Molecular Algorithms

DNA Algorithm for the Graph Coloring Problem
Xiaoming Liu et al.(2005)

Julien A. Nguinkal

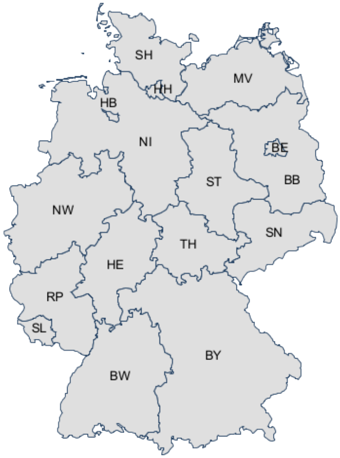
Department of Bioinformatics
Faculty for Mathematics and Computer Science
Friedrich-Schiller University of Jena

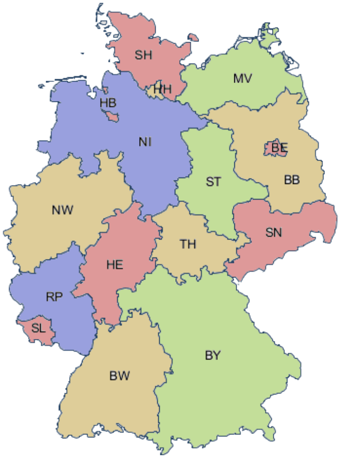
Block Colloquium, Summer 2015

Outline

Outline

Motivation - Map Coloring Problem





Map Coloring Problem...

Coloring the countries of the card with a minimum number of colors so that adjacent Countries do not have the same color

- The most famous graph coloring problem
- Proposed in the nineteenth century and finally solved in 1976.

Outline

Graph Coloring Problem

- A proper coloring of a graph is an assignment of nodes(vertices) with colors so that each 2 adjacent nodes have different colors scheme.

Graph Coloring Problem

- A proper coloring of a graph is an assignment of nodes(vertices) with colors so that each 2 adjacent nodes have different colors scheme.
- A c-coloring is a coloring which uses c different colors.

Graph Coloring Problem

- A proper coloring of a graph is an assignment of nodes(vertices) with colors so that each 2 adjacent nodes have different colors scheme.
- A c-coloring is a coloring which uses c different colors.
- The chromatic number X(G) of a graph G is the smallest number c for which G a c-coloring possesses.

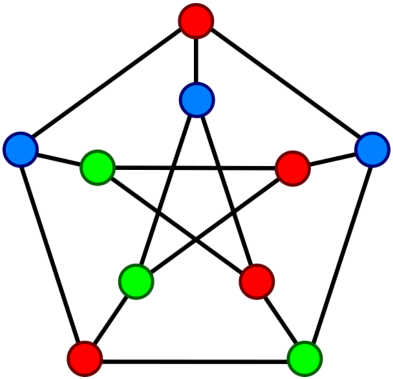


Figure: Petersen graph

Graph Coloring - Formal

Given: Graph G = (V, E)
Sought: Coloring of G with X(G)

Graph Coloring - Formal

Given: Graph G = (V, E)
Sought: Coloring of G with X(G)

$\Longrightarrow$ **can we find a c-coloring for G or is G c colorable?**

Graph Coloring - Formal

Given: Graph G = (V, E)
Sought: Coloring of G with X(G)

$\Longrightarrow$ **can we find a c-coloring for G or is G c colorable?**

$\Longrightarrow Answer\, YES/NO : NP - hard\;\; problem!$

Solution with molecular algorithms : DNA Computing

Outline

Croitorus Algorithm (2002)

Let G=(V,E) be a graph with the vertices set V={ 1, 2,. . . , n} and
S_n *the set of all permutations on the set* $V ==> S_n = n!$

Croitorus Algorithm (2002)

Let G=(V,E) be a graph with the vertices set V={ 1, 2,. . . , n} and S_n *the set of all permutations on the set* $V ==> S_n = n!$

Theorem

For an element $\boldsymbol{e}$ $v = v_1 \dots v_i \dots v_j \dots v_n$ *of* S_n, *if* $e = v_i v_j, i < j$ *and* $v_{i+1} v_{j-1}$ *is a stable set(=independent set), then e is called a bad edge with respect to v*

Let ***b(v)*** *the number of all bad edges in G with respect to v. There is the following theorem:*

$$X(G) = 1 + \min_{v \in S_n} b(v) \quad (1)$$

Croitorus Algorithm (2002)

Let G=(V,E) be a graph with the vertices set V={ 1, 2,. . . , n} and S_n *the set of all permutations on the set* $V ==> S_n = n!$

Theorem

For an element $\boldsymbol{e}$ $v = v_1 \ldots v_i \ldots v_j \ldots v_n$ *of* S_n, *if* $e = v_i v_j, i < j$ *and* $v_{i+1} v_{j-1}$ *is a stable set(=independent set), then e is called a bad edge with respect to v*

*Let **b(v)** the number of all bad edges in G with respect to v. There is the following theorem:*

$$X(G) = 1 + \min_{v \in S_n} b(v) \qquad (1)$$

For a given graph:
Obtaining an optimal c-coloring $\Leftrightarrow$to find an ordering with minimum number of bad edges.

Croitorus Algorithm

- genetic algorithm with 1 crossover operator and 4 mutation operators
- A chromosome represents an ordering of the vertices of the graph
- If the graph has n vertices, the chromosome will be a vector: chrom=$(v_1, v_2 \ldots v_n)$, *where* $v_i \in \{1, 2 \ldots n\}, v_i \neq v_j, i \neq j$.

Outline

Operator : Crossing-over

Let $P_1 = (v_1, v_2 \ldots v_n)$ *and* $P_2 = (u_1, u_2 \ldots u_n)$ be two parents chromosomes.

- Two cutting points $\mathrm{C}_1, C_2 \in \{1, 2 \ldots n-1\}$

are generated randomly for parent1(P_1) *and parent2*(P_2) *respectively*.

- One offspring is obtained by keeping unaltered genetic information from parent1 before C1, the vertices after C1 from the first parent are rearranged using the ordering defined by the second parent

- The second offspring is constructed similarly

Operator : Crossing-over

Example

$$\begin{cases} Parent1 = 316 \downarrow 254 \\ Parent2 = 52 \downarrow 4136 \end{cases} \Longrightarrow \begin{cases} Offsping1 = 316524 \\ Offsping2 = 523164 \end{cases}$$

Operator : Order Mutation

Let $parent = (v_1, v_2 \ldots, v_n)$ be a parent chromosome, the offspring is obtained by exchange 2 vertices located in 2 randomly generated positions.

Example

parent= 316*254* $\Longrightarrow$ *offspring* 314256

Operator : Block Mutation

.

Example

The operator translates blocks of k successive vertices (k is randomly generated). Let parent=$(\mathrm{v}_1, v_2 \ldots, v_n)$ *be a parentchromosome. If* $k = 2$ *and* $i, j \in [1, n-1]$ *are randomly generated, theblock mutation yields offspring* $= (v_1 \ldots v_{i-1} v_{i+2} \ldots v_j v_i v_{i+1} v_{j+1} \ldots)$

Operator : Evaluation by Bad Edge

Will be discussed later...

1. Merge: mixing the contents of two or more test tube into one, denote by T$\leftarrow$ $Merge(T_1, T_2 \ldots T_n)$
2. Detect: testing whether a test tube contains a DNA strand
3. Cut: cutting DNA strands at specific restriction sites, denote by Cut(T,s1|s2).
4. Length: separating DNA strands according to their base length, denoted by T$\leftarrow$ $length(T_0, l)$.
5. Extract: extracting all strands containing certain subsequences, denoted by TâĘŘ$\leftarrow$ $Extract(T_0, (s_1, s_2 \ldots s_n))$.
6. To-Single-Stranded: denature each dsDNA in tube and remove one ssDNA, denoted by T$\leftarrow$ $To - Single - Stranded(T_0)$.
7. To-Double-Stranded: making ssDNA to dsDNA, denoted by T-Double-Stranded(T$_0$).

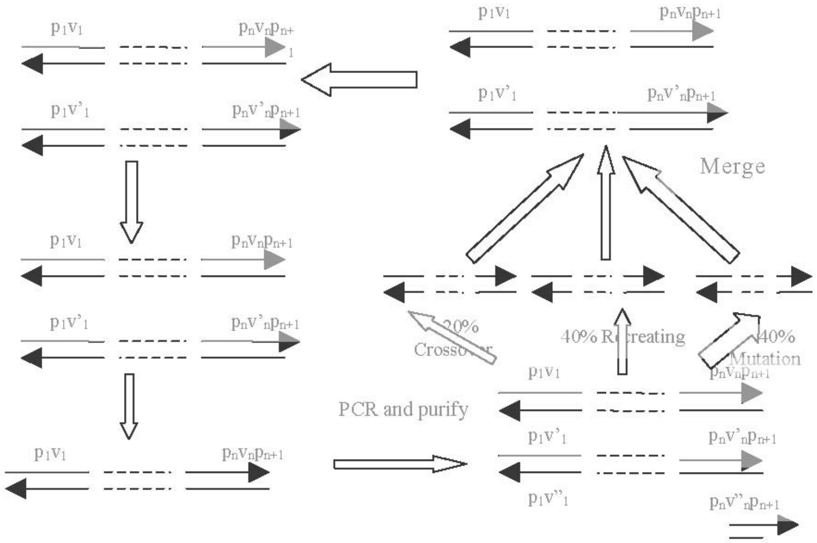
Outline

DNA Implementation of Croitorus Algorithm

Begin with a diverse initial population of candidates.

1. Evaluate the fitness of the candidates.
2. Select and purify more fit candidates.
3. Amplify fit candidates with PCR
4. Reserve some, crossover a part and mutate others.
5. Combine all the candidates from step 4, and obtain a new generation. Repeat.

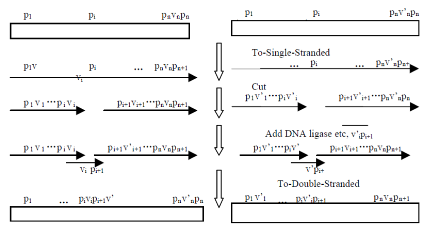
DNA Implementation of Croitorus Algorithm



Encoding Scheme and Generating of Initial Candidate Pool

- Use of dsDNA to encode the permutation of vertices and the encoding looks like $\mathrm{p}_1 v_1 p_2 v_2 \ldots p_n v_n p_{n+1}$.
- |V|=n, |E|=m. For each vertex v_i, *a series of encoding* $p_1 v_i, p_2 v_i \ldots p_n v_i$ *denotes it, where* p_j $(1 \leq j \leq n)$ *means the order of vertex* v_i *in a specific permutation* j.
- the length of a proper permutation will be (2n+1)l
- For a DNA strand representing a permutation of vertices, there are n value sections (v_1 *to* v_n) *sandwiched betweenn* $+ 1$*positionsections*$(p_1$ *to* $p_{n+1})$
- To encode an edge e=$\mathrm{v}_i v_j$, *we use a series of encoding* $v_i p_k v_j (1 \leq k \leq n)$
- To generate an initial candidate pool, we can use POA (Parallel Overlap Assembly)

Implementation of Crossover



```
1.  $T_2 \leftarrow To-Single-Stranded(T_1)$
2.  For each $v_i \in \{v_1,\cdots,v_n\}$, $T_2 \leftarrow Cut(T_2, v_i \mid p_{i+1})$
3.  For j=1 to n add($T_2, \overline{v_j p_{i+1}}$) [In Parallel]
4.  Ligase($T_2$), $T_2 \leftarrow Extract(T_2,(v_1,v_2,\cdots,v_n))$,
    $T_2 \leftarrow Length(T_2,(2n+1)l)$
5.  $T_2 \leftarrow To-Double-Stranded(T_2)$
```

Implementation of Order Mutation

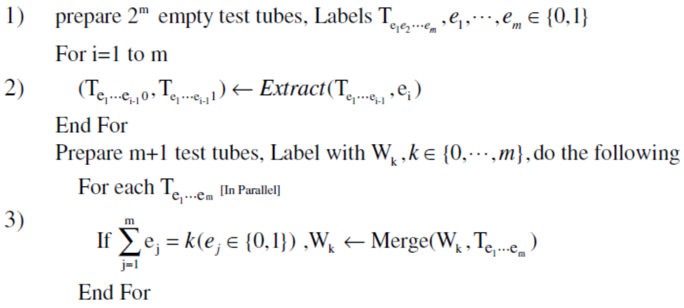
```
1.  $T_1 \leftarrow To-Single-Stranded(T_1)$
    For $v_i, v_j \in \{v_1, \cdots, v_n\}$[In Parallel]
2.          $Cut(T_1, p_i \mid v_i)$          $Cut(T_1, v_i \mid p_{i+1})$
            $Cut(T_1, p_j \mid v_j)$          $Cut(T_1, v_j \mid p_{j+1})$
    End For
3.  $Add(T_1, \overline{p_{i+1}v_i{}'p_i}, \overline{p_{j+1}v_j{}'p_j}), Ligase(T_1)$
4.  $Extract(T_1, (v_1, v_2, \cdots, v_n)), To-Double-Strand(T_1)$
```

Implementation of Block Mutation

Block Mutation: The operator is used to translate blocks of successive vertices. This operation can be implemented by many order mutations.

Implementation of of Evaluation by Bad Edges

=> each stable set consists of successive vertices in the ordering
=> Our aim is to separate chromosomes by their fitness



```
1)  prepare $2^m$ empty test tubes, Labels $T_{e_1 e_2 \cdots e_m}, e_1, \cdots, e_m \in \{0,1\}$
    For i=1 to m
2)      $(T_{e_1 \cdots e_{i-1} 0}, T_{e_1 \cdots e_{i-1} 1}) \leftarrow \mathit{Extract}(T_{e_1 \cdots e_{i-1}}, e_i)$
    End For
    Prepare m+1 test tubes, Label with $W_k, k \in \{0, \cdots, m\}$, do the following
      For each $T_{e_1 \cdots e_m}$ [In Parallel]
3)        If $\sum_{j=1}^{m} e_j = k (e_j \in \{0,1\})$, $W_k \leftarrow \mathrm{Merge}(W_k, T_{e_1 \cdots e_m})$
      End For
```

=> We get k+1 DNA strands sets of different ranks when the algorithm completes
=> We use 2^m test tubes in step 2 and m+1 test tube in step 3

Implementation of of Evaluation by Bad Edges

- Selection is used to keep fit parent chromosomes in child generation and let less fit chromosomes to die
- For a given tube W_k obtained at the end of evaluation, the chromosomes in it have the same fitness rank.
- To let less fit candidates die, we can take a threshold c ($c \leq m$) and discard chromosomes in tube W$_k where c \leq k$.
- To embody the difference of fitness among reserved chromosomes, we can perform different times of PCR for them, making the more fit chromosomes to breed more offspring.
- If after the evaluating, some test tubes W_i are not empty, that is the number of bad edges less k-1.
- then we can say that graph G can be k-colored and answer YES
- We can get concrete colors by decoding these DNA sequences.

Runtime Analysis

=>The initial generation of candidate data pool can be done in O(1) steps.
=>To complete the crossover operation, the number of extracting operations required is O(n).
=> Order mutation requires O(n) extracting
=> The number of extracting operations required for block mutation is O(n^3)
=> Evaluation of candidate based on bad edges is proportional to O(mn).
=> total running time complexity to get a new generation is O($n^3 + mn$)

END

THANKS FOR YOUR ATTENTION