R·I·T

Department of Computer Science

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.

R·I·T
Department of Computer Science

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.

R·I·T
Department of Computer Science

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: $\mathcal{NP}$ 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

for $i = 0$ to $N-1$ **do**
  $P_i = i$
end for
for $i = 0$ to $N-1$ **do**
  $k =$ random_int$(N - i)$
  Interchange P_i with P_{i+k}
end for

This uniformly generates permutations of $\{0, \cdots, N-1\}$

Department of Computer Science

First Example of Ordered Greed (OG)

Permutations Order Placement

The GA creatures are permutation of $(0, 1, \cdots, N-1)$:
individual $= (c_0, c_1, \cdots, c_{N-1})$.
Interpret this as a placement ordering:

for $i = 0$ to $N-1$ **do**
  Place the Queen in row c_j
  in the left-most safe column
end for

Fitness: the number of successfully placed Queens.

R·I·T
Department of Computer Science

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

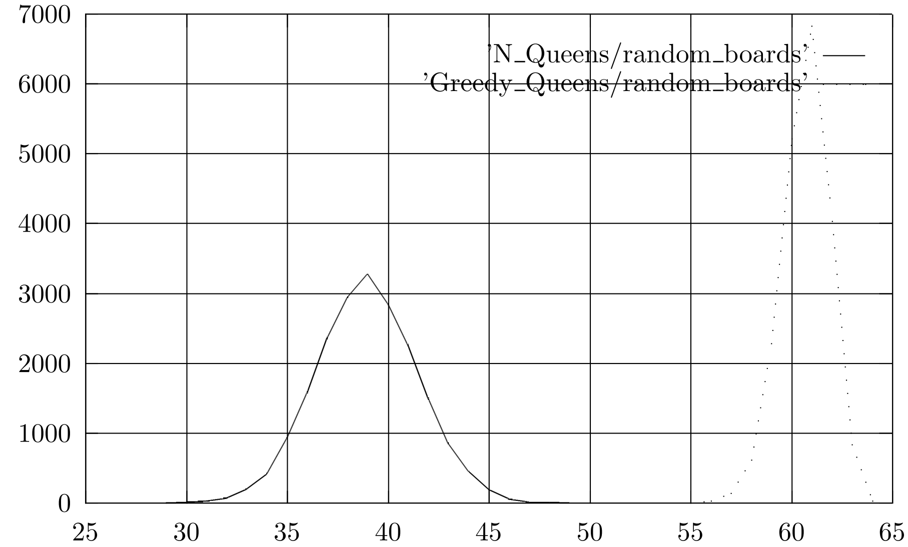
Fitness = 8 = 100%

Greed Ordered by 3 4 1 5 7 0 6 2

	0	1	2	3	4	5	6	7
0						6		
1		3						
2							8	
3	1							
4			2					
5					4			
6								7
7				5				

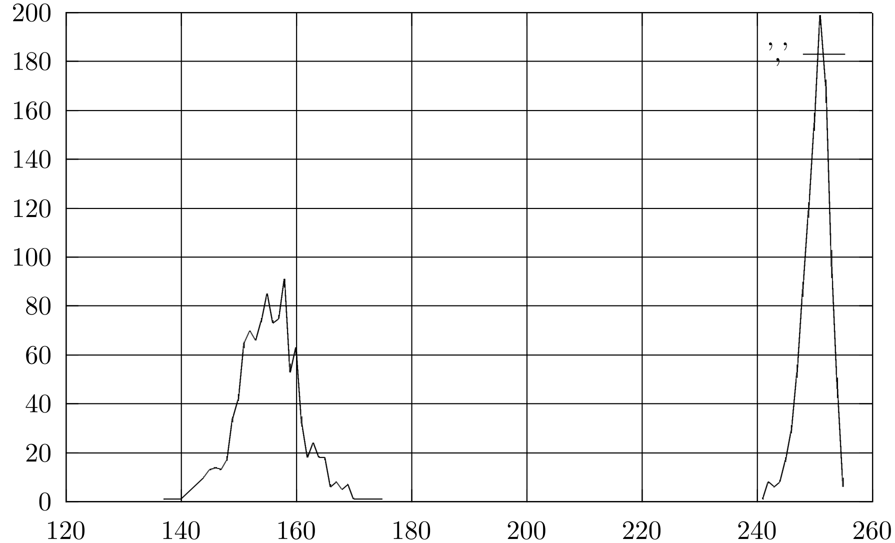
Fitnesses of 20,000 Random 64^2 Boards

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

6	7	4	3	1	0	5	2

0	1	2	3	4	5	6	7

by swapping the indicated substrings.

The results are not permutations:

6	7	2	3	4	5	5	2

0	1	4	3	1	0	6	7

R·I·T
Department of Computer Science

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:

8	2	4	3	7	5	1	0	9	6
4	1	7	6	2	8	3	9	5	0

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

5	2	4	3	7	8	1	0	9	6
4	1	7	6	2	5	3	9	8	0

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.

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

This preserves the some orderings of elements in both parents and position of some in the first parent.

R·I·T
Department of Computer Science

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.

4	1	7	6	2	8	3	9	5	0
3	9	0	1	2	4	6	8	7	5

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

4	1		6		8	3	9		
3	9		1		4	6	8		

R·I·T
Department of Computer Science

CX: Cycle Crossover, Cont.

Both parents have 2 in the same position, so that is fixed.

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

4	1	7	6	2	8	3	9	5	0
3	9	0	1	2	4	6	8	7	5

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

4	1	0	6	2	8	3	9	7	5
3	9	7	1	2	4	6	8	5	0

This looks a lot like uniform crossover—but only certain swaps are allowed.

An Example of Many Cycles

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

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:

-	-	-	5	-	-	6	-	-	7	8	-	-	9	-	-
-	-	-	6	-	-	7	-	-	8	9	-	-	5	-	-

R·I·T Department of Computer Science

An Efficiency Challenge for PMX, OX, and CX

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

Try for $\mathcal{O}(N)$ not $\mathcal{O}(N^2)$ 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.)

R·I·T
Department of Computer Science

Creating a Signature

for $i = 0$ to $N{-}1$ **do**
  $S_i = \text{random\_int}(N - i)$
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.

R·I·T
Department of Computer Science

Converting a Signature to a Permutation

for $i = 0$ to $N{-}1$ **do**
  $P_i = i$
end for
for $i = 0$ to $N{-}1$ **do**
  Interchange P_i with P_{i+S_k}
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 $\mathcal{O}(N)$.

R·I·T
Department of Computer Science

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

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

Extract children: $c_1 = \{2\ \mathbf{3}\ 6\ 7\ 1\ 4\ \mathbf{9\ 0}\ 8\ 5\}$, $c_2 = \{0\ \mathbf{1\ 2}\ 3\ \mathbf{4\ 6\ 8\ 7\ 5}\ 9\}$
(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.

R·I·T
Department of Computer Science

MOX Pseudocode

```
for i = 1 to 2N do
   if random choice = 1 and p_1 not depleted then
      X ← next element of p_1
   else
      X ← next element of p_2
   end if
   if X is not already in c_1 then
      add X to c_1
   else
      add X to c_2
   end if
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 <u>multisets</u>
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.

R·I·T
Department of Computer Science

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.

R·I·T
Department of Computer Science

Even Precedes Odd Results: MOX Wins!

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

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.

R·I·T
Department of Computer Science

500 Queens Results: MOX Wins! (Mostly)

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

The number of fitness evaluations needed to solve the 500-Queens problem for five crossover techniques.

Warnsdorff's Knight Heuristic

See: W. W. Rouse Ball & H. S. M. Coxeter,
Mathematical Recreations & Essays,
University of Toronto Press, 1974.

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.)

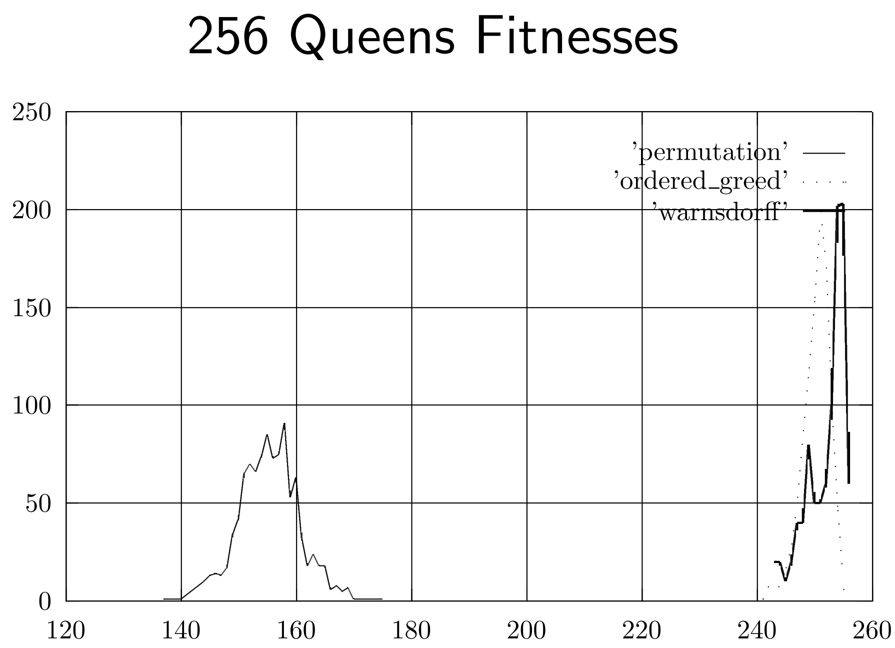
R · I · T
Department of Computer Science

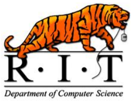
Warnsdorff's N Queens Heuristic

Place the hardest-to-place Queen next.

In case of tie, use our permutation.

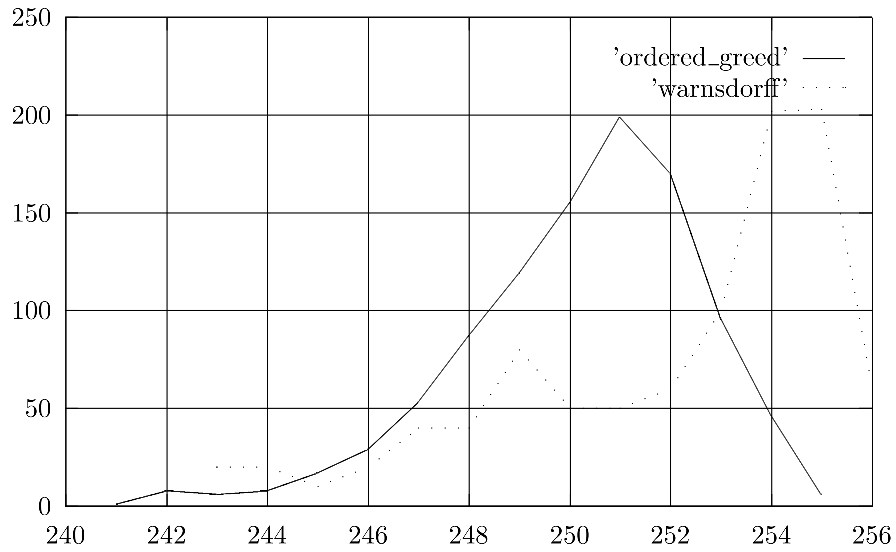
Histograms: W/ & W/O Heuristics





Histograms: The 2 Heuristics

256 Queens Fitnesses



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

	0	1	2	3	4	5	6	7
0								2
1				7				
2	1							
3			8					
4						4		
5		5						
6							3	
7					6			

R·I·T
Department of Computer Science

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

Indiv. #5 fitness = 6

Warnsdorff Uses 1 2 4 5 6 7 0 3

	0	1	2	3	4	5	6	7
(0)								
1	1							
2								2
(3)								
4							3	
5				5				
6		4						
7					6			

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 $= 1.1, 2.2, \cdots, 100.100$.

R·I·T
Department of Computer Science

GA Results, N = 256, Using OG

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

count	tries	fitness
14	277	256
10	331	256
10	593	256
9	637	256
14	948	256
43	2050	255

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)

count	tries	fitness
16	5	256
14	10	256
10	14	256
14	19	256
27	23	256
10	25	256
9	26	256

Results, N = 256 (without OG)

Permutations (total failure)

count	tries	fitness
14	2050	178
10	2050	180
36	2050	181
9	2050	182
16	2050	185
15	2050	187

GA Results, N = 8

Ordered Greed
(all solutions in initial population)

count	tries	fitness
14	1	8
22	2	8
16	3	8
14	7	8
15	13	8
9	14	8
10	16	8

R·I·T
Department of Computer Science

GA Results, N = 8

Ordered Greed with Warnsdorff
(all solutions in initial population)

count	tries	fitness
50	1	8
22	3	8
14	6	8
14	7	8

R·I·T
Department of Computer Science

GA Results, N = 8 (without OG)

Permutations (the GA worked 65%)

count	tries	fitness
10	54	8
15	58	8
16	144	8
10	190	8
14	302	8
35	2050	7

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 $c : V_G \to K$
  (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.

R·I·T
Department of Computer Science

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:

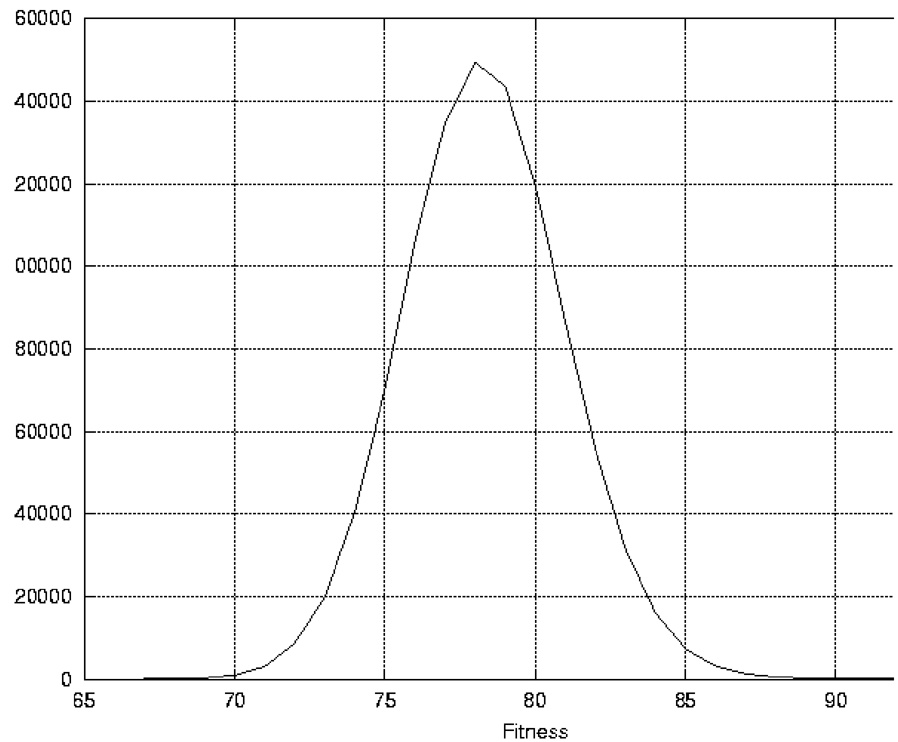
First color the hardest-to-color vertices.

A vertex is <u>hard</u> to color if its neighbors use <u>many</u> colors.

Break ties with the vertex permutation.

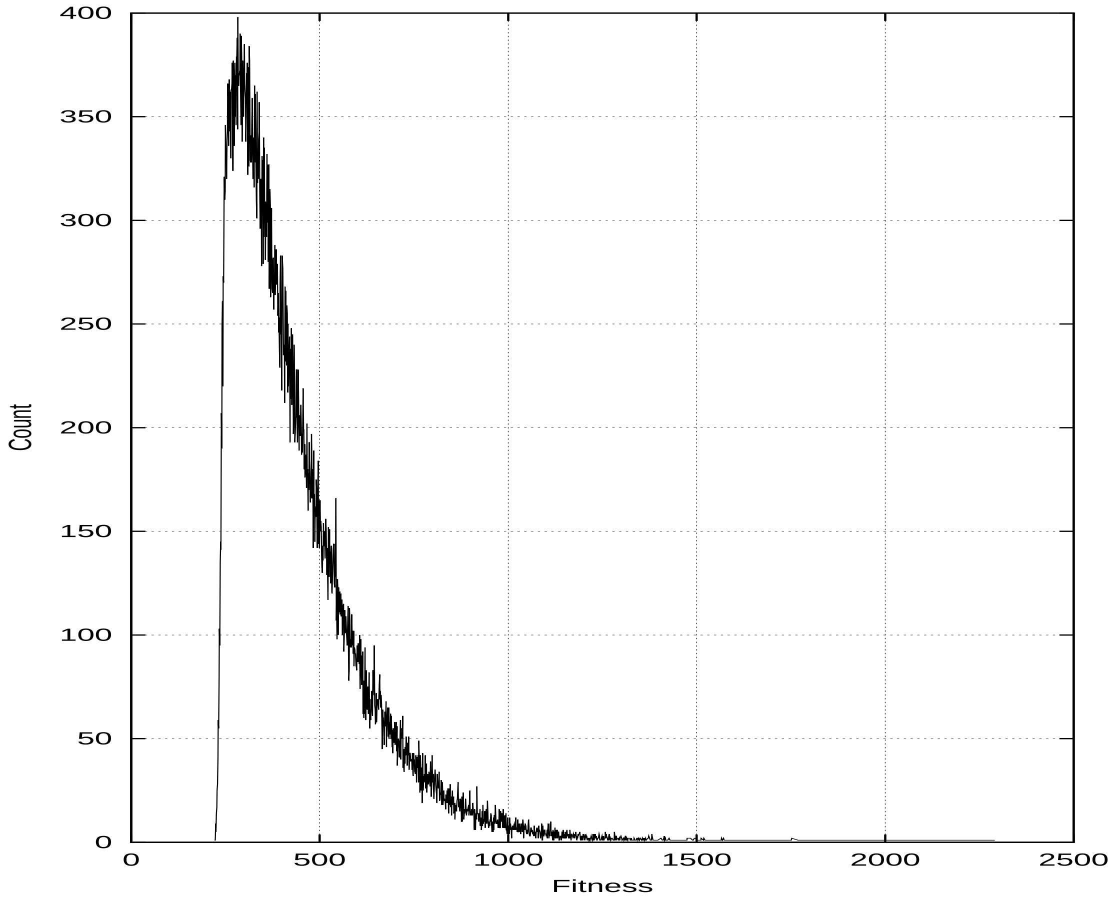
R·I·T
Department of Computer Science

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 $V_G = \{0, 1, 2, 3, \cdots, N-1\}$.

for $v = 0$ to $N-2$ **do**
  for $w = v+1$ to $N-1$ **do**
    if $v \not\equiv w \pmod 3$ **then**
      $(v, w) \in E_G$ with probability p
    end if
  end for
end for

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

R·I·T
Department of Computer Science

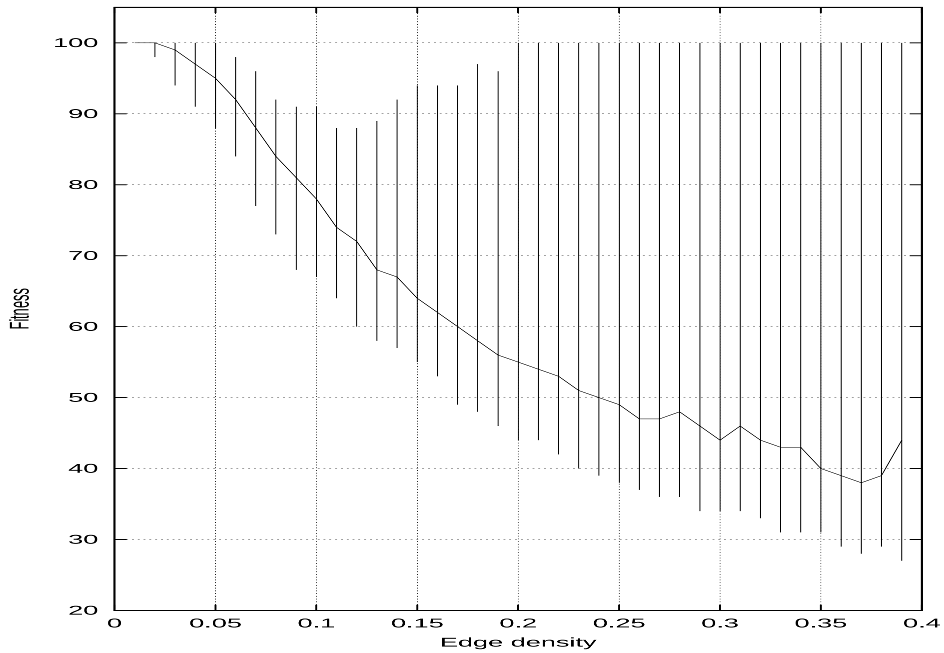
Why Density $p = 0.1$?

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 $p = 0.1$)

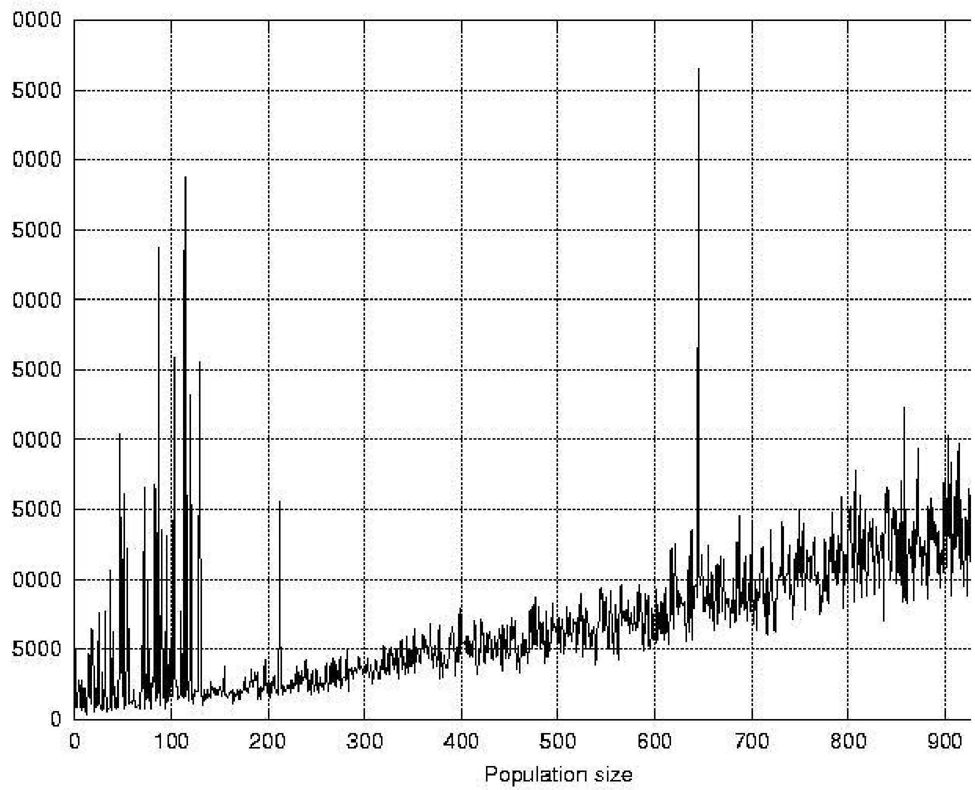
Vary the population size.

Vary the tournament size.

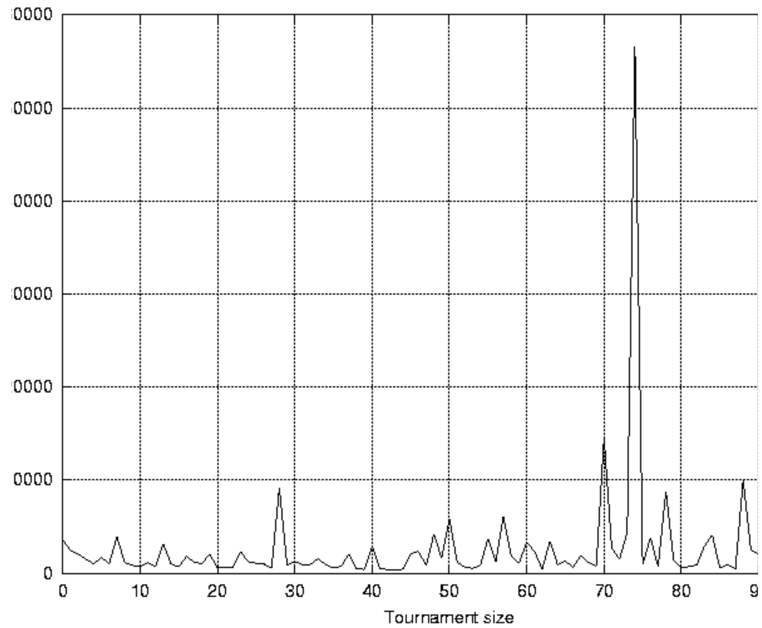
Color the 3-colorable G_{3000}.

R·I·T
Department of Computer Science

Ordered Greed: Fitness Evals for Pop Sizes 10–999

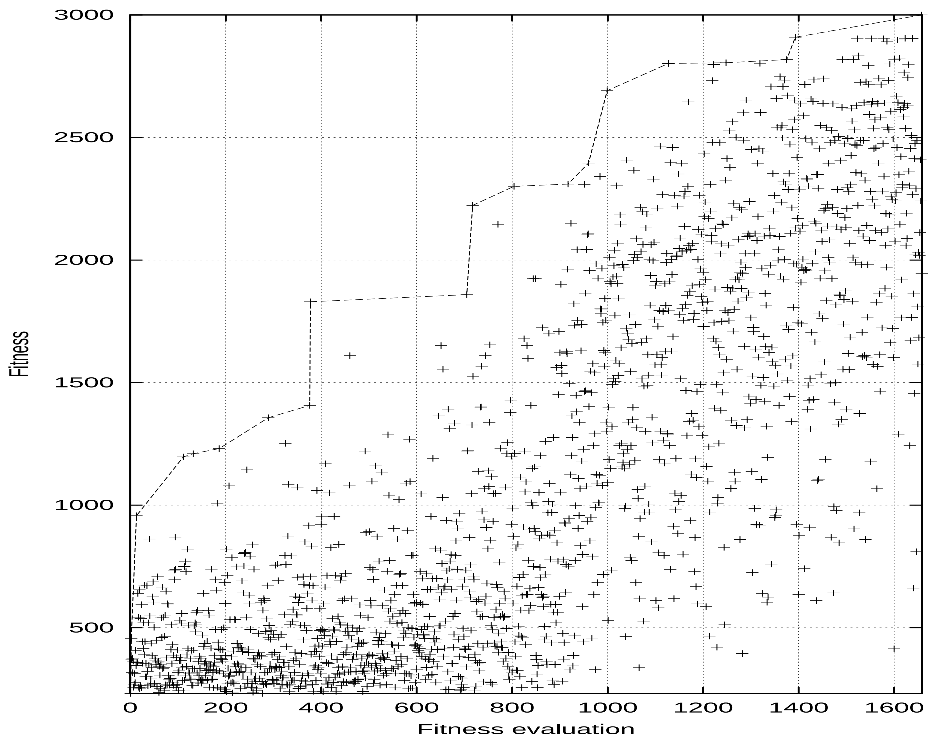


Tournament Sizes 2–99 to Color G_{100}



Average = 2,674. Population size = 200.
Doesn't tournament size matter?

Proof of the Pudding: Color G_{3000}



1,658 fitness evaluations. Population size = 200. Tournament size = 5.

R·I·T
Department of Computer Science

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.

Type 1 graphs

$11 \leq N \leq 30$	all success
$31 \leq N \leq 64$	most success
$65 \leq N \leq 110$	$<$ half success
$111 \leq N \leq 200$	all failed

Type 2 graphs

$11 \leq N \leq 50$	all success
$59 \leq N \leq 171$	most success
$172 \leq N \leq 200$	$<$ half success

Every type 2 graph was colored.

R·I·T
Department of Computer Science

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.

R·I·T
Department of Computer Science

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