Ordered Greed for N SuperQueens

Daniel Kunkle
Computer Science Department
College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623
drk4633@rit.edu
http://www.rit.edu/~drk4633

May 1, 2002

Abstract

This is an extension of the Ordered Greed technique introduced by Anderson and Gustafson in [1]. The Ordered Greed algorithm (OG) is a hybrid greedy algorithm using permutations. OG is applicable to many classical optimization problems, including job assignment, graph coloring, bin packing, and satisfiability. Here, OG is applied to a variant of the N Queens problem, the N SuperQueens problem. The performance of the OG method and that of the Plain Permutation GA in solving the N SuperQueens problem are compared. The results here support those found in [1], that the OG method is far superior to the Plain Permutation GA.

1 N SuperQueens

N SuperQueens is an extension of the classical N Queens problem where the task is to place N chess queens on an $N \times N$ board such that no two queens are attacking each other. The normal queens attack along rows, columns, and diagonals. A SuperQueen has all of the attacking abilities of a normal queen along with those of the chess knight. The task remains to place N

SuperQueens on an $N \times N$ board. The N Queens problem is solvable for any $N \geq 4$, whereas the N SuperQueens problem is solvable for any $N \geq 10$. See figure 1 for N SuperQueens solutions with $N = 10$ and $N = 11$.

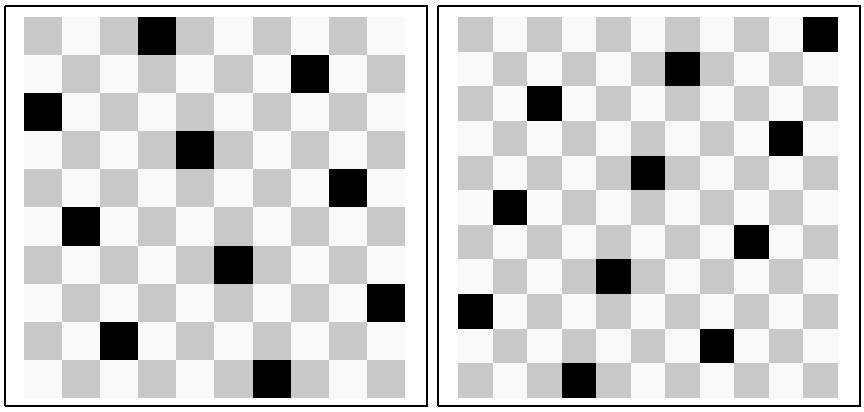

Figure 1: Solutions for the N SuperQueens problem. Black squares represent SuperQueen placements. **left:** $N = 10$. **right:** $N = 11$.

Though many heuristics have been developed to efficiently solve the N Queens problem that could possibly be modified to satisfy the N SuperQueens problem they are not considered here. The goal is to examine and compare two methods of search, Plain Permutation GAs and Ordered Greed. The N SuperQueens problem is used as a comparison problem because of its relative simplicity and it similarity to many hard problems, such as the NP-complete Maximum Independent Set (MIS) problem.

2 Permutations - Order Based GAs

Traditionally, individual GA solutions are represented as a bit string, where each bit can take on a value of 0 or 1. In many problems, however, it is necessary to ensure that each value occurs only once. Most notably, the Traveling Salesman Problem (TSP) specifies that a salesman should visit each city once and only once. In this case a solution is represented by a permutation, $\langle a_0, a_1, a_2, \cdots, a_{N-1} \rangle$ of the numbers $\langle 0, 1, 2, \cdots, N-1 \rangle$.

The use of permutations in GAs presents a problem: the normal genetic operators of crossover and mutation will most likely destroy the permutation properties of the individual, often duplicating some numbers in the set

and removing others. A number of permutation preserving crossover operators have been developed, such as *partially matched crossover* (PMX), *order crossover* (OX), and *cycle crossover* (CX) [2].

In keeping with the methods presented by Anderson and Gustafson in [1], *signatures* are used here as a means of producing order based GAs. A signature is a string of numbers that represent a permutation, but are not themselves a permutation. A signature $S = \langle s_0, s_1, s_2, \ldots, s_{N-1} \rangle$ satisfies the property $0 \leq s_i \leq i$ for all i. S specifies that 0 is in position s_0, 1 is in position s_1 of the remaining $N-1$ positions, 2 is in position s_2 of the $N-2$ remaining positions, and so forth.

The advantages of signatures presented in [1] are:

- Signatures are easy to generate
- Signatures can be easily mutated (regenerate one of the entries) and bred using two-point crossover (signature properties guarantee children are valid signatures).
- There exist straightforward algorithms to convert signatures into their corresponding permutations

3 Plain Permutation GA

The Plain Permutation GA uses the permutation, as specified by the signature, as a direct solution to the problem at hand. In the case of the N SuperQueens problem the permutation $\langle a_0, a_1, a_2, \cdots, a_{N-1} \rangle$ specifies that the SuperQueens are to be placed in row 0, column a_0; row 1, column a_1; $\cdots$; row $N-1$, column a_{N-1}. Any SuperQueens that would be attacked by an earlier placed SuperQueen are not placed on the board at all.

The fitness of a permutation using this method is the number of successfully placed SuperQueens.

4 Ordered Greed

The Ordered Greed method doesn't use the permutation directly, but instead uses the permutation to drive a greedy algorithm. A greedy algorithm is one that makes an optimal choice at each local step but has no global

memory with which to optimize later decisions. One greedy algorithm for N SuperQueens (and N Queens) is to go row by row, placing a Queen in the first non-attacked column. This greedy approach may work on its own for certain board sizes but will fail for most. A permutation can be used to specify the ordering of the rows that the Queens are greedily placed in.

In this way the permutations make up the search space, just as in the Plain Permutation method, but the greedy algorithm makes the best of the situation when placing the Queens. The improvements in performance when using OG over Plain Permutations is dramatic and is described in the next section.

5 Results

These are the results of several experiments to determine the relative effectiveness of Plain Permutations versus Ordered Greed in solving the N SuperQueens problem. These results correspond closely to those found in [1].

5.1 Random Permutation Comparison

As a simple comparison between Plain Permutations and Ordered Greed 10,000 random permutations were created and their fitness evaluated. The fitness of each permutation is the number of successfully placed SuperQueens on a 1024×1024 board. A histogram of the fitnesses for the two methods clearly shows that OG produces superior SuperQueen placements (see figure 2). The average fitness of the random Plain Permutations was 621.7, whereas with OG it was 1013.7. In fact, the problem of placing 1024 SuperQueens was almost solved in the first 10,000 random permutations of Ordered Greed with a maximum fitness of 1023 (see table 3).

5.2 Population Fitness Over Time

When examining a scatter plot of the calculated fitnesses of each individual from the GA it is apparent not only that the OG method produces better fitnesses for random permutations but that the GA using the OG method

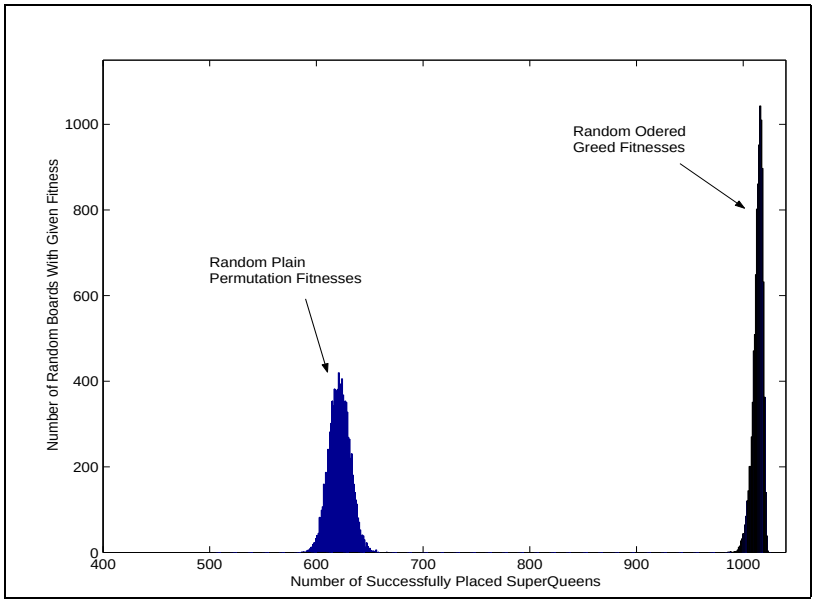

Figure 2: Histogram of fitness values for 10,000 random individuals for the Plain Permutation and Ordered Greed methods as applied to the N SuperQueens problem with $N = 1024$. The OG method is clearly superior.

Method	Mean	Minimum	Maximum
Plain Permutation	621.7($\pm$10.0)	586	666
Ordered Greed	1013.7($\pm$4.6)	986	1023

Figure 3: Fitness value statistics for Plain Permutation and Ordered Greed over 10,000 random individuals as applied to the N SuperQueens problem with $N = 1024$.

has a much better chance of improving over time. Figure 4 shows a comparison of one GA trial for both Plain Permutations and Ordered Greed while attempting to solve N SuperQueens for $N = 4096$. The Plain Permutations method had no substantial increase in fitness over 3000 evaluations, whereas the OG method had a clear improvement in populations fitness over time and found a solution after 594 fitness evaluations.

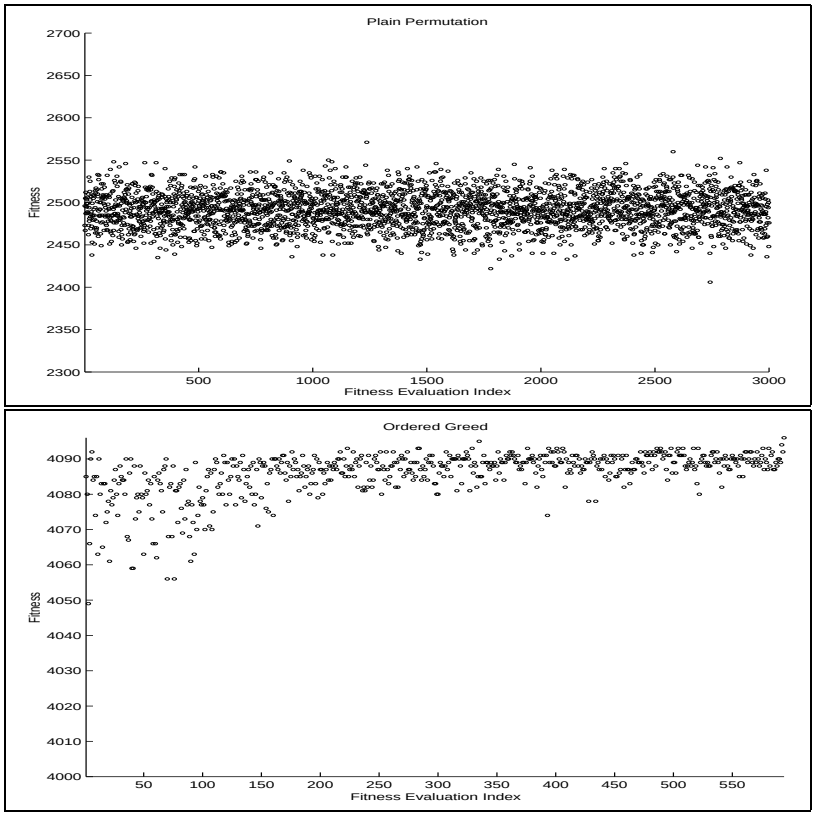

Figure 4: **top:** scatter plot of the first 3000 fitness evaluations of the Plain Permutation method in solving N SuperQueens with N=4096. Almost all fitness values are below 2550 and there is no significant rise in fitness over time. **bottom:** scatter plot of the first 594 fitness evaluations of the Order Greed method of solving N SuperQueens with N=4096. All fitnesses are 4049 or better and a solution was found after 594 fitness evaluations.

5.3 100 Random GA Trials

To further expand on the findings of the one GA trial in the last section, 100 GA trials were conducted for both Plain Permutations and Ordered Greed, using a different random seed each time. Both were trying to solve the N SuperQueens problem with $N = 128$ and were not allowed more than 2100 fitness evaluations (100 in the initial population and 1000 tournaments producing 2 children each). The Plain Permutation GA was never able to solve the problem in the 2100 fitness evaluation and achieved a maximum

fitness of only 93 out of 128 (or 72.6%). The OG GA was able to solve the problem all 100 times, in an average of 517.6 fitness evaluations. Figure 5 details the results.

Method	Average Number of Fitness Evaluations	Average Best Fitness	Maximum Best Fitness
Plain Permutation	2100(±0)	90.11(±0.9)	93
Ordered Greed	517.6(±327.5)	128(±0)	128

Figure 5: Fitness value statistics for Plain Permutation and Ordered Greed over 100 runs of the GA with different random seeds. Parameters were: population size of 100, tournament size 3, uniform crossover, mutation rate of 0.03, and the number of fitness evaluations was held to 2100 or fewer. Applied to the N SuperQueens problem with $N = 128$.

6 Conclusions

The results of applying both Plain Permutations and Ordered Greed Genetic Algorithms to the N SuperQueens problem support the results found in [1]. The speed and likelihood of finding a solution are greatly increased by the use of OG. Any problem that has a greedy approach that can be controlled by permutations should receive the same benefits from OG that N SuperQueens does.

7 Acknowledgments

Thanks to Samuel Inverso for Perl and Unix shell scripts to extract valuable data from raw output.

References

[1] Peter G. Anderson and William T. Gustafson. Ordered greed.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learing.* Addison Weslay, 1989.