are some specific patterns in *BPG* which can be fixed before the distance computation. In this paper, we will extend their result in our optimization methods for *DCJ-Indel-Exemplar(Matching)* distances.

To begin with, we define some terms for future explanation. There are two categories of vertices in a *BPG*: one connects exactly one edge of each edge type (in this paper edge types are expressed by such as dotted, dashed edges etc.), they are called *regular* vertices; the other connects fewer or more than one edges of each edge type, they are called *irregular* vertices. A subgraph in a *BPG* that only contains regular vertices is defined as *regular subgraph*, while one that contains irregular vertices is defined as *irregular subgraph*. In *BPG* with two genomes Γ and Π, vertices and edges of a closed walk form a cycle.

Theorem 1 *In a* BPG, *an irregular subgraph which is a cycle of length 2 can be fixed before computation without losing accuracy.*

Proof Without loss of generality, the proof is sound for both *DCJ-Indel-Exemplar* and *DCJ-Indel-Matching* distances. We prove the theorem under two cases:

(1) For the subgraph in the component which only contains cycles, this is a case that is exactly the same as mentioned in Shao et al. (2014), proof.
(2) For the subgraph in the component which contains paths, since no type of the paths has count more than one (which is the count of a cycle), following the similar proof strategy in Shao et al. (2014), we can get the same conclusion.

□

3.2.2 Adopting morph graph methods to condense BPG

If a gene family has multiple copies of the gene, its corresponding two vertices ($head$ and $tail$) in the *BPG* will have degree of more than one. In contrary, vertex representations of those singleton genes always have degree of one or zero. Once an 'exemplar' or 'matching' is fixed, only edges incident to vertices that have degree of more than one have been changed. We can view the computation of exemplar or matching distance as the process of morphing (or streaming) (Yin et al. 2013) the *BPG* in order to find an ad hoc shape of the *BPG* that achieves optimality. Following this hint, we can bridge out all vertices that are stable and just investigate these dynamically changing vertices without losing accuracy. Suppose there are V vertices in the *BPG*, where V_s are stable and V_d are dynamic, the asymptotic speedup for this morph *BPG* strategy will be $O(\frac{V}{V_d})$.

3.2.3 Harnessing the power of divide-and-conquer approach to reduce the problem space

In the paper by Nguyen et al. (2005), the authors proposed a divide and conquer method to quickly calculate the exemplar distance. Inspired by their idea, we propose the following divide-and-conquer method to compute the above two distances based on the *BPG*. We have the follow observation:

Theorem 2 *The* DCJ-Indel-Exemplar (Matching) *distance is optimal* iff *the choices of exemplar edges (cycle decomposition) in each connected components of* BPG *are optimal.*

Proof Since it's obvious that for regular connected component of *BPG*, there is only one choice of edges, the proof under this case is trivial. For irregular connected component of *BPG*, we prove by contrary: suppose there is another edge selection that can result in a better distance, based on the corresponding *BPG*, there must be at least one connected component that has a better edge selection, replacing it with a better edge selection will result in a better distance, which violates the assumption. □

Combining three optimization methods in tandem with the branch-and-bound framework, we can summarize our algorithm to compute *DCJ-Indel-Exemplar (Matching)* distance as outlined in Algorithm 1, named DCJIndelExem(Matc)Distance respectively.

Algorithm 1: DCJIndelExem(Matc)Distance

```
Input: $G_1$ and $G_2$
Output: Minimum distance $d$
optimization methods on $G_1, G_2$;
$G_1^{'}, G_2^{'}$ ←randomly init exemplar(matching) of all duplicated genes of $G_1, G_2$;
$G_1^{*}, G_2^{*}$ ←remove all duplicated genes of $G_1, G_2$;
$min\_ub \leftarrow DCJIndel(G_1^{'}, G_2^{'})$ ;
$min\_lb \leftarrow DCJIndel(G_1^{*}, G_2^{*})$ ;
Init search list $L$ of size $min\_ub - min\_lb$ and insert $G_1, G_2$;
while $min\_ub > min\_lb$ do
    $G_1^{+}, G_2^{+} \leftarrow$ pop from $L[min\_lb]$;
    for $pair \in$ all mappings of next available duplicated gene do
        $G_1^{+}, G_2^{+} \leftarrow G_1^{+}, G_2^{+}$ fix the exemplar(matching) of $pair$ ;
        $G_1^{+'}, G_2^{+'}$ ←randomly init exemplar(matching) of rest duplicated genes $G_1^{+}, G_2^{+}$;
        $G_1^{+*}, G_2^{+*}$ ←remove rest duplicated genes $G_1^{+}, G_2^{+}$;
        $ub \leftarrow DCJIndel(G_1^{+'}, G_2^{+'})$ ;
        $lb \leftarrow DCJIndel(G_1^{+*}, G_2^{+*})$ ;
        if $lb > min\_ub$ then
            discard $G_1^{+}, G_2^{+}$

        if $ub < min\_ub$ then
            $min\_ub = ub$;

        else if $ub = max\_lb$ then
            return $d = ub$ ;
        else
            insert $G_1^{+}, G_2^{+}$ into $L[lb]$

return $d = min\_lb$;
```

3.3 Adapting *Lin–Kernighan* heuristic to find the median genome

3.3.1 Problem statement

Not surprisingly, finding the median genome that minimizes the *DCJ-Indel-Exemplar (Matching)* distance is challenging. To begin with, given three input genomes, there are multiple choices of possible gene content selections for the median; however, since identifying gene content is simpler and there exists very accurate and fast methods to fulfill the task (Hu et al. 2014), we are more interested on a relaxed version of the median problem that assumes known gene content on the median genome. Which is formally defined as:

3.3.2 Definition

Given the gene content of a median genome, and gene orders of three input genomes. Find an adjacency of the genes of the median genome that minimize the *DCJ-*

Indel-Exemplar(Matching) distance between the median genome and the three input genomes.

The *DCJ-Indel-Exemplar(Matching)* median problem is not even in the class of **NP** because there is no polynomial time algorithm to verify the results. It is hard to design an exact branch-and-bound algorithm for the *DCJ-Indel-Exemplar(Matching)* median problem mainly because the *DCJ-Indel* distance violates the property of triangular inequality which is required for a distance metrics (Yancopoulos and Friedberg 2008). Furthermore, when there are duplicated genes in a genome, it is possible that there are multiple edges of the same type connecting to the same vertex of a *0-matching*, which leads to ambiguity in the edge shrinking step and makes the followed branch-and-bound search process very complicated and extremely hard to implement. To overcome these problems, we provide an adaption of Lin–Kernighan (*LK*) heuristic to help solve this challenging problem.

3.3.3 Design of the Lin–Kernighan heuristic

The *LK* heuristic can generally be divided into two steps: initialize the 0-$matching$ for the median genome, and *LK* search to get the result.

The initialization problem can be described as: given the gene contents of three input genomes, find the gene content of the median genome that minimizes the sum of the number of *Indels* and duplications operations required to transfer the median gene content to the gene contents of the other three genomes. In this paper, we design a very simple rule to initialize the median gene content: given the counts of each gene family occurred with in the three genomes, if two or three counts are the same, we simply select this count as the number of occurrences of the gene family in the median genome; if all three counts are different, we select the median count as the number of occurrences of the gene family in the median genome.

After fixing the gene content for the median genome, we randomly set up the *0-matching* in the *MBG*. The followed *LK* heuristic selects two *0-matching* edges on the *MBG* of a given search node and performs a *DCJ* operation, obtaining the *MBG* of a neighboring search node. We expand the search frontier by keeping all neighboring search nodes to up until the search level $L1$. Then we only examine and add the most promising neighbors to the search list until level $L2$. The search is continued when there is a neighbor solution yielding a better median score. This solution is then accepted and a new search is initialized from scratch. The search will be terminated if there are no improvements to the result as the search level limits have been reached and all possible neighbors have been enumerated. If $L1 = L2 = K$, the algorithm is called *K-OPT* algorithm.

3.3.4 Adopting adequate sub-graphs to simplify problem space

By using the adequate subgraphs (Xu and Sankoff 2008; Xu 2009a), we can prove that they are still applicable for decomposing the graph in the *DCJ-Indel-Exemplar(Matching)* median problem.

Lemma 1 *As long as the irregular vertices do not involve, regular subgraphs are applicable to decompose* MBG.

Proof If there are d number of vertices that contain duplicated edges in *MBG*, we can disambiguate the *MBG* by generating different subgraphs that contain only one of the duplicate edge. We call these subgraphs disambiguate *MBG*, (*d-MBG*), and there are $O(\prod_{i<d} deg(i))$ number of *d-MBG*s. If a regular adequate subgraph exists in the *MBG*, it must also exists in every *d-MBG*. Based on the *0-matching* solution, we can transform every *d-MBG* into completed *d-MBG* (*cd-MBG*) by constructing the optimal completion (Compeau 2012) between *0-matching* and all the other three types of edges. After this step, the adequate subgraphs in every *d-MBG* still exist in every *cd-MBG*, thus we can use these adequate subgraphs to decompose *cd-MBG* for each median problem without losing accuracy. □

3.3.5 Search space reduction methods

The performance bottleneck with the median computation is in the exhaustive search step, because for each search level we need to consider $O(|E|^2)$ possible number of edge pairs, which is $O(|E|^{2L1})$ in total. Unlike the well-studied traveling salesman problem (*TSP*) where it is cheap to find the best neighbor, here we need to compute the *DCJ-Indel-Exemplar(Matching)* problem,*NP*-hard distance, which makes this step extremely expensive to conclude. Noticing that if we search neighbors on edges that are on the same *0-i* color altered connected component (*0-i-comp*), the *DCJ-Indel-Exemplar(Matching)* distance for genome 0 and genome i is more likely to reduce (Yin et al. 2013), thus we can sort each edge pair by how many *0-i-comp* they share. Suppose the number of *0-i-comp* that an edge pair x share is $num\_pair(x)$, when the algorithm is in the exhaustive search step ($currentLevel < L1$), we set a threshold δ and select the edge pairs that satisfy $num\_pair(x) > \delta$ to add into the search list. When it comes to the recursive deepening step, we select the edge pair that satisfy $\underset{x}{\operatorname{argmax}}\ num\_pair(x)$ to add into the search list. This strategy has two merits: (1) some non-promising neighbor solution is eliminated to reduce the search space; (2) the expensive evaluation step which make a function call to *DCJ-Indel-Exemplar(Matching)* distance is postponed to the time when a solution is retrieved from the search list.

The *LK* based median computation algorithm is as Algorithm 2 shows, named DCJIndelExem(Matc)Median respectively.

4 Experimental results

We implement our code with python and C++: the python code realized the optimization methods while the C++ code is implemented on a parallel branch-and-bound framework *OPTKit*. We conducted extensive experiments to evaluate the accuracy and speed of our distance and median algorithms using both simulated and real biological data. Experimental tests ran on a machine with Linux operating system configured with 16 Gb of memory and an Intel(R) Xeon(R) CPU E5530 16 core processors, each

Algorithm 2: DCJINDELEXEM(MATC)MEDIAN

```
Input: MBG $G$, Search Level $L1$ and $L2$
Output: 0-matching of $G$
Init search list $L$ of size $L1$;
Init 0-matching of $G$;
$currentLevel \leftarrow 0$ and $Improved \leftarrow true$;
while $Improved = true$ do
    $currentLevel \leftarrow 0$ and $Improved \leftarrow false$;
    Insert $G$ into $L[0]$;
    while $currentLevel < L2$ do
        $G' \leftarrow$ pop from list $L[currentLevel]$;
        if $G'$ improves the median score then
            $G \leftarrow G'$;
            $Improved \leftarrow true$ and break ;
        if $currentLevel < L1$ then
            for $x \in \forall$ 0-matching pairs of $G$ do
                $G' \leftarrow$ perform DCJ on $G'$ using $x$;
                if $num\_pair(x) > \delta$ then Insert $G'$ into $L[currentLevel + 1]$
        else
            $G' \leftarrow$ perform DCJ on $G'$ using $x = \underset{x}{\operatorname{argmax}}\, num\_pair(x)$ ;
            if $num\_pair(x) > \delta$ then Insert $G'$ into $L[currentLevel + 1]$
        $currentLevel \leftarrow currentLevel + 1$ ;
return 0-matching of $G$;
```

core has 2.4 GHz of speed. All the experiments ran with a single thread. We choose to use g++−4.8.1 as our compiler.

4.1 Distance computation

To the best of our knowledge, there is no software package that can handle both duplications and *Indels*. We compared our *DCJ-Indel-Exemplar (Matching)* distances with *GREDO* (Shao et al. 2014), a software package based on LP that can handle duplications.

4.1.1 Simulated data

The simulated data sets are generated with genomes containing 1000 genes. The *Indels* rate is set (γ) as 5 %, inline with the duplications rate (ϕ) as 10 %. Considering *GREDO* can not process *Indel* data, all *Indels* for *GREDO* are removed. We compared the change of distance estimation with the variation of mutation rate (θ, which grows from 10 to 100 %). The experimental results for simulated data are displayed in Fig. 5.

(1) For computational time, since the results of time spans over a range of thousands of seconds, we display the time with log scale to construe results clearly. When the mutation rate is <50 %, all three methods perform similarly, with the fact

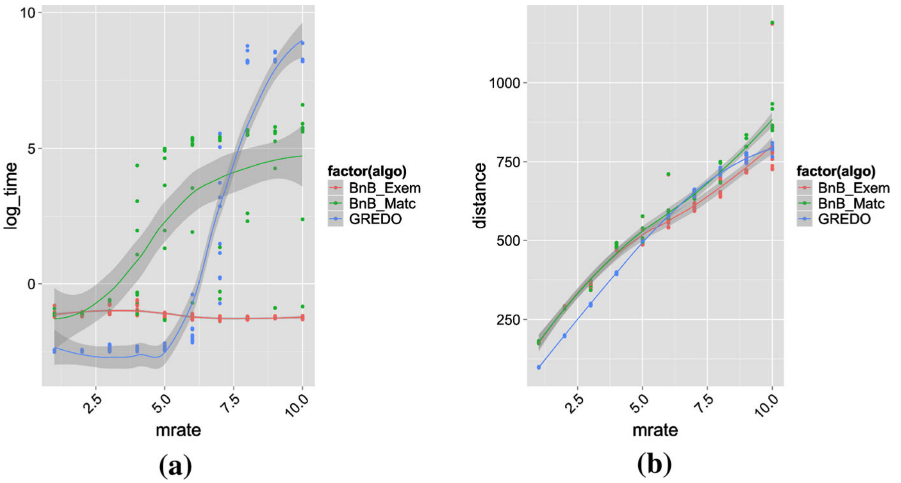

Fig. 5 Experimental results for distance computation using simulated data. **a** Time result for simulated data. **b** Distance result for simulated data

Table 1 Experimental results for distance computation with real data set

Data	Distance results			Time results		
	GREDO	Exem	Matc	GREDO	Exem	Matc
brownrat_chicken	1678	24546	24704	3604.28	172.73	7.45
brownrat_gorilla	1274	17922	17966	5707.13	12.64	12.10
brownrat_human	1083	17858	17900	3725.76	12.14	12.19
brownrat_mouse	790	15433	15445	3725.66	14.51	15.06
chicken_gorilla	1491	16379	16421	3725.62	7.54	7.57
chicken_human	1521	16231	16276	3725.65	7.74	7.47
chicken_mouse	1528	15712	15745	3726.03	9.82	8.16
gorilla_human	486	17798	17798	3607.63	13.94	13.81
gorilla_mouse	860	18914	18935	4816.31	12.60	12.13
human_mouse	749	18126	18144	94.64	12.45	12.61

that *GREDO* is faster than both of our branch-and-bound methods. However, *GREDO* slows down dramatically when the mutation rate is increased, while our branch-and-bound based method takes less increased time to finish.

(2) For computational accuracy, we show the distance results corrected by *EDE* approach which is one of the best true distance estimator. As for simulated data, we can see that when the mutation rate is small ($<50\,\%$) *GREDO* under estimates the distance as opposed to our two branch-and-bound methods; but it will over estimate the distance with the growth of mutation rate.

4.1.2 Real data

We prepare the real data sets using genomes downloaded from Ensenble and processed them following the instructions in Shao et al. (2014). The real data set contains 5 species: brown-rat, chicken, human, mouse and gorilla. For *DCJ-Indel-Exemplar (Matching)* distance, we only convert the Ensenmble format to adapt the data to our program. Meanwhile, just as the simulated data, all *Indels* in real data set for *GREDO* are removed. The results for real data are shown in Table 1.

(1) For computational time, the branch-and-bound method shows orders of magnitudes of speed up compared with *GREDO*. We analyze the data, the reason can be construed as the existence of multiple connected component in *BPG*. So that our method can divide the graph into much smaller size, versus *GREDO* which doesn't have this mechanism.

(2) For computational accuracy, the distance results of the real data gives us a taste of how frequently *Indels* happened in the genome evolution. We can see orders of magnitude of difference between our distance results and *GREDO*, which is mainly due to the large amount of *Indels* in the real data set. Note that we did not change the way *GREDO* compute its distance as in paper (Shao et al. 2014), in the real distance computation, we should consider *Indels* in alignment with duplications.

4.2 Median computation

We simulate the median data of three genomes using the same strategy as in the distance simulation. In our experiments, each genome is "evolved" from a seed genome, which is identity, and they all evolve with the same evolution rate (θ, γ and ϕ). The sequence length in the median experiments are reduced to 50, due to performance issues.

4.2.1 DCJ-Indel-Exemplar median

We analyze the result of using *LK* algorithm with $L1 = 2$ and $L2 = 3$, and the *K-OPT* algorithm of $K = 2$. Search space reduction methods are used, with $\delta = 2$ and $\delta = 3$ respectively.

(1) To begin with, we compared our result along with equal content data, because there are already benchmark programs to help us to perform analysis. We run the exact *DCJ* median solver [we use the one in Yin et al. (2013)] to compare our heuristic with the exact median results. In Fig. 6a, it shows the accuracy of our heuristic versus the exact result. It is shown that when $\theta \leq 60\,\%$, all results of the *LK* and *K-OPT* methods are quite close to the exact solver. For parameter of $\delta = 2$, both *LK* and *K-OPT* methods can generate exactly the same results for most of the cases.

(2) As for the median results for unequal contents, we set both γ and ϕ to 5 % and increase the mutation (inversion) rate θ from 10 to 60 %. We compare our results with the accumulated distance of the three genomes to their simulation seed.

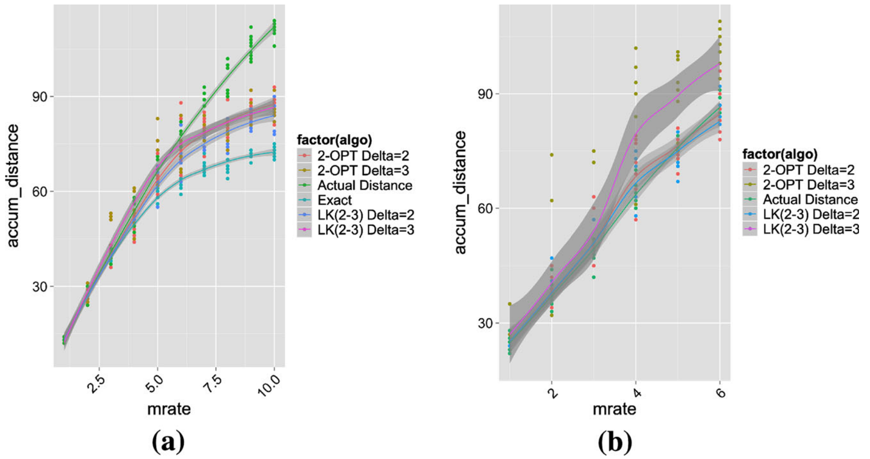

Fig. 6 Experimental results for median computation applying *DCJ-Indel-Exemplar* distance. **a** $\gamma = \phi = 0\,\%$ and θ varies from 10 to 100 %. **b** $\gamma = \phi = 5\,\%$ and θ varies from 10 to 60 %

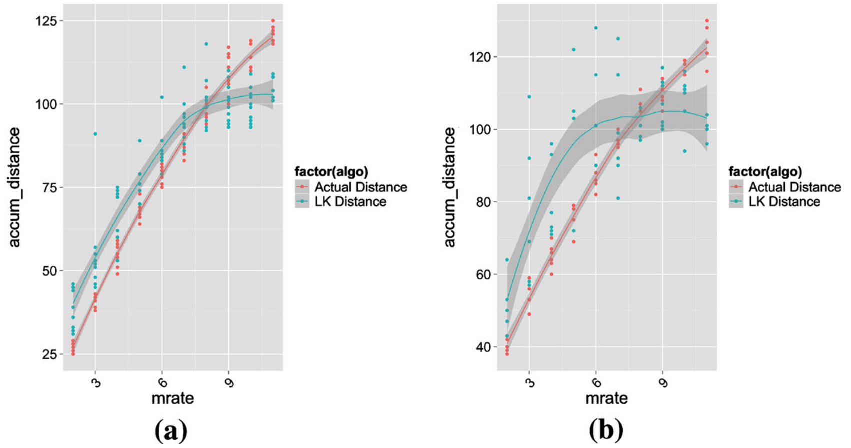

Fig. 7 Experimental results for median computation applying *DCJ-Indel-Matching* distance. **a** $\gamma = \phi = 5\,\%$ and θ varies from 10 to 100 %. **b** $\gamma = \phi = 10\,\%$ and θ varies from 10 to 100 %

Although it can not show the accuracy of our method (since we do not have an exact solver), it can be used as an indicator of how close that our method is to the real evolution. Figure 6b shows that when $\delta = 3$, both the *LK* and *K-OPT* algorithms get results quite close to the real evolutionary distance.

4.2.2 DCJ-Indel-matching median

Because the *DCJ-Indel-Exemplar* median has already given us the result of how *LK* performs against exact solver, and how different parameters of *LK* performs. With these things in mind, we choose to use *LK* with $L1 = 2$ and $L2 = 3$ having $\delta = 2$ as the configuration for our *DCJ-Indel-Matching* median solver. We use the same data as in the previous experiments, and the experimental results are shown in Fig. 7a and b. We can see that in general, the new implementation is quite close to the real result when $\gamma = 5\,\%$ and $\phi = 5\,\%$ and slightly worse than real result when $\gamma = 10\,\%$ and $\phi = 10\,\%$.

5 Conclusion

In this paper, we proposed a new method to compute the distance and median between genomes with unequal contents (with *Indels* and duplications). Our distance method can handle Indels which is ubiquitous in the real data set, and is proved to be more efficient as opposed to *GREDO*. We designed a Lin–Kernighan based method to compute median, which can get close to optimal results in alignment with the exact median solver, and our methods can handle duplications and *Indels* as well.

Acknowledgments This Research was sponsored in part by the NSF OCI-0904461 (Bader), OCI-0904179, IIS-1161586 (Tang) and OCI- 0904166 (Schaeffer).

References

Angibaud S, Fertin G, Rusu I, Thévenin A, Vialette S (2009) On the approximability of comparing genomes with duplicates. J. Graph Algorithms Appl. 13(1):19–53

Bader DA, Moret BME, Yan M (2001) A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. J Comput Biol 8:483–491

Bafna V, Pevzner PA (1998) Sorting by transpositions. SIAM J Discret Math 11(2):224–240

Bergeron A, Mixtacki J, Stoye J (2005) On sorting by translocations. In: Journal of computational biology. Springer, Heidelberg, pp 615–629

Blin G, Chauve C, Fertin G (2004) The breakpoint distance for signed sequences. In: Proceedings of CompBioNets 2004. vol. text in algorithms. King's College, London, pp 3–16

Bourque G, Pevzner PA (2002) Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Res 12(1):26–36

Braga MDV, Willing E, Stoye J (2010) Genomic distance with DCJ and indels. In: Proceedings of the 10th international conference on algorithms in bioinformatics, WABI'10. Springer, Berlin/Heidelberg, pp 90–101

Brewer C, Holloway S, Zawalnyski P, Schinzel A, FitzPatrick D (1999) A chromosomal duplication map of malformations: regions of suspected haplo and triplolethality and tolerance of segmental aneuploidy in humans. Am J Hum Genet 64(6):1702–1708

Caprara A (2003) The reversal median problem. INFORMS J Comput 15(1):93–113

Chauve C, Fertin G, Rizzi R, Vialette S (2006) Genomes containing duplicates are hard to compare. In: Proceedings of international workshop on bioinformatics research and applications (IWBRA), LNCS. Springer, Reading, pp 783–790

Chen X, Zheng J, Fu Z, Nan P, Zhong Y, Lonardi S, Jiang T (2005) Assignment of orthologous genes via genome rearrangement. IEEE/ACM Trans Comput Biol Bioinform 2(4):302–315

Chen Z, Fu B, Zhu B (2012) Erratum: the approximability of the exemplar breakpoint distance problem. In: FAW-AAIM. Springer, Heidelberg, p 368

Compeau PEC (2012) A simplified view of DCJ-indel distance. In: Proceedings of the 12th international conference on algorithms in bioinformatics, WABI'12. Springer, Berlin/Heidelberg, pp 365–377

Gao N, Yang N, Tang J (2013) Ancestral genome inference using a genetic algorithm approach. PLoS One 8(5):e62156

Hu F, Zhou J, Zhou L, Tang J (2014) Probabilistic reconstruction of ancestral gene orders with insertions and deletions. IEEE/ACM Trans Comput Biol Bioinform 11(4):667–672

Lenne R, Solnon C, Stutzle T, Tannier E, Birattari M (2008) Reactive stochastic local search algorithms for the genomic median problem. In: Carlos Cotta JVH (ed) Eighth European conference on evolutionary computation in combinatorial optimisation (EvoCOP). LNCS, Springer, Berlin, pp 266–276

Mabrouk NE (2001) Sorting signed permutations by reversals and insertions/deletions of contiguous segments. J Discret Algorithms 1(1):105–122

Moret BME, Tang J, san Wang L, Warnow Y (2002) Steps toward accurate reconstructions of phylogenies from gene-order data. J Comput Syst Sci 65:508–525

Moret BME, Wang LS, Warnow T, Wyman SK (2001) New approaches for reconstructing phylogenies from gene order data. In: ISMB (Supplement of bioinformatics), pp 165–173

Nguyen CT, Tay YC, Zhang L (2005) Divide-and-conquer approach for the exemplar breakpoint distance. Bioinformatics 21(10):2171–2176

Pe’er I, Shamir R (1998) The median problems for breakpoints are np-complete. Technical Report 71, Electronic Colloquium on Computational Complexity

Rajan V, Xu AW, Lin Y, Swenson KM, Moret BME (2010) Heuristics for the inversion median problem. BMC Bioinform 11(S–1):30

Sankoff D (1999) Genome rearrangement with gene families. Bioinformatics 15(11):909–917

Shao M, Lin Y (2012) Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. BMC Bioinform 13(S–19):S13

Shao M, Lin Y, Moret BME (2014) An exact algorithm to compute the DCJ distance for genomes with duplicate genes. In: RECOMB, Pittsburgh, pp 280–292

Xu AW (2009) DCJ median problems on linear multichromosomal genomes: graph representation and fast exact solutions. In: RECOMB-CG, Budapest, pp 70–83

Xu AW (2009) A fast and exact algorithm for the median of three problem: a graph decomposition approach. J Comput Biol 16(10):1369–1381

Xu AW, Moret BME (2011) Gasts: parsimony scoring under rearrangements. In: WABI. Springer, Berlin, pp 351–363

Xu AW, Sankoff D (2008) Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In: Proceedings of the 8th international workshop on algorithms in bioinformatics, WABI ’08. Springer, Berlin/Heidelberg, pp 25–37

Yancopoulos S, Attie O, Friedberg R (2005) Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics 21(16):3340–3346

Yancopoulos S, Friedberg R (2008) Sorting genomes with insertions, deletions and duplications by DCJ. In: Nelson CE, Vialette S (eds) RECOMB-CG. Lecture notes in computer science, vol 5267. Springer, Berlin, pp 170–183

Yin Z, Tang J, Schaeffer SW, Bader DA (2013) Streaming breakpoint graph analytics for accelerating and parallelizing the computation of DCJ median of three genomes. In: ICCS, pp 561–570