Permutation Based GAs and Ordered Greed

Peter G. Anderson, Rochester Institute of Technology
Daniel Ashlock, University of Guelph

April 9, 2015
ANNIE Outline

• Permutation problems for GA’s to solve.

• The $N$ Queens problem.

• Introducing Ordered Greed.

• Creating permutations.

• Representing permutations with signatures.
ANNIE Outline

• Crossovers with signatures.

• MOX: merging crossover.

• Even preceed odds: a toy problem to compare crossovers.

• Comparing crossovers with $N$ Queens.

• Coloring random planar Hamiltonian graphs.

• Ordered Greed application areas.
N Queens—an Illustration of Permutation-Based GA

Place $N$ mutually un-attacking Queens on an $N \times N$ chess board.

(Queens attack on rows, columns, and diagonals.)

(Noise on the Internet: $NP$ complete, indeed!)
Permutation Is Placement

Let the GA creatures be permutations of $(0, 1, \cdots, N - 1)$:

Individual $= (c_0, c_1, \cdots, c_{N-1})$.

Interpret this as a Queens placement:

The Queen in row $k$ is in column $c_k$.

Fitness: number of Queens unattacked by Queens on previous rows.
How to Make a Permutation

\[
\text{for } i = 0 \text{ to } N-1 \text{ do } \\
P_i = i \\
\text{end for}
\]

\[
\text{for } i = 0 \text{ to } N-1 \text{ do } \\
k = \text{random\_int}(N - i) \\
\text{Interchange } P_i \text{ with } P_{i+k} \\
\text{end for}
\]

This uniformly generates permutations of \{0, \ldots, N - 1\}
First Example of Ordered Greed (OG)

Permutations Order Placement

The GA creatures are permutation of \((0, 1, \cdots, N - 1)\):

\[
\text{individual} = (c_0, c_1, \cdots, c_{N-1}).
\]

Interpret this as a placement ordering:

\[
\text{for } i = 0 \text{ to } N-1 \text{ do}
\]

Place the Queen in row \(c_j\)
in the left-most safe column

\text{end for}

Fitness: the number of successfully placed Queens.
Ordered Greed in Action

Permutation individual: 3 4 1 5 7 0 6 2

Row 3, col 0
Row 4, col 2
Row 1, col 1
Row 5, col 4
Row 7, col 3
Row 0, col 5
Row 6, col 7
Row 2, col 6

Fitness = 8 = 100%
Greed Ordered by 3 4 1 5 7 0 6 2

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fitnesses of 20,000 Random $64^2$ Boards

Successfully placed Queens’ histograms.

Note the efficacy of Ordered Greed vs. random placement.
Fitnesses of Some Random $256^2$ Boards

Note the efficacy of Ordered Greed vs. random placement.
Crossing Over: the Problem

Cross over the two permutations

\[
\begin{array}{ccccccc}
6 & 7 & 4 & 3 & 1 & 0 & 5 & 2 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]

by swapping the indicated substrings.

The results are not permutations:

\[
\begin{array}{ccccccc}
6 & 7 & 2 & 3 & 4 & 5 & 5 & 2 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
0 & 1 & 4 & 3 & 1 & 0 & 6 & 7 \\
\end{array}
\]
Repairing the Crossover Damage

- PMX: “partially matched crossover”
- OX: “ordered crossover”
- CX: “cycle crossover”
PMX: Partially Matched Crossover

Pick an arbitrary position in two parent permutations:

<table>
<thead>
<tr>
<th>8</th>
<th>2</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>5</th>
<th>1</th>
<th>0</th>
<th>9</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

That choice means to interchange 5 with 8 in both parents.

<table>
<thead>
<tr>
<th>5</th>
<th>2</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>8</th>
<th>1</th>
<th>0</th>
<th>9</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

Perform this operation several times, creating children with characteristics of both parents.
OX: Ordered Crossover

Pick about half of the elements of the first parent, (here, we choose 2, 4, 5, 1, and 6) and copy them to the child, preserving the positions. Choose the remaining values (0, 3, 7, 8, and 9) from the second parent, and copy them to the child, preserving the order.

\[
\begin{array}{cccccccccc}
8 & 2 & 4 & 3 & 7 & 5 & 1 & 0 & 9 & 6 \\
4 & 1 & 7 & 6 & 2 & 8 & 3 & 9 & 5 & 0 \\
7 & 2 & 4 & 8 & 3 & 5 & 1 & 9 & 0 & 6 \\
\end{array}
\]

This preserves the some orderings of elements in both parents and position of some in the first parent.
CX: Cycle Crossover

This crossover preserves the position and value of everything. Follow the reasoning: if the first position of $C_1$ is 4, then the first position of $C_2$ must be 3. Then the 3 in $C_1$ must agree with $P_1$, so the 6 in $C_2$ must agree with $P_2$. And so on.

\[
\begin{array}{cccccccccc}
4 & 1 & 7 & 6 & 2 & 8 & 3 & 9 & 5 & 0 \\
3 & 9 & 0 & 1 & 2 & 4 & 6 & 8 & 7 & 5 \\
\end{array}
\]

The consequences of the 4 in the first position of $C_1$ is:

\[
\begin{array}{cccc}
4 & 1 & 6 & 8 & 3 & 9 \\
3 & 9 & 1 & 4 & 6 & 8 \\
\end{array}
\]
Both parents have 2 in the same position, so that is fixed.

The 7–0 pair can be interchanged, with consequences for 5.

\[
\begin{array}{cccccccccc}
4 & 1 & 7 & 6 & 2 & 8 & 3 & 9 & 5 & 0 \\
3 & 9 & 0 & 1 & 2 & 4 & 6 & 8 & 7 & 5 \\
\end{array}
\]

The consequences of the 4 in the first position of \( C_1 \) is:

\[
\begin{array}{cccccccccc}
4 & 1 & 0 & 6 & 2 & 8 & 3 & 9 & 7 & 5 \\
3 & 9 & 7 & 1 & 2 & 4 & 6 & 8 & 5 & 0 \\
\end{array}
\]

This looks a lot like uniform crossover—but only certain swaps are allowed.
An Example of Many Cycles

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 5 & 3 & 10 & 6 & 4 & 11 & 7 & 8 & 12 & 13 & 9 & 14 & 15 \\
1 & 0 & 3 & 6 & 4 & 11 & 7 & 2 & 12 & 8 & 9 & 13 & 14 & 5 & 15 & 10
\end{array}
\]

This pair of parents has four cycles:

\{0, 1\}, \{2, 3, 4\}, \{5, 6, 7, 8, 9\}, \{10, 11, 12, 13, 14, 15\}

Hence they can create a \(2^4 = 16\) possible children.

For example, the pattern below can be copied to children right-side-up or up-side-down:

\[
\begin{array}{cccccccccccccccc}
\end{array}
\]
An Efficiency Challenge for PMX, OX, and CX

How can these operations be performed without resorting to multiple scans of the strings?

Try for $O(N)$ not $O(N^2)$ steps!
Representing Permutations

A permutation’s signature is a list of places.

\( \{p_0, p_1, p_2, \cdots, p_{N-1} \} \) satisfying: \( 0 \leq p_k < N - k \)

Meaning: “Try to put \( k \) in position \( p_k \)”

(There are exactly \( N! \) lists.)
Creating a Signature

\[
\text{for } i = 0 \text{ to } N - 1 \text{ do } \\
S_i = \text{random\_int}(N - i) \\
\text{end for}
\]

This procedure uniformly generates permutation signatures.
Manipulating Signatures

Preserve the rule: \( 0 \leq s_k < N - k \)

Mutate:
- Increment or decrement \( s_k \mod (N - k) \)
- Replace \( s_k \) with random value \( \mod (N - k) \)

Crossover, as usual:
- One point.
- Two point.
- Uniform.
Converting a Signature to a Permutation

\[
\text{for } i = 0 \text{ to } N - 1 \text{ do } \\
P_i = i \\
\text{end for} \\
\text{for } i = 0 \text{ to } N - 1 \text{ do } \\
\quad \text{Interchange } P_i \text{ with } P_{i+S_k} \\
\text{end for}
\]

We require that \( P_{i+S_i} \) does not precede \( P_i \). i.e.: \( 0 \leq S_i < N - i \)

This decoding only costs \( O(N) \).
Inverting: Given a Permutation, What is Its Signature?

A problem for the interested student.

This will demonstrate the 1-1 correspondence between signatures and permutations.
Merging Crossover: MOX

Randomly merge two parents into a $2N$-element list, $L$ (this operation is similar to a riffle shuffle of cards).

The first instance of each value in $L$ gives the first child, and the second instance gives the second child.

Example parents: $p_1 = \{3 9 0 1 2 4 6 8 7 5\}$, $p_2 = \{2 6 7 1 4 8 0 3 5 9\}$

Merge $p_1$ and $p_2$: $L = \{2 3 6 7 1 4 9 0 8 0 1 2 3 4 6 5 8 7 5 9\}$ (the elements from $p_1$ are shown in bold).

Extract children: $c_1 = \{2 3 6 7 1 4 9 0 8 5\}$, $c_2 = \{0 1 2 3 4 6 8 7 5 9\}$ (the $p_1$ contribution is still shown in bold).
Notes and Properties of MOX

The intermediate list, $L$, is not needed, except conceptually.

All we need is a one-element buffer, $X$, that is filled from the initial elements of the two parents, treated as queues, chosen at random.

$X$ is appended to the first child, if $X$ is not already present, otherwise it is appended to the second child.
MOX Pseudocode

\[
\text{for } i = 1 \text{ to } 2N \text{ do} \\
\quad \text{if random choice } = 1 \text{ and } p_1 \text{ not depleted then} \\
\quad \quad X \leftarrow \text{next element of } p_1 \\
\quad \text{else} \\
\quad \quad X \leftarrow \text{next element of } p_2 \\
\quad \text{end if} \\
\quad \text{if } X \text{ is not already in } c_1 \text{ then} \\
\quad \quad \text{add } X \text{ to } c_1 \\
\quad \text{else} \\
\quad \quad \text{add } X \text{ to } c_2 \\
\quad \text{end if} \\
\text{end for}
\]
Notes and Properties of MOX

OG seeks good precedence orders among permutation elements.

Let “$a \prec b$” denote that “$a$ precedes $b$” in a given permutation.

MOX seems particularly suitable for OG because:

$$a \prec b \text{ in both parents } \Rightarrow a \prec b \text{ in both children}$$

If $a \prec b$ in one parent and $b \prec a$ in the other, then both children can have $a \prec b$, both can have $b \prec a$, or the two children can be mixed.

Other permutation crossovers (signatures, CX, OX, PMX) can fail to preserve two parents’ $a \prec b$. 
Adam and Eve

Any permutation and its reverse can produce any permutation as an eventual descendant.

(Binary string one-point crossover can do this starting with any string and its complement.)
MOX and Generalizations of Permutations

OG can deal with permutations of multisets i.e., sets in which some elements appear more than once.

Application: assign several workers (e.g., faculty) to several jobs (classes).

In advance, determine how many jobs each person will do.

A person who must get $k$ jobs appears $k$ times in a list.

MOX can breed such lists.
Even Precedes Odds: A Toy Problem

A permutation-based analogy to the problem maximize the number of 1’s in a binary string.

A perfect string has all the even numbers in the left half.

Partial credit (gives a pretty good gradient towards solutions):
if \( k < N/2 \) and \( P_k \) with the right parity contributes \( N/2 - k \)
if \( k > N/2 \) and \( P_k \) with the right parity contributes \( 1 + k - N/2 \)

Use this problem with \( N = 100 \), population=100, mutation rate=0.001 to compare MOX, PMX, and signatures.

Each problem ran 100 times with different random seeds.
Even Precedes Odd Results: MOX Wins!

<table>
<thead>
<tr>
<th>xover</th>
<th>min</th>
<th>Q1</th>
<th>median</th>
<th>Q3</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOX</td>
<td>3,769</td>
<td>5,647</td>
<td>6,392</td>
<td>7,347</td>
<td>11,028</td>
</tr>
<tr>
<td>PMX</td>
<td>5,239</td>
<td>17,069</td>
<td>30,012</td>
<td>59,263</td>
<td>–</td>
</tr>
<tr>
<td>Sig. 1 Pt.</td>
<td>15,824</td>
<td>31,325</td>
<td>44,534</td>
<td>70,175</td>
<td>–</td>
</tr>
<tr>
<td>Sig. 2 Pt.</td>
<td>12,514</td>
<td>29,298</td>
<td>45,928</td>
<td>68,969</td>
<td>–</td>
</tr>
<tr>
<td>Sig. Unif.</td>
<td>8,648</td>
<td>23,764</td>
<td>38,847</td>
<td>61,547</td>
<td>–</td>
</tr>
</tbody>
</table>

Fitness evaluation counts to solve “evens preceed odds” for five crossovers.

100 tries for each crossover. We report minimum, first quartile, median, third quartile, and maximum number of fitness evaluations.

“–”: Max > 100,000, so the process was stopped.
500 Queens Results: MOX Wins! (Mostly)

<table>
<thead>
<tr>
<th>xover</th>
<th>min</th>
<th>Q1</th>
<th>median</th>
<th>Q3</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOX</td>
<td>162</td>
<td>974</td>
<td>1,444</td>
<td>2,049</td>
<td>5,005</td>
</tr>
<tr>
<td>PMX</td>
<td>184</td>
<td>988</td>
<td>1,722</td>
<td>2,573</td>
<td>7,862</td>
</tr>
<tr>
<td>Sig. 1 Pt.</td>
<td>128</td>
<td>766</td>
<td>1,144</td>
<td>1,627</td>
<td>29,706</td>
</tr>
<tr>
<td>Sig. 2 Pt.</td>
<td>30</td>
<td>763</td>
<td>1,142</td>
<td>1,726</td>
<td>10,885</td>
</tr>
<tr>
<td>Sig. Unif.</td>
<td>204</td>
<td>1,079</td>
<td>1,551</td>
<td>2,539</td>
<td>25,423</td>
</tr>
</tbody>
</table>

The number of fitness evaluations needed to solve the 500-Queens problem for five crossover techniques.
Warnsdorff’s Knight Heuristic


They refer to: H. C. Warnsdorff
*Des Rösselsprunges einfachste und allgemeinste Lösung*, Schamlkalden, 1823.

A knight can tour the chess-board by visiting hardest-to-visit locations first.

A Hamiltonian graph path attempt should try low-degree vertices first.
(Degrees decrease as neighbors are visited.)
Warnsdorff’s N Queens Heuristic

Place the hardest-to-place Queen next.

In case of tie, use our permutation.
Histograms: W/ & W/O Heuristics

256 Queens Fitnesses

[Histogram showing fitnesses for different heuristics: permutation, ordered_greed, and Warnsdorff's rule.]
Histograms: The 2 Heuristics

256 Queens Fitnesses

ordered_greed
warnsdorff
Warnsdorff Uses 2 0 7 6 4 5 1 3

Row 2 has 8 chances
Row 2, col 0
Row 0 has 6 chances
Row 0, col 7
Row 7 has 5 chances
Row 6 has 4 chances
Row 6, col 6
Row 7 has 4 chances
Row 4 has 2 chances
Row 4, col 5
Row 7 has 3 chances
Row 5 has 1 chances
Row 5, col 1
Row 7 has 1 chances
Row 7, col 4
Row 1 has 1 chances
Row 1, col 3
Row 3 has 1 chances
Row 3, col 2

New hero indiv. #1 fitness = 8 = 100.00%
Warnsdorff Uses 2 0 7 6 4 5 1 3

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Warnsdorff Uses 1 2 4 5 6 7 0 3

Row 1 has 8 chances
Row 1, col 0
Row 2 has 6 chances
Row 2, col 7
Row 4 has 4 chances
Row 4, col 6
Row 5 has 3 chances
Row 5 has 1 chances
Row 5, col 3
Row 7 has 1 chances
Row 7, col 4

Indiv. #5 fitness = 6
### Warnsdorff Uses

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rows 0 and 3 received no Queen.
GA Results

We used the following parameters:

- `uniform 1`
- `POP_SIZE 50`
- `LOOPS 1000`
- `tournament_size 2`
- `MUT_RATE 0.001`
- `N 256`

We ran the algorithms 100 times: seed = 1.1, 2.2, · · · , 100.100.
GA Results, N = 256, Using OG

We only allowed 1,000 loops!
The OG GA succeeded in 57% of the trials

<table>
<thead>
<tr>
<th>count</th>
<th>tries</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>277</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>331</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>593</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>637</td>
<td>256</td>
</tr>
<tr>
<td>14</td>
<td>948</td>
<td>256</td>
</tr>
<tr>
<td>43</td>
<td>2050</td>
<td>255</td>
</tr>
</tbody>
</table>

Allowing up to 2,050 trials, random search succeeded in 16% of 100 trials.
GA Results, N = 256

Ordered Greed with Warnsdorff
(all solutions in initial population)

<table>
<thead>
<tr>
<th>count</th>
<th>tries</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>5</td>
<td>256</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>256</td>
</tr>
<tr>
<td>14</td>
<td>19</td>
<td>256</td>
</tr>
<tr>
<td>27</td>
<td>23</td>
<td>256</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>26</td>
<td>256</td>
</tr>
</tbody>
</table>
Results, N = 256 (without OG)

Permutations (total failure)

<table>
<thead>
<tr>
<th>count</th>
<th>tries</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>2050</td>
<td>178</td>
</tr>
<tr>
<td>10</td>
<td>2050</td>
<td>180</td>
</tr>
<tr>
<td>36</td>
<td>2050</td>
<td>181</td>
</tr>
<tr>
<td>9</td>
<td>2050</td>
<td>182</td>
</tr>
<tr>
<td>16</td>
<td>2050</td>
<td>185</td>
</tr>
<tr>
<td>15</td>
<td>2050</td>
<td>187</td>
</tr>
</tbody>
</table>
## GA Results, $N = 8$

Ordered Greed
(all solutions in initial population)

<table>
<thead>
<tr>
<th>count</th>
<th>tries</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>
GA Results, N = 8

Ordered Greed with Warnsdorff
(all solutions in initial population)

<table>
<thead>
<tr>
<th>count</th>
<th>tries</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
GA Results, $N = 8$ (without OG)

Permutations (the GA worked 65%)

<table>
<thead>
<tr>
<th>count</th>
<th>tries</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>58</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>144</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>190</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>302</td>
<td>8</td>
</tr>
<tr>
<td>35</td>
<td>2050</td>
<td>7</td>
</tr>
</tbody>
</table>

This is marginally better than random search.
Graph Coloring Problems

• $G = (V_G, E_G)$ is a set of vertices, $V_G$, and a set of edges, $E_G$.

• An edge, $(v, w)$, connects the vertices $v$ and $w$. Vertices $v$ and $w$, are adjacent in $G$.

• A coloring of $G$ is a function $c : V_G \rightarrow K$ ($K$ is the set of colors) such that $(v, w) \in E_G \Rightarrow c(v) \neq c(w)$.

  $G$ is $k$-colorable if it has a coloring: $|K| = k$.

• The smallest $k$ is the chromatic number of $G$. 
Graph Coloring

Famous graph coloring problem: map coloring.

Vertices are countries. Edges connect countries touch.

Touching countries must be colored differently.

Graph coloring also models:
exam scheduling, process scheduling, memory allocation, ...

Graph coloring is NP-complete: there is no known efficient algorithm. Fast approximations are desirable.

We color random 3-colorable graphs with edge density $p = 0.1$. 
Kubale’s Sequential Coloring Algorithm

Given: a permutation of $V_G$: $v_0, v_1, v_2, \cdots, v_{N-1}$

for $k = 0$ to $N-1$ do
    Color $v_k$: use the smallest color number not assigned to a vertex adjacent to $v_k$.
end for

Different permutations of $V_G$ can give different color assignment. The best coloring is clearly achievable!
Heuristics?

Warnsdorff may improve the performance:

First color the hardest-to-color vertices.

A vertex is hard to color if its neighbors use many colors.

Break ties with the vertex permutation.
1,000,000 Tries to Color $G_{100}$ by the Sequential Method

Fitness is the number of vertices colored with three colors.
100,000 Tries to Color $G_{3000}$ by the Sequential Method
Building Random Graphs $G_{100}$ and $G_{3000}$

Build a graph with $V_G = \{0, 1, 2, 3, \cdots, N - 1\}$.

\[
\begin{align*}
\text{for } v = 0 & \text{ to } N - 2 \text{ do} \\
\text{for } w = v + 1 & \text{ to } N - 1 \text{ do} \\
\text{if } v \not\equiv w \pmod{3} & \text{ then} \\
(v, w) & \in E_G \text{ with probability } p \\
\text{end if} \\
\text{end for} \\
\text{end for}
\end{align*}
\]

Edge density is $p = 0.1$ for all 3-coloring experiments.
Why Density $p = 0.1$?

We did 10,000 coloring attempts with graphs of 100 vertices & edge densities of 0.01–0.40.

Small density graphs have many small connected components, thus easy to color.

Large density graphs have many triangles. An easy coloring strategy: locate and color the vertices in a triangle, then color the vertices in triangles with edges in common with a colored triangle.
Why $p = 0.1$? The Experiments: $p = 0.01 \cdots 0.40$

Minimum, mode, & maximum fitness.  
The sequential method attempts to 3-color 100-vertex graph.  
10,000 fitness evaluations for each density
The Experiments

Use 3-colorable $G_{100}$ (100 vertices, edge density $p = 0.1$)

Vary the population size.

Vary the tournament size.

Color the 3-colorable $G_{3000}$. 
Ordered Greed: Fitness Evals for Pop Sizes 10–999
Tournament Sizes 2–99 to Color $G_{100}$

Average = 2,674. Population size = 200. Doesn't tournament size matter?
Coloring Hamiltonian Planar Graphs

The vertices are on a sphere’s equator.

The edges are randomly drawn to triangulate each hemisphere.

Type 1 graphs: only use the edges in the northern hemisphere. These are 3-colorable, uniquely.

Type 2 graphs: use the edges in both hemispheres. These are 4-colorable, right?

Type 1 graphs proved harder to color!
Experiments & Results

190 random graphs of each type, $11 \leq N \leq 200$.
21 attempts to color each.

MOX, pop=20, mut-rate=0.01, max 100,000 fitness evaluations.

<table>
<thead>
<tr>
<th>Type 1 graphs</th>
<th>Type 2 graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$11 \leq N \leq 30$</td>
<td>all success</td>
</tr>
<tr>
<td>$31 \leq N \leq 64$</td>
<td>most success</td>
</tr>
<tr>
<td>$65 \leq N \leq 110$</td>
<td>&lt; half success</td>
</tr>
<tr>
<td>$111 \leq N \leq 200$</td>
<td>all failed</td>
</tr>
<tr>
<td>$11 \leq N \leq 50$</td>
<td>all success</td>
</tr>
<tr>
<td>$59 \leq N \leq 171$</td>
<td>most success</td>
</tr>
<tr>
<td>$172 \leq N \leq 200$</td>
<td>&lt; half success</td>
</tr>
</tbody>
</table>

Every type 2 graph was colored.
Using Warnsdorff to Help Color Graphs

OG + Warnsdorff: color the hardest-to-color first.

Break ties using the permutation.

Result for type 1 graphs: The first attempt to color each one worked.

Type 2 graphs: work in progress.
An OG Applications Sampler

O. G. is natural & appropriate for a variety of problems.

A necessary condition: The optimal solution can be found by a greedy algorithm and the right permutation.

Many permutations will produce the same, or equivalent, answer.
An OG Applications Sampler

• N Queens
• Graph vertex & edge coloring
• Scheduling in general
• Exam scheduling
• Sports tournaments scheduling
• Multiprocessors scheduling
• Faculty teaching assignments

• Job assignments
• Matching
• Traveling salesman
• Bin packing
• 2D board cutting
• Pentominos
• SAT, 3SAT