Sorting Genomes with Rearrangements and Segmental Duplications through Trajectory Graphs

Mingfu Shao  Yu Lin  Bernard Moret

Swiss Federal Institute of Technology, Lausanne (EPFL)

October 19, 2013
Model of Genomes

- Each genome is modeled as a set of chromosomes.
- Each chromosome is modeled as a (linear/circular) sequence of genes.
- Terminology: extremity, adjacency
Model of Genomic Evolution

- Evolutionary events:
  - Rearrangements: Double-Cut-and-Join (DCJ) operation
  - Gene-content modifying events: Segmental duplication
Model of Genomic Evolution

- Evolutionary events:
  - Rearrangements: Double-Cut-and-Join (DCJ) operation
  - Gene-content modifying events: Segmental duplication

- Optimization Problem: Transform $G_1$ to $G_2$ with minimum number of DCJ operations and segmental duplications.
Evolutionary events:
- Rearrangements: Double-Cut-and-Join (DCJ) operation
- Gene-content modifying events: Segmental duplication

Optimization Problem: Transform $G_1$ to $G_2$ with minimum number of DCJ operations and segmental duplications.

Strategy: Use the **trajectory graph** to represent a given initial sorting path, and then simplify it.
DCJ Operation

\[
\begin{array}{cccc}
  a & b & c & d \\
\end{array}
\]
DCJ Operation

\[ \{a_h, b_t\} \quad \{c_h, d_t\} \]
DCJ Operation

\[
\{a_h, b_t\} - \{c_h, d_t\}
\]

inversion

\[
\{a_h, c_h\} - \{b_t, d_t\}
\]
DCJ Operation

\[
\begin{align*}
\{a_h, b_t\} & \quad \{c_h, d_t\} \\
\{a_h, c_h\} & \quad \{b_t, d_t\}
\end{align*}
\]

inversion

fission

\[
\begin{align*}
\{a_h, d_t\} & \quad \{b_t, c_h\}
\end{align*}
\]
DCJ Operation
DCJ Operation

\[ \{b_h, c_t\} \]

\[ \{f_h, g_t\} \]
DCJ Operation

translocation

Mingfu Shao, Yu Lin, Bernard Moret
October 19, 2013
DCJ as Function of Adjacencies

- **Input:** Two adjacencies $\{a_h, b_t\}$ and $\{c_h, d_t\}$.
- **Output:** Two new adjacencies $\{a_h, c_h\}$ and $\{b_t, d_t\}$, or $\{a_h, d_t\}$ and $\{b_t, c_h\}$.

All four edges are defined as **active edges**.
Segmental Duplication

\[ \{a_h, b_l\} \]
Segmental Duplication

\[ \{a_h, b_l\} \]
Segmental Duplication

\[ \{a_h, b_t\} \]
Segmental Duplication

\[
\begin{align*}
\{a_h, b_t\} & \quad \{a_h, c_t\} & \quad \{e_h, b_t\} \\
\{a_h, e_h\} & \quad \{c_t, b_t\}
\end{align*}
\]
Segmental Duplication

\[
\begin{align*}
(a, b) & \quad \{a_h, b_t\} \\
(c, d, e) & \\
(a, -e', -d', -c') & \quad \{a_h, e_h\} \quad \{c_t, b_t\} \\
\end{align*}
\]
Segmental Duplication as Function of Adjacencies

- **Input:** One adjacencies (inserting position), and a segment with \( n \) genes, which is represented by \( n - 1 \) adjacencies.
- **Output:** \( n + 1 \) adjacencies representing the new segment, plus the old \( n - 1 \) adjacencies.
**Input:** One adjacencies (inserting position), and a segment with \( n \) genes, which is represented by \( n - 1 \) adjacencies.

**Output:** \( n + 1 \) adjacencies representing the new segment, plus the old \( n - 1 \) adjacencies.
Segmental Duplication as Function of Adjacencies

- **Input:** One adjacencies (inserting position), and a segment with \( n \) genes, which is represented by \( n - 1 \) adjacencies.
- **Output:** \( n + 1 \) adjacencies representing the new segment, plus the old \( n - 1 \) adjacencies.
Segmental Duplication as Function of Adjacencies

- **Input**: One adjacencies (inserting position), and a segment with \( n \) genes, which is represented by \( n - 1 \) adjacencies.

- **Output**: \( n + 1 \) adjacencies representing the new segment, plus the old \( n - 1 \) adjacencies.
Segmental Duplication as Function of Adjacencies

- **Input:** One adjacencies (inserting position), and a segment with \( n \) genes, which is represented by \( n - 1 \) adjacencies.
- **Output:** \( n + 1 \) adjacencies representing the new segment, plus the old \( n - 1 \) adjacencies.

The three edges related with the inserting position are active.
Segmental Duplication as Function of Adjacencies

- **Input:** One adjacencies (inserting position), and a segment with $n$ genes, which is represented by $n - 1$ adjacencies.
- **Output:** $n + 1$ adjacencies representing the new segment, plus the old $n - 1$ adjacencies.
Segmental Duplication as Function of Adjacencies

- **Input:** One adjacencies (inserting position), and a segment with $n$ genes, which is represented by $n - 1$ adjacencies.

- **Output:** $n + 1$ adjacencies representing the new segment, plus the old $n - 1$ adjacencies.

The three edges related with the inserting position are **active**.
Segmental Duplication as Function of Adjacencies

- **Input:** One adjacencies (inserting position), and a segment with \( n \) genes, which is represented by \( n - 1 \) adjacencies.
- **Output:** \( n + 1 \) adjacencies representing the new segment, plus the old \( n - 1 \) adjacencies.

The three edges related with the inserting position are active.
Trajectory Graph

\[ G_1 \rightarrow a \rightarrow b \rightarrow c \rightarrow d \]
Trajectory Graph

\[ G_1 \]

\[ 0, a \quad b, a_{t}, b \quad b, c \quad c_{t}, d \quad d, 0 \]
Trajectory Graph

$G_1 \rightarrow a \rightarrow b \rightarrow c \rightarrow d$

$\{c_h, d_t\}$

$\rightarrow a \rightarrow b \rightarrow c \rightarrow d'$

$0, a_t \quad a_h, b_t \quad b_h, c_t \quad c_h, d_t \quad d_h, 0$
Trajectory Graph

\[ G_1 \]

\[ \{c_h, d_t\} \]

\[ 0, a_t \quad a_h, b_t \quad b_h, c_t \quad c_h, d_t \quad d_h, 0 \]

DUP

\[ a_h, b_t \quad c_h, a'_t \quad a'_h, b'_t \quad b'_h, d_t \]
Trajectory Graph

$G_1$ with edges $\{a, b, c, d\}$ and $\{c_h, d_t\}$.

Additional graph with edges $\{0, a_t\}$, $\{c_h, a'_t\}$, and $\{-c, -b, -a, a', b', d\}$.

DUP node connected to $\{0, a_t\}$, $\{a_h, b_t\}$, $\{b_h, c_t\}$, $\{c_h, d_t\}$, $\{d_h, 0\}$, $\{a_h, b_t\}$, $\{c_h, a'_t\}$, $\{a'_h, b'_t\}$, $\{b'_h, d_t\}$. 
Trajectory Graph

\[ G_1 \]

\[
\begin{align*}
\{0, a_t\} & \quad \{c_h, a'_t\} \\
-c & \quad -b & \quad -a & \quad a' & \quad b' & \quad d
\end{align*}
\]

\[
\begin{align*}
\{c_h, d_t\} \\
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0
\end{align*}
\]

DUP

DCJ-1

\[
\begin{align*}
0, c_h & \quad a_t, a'_t
\end{align*}
\]
Trajectory Graph

\[ G_1 \]

\[ a \rightarrow b \rightarrow c \rightarrow d \]

\{c_h, d_t\}

\[ a \rightarrow b \rightarrow c \rightarrow a' \rightarrow b' \rightarrow d \]

\{0, a_t\}, \{c_h, a'_t\}

\[ -c \rightarrow -b \rightarrow -a \rightarrow a' \rightarrow b' \rightarrow d \]

\{b_t, a'_h\}, \{b'_h, d_t\}

\[ -c \rightarrow -b \rightarrow -b' \rightarrow -a' \rightarrow a \rightarrow d \]

\[ 0, a_t \rightarrow a_h, b_t \rightarrow b_h, c_t \rightarrow c_h, d_t \rightarrow d_h, 0 \]

DUP

DCJ-1

\[ 0, c_h \rightarrow a_t, a'_t \]

\[ a_h, b_t \rightarrow c_h, a'_t \rightarrow a'_h, b'_t \rightarrow b'_h, d_t \]
Trajectory Graph

$$G_1$$

$\{0, a_t\} \rightarrow \{c_h, a'_t\} \rightarrow \{b_t, a'_h\} \rightarrow \{b'_h, d_t\} \rightarrow \{0, c_h, a'_t, b_t, b'_h, a_h, d_t\}$

$\{c_h, d_t\}$

$0, a_t \leftarrow a, b, c, d \leftarrow a', b', c, d \leftarrow a', b', c, d$
Trajectory Graph

$G_1$

$G_2$

$0, a_t$ $a_h, b_t$ $b_h, c_t$ $c_h, d_t$ $d_h, 0$

DUP

DCJ-1

DCJ-2

$0, c_h$ $a_t, a_t'$ $b_t, b_h'$ $a_h, d_t$
Trajectory Graph

\[ G_1 \]

\[ a \rightarrow b \rightarrow c \rightarrow d \]
\[ \{c_h, d_t\} \]

\[ G_2 \]

\[ a' \rightarrow b' \rightarrow a \rightarrow b' \rightarrow d \]
\[ \{b_t, a_h'\} \]

\[ \{a_t', a_t\} \]

\[ D \]

\[ DUP \]

\[ DCJ-1 \]

\[ 0, a_t \]
\[ a_h, b_t \]
\[ b_h, c_t \]
\[ c_h, d_t \]
\[ d_h, 0 \]

\[ DCJ-2 \]

\[ a_h, b_t \]
\[ c_h, a_t' \]
\[ a_h', b_t' \]
\[ b_h', d_t \]

\[ DCJ-3 \]

\[ 0, c_h \]
\[ a_t, a_t' \]
\[ b_t, b_t' \]
\[ a_h, d_t \]
Trajectory Graph

$G_1$

$G_2$

DUP

DCJ-1

DCJ-2

DCJ-3
Optimization Algorithm

Active Cycle Theorem
If there exists any active cycle in the trajectory graph, then we can reduce the number of DCJ operations at least by 1.

Algorithm
Iteratively exchange the bottom DCJ operation with one of the above operation node, to reduce the size of the active cycle.
Exchange Two Operations—Step 1
Exchange Two Operations—Step 1
Exchange Two Operations—Step 1

0, a_t

DUP

DCJ-1

DCJ-3

0, c_h

a_t, b'_h

a'_t, b_t

0, c_h

a_t, b'_h

a'_t, b_t

DUP

DCJ-2

DUP

DCJ-2

b_t, b'_h

a_h, d_t

0, a_t

a_h, b_t

b_h, c_t

a'_h, d'_t

a'_h, b'_t

b'_h, d_t

0, a_t

a_h, b_t

b_h, c_t

c_h, d_t

d_h, 0

a'_h, b'_t

b'_h, d_t

b_t, b'_h

a_h, d_t
Exchange Two Operations—Step 1

0, a_t, a_h, b_t, b_h, c_t, c_h, d_t, d_h, 0

DUP

DCJ-1

DCJ-2

DCJ-3

0, a_t

a_h, b_t, b_h, c_t, c_h, d_t, d_h, 0

DUP

DCJ-2

DCJ-3

0, c_h, a_t, b_h, a'_t, b_t, c_h, b'_h

0, c_h, a_t, b'_h, a'_t, b_t

Mingfu Shao, Yu Lin, Bernard Moret

October 19, 2013
Exchange Two Operations—Step 1

![Diagram showing the exchange of two operations between two sets of nodes labeled with variables.]
Exchange Two Operations—Step 2

\[
\begin{array}{cccc}
0, a_t & a_h, b_t & b_h, c_t & c_h, d_t & d_h, 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
DUP & b_t, b'_h & b_h, 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
DCJ-2 & b_h, d_t & a_h, b_t \\
\end{array}
\]

\[
\begin{array}{ccc}
DCJ-3 & a_t, b'_h & c_h, b'_h \\
\end{array}
\]

\[
\begin{array}{ccc}
DCJ-1 & 0, c_h & a_h, d_t \\
\end{array}
\]
Exchange Two Operations—Step 2

0, a_t, a_h, b_t, b_h, c_t, c_h, d_t, d_h, 0

DUP

0, a_h, b_t, c_h, d_t, b_h, d_t, 0

DCJ-2

b_t, b_h

DCJ-3

b_t, b_h

DCJ-1

0, c_h, a_t, b_t, a_t, b_t, a_h, d_t, a_h, d_t, c_h, b_h, c_h, b_h

Mingfu Shao, Yu Lin, Bernard Moret October 19, 2013
Exchange Two Operations—Step 2

0, \(a_t\)  \(a_h, b_t\)  \(b_h, c_t\)  \(c_h, d_t\)  \(d_h, 0\)

DUP

\(a_h, b_t\)  \(c_h, a'_t\)  \(a'_h, b'_t\)  \(b'_h, d_t\)

DCJ-2

\(b_t, b'_h\)

DCJ-3

\(c_h, b'_h\)

DCJ-1

\(0, c_h\)  \(a_t, b'_h\)  \(a'_t, b_t\)

\(0, a_t\)  \(a_h, b_t\)  \(b_h, c_t\)  \(c_h, d_t\)  \(d_h, 0\)

DUP

\(a_h, b_t\)  \(c_h, a'_t\)  \(a'_h, b'_t\)  \(b'_h, d_t\)

\(b'_h, c_h\)

DCJ-1

\(0, c_h\)  \(a_t, b'_h\)  \(a'_t, b_t\)
Exchange Two Operations—Step 2

\[
\begin{align*}
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
\end{align*}
\]

DUP

\[
\begin{align*}
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
\end{align*}
\]

DCJ-2

\[
\begin{align*}
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
\end{align*}
\]

DCJ-3

\[
\begin{align*}
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
\end{align*}
\]

DCJ-1

\[
\begin{align*}
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
\end{align*}
\]
Exchange Two Operations—Step 2

0, a_t \rightarrow \begin{array}{l}
0, c_h \\
a_t, b'_h \\
a'_t, b_t
\end{array}

\begin{array}{l}
b_t, b'_h
\end{array}

\begin{array}{l}
d_h, 0
\end{array}

\begin{array}{l}
0, a_t \\
a_h, b_t \\
b_h, c_t \\
c_h, d_t \\
d_h, 0
\end{array}

\begin{array}{l}
dcj-1
\end{array}

\begin{array}{l}
dcj-2
\end{array}

\begin{array}{l}
dcj-3
\end{array}

\begin{array}{l}
 DUP
\end{array}

\begin{array}{l}
dcj-1
\end{array}

\begin{array}{l}
dcj-2
\end{array}

\begin{array}{l}
dcj-3
\end{array}

0, a_t 

\begin{array}{l}
0, c_h \\
a_t, b'_h \\
a'_t, b_t
\end{array}

\begin{array}{l}
b_t, b'_h \\
d_h, 0
\end{array}

\begin{array}{l}
0, a_t \\
a_h, b_t \\
b_h, c_t \\
c_h, d_t \\
d_h, 0
\end{array}

\begin{array}{l}
dcj-1
\end{array}

\begin{array}{l}
dcj-2
\end{array}

\begin{array}{l}
dcj-3
\end{array}

0, a_t 

\begin{array}{l}
0, c_h \\
a_t, b'_h \\
a'_t, b_t
\end{array}

\begin{array}{l}
DUP
\end{array}

\begin{array}{l}
dcj-1
\end{array}

\begin{array}{l}
dcj-2
\end{array}

\begin{array}{l}
dcj-3
\end{array}
Merge Segmental Duplication and DCJ

0, a_t, a_h, b_t, b_h, c_t, c_h, d_t, d_h, 0

DUP

a_h, b_t, c_h, a'_t, a'_h, b'_t, b'_h, d_t

DCJ-3

c_h, b'_h

d'_t, d_t

DCJ-1

0, c_h, a_t, b'_h

DCJ-2

a'_t, b_t, a_h, d_t
Merge Segmental Duplication and DCJ

\[0, a_t \quad a_h, b_t \quad b_h, c_t \quad c_h, d_t \quad d_h, 0\]

DUP

\[a_h, b_t \quad c_h, a'_t \quad a'_h, b'_t \quad b'_h, d_t\]

DCJ-3

\[c_h, b'_h \quad d'_t, d_t\]

DCJ-1

0, c_h

\[a_t, b'_h\]

DCJ-2

\[a'_t, b_t \quad a'_h, d_t\]
Merge Segmental Duplication and DCJ

0, \(a_t\) \(a_h, b_t\) \(b_h, c_t\) \(c_h, d_t\) \(d_h, 0\)

\(0, a_t\) \(a_h, b_t\) \(b_h, c_t\) \(c_h, d_t\) \(d_h, 0\)

\(c_h, b'_h\) \(a'_t, d_t\)

\(d_t\)

\(0, c_h\) \(a_t, b'_h\) \(a'_t, b_t\) \(a_h, d_t\)

\(0, c_h\) \(a_t, b'_h\) \(a'_t, b_t\) \(a_h, d_t\)

\(0, a_t\) \(a_h, b_t\) \(b_h, c_t\) \(c_h, d_t\) \(d_h, 0\)

\(0, a_t\) \(a_h, b_t\) \(b_h, c_t\) \(c_h, d_t\) \(d_h, 0\)

\(c_h, b'_h\) \(a'_t, d_t\)

\(0, c_h\) \(a_t, b'_h\) \(a'_t, b_t\) \(a_h, d_t\)
Merge Segmental Duplication and DCJ

\[
\begin{align*}
0, a_t & \quad a_h, b_t & b_h, c_t & c_h, d_t & d_h, 0 \\
& \quad a_h, b_t & c_h, a'_t & a'_h, b'_t & b'_h, d_t \\
& \quad c_h, b'_h & a'_t, d_t & & \\
& & DCJ-1 & DCJ-2 & \\
0, c_h & a_t, b'_h & a'_t, b_t & a_h, d_t & \\
\end{align*}
\]
Merge Segmental Duplication and DCJ

Diagram showing the process of merging segmental duplication and DCJ operations.

DUP node with edges leading to DCJ-1 and DCJ-2 nodes.

- DCJ-1: 0, c, a, b', a', b, a, d
- DCJ-2: 0, c, a, b', a', b, a, d

DCJ-3: 0, a, b, c, d

Operations:
- c × a' × b' × d
- {c, a'} × {b', d}

Mingfu Shao, Yu Lin, Bernard Moret
October 19, 2013
Merge Segmental Duplication and DCJ

\[
\begin{align*}
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
& \quad a_h, b_t & \quad c_h, a_t' & \quad a_h', b_t' & \quad b_h', d_t \\
& \quad c_h, b_t' & \quad a_t', d_t & \quad b_h', c_t & \quad d_h, 0 \\
0, a_t & \quad a_h, b_t & \quad b_h, c_t & \quad c_h, d_t & \quad d_h, 0 \\
& \quad a_h, b_t & \quad c_h, b_t' & \quad a_t', b_t' & \quad b_h', d_t \\
& \quad a_h', b_t' & \quad c_h, b_t' & \quad a_t', d_t & \quad b_h', c_t & \quad d_h, 0 \\
\end{align*}
\]
Theorem

If the corresponding trajectory graph consists of only trees, then the sorting path is optimal.
Theorem

If the corresponding trajectory graph consists of only trees, then the sorting path is optimal.
Restricted on DCJ Model

**Theorem**

If the corresponding trajectory graph consists of only trees, then the sorting path is optimal.

\[ \#\text{DCJ} = n - 1 \]
Restricted on DCJ Model

Theorem
If the corresponding trajectory graph consists of only trees, then the sorting path is optimal.

\[ \text{DCJ-1} \]

\[ \text{DCJ-2} \]

\[ \text{DCJ-3} \]

\[ \#\text{DCJ} = n - 1 \]
Restricted on DCJ Model

**Theorem**

If the corresponding trajectory graph consists of only trees, then the sorting path is optimal.

\[
\text{#DCJ} = n - 1
\]

\[
\text{#OPT-DCJ} = n - c
\]
## Theorem

If the corresponding trajectory graph consists of only trees, then the sorting path is optimal.

\[ \text{#DCJ} = n - 1 \]

\[ \text{#OPT-DCJ} = n - c \]

We need to prove that \( c = 1 \).
Proof of the Theorem

Lemma
The adjacency graph corresponding to a single tree consists of exactly one cycle.

Proof: by contradiction.
Proof of the Theorem (cont.)

\[ \begin{align*}
  a_h, b_t & \rightarrow b_h, c_t & c_h, d_t & \rightarrow d_h, e_t & e_h, a_t \\
  b_h, c_h & \rightarrow c_t, d_t & d_h, e_h & \rightarrow e_t, a_t \\
  b_t, b_h & \rightarrow d_t, e_h & c_t, d_h & \\
  b_t, d_t & \rightarrow b_h, e_h
\end{align*} \]
Proof of the Theorem (cont.)

\[ \begin{align*}
\{R, R\} & \cup \{R, B\} \rightarrow \{R, R\} \cup \{R, B\} \\
\{R, R\} & \cup \{R, B\} \rightarrow \{R, R\} \cup \{R, B\} \\
\{R, R\} & \cup \{R, B\} \rightarrow \{R, R\} \cup \{R, B\}
\end{align*} \]
1  \( \{R,R\} + \{R,R\} \rightarrow \{R,R\} + \{R,R\} \) (degree = 0)
Proof of the Theorem (cont.)

1. \( \{R, R\} + \{R, R\} \rightarrow \{R, R\} + \{R, R\} \) (degree = 0)
2. \( \{R, R\} + \{B, B\} \rightarrow \{R, B\} + \{R, B\} \) (degree = 2)
Proof of the Theorem (cont.)

1. \( \{R, R\} + \{R, R\} \rightarrow \{R, R\} + \{R, R\} \) (degree = 0)
2. \( \{R, R\} + \{B, B\} \rightarrow \{R, B\} + \{R, B\} \) (degree = 2)
3. \( \{R, R\} + \{R, B\} \rightarrow \{R, R\} + \{R, B\} \) (degree = 2)
Proof of the Theorem (cont.)

1. \( \{R, R\} + \{R, R\} \rightarrow \{R, R\} + \{R, R\} \) (degree = 0)
2. \( \{R, R\} + \{B, B\} \rightarrow \{R, B\} + \{R, B\} \) (degree = 2)
3. \( \{R, R\} + \{R, B\} \rightarrow \{R, R\} + \{R, B\} \) (degree = 2)
4. \( \{R, B\} + \{R, B\} \rightarrow \{R, R\} + \{B, B\} \) (degree = 2)
Proof of the Theorem (cont.)

1. \{R, R\} + \{R, R\} → \{R, R\} + \{R, R\} (degree = 0)
2. \{R, R\} + \{B, B\} → \{R, B\} + \{R, B\} (degree = 2)
3. \{R, R\} + \{R, B\} → \{R, R\} + \{R, B\} (degree = 2)
4. \{R, B\} + \{R, B\} → \{R, R\} + \{B, B\} (degree = 2)
5. \{R, B\} + \{R, B\} → \{R, B\} + \{R, B\} (degree = 4) □
Conclusion

- We proposed a new graphical data structure, trajectory graph, that can naturally combine rearrangements and segmental duplications in a single model.

- We gave an iterative algorithm to reduce redundant DCJ operations.

- We proved that our algorithm converges to the optimal solution from any initial trajectory when the model is restricted to rearrangements.
Future Work

- Extend the trajectory to include more evolutionary events, for example, gene insertion and gene loss.

- Design new optimization algorithms for the trajectory graphs, for example, merging duplications.

- Find good initial solutions.

Thank You!