SCANDINAVIAN FOREST ECONOMICS
No. 40, 2004

Proceedings
of the Biennial Meeting of the
Scandinavian Society of Forest Economics
Vantaa, Finland, 12th-15th May, 2004

Heikki Pajuoja and Heimo Karppinen (eds.)

Vantaa

This on-line version differs from the printed Proceedings 2004.
Ragnar Jonsson's paper is included in this version, but is missing from the paper copy.

A Hybrid Heuristic Algorithm for Harvest Decision of Mixed Species Stand under Price Risk

Fadian Lu
Department of Forest Economics
Swedish University of Agricultural Sciences
90183, Umeå, Sweden

Abstract

In this article, a hybrid heuristic algorithm based on Genetic Algorithm and Hooke and Jeeves is described for solving a complicated forest harvest decision problem, which involves optimization of thinning and final felling under price risk for a mixed species stand of spruce and pine. The strategy consists of two optimal stocking level functions and one reservation price function; in which, there are ten variables need to be optimized. The hybrid heuristic algorithm consists of two stages. At the first stage, Genetic Algorithm is applied to generate candidate initial solutions. At the second stage, the Hooke and Jeeves is applied to find the optimal solutions using these initial solutions. As benchmark, a pure Genetic Algorithm, Hooke and Jeeves, and Powell search are also tested. Results show that the hybrid heuristic algorithm is the best one among all of the tested algorithms. Genetic Algorithm ranks second, Hooke and Jeeves the third and Powell search is the worst.

Keywords: Harvesting decision, Genetic Algorithm, Hooke and Jeeves, Powell search

1. Introduction

The optimization of decision in forest management is usually complex due to the long planning horizon and high dimensionality. Consideration of uncertainty and multi-functionality of forest makes the problem becomes even more complex. The optimization of thinning and harvest decision under price uncertainty is one typical example. In such a case, the traditional optimization techniques usually can not be applied to solve the problem within a reasonable computation time. Simplification is one way to deal with this kind of problem, which, of course, will reduce the accuracy and correctness of the model for the real world problem. An alternative is using heuristic optimization techniques instead, to obtain the near optimal solution from complicated model which is more near the reality.

The aim of this study is to develop and test a heuristic approach for harvest decision problem (including thinning) of a mixed species stand of pine and spruce, when price uncertainty was considered. This approach is based on modified genetic algorithm and a traditional nonlinear optimization technique-Hooke and Jeeve.

The testing problem is come from a previous study of Lu and Gong (2004), in which an optimal thinning and final harvest strategy for mixed species stand of pine and spruce with price risk was developed. This strategy includes two optimal stocking level functions to guide the thinning decision for each species and a reservation price function for the final harvest decision. With this strategy, the thinning and final harvesting decision could be optimized simultaneously without any limitation on the number and intensity of thinning. Only the ten coefficients in the three functions need to be optimized through simulation. This was considered as an advantage compared with stochastic dynamic programming. In that study, a traditional non-linear optimization technique, Powell Search, was applied to optimize the coefficients.

However, it is not an easy task to optimize the ten coefficients in the three functions, due to the high dimensionality. Meanwhile, the computation burden for each iteration (objective function evaluation) in this problem is big, because of the complexity of the model. The forest stand growth is simulated by the single tree growth model of Söderberg (1986), which is the most complicated one among the stand growth simulation models. Furthermore, the thinning effect model of Jonsson (1980) and the natural mortality model of Bengtsson (1981) were also applied for the growth simulation, which, of course, make the model even more complicated. In addition to the complex stand growth simulation, 100 price scenarios were applied to simulate the stochastic price process, which means the stand growth simulation will be repeated 100 times for each iteration.

Using Powell Search, the satisfied result can only be obtained through certain numbers of repetitions with different initial guess. If some other kinds of risks (e.g. growth risk) are included, and /or more tree species are considered, the optimization problem would be more complicated. The function form of the optimal stocking level functions and the reservation price function used in the previous study is some how 'ad hoc'. More suitable function form could be found by further study, which possibly has more coefficients to be optimized, and therefore, an even higher dimensions problem needs to be solved. So a more efficient optimization approach is under high demanding.

2. Method

Before discuss the hybrid heuristic algorithm, we first illustrate the implementation of Genetic Algorithm, which is the main part of the hybrid algorithm, to the problem.

2.1 Implementation of Genetic Algorithm

Genetic algorithm (GA) is a heuristic optimization technique developed by Holand (1975). In the last twenty years, it has been applied to a wide range of areas. Together with other popular heuristic methods (Tabu Search, Simulated annealing, etc), GA also has been successfully applied to forestry to solve decision problems (Lu & Eriksson 2000, Öhman & Eriksson 1998, Wikström 2000). Based on a randomly generated initial population of solutions, GA improves these solutions by applying the basic genetic law: selection, crossover and mutation from generation to generation until certain stop criterion is met. The performance of a GA usually depends on the implementation details. Still, according to the no free lunch theory, GA does not necessarily perform better than other methods in all kinds of problems. In this study, in order to improve the performance of GA, a number of modifications were made, which will be illustrated in the following sections.

2.2 Coding of parameters

GA requires the natural parameter set of the optimization problem to be coded as a finite-length string over some finite alphabet (Goldberg, 1989). There are different ways for parameter coding, e.g. binary coding, real number coding, grey coding, etc. The binary coding method was used in this study. The ten parameters in the testing problem are continuous variables, so the coding of these parameters is also a process of discretion. In previous study, we determined the search scope for each of the ten parameters. If we let a binary string with length l to code one parameter with specified interval [U_{min}, U_{max}] , then the map between the cod and the value of the parameter is:

$$\underbrace{000..........0}_{l} \rightarrow U_{\min}$$

$$\underbrace{111..........1}_{l} \rightarrow U_{\max}$$

Others map linearly in between. The precision of this mapped coding is:

$$\pi = \frac{U_{\max} - U_{\min}}{2^l - 1}$$

When $U_{\min}$ and $U_{\max}$ is fixed, the precision could be controlled by choosing different value of (Goldberg, 1989). A smaller value of will reduce the search space and make it easier to find the optimal solution, but at the cost of lower precision. It will be the opposite if the value of is higher. In order to solve this problem, a multi-stage approach was developed in this study. At the first stage, with the original search scope [U_{min}, U_{max}], a relatively lower value of was chosen. This will simplify the problem but the precision was lower than the requirement. After the optimal solution was found, the second stage was started. and were adjusted accordingly and the interval was narrowed to half of the previous stage with the center of the optimal points found in previous stage. With the same value of, the precision will be improved in each succeeding stage . This process will be continued until certain criterion for the precision was met. In this study, we set for each of the ten coefficients. Therefore, the total length of the string is 70.

2.3 Generation of initial population

Finding a proper population size is important for the performance of GA. With a too small population, the algorithm tends to converge quickly and stop at local optima, while a too large population will increase the burden of calculation. A population size of 100 was chosen for our study problem after preliminary test. The initial population was created randomly, that is every position on the string has an equal probability of being 1 or 0. Each of the 100 strings will correspond to a set value of the ten parameters, a solution of our optimization problem.

2.4 Local improvement

The fitness of every string is evaluated according to its objective function value (directly or indirectly). The simple and straightforward principle is: the higher of string's objective function value, the fitter of it. However, this evaluation method will neglect the potential value of each string, especially at the initial search stage. This can be illustrated through a simple example of a one-dimensional maximization problem (figure1). The solution point P_1 is obviously better than point P_2 if their fitness is evaluated in terms of the objective function value. Nevertheless, the solution point P_2 is potentially better since it is near the global optimal point, and this fact will be neglected by normal fitness evaluation and selection. This can be improved to some extents by local improvement. For each string, we set a narrow search scope [U_1, U_2] (one tenth of the original search scope) and apply a simple random search with only five randomly generated solutions within this scope. Using the best one among the five solutions replaces the original strings if it has a higher value, otherwise, the original string keeps unchanged.

Local improvement will increase the burden of computation; therefore it is only applied in the first two stages. At later stages of the search process, it tends to converge to an optimal point (local or global), and all the candidates will gather at a smaller scope. Therefore, the local improvement is not as necessary.

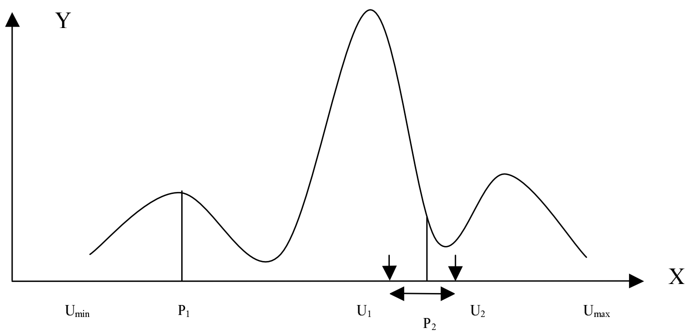

Figure 1. illustration of local improvement for one dimension maximization problem

2.5 Fitness scaling

Each string corresponds to a solution with an objective function value. Using the objective function value directly to evaluate the fitness of each string is usually not effective due to the following three reasons:

1. At initial stage, some strings with high objective function value could dominated the population; the others with low objective function value could be kicked off too early in the search process. This would be the reason for early convergence and stop at local optimal

2. At the final stage, the difference among the strings is small. The good string with high objective function value could not be favor enough and this will slow down the convergence process.

3. Some strings are infeasible solutions and their objective function values are zero; yet they still could carry useful information (genes), which could be used to create good new strings with other strings through crossover. If the fitness is evaluated by the objective function value directly, this kind of strings will be eliminated directly.

In this study, the following strategy was used for fitness scaling:

1. Determine the rank number N_i of string i in the population according to its objective function value. The best string (with the highest objective value) is assigned the rank number 1, while the worst string is assigned the rank number 100.

2. Using the following equation assigns new fitness value for each string:

$$Z_i = X^{100-N_i+1}$$

Z_i is the new value for each string.
X is adjustable coefficient. It ranges from 1 to 1.1.

The new value of a string not only depends on its rank number *N*, but also depends on the coefficient *X*.

For example, if we set the value of *X* to be 1.02, then the new value for the best string would be: 1.02^{100} while that for the worst string is: 1.02.

2.6 Selection

Roulette selection approach was applied in our study based on the new value for each string. The strings with higher value have high probability of being selected as candidate parents for the next generation. Since we use the new value to evaluate the fitness for each string, we can adjust the extent to favor the high value string by choosing different values of *X* during the selection. It is not difficult to find the most suitable value of *X* for selection. However, fixed value of *X* is not the optimal way for selection. In order to avoid the premature result, at the initial stage, it is suitable not to favor the good strings too much, and to keep the search in a wider range. At the final stage, in order to speed up the convergence process, the good strings should be favored more. Therefore, at different stages of the search process, different strategies should be applied. In stead of a fixed value of *X*, we designed a dynamic selection strategy which uses a relatively small value of *X* at the first stage and a higher value of *X* at the final stage. It is difficult to determine the value of *X* during different stages. The number of generation is not an accurate indicator of the process of convergence. We use the following equation to control the value of *X*:

$$X = X_1 + X_2 * \delta$$
$$\delta = (RE_2 - RE_1) / RE_1$$

in which
X_1 and X_2 are coefficients to determine the relation between *X* and δ
δ is the relative improvement of objective function value
ob^1 is the initial objective function value
ob^1 is the latest objective function value

During the selection, the ‘reposition of the best’ strategy was applied. That is, the best string in all generations so far will go to the next generation automatically. This is more important during the initial stage, when the diversity is high and the value of *X* is low (which mean that the good strings are not favored very much).

2.7 Crossover

In order to create new strings with the selected strings through selection, crossover was performed in hoping that the new strings could have higher fitness. The pair of parents was randomly chosen from the selected population, and with a pre-assigned probability P_c (crossover rate), the parents performed the crossover and created two new strings. If no crossover takes place (with probability 1- P_c), then form two new strings that are exact copies of the two parents.

We tested five different crossover strategies with different numbers of splice point through 1-5. The result showed that the four point crossover is the best one. The value of crossover rate P_c was also was decided by preliminary test.

2.8 Mutation

Mutation is considered as a secondary mechanism of GA, which performed after selection and crossover. Crossover only shuffles the substrings contained in the population. The substrings of the optimum have to be present in the population; otherwise a search by recombination is unable to locate the optimum (Muhlenbein, 1997). Therefore mutation is essential since its role is to recover some potentially useful genetic material (1's or 0's at particular locations) which is not available in the initial population (Goldberg, 1989). Since we use binary coding in our problem, the mutation is simply occasional changing from 1 to 0 and vice versa. The probability of this changing for each position is called the rate of mutation.

2.9 Stop Criterion

As mentioned in the coding section, in order to meet the precision requirement and keep the length of the string short, we divided the search process into stages, with the search scope being narrowed at each succeeding stage. The stop criterion for the whole process is that a pre-specified precision being met. And for each stage, we set a specified number of iteration without improvement as the stop criterion to end that stage.

The results of this GA strategy developed here will be presented, discussed and compared with others later.

2.10 Designing the Hybrid Heuristic Algorithm

GA has the population based structure and is a parallel random search with centralized control through selection schedule. Instead of improving one solution points step by step, GA evolves a certain number of solutions- the population simultaneously. This character makes GA powerful to locate the optimal point. However, due to the same character, GA also has the drawback of slower convergence after the optimal point being located, compared with some traditional hill climbing methods. This fact motivates the study on hybrid algorithm of GA and other optimization approaches. The idea is using GA to locate the mountain (the rough position of the optimal points), and let others to climb the mountain. In this study, we designed hybrid heuristic algorithm, which is GA plus Hooke and Jeeve (HJ). We also tested GA plus Powell Search, which is not as good as GA plus HJ.

There are two important things for this hybrid algorithm. One is choosing the right time of switching from GA to HJ; the other is selecting the solutions among the populations of GA to be used as the initial solution points for HJ.

It is difficult to deicide the right time to stop GA and start HJ. There is no sign to indicate if GA has really found the position of optimal solution or not. By examining the performance of GA developed previously, we can see that the improvement of the objective function value is mainly made during the initial stage of GA. This gives us a hint that we can stop the GA after the main improvement. In this study, we set a limited number of generations for GA and after this limitation is reached, GA will be stopped. 50 generations was used as the limitation for GA. That is to say, we let GA run 50 generations, and then switch to HJ.

Then the second question is which of the solution among the population should be used as the initial solution for HJ. Intuitively, the best solution among the population should be chosen as the initial solution. However, is there any value of the other solutions? Would it be helpful to take average of some of the other solutions? In order to answer these questions, we tested ten candidates solutions selected from the population. The first five is the five best solutions among the population, then, the sixth is the average of the five best solutions, the seventh is the average of the ten best solutions, the eighth is average of the twenty best solutions, the ninth is the average of the fifty best solutions, and the tenth is the average of all the population (100 solutions).

3. Result

Besides GA and the hybrid algorithm, we also tested random search, Hooke and Jeeve, and power search, and used their results as benchmarks to check the performance of the first two algorithms.

The parameters of GA were determined after preliminary test. The mutation rate is 0.01, crossover rate is 0.8, and the value of the two parameters X_1, X_2, which were used to determine the selection rate, is 1.005 and 0.06, respectively.

Through examining the convergence process of GA, we found that, in most of the situations, the improvement is mainly made during the first 50 generation. Therefore, for the hybrid algorithm, the GA will be stopped and switch to HJ after 50 generations. Of course, if GA is given more generations before switching to HJ, the quality of the initial solutions for HJ will be higher, and the finial results for each run would be better. However, this will increase the time for each run, and decrease the overall efficiency.

Since it is impossible to get the real optimal results, and in order to compare the performances of different algorithms, we set the same fixed computation time for each of the algorithms and just take the ten best results from them. Due to the same reason, the relative deviation index (RDI) (Kim and Kim, 1996) was used to compare the results of different algorithms. The definition of RDI is:

$$RDI = (TI - TW)/(TB - TW)$$

Where *TI, TW, TB* are respectively, the objective function value of result I, the worst result and the best result.

The results were shown in table2:

Table 2. The mean RDI of the ten best solutions determined using different algorithms

algorithm	RM	PW	HJ	GA	HB1	HB2
Ten best results	0.346	0.935	0.953	0.940	0.964	1.000
	0.283	0.820	.0775	0.920	0.964	0.962
	0.219	0.385	0.733	0.847	0.937	0.941
	0.187	0.363	0.672	0.791	0.933	0.925
	0.110	0.363	0.653	0.748	0.924	0.920
	0.071	0.355	0.466	0.461	0.918	0.919
	0.068	0.327	0.449	0.443	0.762	0.916
	0.027	0.318	0.303	0.416	0.740	0.885
	0.014	0.317	0.292	0.369	0.701	0.885
	0.000	0.308	0.279	0.355	0.560	0.885
Average	0.133	0.449	0.558	0.629	0.840	0.924

The fixed computation time is the time needed by GA for ten runs. Since other algorithms cost less time for each run compared to GA, the ten results listed here is the ten best results among their total results. From the results in table 2, it is clear that the hybrid algorithm performs best among the five tested algorithms, while random search is the worst one. The modified GA developed in this study is better than the two traditional optimization algorithms, HJ and PS; while HJ is better than PS.

As mentioned before, for the hybrid algorithm, ten solutions from the GA stage were used as the initial solutions for the HJ stage. We run 150 times for the hybrid algorithm in order to test the performances of different initial solutions. The frequency of the best result occurred from different initial solutions was presented in table 3.

Table 3. Frequency of the best result occurred from 10 initial solutions

No. of initial solution	1	2	3	4	5	6	7	8	9	10
Frequency of best result	43	18	16	19	21	6	4	10	7	6
Percentage (%)	28	12	10	13	14	4	3	7	5	4

It is not surprising to see the highest frequency of the best result occurred when the best solution of GA stage was used as the initial solution for the following HJ stage. However, there is still a very high percentage (72%) of the best solution being found when other solutions from the GA stage was used as initial solution for HJ stage. This is to say, the best point (the point with the highest objective function value) got from GA is not necessarily the best starting point for HJ to climb the top of the mountain. Since each run of HJ takes relatively little time, it is worthwhile to try multiple initial starting points, in order to improve the performance of this hybrid algorithm.

4. Discussion and Conclusion

In this study, a hybrid heuristic algorithm, which is based on Genetic Algorithm and Hooke and Jeeve, was tested for a forest harvest decision problem under price risk. Genetic Algorithm, random search, Hooke and Jeeve, and Power Search were also applied to the problem. The results show that if properly implemented, GA will perform better than the traditional algorithms tested in this study for continuous variables optimization problems. An even better performance was observed for the hybrid algorithm based on Genetic Algorithm and Hooke and Jeeve. This indicates that heuristic optimization approach is promising for complicated forestry decision problems.

The performance of GA is problem specific and a successful application is highly based on the preliminary work for the right way to implement it. Besides the work for finding the right value of parameters for different operations (selection, crossover, mutation), it is also very important to interpret the problem properly, that is, finding the suitable way to code parameters in the problem. In this study, we use the traditional binary coding system to cod the variables. Since the variables are continuous, to code them is also the process of discretion. A dynamic search scope strategy was developed in order to keep the length of strings relatively short and meanwhile satisfied the precision requirement. The speed of narrowing the search scope as stage goes on is low, (the scope of any stage is half of that of previous stage), to avoid missing of the real optimal point.

A strategy to map the objective function value to a new fitness unit was developed. In this strategy, the fitness of each string will depend on its position in the order which made according to its original objective function values. By this strategy, the relative fitness of each string is adjustable, and this gives us the chance to develop a dynamic selection strategy. That is, at the initial stage, the good strings (with high objective function value) are favored less compared to that in the final stage. This is controlled by adjusting the relative fitness in deferent stages, and the relative improvement of the objective function value was used to indicate the stage process.

Different ways to combine GA and Hooke and Jeeve to form the hybrid algorithm were tested. It turns out that ending the GA at its early stage (after 50 generations) and then switch to Hooke and Jeeve is the most efficient way. This result is due to the fact that GA is more capable of locating the rough position of the global optimal point and Hooke and Jeeve is faster in convergence to local optimal points. The five best solutions from the last generation (generation 50) of GA were used as the initial solution of HJ. Besides that, another five solutions were derived from the GA solutions (the means of the five best solutions, ten best solutions, twenty best solutions, fifty best solutions and total solutions respectively) were also tested. The result showed that using the best solution from GA as the initial solutions of HJ has the highest tendency to reach the final best solution. However, among the 150 tested cases, 72% of the best final solutions were obtained using the other initial solutions. The overall efficiency was improved by using multiple initial solutions for HJ instead of one initial solution.

References

BENGTSSON, G. 1981. Beräkning av den naturliga avgången ur virkesförrådet I HUGIN-systemet (Stecil). Department of Forest Survey, Swedish University of Agricultural Sciences.

GOLDBERG, D.E. 1989. Genetic Algorithms in search, Optimization, and Machine Learning. Addison Wesley, Reading, Massachusetts.

HOLLAND, J. H. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press.

JONSSON, B. 1980. Functions for long-term forecasting of the size and structure of timber yields (in Swedish with English summary). Report 7, Department of Biometry and Forest Management, Swedish University of Agricultural Sciences, Umeå.

KIM, J.U., Kim, Y.D., 1996. Simulated annealing and genetic algorithms for scheduling products with multi-level product structure. Computers Ops. Res. 23, 857-868.

LU, F. and Gong, P. 2004. Adaptive Thinning Strategy for Mixed-Species Stand Management with Stochastic Prices. Working paper, Department of Forest Economics, Swedish University of Agricultural Science, Umeå

LU, F. and Eriksson, L.O. 2000. Formation of Harvest Unit with Genetic Algorithms. Forest Ecology and Management 130: 57-67

MUHLENBEIN, H. 1997. Local Search in Combinatorial Optimization. Wiley

SÖDERBERG, U. 1986. Functions for forecasting timber yield: increment and form height of individual trees of native tree species in Sweden (in Swedish with English summary). Report 14, Department of Biometry and Forest Management, Swedish University of Agricultural Sciences, Umeå.

WIKSTRÖM, P., Eriksson, L.O. 2000. Solving the stand management problem uncer biodiversity-related considerations. Forest Ecology and Management 126: 361-376

ÖHMAN, K. and ERIKSSON, L.O. 1998. The core area concept in forming contiguous areas for long term forest planning. Canadian Journal of Forest Research 28: 1032-1039