Eur J Forest Res (2016) 135:765–779
DOI 10.1007/s10342-016-0971-x

ORIGINAL PAPER

Fine-tuning heuristic methods for combinatorial optimization in forest planning

Xingji Jin[1] · Timo Pukkala[2] · Fengri Li[1]

Received: 6 January 2016 / Revised: 8 April 2016 / Accepted: 25 May 2016 / Published online: 13 June 2016

Abstract Heuristic methods are commonly used in complicated spatial forest planning problems to find the best combination of management alternatives for stands. The performance of heuristic methods depends on the parameters that guide their search processes. This study used numerical optimization to find the optimal parameter values for simulated annealing (SA), threshold accepting (TA), great deluge (GD), tabu search (TS), genetic algorithm (GA) and ant colony optimization (AC) when they are used for combinatorial optimization in forest planning. Ant colony optimization was implemented using the Max–Min Ant System, which was applied for the first time to forest planning problem. Solutions found by different heuristic methods for a non-spatial and a spatial forest planning problem were compared in a situation where the search time was restricted. The comparisons revealed that SA and TA were the best methods for fast search in both non-spatial and spatial problems. GA and AC were the least satisfactory methods, and GD and TS were between the best and the worst heuristics. The main reason for the poor performance of GA and AC was their slow search process. Differences between heuristic methods decreased when the allowed search time increased.

Communicated by Martin Moog.

✉ Fengri Li
fengrili@126.com

[1] Department of Forest Management, School of Forestry, Northeast Forestry University, Harbin 150040, Heilongjiang, People's Republic of China

[2] University of Eastern Finland, P.O. Box 111, 80101 Joensuu, Finland

Keywords Simulated annealing · Threshold accepting · Great deluge · Tabu search · Genetic algorithm · Ant colony optimization · Max–Min Ant System · Hooke and Jeeves · Heuristic search

Introduction

A typical forest planning problem consists of finding the optimal combination of treatment schedules for the stands of a forest. Combinatorial optimization is commonly used in this task. The methods for combinatorial optimization fall into two categories: mathematical programming and heuristics (e.g., Bettinger et al. 2009). Heuristic methods have been increasingly used as an alternative to mathematical programming (e.g., Pukkala and Kangas 1993; Boston and Bettinger 1999, 2002; Falcão and Borges 2002; Liu et al. 2006). Other methods such as dynamic programming and nonlinear optimization are frequently used in stand management optimization where the optimized variables are continuous (e.g., Haight et al. 1985; Valsta 1992; Pukkala 2007).

Most of the used heuristics fall to the category of global, or centralized, methods since the quality of a solution and the effects of changes that are made in the solution are evaluated at the forest level (globally). Decentralized methods optimize the management of each individual stand, but, when doing this, they are able to take into account forest-level constraints. Examples of decentralized methods are cellular automata (Strange et al. 2002; Heinonen and Pukkala 2007; Mathey et al. 2007) and the method of reduced costs (Hoganson and Rose 1984; Pukkala et al. 2009).

The most common global heuristics are simulated annealing, tabu search and genetic algorithm (Reeves 1993;

Borges et al. 2002). Simulated annealing (SA) is a cooling method since it mimics the cooling process of melted metal. Other heuristics that can also be called as cooling methods are threshold accepting (TA) and great deluge (GD, Bettinger et al. 2002). All these methods accept also inferior moves (such changes in the solution that make it worse), but the probability of accepting inferior moves decreases when the optimization proceeds and the process cools. The purpose of accepting inferior moves is to provide a way to get away from local optima.

Tabu search (TS) uses tabu lists for the same purpose. Recent moves are temporally forbidden, which forces the method to explore a wider set of alternative solutions. Typical of tabu search is that several candidate moves are generated, and the best non-tabu candidate is accepted (Reeves 1993). While tabu search and all cooling methods operate with one solution, genetic algorithms (GA) operate with a population of solutions, which are combined to obtain a new generation of solutions (Reeves 1993). The solutions are called chromosomes. The objective function value calculated for a solution is called fitness.

Ant colony optimization (AC) is another heuristic, which operates with more than one solution (Zeng et al. 2007). In AC, solutions are called ants. Pheromone tracks left by the ants affect the way in which new ants select treatment schedules for stands when they construct their solutions. AC is a widely used heuristic method (Blum 2005; Dorigo et al. 2006), but it has been used only once in forest management planning (Zeng et al. 2007).

The advantage of heuristics is their flexibility. They can be used in problems in which the additivity and proportionality requirements of linear programming are not met. The objective variables can be spatial or non-spatial. The only requirements of heuristics are that the objective variable is numerical and there is a method for calculating its value. Heuristics are easy to use in multiobjective optimization. The relationships between objective variables and objective function (OF) can be linear or nonlinear, and the effects of different objective variables on the objective function value can be additive, multiplicative or a combination of both. Although mathematical programming methods can be used in complicated spatial (Baskent and Keles 2005; Constantino et al. 2008; Könnyű et al. 2014.) and stochastic (Eyvidson and Kangas 2014) problems, it is likely that the use of heuristic methods continues to increase since the complexity of forest management problems is increasing due to the increasing number of products and services that should be considered in forest planning.

The problem of heuristic methods is that they do not necessarily find the optimal solution. Usually, they find a good solution, but it is not assured that the solution is the global optimum. However, the practical significance of this limitation may be small since there are much bigger sources of uncertainty in forest planning. These uncertainties include errors in inventory data and growth models, as well as uncertainties related to future timber prices, regeneration of trees, pest outbreaks, diseases and various abiotic hazards.

Another problem is the difficulty to find the best parameters for the heuristics. Each method has a few parameters which greatly affect the performance and time consumption of the method. The optimal parameter values depend on the type of the problem, which means that different parameter values should be used in different problems. It is extremely tedious to systematically analyze the effect of different parameter combinations on the performance of the heuristic. For example, if the heuristic has 5 parameters and 10 different values of each parameter are compared, the number of parameter combinations is $10^5 = 100{,}000$. Therefore, when seeking good parameter values for the comparison of different heuristic methods (Bettinger et al. 2002; Pukkala and Kurttila 2005), it is common to examine one parameter at a time and ignore the interactions of parameters. However, due to the interrelationships between parameters, this approach does not guarantee that the best parameter values were found. This limitation has also an effect on the comparisons of alternative heuristic methods since the parameters used may be more optimal for certain heuristic than for the others.

An alternative to trial and error is to use optimization in the search for optimal parameter values. This would make the comparisons between different heuristic methods more 'fair' since none of the methods is impaired by poorly set parameters. There is already one study that used optimization to find the best parameter values for heuristic methods (Pukkala and Heinonen 2006). However, several of the well-known heuristic techniques (TS, GA, GD, AC) were not included in the optimizations.

This study aimed at providing a fair comparison of the most common global heuristics in non-spatial and spatial forest planning problems. The analysis consisted of two steps. First, the parameters of each method were optimized on the constraint that each method had the same time to find the solution. Second, the OF values produced by different heuristics with the optimal parameter values were compared.

Methods

The heuristic methods were used to find the optimal combination of treatment schedules generated for the stands of a forest. The treatment schedules were produced before running the heuristics. The CMForest software (Jin et al. 2016) was used to produce the schedules. In the case study area used in this study the average number of treatment schedules was 14.5 per stand, and the number of stands was 888. This means that the magnitude of the number of different combinations of treatment schedules (size of decision space) was 14.5^{888}, which is a huge number. The task of the heuristic was to find the best of these combinations.

Each heuristic can be implemented in different ways. In this study SA, TS and GA were used as explained in Pukkala and Heinonen (2006), who in turn based their implementations on the advices given in Reeves (1993). TA and GD were implemented as described in Bettinger et al. (2002). Because there is plenty of literature on all these methods (e.g., Reeves 1993; Bettinger et al. 2002; Borges et al. 2002; Pukkala and Kurttila 2005), they are described only briefly below. The AC heuristic used in this study is based on the descriptions of Blum (2005) and Dorigo et al. (2006).

Simulated annealing

SA begins with an initial solution. The initial solution is the best of several random combinations of treatment schedules of stands. Moves (changes) are made to the initial solution by selecting first a random stand and then a random schedule for this stand from those produced beforehand. The schedule that is currently in the solution is replaced by the randomly selected alternative schedule. The replacement (move) is maintained if it improves the OF value; otherwise, it is maintained with the following probability:

$$p = \exp\left[\frac{(\mathrm{OF}_{\mathrm{Candidate}} - \mathrm{OF}_{\mathrm{Current}})}{T}\right] \qquad (1)$$

where $\mathrm{OF}_{\mathrm{Current}}$ is the OF value of the solution before implementing the move, $\mathrm{OF}_{\mathrm{Candidate}}$ is the OF value after the move, and T is 'temperature' which determines the probability of accepting inferior moves. The temperature is decreased toward the end of the search, which means that the probability to accept inferior moves also decreases. Several candidate moves are produced at each temperature. The search is stopped when the temperature reaches a predefined low value, which is called freezing temperature. A new temperature is obtained by multiplying the current temperature by a multiplier smaller than 1. It is possible to increase or decrease the number of candidate moves per temperature when the process cools. An increase means that the search is intensified toward the end of the search.

SA has the following parameters:

- P1: number of random searches done to obtain the initial solution
- P2: starting temperature
- P3: freezing temperature
- P4: cooling multiplier (<1)
- P5: search intensity at initial temperature, expressed as the proportion of the number of stands (1.0 means that the number of evaluated candidate moves is equal to the number of stands)
- P6: multiplier of search intensity, to obtain the search intensity in the next temperature

Threshold accepting

TA is fairly similar to SA, and in this study, it was implemented in a nearly similar way as SA. The difference is that whereas SA accepts inferior moves with certain probability, TA accepts all moves which produce an OF value greater than the current value minus a threshold. For example, if the current OF value is 0.9 and the threshold is 0.05, all moves that produce a solution with OF value greater than 0.85 are accepted. Cooling is implemented so that the threshold is reduced toward the end of the search. Search is terminated once the threshold value is close zero. This threshold may be called as freezing threshold.

TA has the following parameters:

- P1: number of random searches done to obtain the initial solution
- P2: starting threshold
- P3: freezing threshold
- P4: cooling multiplier (<1), to obtain the next threshold
- P5: search intensity at initial threshold, expressed as the proportion of the number of stands
- P6: multiplier for search intensity, to obtain the search intensity at the next threshold

Great deluge

GD can also be classified as cooling method. Cooling is implemented via rising water level. Water level is the lowest accepted OF value. Water has an initial level. Always when a move improves the solution, there is a rain event, which rises the water level. The amount of rain gets smaller when the search continues. This type of search stops when the water level is equal to the current OF value. The search continues with another search mode, namely direct ascent (hill climbing). The direct ascent search mode selects a random stand and replaces its current treatment schedule by another, randomly selected schedule simulated for the same stand. All improving moves are accepted, and all moves which do not improve the solution are rejected. Direct ascent steps are repeated for a certain number of iterations.

GD has five parameters:

- P1: number of random searches done to obtain the initial solution
- P2: initial water level
- P3: initial rain
- P4: rain multiplier (<1), to obtain the magnitude of the next rain
- P5: number of direct ascent iterations at the end of the search

Tabu search

TS is somewhat different from the cooling methods. The first difference is that several candidate moves are produced and only one of them is accepted. The number of candidates can be anything from two to the highest possible value (total number of schedules minus number of stands). The best non-tabu candidate is accepted even if it produces an OF value lower than the current value. If all candidate moves are in the tabu list (forbidden), the move that has the shortest tabu tenure is accepted (shortest tabu tenure means that the move will be the first to be removed from the tabu list). Moreover, if a move would produce a better OF value than the best found so far, the move is accepted even if it is tabu.

Another feature is the use of tabu lists, which prevent the search process from repeating recent moves. The two schedules that participate in a move go to a tabu list in which they stay for a certain number of iterations. Iteration refers to the generation of a set of candidate moves and selecting the best of them. For example, an initial tabu tenure 10 means that the schedule which goes to the list is tabu for 10 iterations. Every iteration reduces the remaining tabu tenures of all listed schedules by 1.

The tabu search heuristic used in this study has two tabu lists, one for the schedules that are replaced by another schedule (schedules that leave the solution) and the other list for schedules that replace the removed schedule (schedules that enter the solution). The whole stand is tabu as long as the entering schedule is tabu. Tabu search is terminated when a certain number of iterations are completed.

The parameters of TS are:

- P1: number of iterations
- P2: number of candidate moves produced per iteration
- P3: length of the tabu tenure of leaving schedules
- P4: length of the tabu tenure of entering schedules

Genetic algorithm

GA can be implemented in several different ways. Common to all implementations is that, opposite to all previous heuristics, they operate with several solutions. These solutions are called chromosomes. New chromosomes (off-spring) are produced by combining two parent chromosomes using crossing-over. The new chromosome may have mutations, i.e., small changes somewhere in the chromosome. When GA is applied to forest planning, a gene corresponds to a stand, and different treatment schedules of the stand are the alleles of the gene. A chromosome is a list of schedules which are included in the solution (Fig. 1).

In this study, only one off-spring was produced per iteration (generation). Two parent chromosomes were selected from the current population. For the first parent, the probability of selection depended on the fitness value (OF value) of the chromosome, so that the best chromosome had the highest probability to become selected. The second parent was selected randomly, using the same probability for all chromosomes.

Then, two crossing-over places were selected randomly. The genes located in the chromosome before the first and after the second point were taken from the first parent, and the remaining genes were taken from the second parent (Fig. 1). For example, if the solution (chromosome) consisted of treatment schedules for 100 stands and the crossing-over places were 25 and 66, the schedules for stands 1–24 and 67–100 were taken from the first solution and the schedules of stands 25–66 were taken from the second solution.

Mutations were applied to the new off-spring. A mutation means that a random gene (stand) is selected, and its schedule is replaced by another, randomly selected schedule simulated for the same stand. Mutation rate changed during the GA search process.

After a new off-spring was produced, one chromosome was removed from the population. The probability of removal was inversely proportional to the ranking of the chromosome (worst chromosomes had the highest probability of removal). The search was stopped when a certain pre-defined number of generations were completed.

The parameters of GA are:

- P1: number of chromosomes
- P2: number of random searches that are used to produce an initial chromosome
- P3: number of generations
- P4: initial mutation rate (at first generation)
- P5: final mutation rate (at last generation)

Mutation rate was changed gradually from the initial to the final rate. Rate 2.5 means that two mutations are made with certainty and a third mutation is made with the probability of 0.5.

Ant colony optimization

The original plan in this study was to use the AC algorithm proposed for forest planning by Zeng et al. (2007).

Stand	1	2	3	4	5	6	7	8	9	10
Parent 1	3	2	11	13	7	15	1	9	4	6
Parent 2	12	5	10	4	5	7	7	8	11	1
Off-spring after crossing-over	3	2	11	4	**5**	7	7	9	4	6
Off-spring after mutation	3	2	11	4	**11**	7	7	9	4	6

Fig. 1 Production of off-spring in genetic algorithm. Two parent chromosomes (solutions) and crossing-over points (*thick vertical lines*) are selected. The *middle part* of the off-spring is taken from the second parent, and the rest is taken from the first parent. Mutations can be made in the newly produced off-spring (the *gray-shaded cells* with *boldface numbers* show one mutation). The *numbers* in the chromosomes indicate the number of the treatment schedule which is included in the solution represented by the chromosome

However, preliminary analyses revealed that this algorithm was not working satisfactorily in the forest planning problems analyzed in this study. Therefore, each of the three variants of ant colony optimization described in Dorigo et al. (2006), namely Ant System, Max–Min Ant System, and Ant Colony System, was adapted to forest planning and tested. It turned out that the Ant System was inferior to the other variants, and the Ant Colony System was not better than the Max–Min Ant System. Therefore, the Max–Min Ant System was selected for parameter optimization.

In the Max–Min Ant System used in this study, all treatment schedules are given the same initial pheromone level. Each ant of the first ant iteration constructs a solution by selecting a random treatment schedule for each stand. The objective function value is calculated for each ant.

The pheromone levels of all treatment schedules are updated at the end of each iteration. In the Max–Min Ant System only the best ant updates the pheromones. This ant can be the best ant of the last iteration (iteration-best) or the best ant of all iterations completed so far (best-so-far). In our study, both the iteration-best ant and the best-so-far ant updated the pheromones of the treatment schedules. The pheromone update was done as follows:

$$\tau_{ij} \leftarrow \left[(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{\text{best}}\right]_{\tau_{\min}}^{\tau_{\max}} \tag{2}$$

where τ_{ij} is the pheromone level of treatment schedule i of stand j, ρ is the pheromone evaporation rate, and $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum pheromone levels, respectively. The amount of pheromone addition was equal to half of the objective function value of the solution generated by the iteration-best or best-so-far ant:

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} OF_{\text{best}}/2 & \text{if } i \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

After completing the first iteration and updating the pheromone levels of all treatment schedules the same number of ants constructed new solutions so that the probability of selecting a schedule depended on the pheromone levels of the treatment schedules as follows:

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha}}{\sum_{i=1}^{n_j} \tau_{ij}^{\alpha}} \tag{4}$$

where p_{ij}^{k} is the probability that ant k selects schedule i for stand j, τ_{ij} is the pheromone of schedule i of stand j, and n_j is the number of treatment schedules simulated for stand j. The heuristic information included in most ant colony algorithms (Dorigo et al. 2006) was not used in the AC heuristic tested in this study.

The same process of constructing solutions and updating pheromones was repeated for a certain number of ant iterations. The best solution constructed during the whole search process (best-so-far) was the solution found by the AC heuristic.

The parameters of AC are:

- P1: number of ants
- P2: number of iterations
- P3: evaporation rate (ρ in Eq. 2)
- P4: probability exponent (α in Eq. 4)
- P5: initial pheromone
- P6: minimum pheromone ($\tau_{\min}$)
- P7: maximum pheromone ($\tau_{\max}$)

Case study problems

The heuristics were optimized and compared using data from the Mengjiagang Forest, located in the Heilongjiang Province in Northeast China. The forest has been divided into 93 compartments, and each compartment has been further divided into subcompartments so that the total number of subcompartments is 888. Subcompartment, which corresponds to stand, was the calculation unit in the simulation of treatment alternatives. The total area of the case study forest is 15,533 hectares.

The CMForest software (Jin et al. 2016) was used to simulate alternative treatment schedules for the stands for three 10-year periods. The total number of alternatives simulated for the 888 stands was 12,940, which gives an average of 14.57 schedules per stand. Schedules representing even-aged management were simulated for planted stands, and selective cuttings were simulated for all other stands. Details of the automated simulation tool of CMForest can be found in Jin et al. (2016).

The parameters of the heuristics were optimized for a non-spatial and spatial problem. The forest planning problems were formulated in a utility theoretic way as follows:

$$\max U = \sum_{i=1}^{I} a_i u_i(q_i) \tag{5}$$

subject to

$$q_i = Q_i(\mathbf{x}), \quad i = 1, \ldots, I \tag{6}$$

where U is total utility, a_i is the weight, u_i is the subutility function, and q_i is the quantity of objective variable i. Q_i is the procedure which calculates the value of objective variable i from the information of those treatment schedules that are included in the solution, and $\mathbf{x}$ is a vector which indicates the ID numbers of those schedules which are included in the solution (see Fig. 1).

Subutility functions scale all objective variables to range 0–1. In addition, a subutility function tells how the utility depends on the quantity of the objective variable. The weights (a_i) were scaled so that their sum was equal to 1. Therefore, the total utility also ranged from 0 to 1.

The utility function of the non-spatial problem was as follows

$$U = 0.25u_1(H_{\mathrm{I}}) + 0.25u_2(H_{\mathrm{II}}) + 0.25u_3(H_{\mathrm{III}}) + 0.25u_4(V_{30}) \tag{7}$$

where H_{I}, H_{II} and H_{III} are the harvested volumes of the first, second and third 10-year period and V_{30} is the total growing stock volume at the end of the third 10-year period. The spatial problem had two additional objective variables:

$$U = 0.1667u_1(H_{\mathrm{I}}) + 0.1667u_2(H_{\mathrm{II}}) + 0.1667u_3(H_{\mathrm{III}}) + 0.1667u_4(V_{30}) + 0.1667u_5(\mathrm{CC}) + 0.1667u_5(\mathrm{CNC}) \tag{8}$$

where CC is the length of the common boundary between such adjacent stands which are both cut during the same 10-year period, and CNC is the boundary between such adjacent stands of which one is cut and the other is not cut during a certain 10-year period (Kurttila et al. 2002; Palahí et al. 2004). Maximization of CC leads to aggregated harvest blocks and large total area of harvested stands, and minimization of CNC leads to compact harvest blocks and small total area of harvested stands. Previous research has shown that simultaneous use of these two variables as management objectives leads to good aggregation of harvest blocks (Islam et al. 2010).

The utility functions for harvested volumes were formulated so that a removal of 300,000 m^3 gave the maximal subutility, and deviations to both directions decreased utility. The cutting target of 300,000 m^3 is close to the 10-year volume increment of the case study forest. Linear subutility functions were used for V_{30} and CC. The maximum possible value gave subutility 1, and if the value of the objective variable was zero, subutility was also zero. The subutility function of CNC was also linear, but now the maximum possible value of CNC resulted in subutility 0, and the lowest possible value of cut-uncut border (0 m) resulted in subutility 1. This means that CNC was minimized.

Optimization method

The method of Hooke and Jeeves (1961) was used to find the optimal values of the parameters of each heuristic. The direct search of the Hooke and Jeeves method consists of alternating steps of exploratory and pattern searches. The exploratory search changes the value of one variable at a time, and all changes that improve the objective function are accepted. After examining all optimized variables in the exploratory search the algorithm goes to pattern search, which makes simultaneous changes in more than one variable. Then the step size (magnitude of change) is halved, and the sequence of exploratory and pattern searches is repeated. This process is continued until the step size becomes smaller than the stopping criterion of the algorithm.

Since the Hooke and Jeeves direct search algorithm does not necessarily converge to the global optimum, the search was repeated a few times, starting from different initial solutions (different initial combinations of the parameters of the heuristic). The parameter optimizations for the non-spatial forest planning problem were repeated 5 times, and 2 or 3 repeated direct searches were used when the parameters were optimized for the spatial problem, for which the heuristic search was more time-consuming.

The starting point of the first direct search was an initial guess by the authors (‘Start’ in Table 1). The other direct searches started from the best of 100 random combinations of the parameters of the heuristic. The random parameter values were drawn from uniform distributions specified by the minimum and maximum values of the respective parameter (Low and High in Table 1). The initial step size of the direct search was 10 % of the range of variation (Low–High). The direct search was terminated when the step size was less than 1 % of the initial step size. Since the direct search was allowed to move beyond the Low–High range, every parameter was also given a minimum allowed and maximum allowed value (Min and Max in Table 1).

Table 1 Ranges and starting values for the optimized parameters of the heuristic algorithms

Parameter	Min	Low	Start	High	Max
Simulated annealing					
Initial random searches (no.)	1	1	100	1000	10,000
Initial temperature	0.00001	0.0001	0.001	0.01	0.1
Cooling multiplier	0.5	0.8	0.9	0.925	0.95
Freezing temperature	0.00000001	0.0000001	0.00001	0.0001	0.001
Initial search intensity	0.1	0.5	1.0	2.0	10.0
Search intensity multiplier	0.1	0.8	1.05	1.1	1.2
Threshold accepting					
Initial random searches (no.)	1	1	100	1000	10000
Initial threshold	0.00001	0.0001	0.001	0.01	0.1
Cooling multiplier	0.5	0.8	0.9	0.93	0.96
Freezing threshold	0.00000001	0.0000001	0.00001	0.0001	0.001
Initial search intensity	0.1	0.5	1.0	5.0	10.0
Search intensity multiplier	0.1	0.8	1.05	1.1	1.3
Great deluge					
Initial random searches (no.)	1	1	100	1000	10,000
Initial relative water level	0.4	0.6	0.8	0.9	0.95
Initial rain	0.000001	0.00001	0.001	0.01	0.1
Rain multiplier	0.8	0.9	0.99	0.999	0.99999
Final direct accent searches	1	1	100	1000	10,000
Tabu search					
Number of iterations	1	100	1000	50,000	100,000
Candidates per iteration	1	1	10	100	1000
Tabu tenure for leaving schedule	0	1	10	50	100
Tabu tenure for entering schedule	0	0	5	50	100
Genetic algorithm					
Number of chromosomes	2	4	20	100	300
Random searches per chromosome	1	1	10	100	1000
Number of generations	1	50	100	2000	10,000
Starting mutation rate	0	0	1	10	100
Ending mutation rate	0	0	1	10	100
Ant colony optimization					
Number of ants	2	5	30	60	500
Number of iterations	3	10	100	1000	5000
Pheromone evaporation rate	0.01	0.05	0.2	0.7	0.9
Probability exponent	0.1	1.0	1.5	4.0	10.0
Initial pheromone	0.01	0.1	1.0	6.0	10.0
Minimum pheromone	0.001	0.01	0.1	1.0	5.0
Maximum pheromone	0.1	1.0	4.0	8.0	20.0

'Start' is the starting value of the first Hooke and Jeeves direct search. Low–High is the range of the random searches that were conducted to find the starting point for subsequent direct searches. Min–Max is the range within which the direct search was restricted

All values beyond the Min–Max range were replaced by Min or Max.

All tested parameter combinations were passed to the heuristic search algorithm, which solved the forest planning problem using these parameter values. The utility value of the management plan was passed to the Hooke and Jeeves algorithm. Based on this feedback the Hooke and Jeeves algorithm either accepted or rejected the change it made to the parameters of the heuristic.

Each heuristic method was given the same search time. The time limitation was implemented by using a penalty function. The penalty was zero if the used search time was less than the allowed limit and increased linearly up to one when the search time was twice the allowed time. This means that the objective function value (Utility–Penalty) was zero if the search time was equal to twice the allowed search time. All calculations were done with Lenovo ThinkCentre M8500t computer (CPU I7-3770, memory 24 GB).

The allowed time was found by applying all heuristics to the non-spatial and spatial forest planning problem, using parameters recommended in earlier literature (Reeves 1993; Bettinger et al. 2002; Pukkala and Kurttila 2005; Pukkala and Heinonen 2006). Based on these searches, two allowed search times were defined for the non-spatial and spatial problem. The first time, leading to 'fast search,' was the time that the quickest heuristics needed for the search, and the second, leading to 'slow search,' was the time that the slowest heuristics used to complete their search process. In the non-spatial problem, the fast time was 1 s and the slow time was 7 s. In the spatial problems, the fast search time was 10 s, and slow search time was 60 s.

Results

Optimal parameter values of the heuristics

Simulated annealing

For the fast solving of the non-spatial problem, the optimal parameters of SA were 44 initial random searches (the starting point was the best of 44 random solutions) with an initial temperature on 0.002674, freezing temperature of 0.000018 and cooling multiplier of 0.875 (Table 2, Non-spatial, fast). The number of generated candidate moves in the initial temperature was 0.740 times the number of stands (888) resulting in 657 candidates. The search intensity was increased by 5.1 % in every new temperature (iteration multiplier was 1.051). However, there was some variation between the five repeated Hooke and Jeeves optimizations. In the three best ones (2nd, 3rd and 4th solution in Table 2) the number of initial random searches was 18–149, initial temperature ranged from 0.001728 to 0.003150, and freezing temperature ranged from 0.000018 to 0.000098. The search intensity also varied and the two parameters related to search intensity were interrelated so that when the initial intensity was high, it decreased toward the end of the search, and vice versa.

When more time was allowed for the search, the number of initial random searches increased, initial temperature increased, freezing temperature was not much affected, cooling became slightly slower, and the search intensity became slightly higher (Table 2, Non-spatial, slow). These conclusions can be made when looking at the results of all repeated Hooke and Jeeves optimizations.

Table 2 Optimal parameter values of simulated annealing for different problems and solution times found in two to five repeated Hooke and Jeeves optimizations

Utility	P1	P2	P3	P4	P5	P6
Non-spatial, fast (1 s)						
0.871563	*100*	*0.000010*	*0.000005*	*0.914*	*0.550*	*1.095*
0.873159	18	0.003150	0.000098	0.925	1.654	1.018
0.872876	149	0.001728	0.000075	0.896	2.826	0.991
0.873618	**44**	**0.002674**	**0.000018**	**0.875**	**0.740**	**1.051**
0.872072	*142*	*0.006262*	*0.000101*	*0.820*	*2.196*	*1.061*
Non-spatial, slow (7 s)						
0.876482	150	0.002485	0.000010	0.912	1.150	1.080
0.875883	495	0.008804	0.000010	0.894	1.242	1.073
0.876303	196	0.003274	0.000065	0.915	1.683	1.093
0.876773	238	0.008150	0.000019	0.898	1.968	1.067
0.876936	**334**	**0.005014**	**0.000052**	**0.911**	**1.618**	**1.071**
Spatial, fast (10 s)						
0.773342	**1**	**0.001000**	**0.0000120**	**0.916**	**1.338**	**1.050**
0.766615	84	0.005682	0.00000001	0.873	3.603	0.976
0.769635	27	0.003712	0.00000001	0.842	2.817	1.012
Spatial, slow (60 s)						
0.783857	**1**	**0.001990**	**0.000012**	**0.912**	**1.000**	**1.095**
0.780256	*395*	*0.009051*	*0.000010*	*0.894*	*1.236*	*1.073*

The best combination is in boldface, and combinations which were interpreted as inferior are in italics

P1: number of random searches done to obtain the initial solution; P2: starting temperature; P3: freezing temperature; P4: cooling multiplier; P5: search intensity at initial temperature expressed as the proportion of the number of stands; P6: multiplier for search intensity to obtain iteration intensity in the next temperature

The number of initial random searches was smaller in the spatial problems (except in the second optimization for 'Spatial, slow,' which resulted in low utility value), and freezing temperatures were lower than in the non-spatial problems. The main difference between fast and slow search for the spatial problem was slightly higher initial temperature and more quickly increasing search intensity in the slow search.

Threshold accepting

Typical to TA was high variation in the number of initial random searches, suggesting that the search result is not sensitive to this parameter (Table 3). In the fast search for non-spatial problem, the initial threshold was 0.004437–0.009934 and the cooling multiplier was 0.814–0.867. A clear difference to SA was about two times more intensive search at the initial threshold with almost constant search intensity during the whole search process. Increasing search

Table 3 Optimal parameter values of threshold accepting for different problems and solution times found in two to five repeated Hooke and Jeeves optimizations

Utility	P1	P2	P3	P4	P5	P6
Non-spatial, fast (1 s)						
0.871326	*100*	*0.000010*	*0.00000001*	*0.888*	*0.550*	*1.050*
0.873143	140	0.004684	0.000014	0.855	4.053	0.961
0.873014	26	0.005653	0.00000001	0.830	2.814	0.997
0.874443	**1**	**0.009934**	**0.000046**	**0.867**	**4.852**	**0.991**
0.873516	106	0.004437	0.000025	0.814	2.716	1.034
Non-spatial, slow (7 s)						
0.877368	**1**	**0.004217**	**0.000032**	**0.903**	**2.814**	**1.080**
0.877147	731	0.007558	0.000057	0.840	4.737	1.114
0.877256	527	0.008241	0.000046	0.898	4.651	1.053
0.877316	324	0.006659	0.000040	0.862	5.328	1.089
0.877064	85	0.007242	0.000099	0.896	3.068	1.093
Spatial, fast (10 s)						
0.772545	**100**	**0.001247**	**0.000010**	**0.900**	**1.000**	**1.050**
0.769973	1	0.002642	0.00000001	0.864	4.503	0.991
0.770380	27	0.002714	0.0000001	0.866	3.264	0.997
Spatial, slow (60 s)						
0.785600	512	0.002732	0.000020	0.922500	1.112	1.082
0.786061	**596**	**0.004294**	**0.000032**	**0.926779**	**1.956**	**1.063**

The best combination is in boldface, and combinations which were interpreted as inferior are in italics

P1: number of random searches done to obtain the initial solution; P2: starting threshold; P3: freezing threshold; P4: cooling multiplier (<1) to obtain the next threshold; P5: search intensity at initial threshold expressed as the proportion of the number of stands; P6: multiplier of search intensity to obtain iteration intensity at the next threshold

Table 4 Optimal parameter values of great deluge for different problems and solution times found in two to five repeated Hooke and Jeeves optimizations

Utility	P1	P2	P3	P4	P5
Non-spatial, fast (1 s)					
0.870343	**200**	**0.815000**	**0.000134**	**0.998**	**112**
0.869558	114	0.701618	0.000396	0.978	321
0.868180	265	0.878283	0.000464	0.963	901
0.868224	131	0.663305	0.000412	0.965	616
0.868948	189	0.898819	0.000169	0.908	235
Non-spatial, slow (7 s)					
0.873074	199	0.800000	0.00001	0.990	1
0.872446	114	0.716618	0.00001	0.961	446
0.871250	*1077*	*0.742962*	*0.000047*	*0.952*	*275*
0.868355	*836*	*0.831954*	*0.000424*	*0.972*	*923*
0.873127	**214**	**0.883819**	**0.00001**	**0.908**	**360**
Spatial, fast (10 s)					
0.768642	**64**	**0.789687**	**0.000056**	**0.996**	**122**
0.760376	*126*	*0.686618*	*0.000086*	*0.971*	*353*
0.727339	251	0.740539	0.000328	0.960	403
Spatial, slow (60 s)					
0.783397	**100**	**0.845000**	**0.000010**	**0.980**	**200**
0.775985	152	0.682860	0.000017	0.983	359

The best combination is in boldface, and combinations which were interpreted as inferior are in italics

P1: number of random searches done to obtain the initial solution; P2: initial water level; P3: initial rain; P4: rain multiplier; P5: number of direct access iterations at the end of the search

time led to higher number of initial random searches, slower decrease in threshold and more intensive search.

The optimal parameters for a fast search in the spatial problem did not differ much from those for a fast search in the non-spatial problem. The starting threshold was systematically lower, and the freezing threshold was always low in the spatial problem. For the slow search in spatial problem, TA adopted a higher number of initial random searches, slower cooling and gradually increasing search intensity.

Great deluge

In GD, fast search for the non-spatial problem used 114–200 initial random searches and 112–901 direct ascent search steps at the end of optimization (Table 4, Non-spatial, fast). Between them, the great deluge itself started from an initial water level of 0.663–0.899, initial rain of 0.000134–0.000464 and rain multiplier of 0.908–0.998. When more time was allowed for the search, the main and most systematic difference was smaller initial rain, which resulted in slower cooling (Table 4, Non-spatial, slow). The optimal parameters for the spatial problem were not systematically different from the optimal parameters for non-spatial problem. The same difference was found between fast and slow search: Initial rain was smaller when more time was allowed for the search, resulting in slower cooling.

Tabu search

In fast tabu search for the non-spatial problem, TS used 5000–7000 iterations, produced 4–12 candidate moves per iteration and forbade the treatment schedules that participated in the last move from participating in another move for 18 to 43 iterations (Table 5). Note that when the tabu tenure is longer for the entering schedule (P4 in Table 5) than for the leaving schedule (P3), parameter P3 has no effect on the search process since all schedules of the stands are tabu as long as the entering schedule is in the tabu list.

The number of iterations increased when 7 s was allowed for the search instead of 1 s. The other parameters

Table 5 Optimal parameter values of tabu search for different problems and solution times found in two to five repeated Hooke and Jeeves optimizations

Utility	P1	P2	P3	P4
Non-spatial, fast (1 s)				
0.871251	3456	6	10	5
0.871239	3801	10	15	18
0.871146	4470	8	9	0
0.871922	**4996**	**8**	**(31)**	**49**
0.870744	*703*	*100*	*53*	*32*
Non-spatial, slow (7 s)				
0.871831	*8485*	*5*	*20*	*7*
0.873206	41,237	4	48	20
0.873884	**38,010**	**4**	**(35)**	**47**
0.873486	38,854	4	6	6
0.873064	32,439	3	7	22
Spatial, fast (10 s)				
0.758833	5990	19	14	5
0.763130	**10,646**	**13**	**(31)**	**51**
0.759913	11,877	10	(7)	12
Spatial, slow (60 s)				
0.763935	15,970	10	(5)	15
0.767338	**41,024**	**11**	**(2)**	**6**

The best combination is in boldface, and combinations which were interpreted as inferior are in italics. When the tabu tenure is longer for the entering schedule (P4) than for the leaving schedule (P3), parameter P3 has no effect on the search. These cases are in parentheses

P1: number of iterations; P2: number of candidate moves produced per iteration; P3: length of the tabu tenure for leaving schedules; P4: length of the tabu tenure for entering schedules

Table 6 Optimal parameter values of genetic algorithm for different problems and solution times found in two to five repeated Hooke and Jeeves optimizations

Utility	P1	P2	P3	P4	P5
Non-spatial, fast (1 s)					
0.777793	*15*	*10*	*100*	*1.000*	*0.000*
0.798590	12	12	116	5.278	7.552
0.811681	2	30	237	4.609	5.050
0.824934	**2**	**13**	**160**	**9.374**	**9.368**
0.777383	*44*	*1*	*248*	*8.502*	*2.008*
Non-spatial, slow (7 s)					
0.861927	2	50	1050	3.000	5.000
0.868495	2	74	1151	2.899	11.287
0.866467	2	97	954	10.024	3.056
0.869369	**2**	**88**	**1207**	**1.221**	**11.407**
0.869210	2	91	1453	5.558	8.448
Spatial, fast (10 s)					
0.672784	**4**	**15**	**213**	**8.859**	**6.050**
0.652924	11	10	148	2.125	1.500
0.651844	14	14	65	5.216	7.302
Spatial, slow (60 s)					
0.719818	**5**	**89**	**863**	**8.975**	**4.237**
0.719461	2	60	961	8.000	4.000

The best combination is in boldface, and combinations which were interpreted as inferior are in italics

P1: number of chromosomes; P2: number of random searches that are used to produce an initial chromosome; P3: number of generations; P4: initial mutation rate (at first generation); P5: final mutation rate (at last generation)

were not systematically affected. The spatial problem seems to need a higher number of candidate moves and longer tabu tenures for those treatment schedules that a removed from the solution. The main difference between fast and slow search was higher number of iterations in the slow search.

Genetic algorithm

The overall observation that characterized the optimal parameter values of GA was very low number of chromosomes, which was the lowest possible (2) in a majority of optimizations (Table 6). This means that the algorithm operated with only two solutions that were combined to obtain an off-spring, after which one of the three solutions was removed from the population. Most probably, the reason for such a low number of solutions was the short allowed search time. With a 'normal' population size, GA would be clearly slower than the other heuristics analyzed in this study (except AC).

In the fast search processes (both non-spatial and spatial), the number of random searches that were used to produce each of the chromosomes was 10–30, and the number of generations was 65–237. The number of mutated 'genes' (stands) was 1–10, which means that the treatment schedules of 1–10 stands out of the 888 stands of the case study forest were changed after the first version of the off-spring was produced by crossing-over. The mutation rate sometimes increased (final mutation rate, P5 was higher than P4) and sometimes decreased (P4 was higher than P5) during the search process. This suggests that it would be enough to use a constant mutation rate (of about 5/888) during the whole search.

Increasing search time increased the number of generations. Also the number of random searches that were used to produce an initial solution increased. These changes were fairly systematic.

Ant colony optimization

For fast AC search either the number of ants or the number of ant iterations was low (Table 7). It seems that fast search

Table 7 Optimal parameter values of ant colony optimization for different problems and solution times found in two to five repeated Hooke and Jeeves optimizations

Utility	P1	P2	P3	P4	P5	P6	P7
Non-spatial, fast (1 s)							
0.789248	*30*	*10*	*0.200*	*1.500*	*1.000*	*0.100*	*4.000*
0.771419	*18*	*5*	*0.660*	*1.007*	*1.795*	*0.352*	*2.361*
0.764368	*46*	*3*	*0.384*	*1.135*	*0.184*	*0.001*	*6.480*
0.767516	*13*	*5*	*0.588*	*1.199*	*3.396*	*0.596*	*6.615*
0.838710	**2**	**150**	**0.569**	**1.118**	**4.319**	**0.001**	**4.249**
Non-spatial, slow (7 s)							
0.777701	*19*	*5*	*0.135*	*1.500*	*0.410*	*0.100*	*4.000*
0.867802	**43**	**45**	**0.578**	**1.082**	**2.699**	**0.001**	**4.614**
0.783820	*27*	*30*	*0.259*	*0.985*	*1.678*	*0.767*	*7.840*
0.779662	*11*	*41*	*0.474*	*1.015*	*3.729*	*0.537*	*6.727*
0.782094	*9*	*68*	*0.693*	*1.063*	*1.772*	*0.681*	*3.625*
Non-spatial, very slow (60 s)							
0.869057	**30**	**100**	**0.135**	**1.650**	**1.885**	**0.051**	**4.000**
0.868354	22	89	0.367	1.931	2.433	0.039	6.810
0.866709	12	244	0.048	1.672	2.265	0.151	6.626
Spatial, fast (10 s)							
0.660024	**14**	**100**	**0.070**	**1.350**	**0.010**	**0.001**	**3.650**
0.640932	7	5	0.578	1.232	2.385	0.253	2.361
0.647730	34	5	0.200	1.182	4.975	0.489	3.783
Spatial, slow (60 s)							
0.712679	30	100	0.143	1.800	0.410	0.100	3.300
0.713957	**33**	**89**	**0.424**	**1.931**	**5.383**	**0.039**	**6.110**
0.710568	12	244	0.113	1.972	2.264	0.151	7.283
Spatial, very slow (120 s)							
0.712785	41	100	0.200	2.100	1.000	0.100	3.300
0.721106	47	175	0.146	1.251	2.550	0.012	7.041
0.684041	9	317	0.058	2.592	5.416	0.299	6.110

The best combination is in boldface, and combinations which were interpreted as inferior are in italics

P1: number of ants; P2: number of iterations; P3: evaporation rate (ρ in Eq. 2); P4: probability exponent (α in Eq. 4); P5: initial pheromone; P6: minimum pheromone; P7: maximum pheromone

with reasonable results is only possible with a very low number of ants. In this respect, the results correspond to those obtained for GA: Small population size (chromosomes or ants) must be used in fast search. Both the number of ants and the number of iterations increased when more time was allowed for the search.

Since the results still varied a lot between repeated optimizations, especially in terms of utility value, additional optimizations were conducted with a longer search time (very slow in Table 7) to see whether the results of repeated optimizations would stabilize, making it easier to make conclusions about suitable parameter values. Now the utility values stabilized, and the results suggested 12–30 ants with 89–244 ant iterations. If all optimizations for the non-spatial problem that resulted in good utility values are looked together, it may be concluded that a suitable maximum pheromone is 4–6, minimum pheromone is 0.001–0.15, and the initial pheromone should be 2–4. Parameter α of Eq. 4 should be 1.1–1.9. It seems that longer allowed search times lead to higher number of ants and iterations and lower evaporation rate.

Also in the spatial problem the number of ants or ant iterations tended to increase with increasing allowed search time. However, there seems to be much uncertainty about the optimal values of some parameters. This may be because of the interrelationships between parameters. For example, evaporation rate was often low when the initial pheromone was low and higher evaporation rates often occurred with high initial pheromone levels.

Search process with optimal parameter values

Figure 2 shows examples of the development of the utility value in the search process for the non-spatial problem in the slow search (7 s). The best parameter values (boldface in Tables 2–6) were used. Typical of all cooling methods (SA, TA and GD) was the high but decreasing oscillation of the utility value of the current solution and a gradual increase in the utility of the best solution found so far. The reason for the decreasing oscillation was decreasing probability to accept inferior moves, i.e., cooling. In GD, the lower limit of the range of oscillation increased steadily, which was a result of rising water level. The utility of SA reached a high level later than in TA, which is because of higher number of initial random searches in SA (334, see Table 2) than in TA (1, Table 3). However, the number of initial random searches varied a lot between repeated Hooke and Jeeves optimizations for both SA and TA, which means that since both of these methods are fast, they can spend a part of the allowed search time for initial random searches although these searches are not critically important for the outcome of the search.

The corresponding diagram looked quite different for TS (Fig. 2, Tabu search), which quickly found a good solution, which could not be improved much during the rest of the search process. The range of oscillation did not decrease during the search process because there was no cooling. Inferior moves were to be accepted during the whole search process because of constant tabu tenures.

In GA, the utility of the best chromosome and the average utility of all chromosomes increased almost linearly during the whole search process. This means that GA needed the whole allowed time to find a good solution.

Figure 3 shows the development of the utility of the best-so-far and iteration-best ants and the mean utility of all ants of the current iteration in fast, slow and very slow AC search for the non-spatial problem. When the number of

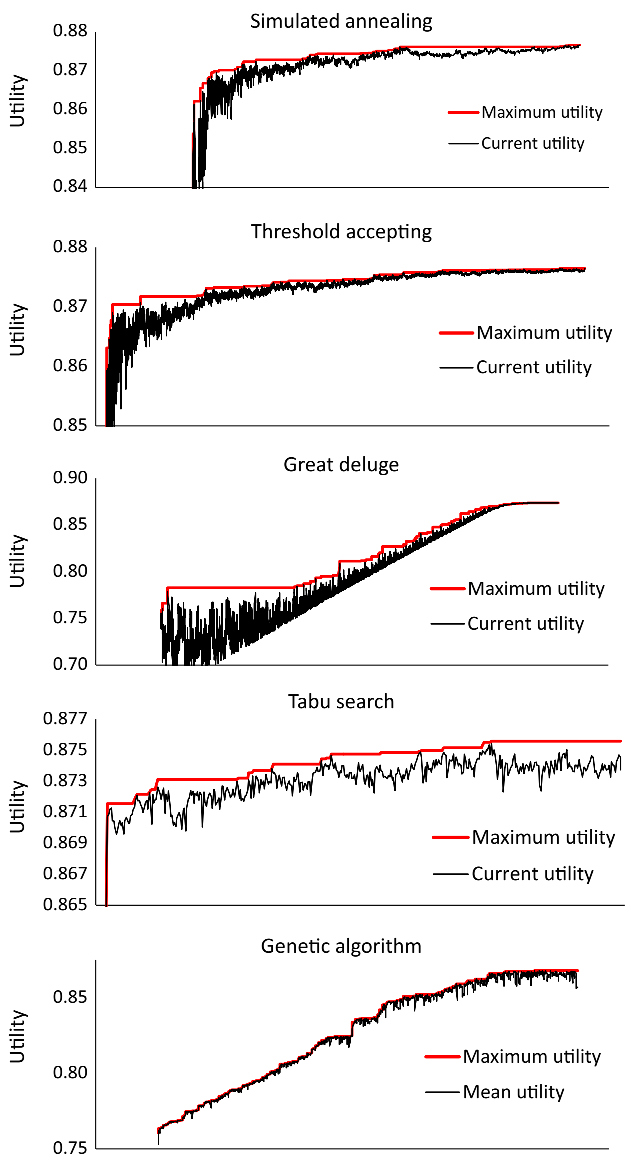

Fig. 2 Development of the utility of the current solution and the best solution found so far in the slow search for non-spatial problem when optimal parameter values are used for each heuristic. In genetic algorithm, which operates with several solutions, the mean utility of the current population of solution vectors is shown

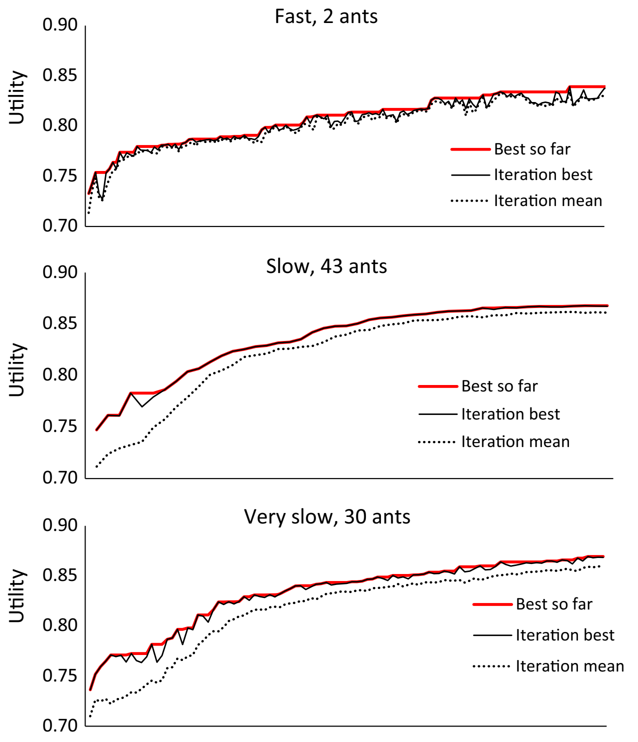

Fig. 3 Development of the utility of the best-so-far and iteration-best ant and the mean utility of all ants of the current iteration in the fast, slow and very slow search for non-spatial problem when optimal parameter values are used

ants was 46, the iteration-best became almost always the new best-so-far ant. With fewer ants, the curve for the iteration-best ant often differed from the best-so-far curve, and sometimes it took several iterations to improve the best-so-far solution. When there were more than two ants, differences in their utility values decreased gradually and the mean utility approached the utility of the best ant.

The development of the 10-year harvests during the search was similar in SA and TA (Fig. 4) after the end of the random searches, which lasted longer in SA. The pattern was completely different in GD, which needed almost the whole allowed time to find solutions in which the harvested volume was close to 300,000 m^3 for all three 10-year periods. In this respect, GD and GA were similar, but there was less oscillation in GA (Fig. 4 bottom). Common to both methods was that they used the whole allowed computing time to reach a solution which satisfied the cutting targets and had a good utility value.

TS very quickly found a solution in which the 10-year harvests were near 300,000 m^3, after which they varied only little during the rest of the search. Common to all methods was the gradually increasing harvested volume of the first 10-year period and decreasing harvest of the third period.

In the fast AC search the harvest of the first 10-year period did not reach the required 300,000 m^3 suggesting that 1 s was a too short search time for the AC heuristic to find good solutions (Fig. 5). The required harvested volumes were reached in the slow and very slow search, and the development of the periodical harvests resembled to those in GA. The smoother progress of the harvested volume in the slow search can be explained by the high number of ants resulting in a new best-so-far solution at every iteration.

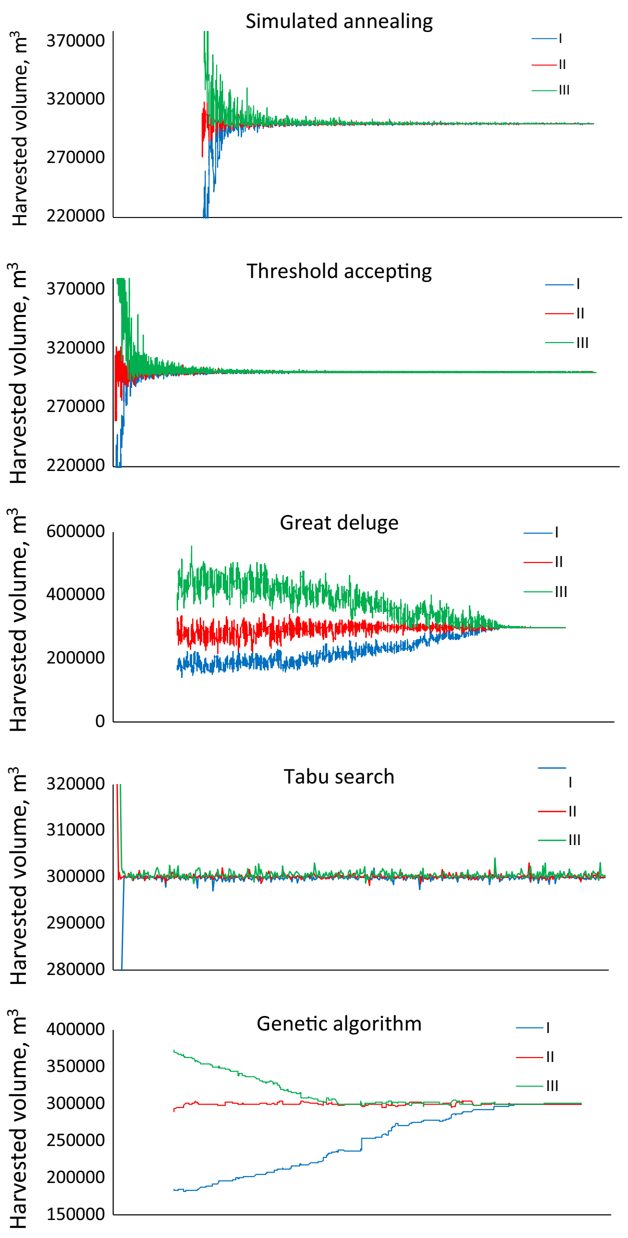

Fig. 4 Development of 10-year harvests in the current solution in the slow search for non-spatial problem when optimal parameter values are used for each heuristic. For genetic algorithm, the harvested volumes of the best solution of the current population are shown. The *Roman number* (*I*, *II* and *III*) refers to the 10-year period

Ranking of the heuristics

In the fast search for the non-spatial problem, TA, SA and TS were the best methods and GA and AC were clearly worse than the other heuristics (Fig. 6, top). All methods benefitted from increased computing time, but the improvement of utility was much larger in GA and AC than in the other methods. SA was the best method with 7-s computing time, but TA was nearly as good. GA was again

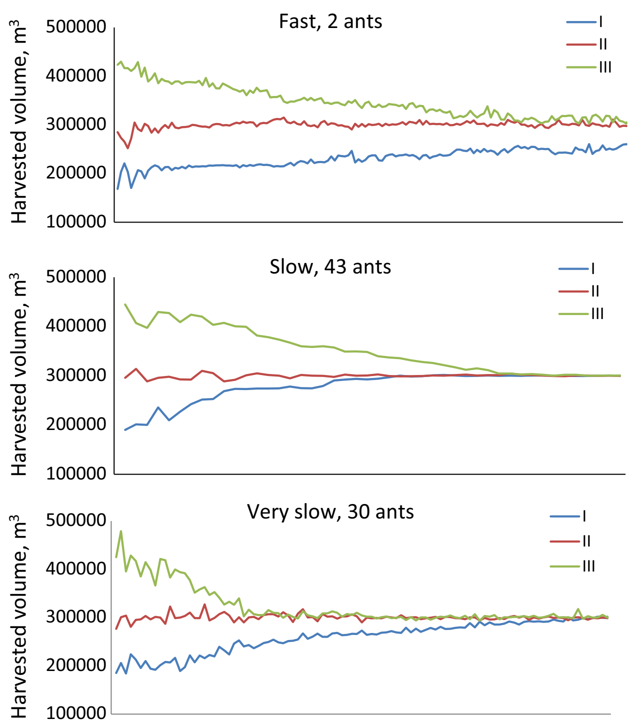

Fig. 5 Development of 10-year harvests for the iteration-best ant in the fast, slow and very slow AC search for non-spatial problem when optimal parameter values are used. The *Roman number* (*I*, *II* and *III*) refers to the 10-year period

the worst method, and GD and TS were halfway between the two worst (GA and AC) and the two best (SA and TA) methods.

In the spatial problem, SA was the best method in the fast search (10 s) and TA was the second, but the order was reversed when the search process was allowed to last for 60 s (Fig. 6, bottom). GA and AC were again clearly weaker than the other methods, and GD was better than TS. Based on all the optimizations, it is easy to rank the five tested heuristics as follows: SA and TA are the best methods, GD is the third in ranking, TS is the fourth, and GA and AC are the least satisfactory heuristics when the same computing time is allowed for all heuristics. AC was slightly better than GA.

Discussion

This study is the second one in which optimization was used to find the best combinations of parameters that guide the search processes of heuristic methods used in forest management planning (the first is Pukkala and Heinonen 2006). The study is the first one that optimized the parameters of great deluge, genetic algorithm and ant colony optimization. The Min–Max Ant System was used for first time in forest management planning.

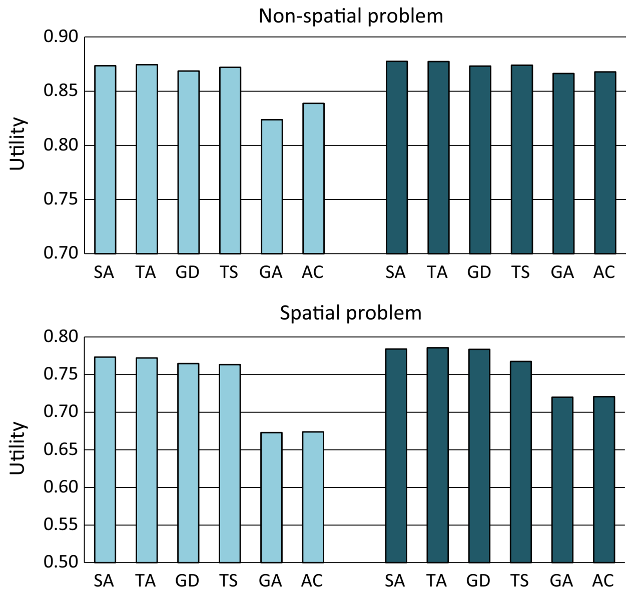

Fig. 6 Utility values obtained with the optimal parameter values found for different heuristic methods in the fast (*left*, *light tone*) and slow (*right*, *dark tone*) search for the spatial and non-spatial forest planning problem

The purpose of parameter optimization was to help other users of heuristic methods to set their parameters so that good solutions are found in reasonable computing time. Although the results of repeated optimizations varied, some conclusions on recommendable parameter values for similar forest planning problems can be given. For example, the cooling multiplier of SA should be about 0.9. This is a significant result since very fast cooling (e.g., 0.7 or 0.8) and very slow cooling (e.g., 0.95 or 0.99) could be ruled out. A suitable number of iterations in each temperature were 1–2 times the number of stands. If the search time does not need to be very short, the number of iterations could be increased by 5 % in every new temperature. The number of random searches that are done to obtain the initial solution for the SA heuristics is not an important parameter.

The results were equally clear for the TA heuristic. More problematic parameters, from the generalization point of view, are the initial and freezing temperatures of SA, and the initial and freezing thresholds of TA. However, generalization becomes easier if these parameters are related to the unit and range of variation of the objective function and the influence that one move (or stand) can have on the objective function value. For example, the optimal initial temperature of SA was found to be approximately 0.002 in most cases. In the utility maximization of this study, the maximum influence of a stand on the OF value of the solution is approximately 1/888 = 0.001126, i.e., the maximum possible utility divided by the number of stands. A suitable initial temperature was equal to the maximum value of the objective function divided by the number of stands and multiplied by two.

The results on starting and cooling temperature may be applied to other cases as follows. If the maximized variable is net present value, its maximum value is approximately 2,500,000 USD, and there are 1500 stands, a suitable starting temperature would be $2 \times 2{,}500{,}000/1500 = 3333$. The magnitude of freezing temperature should 0.5–1 % of the initial temperature, i.e., around 15–30. Similar rules apply to the initial and freezing thresholds of the TA heuristic. This shows that although the optimal parameters of the heuristics are problem-specific, some generalizations to similar forest planning problems can be made from the results of this study. Therefore, the results of this study make it easier for forest managers to find suitable parameters for the heuristic methods they use to solve forest planning problems.

Generalization would be easier if parameter optimizations are made for several forests and forest planning problems varying in size, objective function and constraints. This would make it possible to develop rules or even models for the optimal parameter values of the heuristics.

The optimal parameter values for the SA, TA and TS heuristics were reasonably close to those obtained by Pukkala and Heinonen (2006). This earlier study suggests slightly slower cooling in SA and TA (a cooling multiplier of 0.92–0.94) than found in our study (0.9). To compensate for the increased search time due to slower cooling, the number of iterations at the initial temperature or threshold was lower in the study of Pukkala and Heinonen (2006). They found that the number of candidate moves that are generated at each iteration of tabu search was mostly 1–5 % of the number of stands, whereas the corresponding proportion in our study was mostly 0.5–1.2 % of stands. The length of the tabu tenure was of the same magnitude in both studies.

Our study suggests that SA and TA are the best heuristics for typical combinatorial forest planning problems if the search time cannot be very long. Also in Pukkala and Heinonen (2006) TS was slightly inferior to SA and TA. Similarly to Zeng et al. (2007), our study suggests that SA works better in forest planning problems than the GA and AC heuristics. Bettinger et al. (2002) found that all cooling methods (SA, TA and GD) worked better than TS and GA, which is in line with our study. However, when tabu search was used with 2-opt moves, it was equally good as the cooling methods. A 2-opt move means that the move consists of a simultaneous change in the treatment schedule of two stands. Pukkala and Kurttila (2005) found that SA was better than TS and GA in simple forest planning problems, but GA was the best method in the most complicated problem that included two hierarchical levels and both spatial and non-spatial objective variables. However, similarly to our study, GA needed a longer time to find good solutions. Liu et al. (2006) found SA to be a better heuristic than the GA and hill climbing (HC) algorithms for solving a spatially constrained forest planning problem. SA and HC were approximately 10 times faster than GA.

As a conclusion, our study suggests that the cooling methods, especially SA and TA, are the most recommendable heuristics for solving combinatorial forest planning problems when short search time is important. When the solutions must be found quickly, for instance in an iterative analysis of the trade-offs between different objectives, the population-based heuristics GA and AC may not be good choices. However, if search time is not an issue, none of the heuristics should be ruled out. The results obtained for different heuristics depend on the way the methods are implemented as an algorithm and a computer program. There is much flexibility in the implementation of the heuristics, especially TS, GA and AC. Therefore, the results obtained for them in this study would improve if more efficient implementations are developed.

Acknowledgments This research was financially supported by the Ministry of Science and Technology of the People's Republic of China (2015BAD09B01) and the Fundamental Research Funds for the Central Universities of the People's Republic of China (2572014BA09). The authors thank the teachers and students of the Department of Forest Management, Northeast Forestry University (NEFU), PR China, who collected the data for this study.

References

Baskent EZ, Keles S (2005) Spatial forest planning: a review. Ecol Model 188:145–173

Bettinger P, Graetz D, Boston K, Sessions J, Chung W (2002) Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. Silva Fennica 36:561–584

Bettinger P, Boston K, Siry JP, Grebner DL (2009) Forest management and planning. Academic Press Elsevier, Amsterdam **(ISBN 978-0-12-374304-6)**

Blum C (2005) Ant colony optimization: introduction and recent trends. Physics of Life Reviews 2:353–373

Borges JG, Hoganson HM, Falcão AO (2002) Heuristics in multi-objective forest management. In: Pukkala T (ed) Multi-objective forest planning. Kluwer Academic Publisher, Dordrecht **(ISBN 1-4020-1097-4)**

Boston K, Bettinger P (1999) An analysis of Monte Carlo Integer programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. Forest Science 45:292–301

Boston K, Bettinger P (2002) Combining tabu search and genetic algorithm heuristic techniques to solve spatial harvest scheduling problems. Forest Science 48:35–46

Constantino M, Martins I, Borges JG (2008) A new mixed integer programming model for harvest scheduling subject to maximum area restrictions. Oper Res 56:542–551

Dorigo M, Birattari M, Stützle T (2006) Ant colony optimization. Artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine, November 2006: 28-39

Eyvidson K, Kangas A (2014) Stochastic goal programming in forest planning. Can J For Res 44(10):1274–1280

Falcão AO, Borges JG (2002) Combining random and systematic search heuristic procedures for solving spatially constrained forest management scheduling problems. Forest Science 48:608–621

Haight RG, Brodie JD, Adams DM (1985) Optimizing the sequence of diameter distributions and selection harvests for uneven-aged stand management. Forest Science 31(2):451–462

Heinonen T, Pukkala T (2007) The use of cellular automaton approach in forest planning. Can J For Res 37:2188–2200

Hoganson HM, Rose DW (1984) A simulation approach for optimal timber management scheduling. Forest Science 30:220–238

Hooke R, Jeeves TA (1961) "Direct search" solution of numerical and statistical problems. Journal of the Association for Computing Machinery 8:212–229

Islam NMd, Kurttila M, Mehtätalo L, Pukkala T (2010) Inoptimality losses in forest management decisions caused by errors in an inventory based on airborne laser scanning and aerial photographs. Can J For Res 40:2427–2438

Jin X, Pukkala T, Li F (2016) A management planning system for even-aged and uneven-aged forests in northeast China. Journal of Forestry Research (in print)

Könnyű N, Tóth SF, McDill ME, Rajasekaran B (2014) Temporal Connectivity of Mature Patches in Forest Planning Models. Forest Science 60:1089–1099

Kurttila M, Pukkala T, Loikkanen J (2002) The performance of alternative spatial objective types in forest planning calculations: a case for flying squirrel and moose. For Ecol Manage 166:245–260

Liu G, Han S, Zhao X, Nelson JD, Wang H, Wang W (2006) Optimization algorithms for spatially constrained forest planning. Ecol Model 194:421–428

Mathey A-H, Krcmar E, Tait D, Vertinsky I, Innes J (2007) Forest planning using co-evolutionary cellular automata. For Ecol Manage 239(1–3):45–56

Palahí M, Pukkala T, Pascual L, Trasobares A (2004) Examining alternative landscape metrics in ecological forest landscape planning: a case for capercaillie in Catalonia Investigaciones Agrarias: Sistemas de Recursos Forestales 13(3):527–538

Pukkala T (2007) Population-based methods in the optimization of stand management. Silva Fennica 43(2):261–274

Pukkala T, Heinonen T (2006) Optimizing heuristic search in forest planning. Nonlinear Analysis: Real World Applications 7:1284–1297

Pukkala T, Kangas J (1993) A heuristic optimization method for forest planning and decision making. Scand J For Res 8:560–570

Pukkala T, Kurttila M (2005) Examining the performance of six heuristic optimization techniques in different forest planning problems. Silva Fennica 39(1):67–80

Pukkala T, Heinonen T, Kurttila M (2009) An application of the reduced cost approach to spatial forest planning. Forest Science 55(1):13–22

Reeves CR (ed) (1993) Modern heuristic techniques for combinatorial problems. Blackwell Scientific Publications, Oxford

Strange N, Meilby H, Thorsen JT (2002) Optimizing land use in afforestation areas using evolutionary self-organization. Forest Science 48(3):543–555

Valsta L (1992) An optimization model for Norway spruce management based on individual-tree growth models. Acta Forestalia Fennica 232

Zeng H, Pukkala T, Peltola H, Kellomäki S (2007) Application of ant colony optimization for the risk management of wind damage in forest planning. Silva Fennica 41(2):315–332