A Novel Crossover Operator for Harvest Volume Optimization GAs

Drew Becker
FORST 8450
Dr. Pete Bettinger
UGA Institute for Artificial Intelligence

1. Introduction:

Developing a harvest scheduling plan is an important aspect of the forestry industry. Various problem-solving techniques have been used, with linear programming and mixed integer programming being the most popular programming methods today (Bettinger & Chung, 2004). Praised for their simplicity and ease of understanding, these methods have been established since the middle of the 20th century. However, as more environmental regulations and data analysis tools become commonplace, there remains the possibility of more complex problems in forestry, and thus more complex algorithms for solving these problems.

Evolutionary computation is a form of p-based metaheuristic that is modeled after natural processes, specifically, the evolution of species. The basics of an evolutionary computation algorithm are as follows. The goal of evolutionary computing is to evolve a population of individuals until an individual produces the global optimum for a given problem. In an evolutionary algorithm every individual corresponds to a full solution of the problem. Each individual is assigned a "fitness" value corresponding to the objective function. These Fitness values may also include penalties from violating constraints in the problem definition. Oftentimes, developing the fitness function for the evolutionary algorithm is the most intensive and complex part of the algorithm. Each individual undergoes a series of operators called *crossover* and *mutation*, that result in a new population of individuals each with a new fitness function. A selection method is also chosen to ensure competition in the population, that is, individuals with the best fitness values have the best chance for survival and reproduction. Generally, a heuristic algorithm strikes a balance between exploration of the search space and exploitation of good solutions. *Mutation* ensures that the search space is Thoroughly explored in evolutionary computing, and *crossover* and selection ensure exploitation of good solutions by combining genes from fit individuals.

Although unpredictable and difficult to explain, evolutionary computing often produces good results on problems that prove to be too difficult for other methods to solve. Evolutionary Computing is also a prime candidate for solving the types of problems called map coloring problems, of which harvest scheduling problems fall (Fotakis et. al 2012). In a map coloring problem, segments of the map must be colored with different treatments, subject to adjacency constraints and other constraints. A challenge this poses for evolutionary computing is that in sufficiently complex problems, the feasible solution space may be small, a problem I address in this paper.

Another challenge for evolutionary computing in harvest scheduling problems is that oftentimes the simple representation of solutions does not adequately capture the true adjacencies

of the management units in a tract of land. Since these are often labeled somewhat out of order, this may pose an additional problem for crossover operators in genetic algorithms. To combat this potential difficulty, in this paper, I propose two new types of crossover for handling adjacency constraints in harvest scheduling problems and compare them to standard methods.

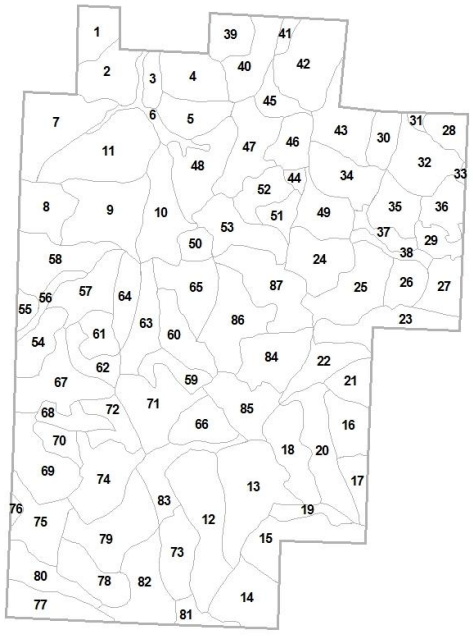
2. Methodology

2.1: The Harvest Scheduling Problem:

The problem to be optimized is a relatively simple one that can be solved with simpler algorithms. There is only one objective, to maximize the harvest volume (HV) in thousand board-feet (MBF) of the Lincoln Tract over a 30 year time span. The overall timeline is divided into 6 management periods, each five years long. The Lincoln tract itself is divided into 87 management units (stands)[fig. 1]. Each stand can receive one of 7 management treatments, clear cut in time period n=[1, 6], or no clear cut at all. In addition to this objective, there are a variety of constraints placed on the harvest scheduling problem. These constraints are: (1) *Adjacency:* no two stands that are adjacent may be cut in the same time period, (2) *Even Flow of Harvest:* the total MBF harvested across all time periods must be relatively consistent, (3) *Maximum Harvest Volume:* no more than 13,950 MBF may be harvested in any time period, (4) *Minimum Harvest Age:* no stands younger than 35 years of age may be harvested, and (5) *Maximum Harvest Size:* no more than 120 acres of continuous land may be harvested at a time. These constraints are in compliance with the *Oregon Department of Forestry Forest Practice Administrative Rules and Forest Practices Act*, however, they are not the only constraints to be used in a real-world harvest scheduling problem. Other constraints, such as those regarding roads, streams, and wildlife are not considered for the purposes of this problem.

2.2 Optimization Procedure

An optimization procedure was applied to find a maximum harvest volume for the Lincoln Tract subject to the constraints listed previously. The procedure under evaluation is a standard genetic algorithm that implements crossover, mutation and selection to evolve a population of solutions to find a single best solution (global optimum).



The data for this project consisted of two csv format *.txt* files, one of which contained information about the ID, age, size, and volume per acre of each management unit in the lincoln tract, and one which contained a list of unit adjacencies to define the structure of the tract. It is important to note that both the *Minimum Harvest Age* and *Maximum Harvest Size*

constraints are accounted for in the data preprocessing, and thus the GA does not have to operate on these constraints directly.

Individual Representation: Each individual is a completed harvest scheduling plan. In the algorithm, the individuals are represented by an integer array of length 87, with each index corresponding to a fixed management unit, listed in order of identification number. The integer values correspond to either no treatment (0), or cut in the respective time period (1-6). These lists of integers are the representations that undergo crossover and mutation, but the correspondence of an index to its respective management unit remains the same throughout the algorithm.

Fitness Function: The objective to be maximized is the overall harvest flow from the Lincoln Tract. I also levied penalties for violated constraints, and weighted each constraint in order of importance. Any individual may have multiple constraints violated, and multiple penalties levied to them. The order of harshness of penalties from least harsh to most harsh go as follows: adjacency constraints, maximum flow constraints, even flow constraints. Penalizing these constraints allow for the algorithm to search through the infeasible solution space to find its way to the feasible solution space.

Crossover: There were four crossover types used for comparison in this optimization procedure. Two of these crossovers, *one point* and *two point,* are traditional crossover operators for GAs with permutation, binary, or integer representations. The other two crossovers, the ones primarily under scrutiny in this paper are novel. They are called *Geographic Crossover* and *Smart Geographic Crossover.* The standard procedure for *geographic crossover* is as follows. This crossover begins with two parent individuals (*P1* and *P2*) and results in two child individuals (*C1* and *C2*). A starting management unit, ***i,*** is selected at random among the genes of parent 1 (*P1*). A selection size, $0 \leq \boldsymbol{k} \leq 87$ is chosen at random. Then, data from the adjacency matrix is gathered to select all the adjacent stands to stand ***i***. The genes corresponding to these stands are added from *P1* into *C1* at their respective locations. This process is repeated for each adjacency matrix of each stand added to *C1*, in the same order they were added until ***k*** number of genes are transferred from *P1* to *C1.* Then the rest of the genes in *C1* are filled in from *P2*. After *C1* is created, the process gets repeated on *P2,* adding genes from ***i*** until ***k*** genes are transferred to *C2,* then the rest of the genes are filled in from *P1*.

Smart geographic crossover works very similarly to *geographic* crossover, with the main differences being: *smart* geographic crossover starts at a stand that necessarily has no adjacency violations, that is, stand ***i*** + ***n,*** where ***n*** is the number of stands it takes from stand ***i*** to find a stand with no adjacency violations. Then, stands from the adjacency matrix of stand ***i+n***
Note that adjacent stands may be represented as adjacent indices in the integer representation if their identification numbers are also adjacent. However, if the ID numbers are not adjacent, these stands are not adjacent in the internal representation, and this is the key fact that separates Geographic crossover from *two-point* crossover. In each run, the crossover probability was 0.9 to ensure fast convergence.

Mutation: To promote diversity and ensure that new genes can be introduced into the solution, I implemented uniform mutation, which randomly selects ***n*** genes from the solution,

and replaces their values with randomly selected values from the gene pool. This mutation was kept standard across all conditions to ensure the same exploration through the search space. Each individual had a 0.2 probability of being mutated to allow for escape from local maxima.

Selection: Parents were selected from the population using tournament-style selection, with a tournament size of 20. A large tournament size was adopted to increase selection pressure in the population. All children were kept each generation.

Termination Criteria: The algorithm terminated after 10,000 fitness evaluations, or 50 generations of population size 200.

2.3 Experiment

In this paper my goal is to compare geographic crossover and smart geographic crossover with two standard forms of crossover and no crossover at all. To do this, I ran the algorithm described above with each crossover 30 times on the optimization problem, for a total of 150 runs. Every run terminated after 10,000 fitness evaluations to test for speedy convergence to a global optimum. I recorded the best individuals and their fitness values from every run. Individuals that violated constraints were kept in the best solutions, although their fitness scores included the penalties from violating the constraints. In this paper, I observe the best and mean best fitness values of each algorithm, while ignoring the success rate, because only one best solution is needed.

3. Results:

Fig 1 and Fig 3 show a table and a graphical representation of the results from this experiment. Each type of crossover was implemented on 30 parallel runs of the optimization procedure. The best crossover operator was two point crossover, with a mean best fitness of 79,538 MBF and best fitness of 83,212 MBF. The next best operator was one point crossover, with a mean best fitness of 78,714 MBF and a best fitness of 82,440 MBF. Geographic crossover and Smart geographic crossover performed similarly to each other, netting a mean best fitness of 78,292 and 77,030 respectively, and best fitness values of 82,023 and 82,220 respectively. I also included data from 30 runs without crossover as a baseline for comparison.

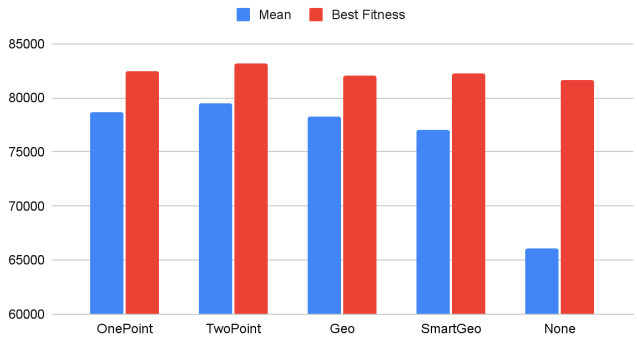
Fig 2 shows the t-test scores for each comparison. According to this chart, it appears that geographic crossover performed comparably with one point crossover and two point crossover, but smart geographic crossover varied more greatly from these two methods. Both of the novel methods, however, performed greatly better than no crossover.

Fig 1. Data Table	Mean	Best Fitness

OnePoint	78714.65367	82440.9019
TwoPoint	79538.08351	83212.3583
Geo	78292.72047	82023.8593
SmartGeo	77030.97146	82220.8453
None	66048.43543	81633.3894

Fig 2	Geo/ onepoint	geo/two point	smartgeo/ onepoint	smartgeo/ twopoint	onepoint/ twopoint	geo/smart geo	smartgeo/ none	geo/none
t-test	0.4746535	0.214857	0.152179	0.095134	0.400354	0.325144	0.000988	0.000338

Fig 3. Mean Fitness and Best Fitness for LTOP



Discussion:

From the results, I am left to conclude that geographic crossover and smart geographic crossover compare competitively with one-point and two-point crossover, if slightly worse. To extend this research further, these crossover methods should be used on more complex harvest scheduling problems, potentially those involving stream and waterway constraints, or those with multiple objectives. In addition, different tracts with more adjacency violations or a randomly generated set of adjacencies could be further used to validate the performance of these new crossovers. It is possible that the simplicity of the lincoln tract problem leads to greater success for simple operators, or it could also be the case that penalizing adjacency constraints harshly allowed for simple natural selection to do away with most of the adjacency constraints, without the need for keeping segments of land together during the crossover stage. Regardless, the level of success of these new crossover operators is promising enough to warrant future work on finding the benefits and drawbacks of these operators.

References

Bettinger, P., & Chung, W. (2004). The key literature of, and trends in, forest-level management planning in North America, 1950–2001. International Forestry Review, 6(1), 40–50. https://doi.org/10.1505/ifor.6.1.40.32061

Fotakis, D. G., Sidiropoulos, E., Myronidis, D., & Ioannou, K. (2012). Spatial genetic algorithm for multi-objective Forest Planning. Forest Policy and Economics, 21, 12–19. https://doi.org/10.1016/j.forpol.2012.04.002